Methodology article

Organization Studies
1–24

DOI: 10.1177/01708406211069438
www.egosnet.org/os


The effect of repertoire, routinization and enacted complexity: Explaining task performance through patterns of action

Mathias Hansson
Inland University of Applied Sciences and BI Norwegian Business School, Norway

Thorvald Hærem
BI Norwegian Business School, Norway

Brian T. Pentland
Michigan State University, USA

Abstract

We use pattern mining tools from computer science to engage a classic problem in organizational theory: the relation between routinization and task performance. We develop and operationalize new measures of two key characteristics of organizational routines: repertoire and routinization. Repertoire refers to the number of recognizable patterns within a stream of actions used to perform the task, and routinization refers to the degree to which a stream of actions follows recognizable patterns. We use these measures to develop a novel theory that predicts task performance based on the size of repertoire, the degree of routinization, and enacted complexity. We test this theory in two settings that differ in their programmability: crisis management and invoice management. We find that repertoire and routinization are important determinants of task performance in both settings, but with opposite effects. In both settings, however, the effect of repertoire and routinization is mediated by enacted complexity. This theoretical contribution is enabled by the methodological innovation of pattern mining, which allows us to treat routines as a collection of sequential patterns or paths. This innovation also allows us to clarify the relation of routinization and complexity, which are often confused because the terms routine and routinization connote simplicity. We demonstrate that routinization and enacted complexity are distinct constructs, conceptually and empirically. It is possible to have a high degree of routinization and complex enactments that vary each time a task is performed. This is because enacted complexity depends on the repertoire of patterns and how those patterns are combined to enact a task.

Corresponding author:
Mathias Hansson, Inland Norway University of Applied Sciences, Postbox 400, Elverum, 2418, Norway.
Email: mathias.hansson@inn.no

Keywords
enacted complexity, organizational routines, pattern mining, patterns of action, performance programs, programmability, task performance

Introduction

The idea of creating performance programs to improve efficiency, quality, and managerial control has its roots in the classic works of organization theory (e.g., Cyert & March, 1963; March & Simon, 1958). Now, it is woven into the fabric of contemporary managerial practice. ISO 9000, total quality management (TQM), and business process management (BPM) all represent efforts to *program* or *routinize* decisions, tasks, and processes (Benner & Tushman, 2003). BPM has become a multi-billion-dollar market that is growing rapidly (Srivastava, Jain, & Rashid, 2020). As this technology continues to expand its scope and influence, it is useful to reexamine the concepts of programmability and routinization in a theoretically grounded way. In principle, programmable tasks should be easier to routinize. The digital society's increasing ability to program task resolutions means that more tasks are getting programmed. Still, we lack the tools to operationalize routinization and test its relation to task performance.

To help address this gap, we introduce and test a novel theoretical model that predicts task performance based on three concepts: repertoire, routinization, and enacted complexity. *Repertoire* is the number of distinct, recognizable patterns within a stream of actions used to perform the task. *Routinization* is the degree to which a stream of actions follows recognizable patterns. *Enacted complexity* (Hærem, Pentland, & Miller, 2015; Danner-Schröder & Ostermann, 2020) reflects the number of possible ways the task can be performed, given the enacted network of actions. We test this model in two settings - one more programmable and one less programmable.

We focus on a concrete research question: *What is the effect of repertoire, routinization, and enacted complexity on task performance?* Based on observable behaviors, we use pattern mining tools from computer science (Fournier-Viger, Lin, Kiran, Koh, & Thomas, 2017) to provide measures of routinization and repertoire. Using one example from a more programmable setting (invoice management) and one from a less programmable setting (crisis management), we demonstrate that routinization and repertoire influence performance via a third concept: enacted complexity (Hærem et al., 2015). As expected, we find that a larger repertoire and lower routinization are beneficial in crisis management, but the opposite is true in invoice processing. We find that the repertoire of action patterns and the degree of routinization influence performance both directly and indirectly through enacted complexity, but that their effects are opposite depending on the programmability of the task.

This article offers three contributions. The first contribution is methodological: we provide an objective way to conceptualize and operationalize repertoire and routinization. These terms have been used for decades but never clearly defined or operationalized in terms of action patterns. We offer a behavioral measure of routinization that is based on the percentage of actions that fit recognizable patterns. Because they are based on observable behavior, these measures are less subject to the biases that influence perceptual self-report measures (Hærem & Rau, 2007) and enable us to describe performances in a new, systematic manner. Like a new kind of microscope, pattern mining allows us to see phenomena we could not see before.

The second contribution is theoretical: we use these constructs to create and test a simple model that connects the observable properties of routines (repertoire, routinization, and enacted complexity) to measurable task outcomes (speed and accuracy). The model explains why the effects of repertoire and routinization are contingent on the programmability of the task by accounting for the

effects of action patterns. This theoretical innovation would not have been possible without the methodological innovation of pattern mining.

Finally, we revisit the concept of performance programs with current theories of routine dynamics (Feldman et al., 2022). In doing so, we revitalize the abandoned concept of programmability. March and Simon (1958) remains one of the most influential books in organization theory, and the concepts of programs and routines are among its most enduring ideas (Anderson & Lemken, 2019). As we enter the age of machine learning, robotics, and artificial intelligence, the concepts of programmability and routines take on new meanings and new relevance for work and organization.

Theory

Programs versus routines

As Anderson and Lemken (2019) point out, "routines" and "performance programs" have been deeply intertwined in the organizational literature since March and Simon joined them together sixty years ago. Simon (1977, p. 46) argued that "Decisions are programmed to the extent that they are repetitive and routine, to the extent that a definite procedure has been worked out for handling them so that they don't have to be treated from scratch each time they occur." This quote shows a tendency to equate "programmed" with "repetitive and routine."

However, a long line of field research on organizational routines shows that enacted behavior (the routine) does not always match the program (Berente, Lyytinen, Yoo, & King, 2016). Thus, it is best to treat routines and programs as distinct.[1] Blurring the distinction may lead to some of the confusion documented by Kay (2018). Following the usage in March and Simon (1958), economists tend to use "programs," "routines," and "standard operating procedures" as synonyms. In contrast, the literature on routine dynamics (Feldman et al., 2022) generally treats "routines" and "programs" as distinct concepts, as we do in this article. The recent developments in pattern mining techniques make it possible to characterize routines by the specific patterns of actions that were performed. These developments, therefore, reinforce the importance of distinguishing between routines and programs.

With this distinction in mind, the question naturally arises: how much of an enacted pattern of behavior should we try to codify into a program? As technology continues to advance, it seems feasible to encode more and more behavior into programs, but when will that be desirable? Our ambition is not to provide a definitive answer, but rather to contribute to a foundation to address such questions. To do that we start back at the roots of these concepts.

Programming behavior

The idea that organizational behavior can be codified into performance programs originated in the early work of the Carnegie School (Simon, 1947; March & Simon, 1958). Table 1 lists a range of factors that influence the programmability of a task.

March and Simon (1958) argued that tasks that are repeated frequently and that require little search are most programmable. Perrow (1967) focused on exceptions and analyzability as the primary drivers of search. When exceptions are rare and easy to analyze, the task is more programmable. With low analyzability of exceptions, search will be unsystematic and characterized by guesswork. Frequent exceptions and low analyzability increase search, making a task less programmable. Unsystematic guesswork can also be caused by a lack of clarity in means–ends relations (Thompson, 1967; Newell & Simon, 1972), by high equivocality (Daft & Weick, 1984), or a

Table 1. Factors that influence task programmability.

	More programmable	Less programmable	Foundational references
Frequency of task repetition (access to experience)	High	Low	March & Simon (1958)
Required search	Low	High	March & Simon (1958)
Number of exceptions	Few	Many	Perrow (1967)
Analyzability of exceptions	High	Low	Perrow (1967)
Means–ends relations	Clear	Unclear	Thompson (1967); Newell & Simon (1972)
Quality of feedback environment	Kind	Wicked	Hogarth (2001)

wicked feedback environment (Hogarth, 2001; Rice & Cooper, 2010). In any event, tasks with such characteristics will be less programmable.

Example 1: Crisis management. Crisis management exemplifies a less programmable task. A crisis "is characterized by ambiguity of cause, effect, and means of resolution, as well as by a belief that decisions must be made swiftly" (Pearson & Clair, 1998, p. 61). As a crisis unfolds, high levels of uncertainty and interdependence make it difficult to know which actions will be effective (Perrow, 2011; Weick & Sutcliffe, 2015). In a crisis, *programmed* responses—"fixed responses to defined stimuli" (March & Simon, 1958, p. 142)—will probably not result in desirable outcomes. Professionals in many settings, for example, medical doctors, police officers, military personnel, firefighters, and even business managers, must handle unexpected situations that may turn into a crisis (e.g., Rudolph, Morrison, & Carroll, 2009; Weick & Sutcliffe, 2015). We can try to program responses for data breaches, terrorist threats, and pandemics, but as Field Marshal Helmuth Karl Bernhard Graf von Moltke observed, no plan survives first contact with the enemy.[2]

Example 2: Invoice management. Even during a crisis, organizations deal with flows of ordinary, repetitive transactions. For example, supply chains generate rivers of orders, invoices, and payments that need to be matched, approved, and audited. A programmed response is not only effective, it is essential. In most organizations, such tasks are literally programmed, in the sense that computerized systems implement the processes and business rules that are needed to support or even automate the work (Dumas, La Rosa, Mendling, & Reijers, 2013). In between these extremes, organizations struggle with a range of more or less programmable tasks that need to be managed efficiently and effectively.

Taking Measure of Enactments

Pattern mining provides a way to define and analyze key properties of routines as they are enacted. Enactment corresponds to the performative aspect of a routine, as defined by Feldman and Pentland (2003). In this section, we define three measures based on enactments: repertoire, routineness, and enacted complexity.

Mining sequential patterns

Pattern mining algorithms depend strongly on repetition. Observing a sequence once is not sufficient evidence to recognize it as a pattern. This aligns with the standard definition of a routine as a

"repetitive, recognizable pattern" (Feldman & Pentland, 2003, p. 96), but we view a routine as a dynamic collection of smaller patterns, rather than a single, big pattern, as implied in the standard definition.

If we observe the same sequence in every performance of a routine, we have strong evidence for recognizing it as a pattern. Frequency of repetition is explicitly operationalized in the pattern recognition algorithm we use here (Fournier-Viger et al., 2017; Fournier-Viger, Wu, Gomariz, & Tseng, 2014b). It is called support.

Consider the example of invoice processing in an accounting department. We might recognize several patterns of action (e.g., "receive an invoice, capture the information on the invoice, send to the person who ordered for approval, check amount, check goods, approve, pay…"). However, the identical pattern does not occur in every transaction. In practice, they are adjusted to address exceptions, errors, interruptions, and other factors, so some patterns receive more support than others. By definition, repetitive patterns are supported more strongly in the data, and repetitive patterns are easier to recognize.

The application of pattern mining methods to research on routines is still quite rare. Cohen and Bacdayan (1994) examined patterns of action in a card game but without the use of computerized tools for pattern mining. Stachowski, Kaplan, and Waller (2009) studied team interaction patterns using a pattern-mining tool called THEME. Su, Brdiczka, and Begole (2013) studied temporal patterns in the individual use of computer software. However, these studies emphasized clock time—how quickly one event follows the next—as a key indicator of routinization. Here, more in line with the overall conceptualization of organizational routines (Feldman & Pentland, 2003), we rely on sequence (event time) to recognize patterns, since sequence has been used in the preponderance of field studies on routines (Howard-Grenville & Rerup, 2016; Turner & Rindova, 2018).

Repertoire: How many unique patterns are recognized?

The repertoire of routines is theorized as a critical resource for organizational responsiveness (Feldman, 2000). For example, Weick, Sutcliffe, and Obstfeld (2008, p. 37) argue that "Action repertoire in high reliability organizations is a key to their effectiveness." Despite its prominent position in theory, the repertoire of action patterns has not been operationalized in empirical research.

We define the size of the repertoire as the number of unique action patterns recognized in observed performances. In our study, the algorithm inductively recognizes all patterns across all performances of the task, given the predefined support level. As we discuss below, the specific number of recognized patterns will depend on the support level chosen. Conceptually, the definition is simple: How many unique patterns are recognized?

Routinization: What proportion of behavior is made up of recognized patterns?

March and Simon (1958) suggest that routinization reflects the tendency to produce a standardized response to a particular stimulus. Thus, we define routinization as the proportion of total behavior that conforms to a recognizable pattern. This measure of routineness correlates with the idea of "fitness" in process mining, which describes "the fraction of the behavior in the event log that can be replayed by the process model" (Buijs, van Dongen, & van der Aalst, 2014, p. 3). If all actions conform to recognized patterns, that behavior is highly routinized. For example, the "green, yellow, red" sequence of a traffic light is perfectly routinized. In invoice management, some behavior will fit the recognized patterns, but some will not. In crisis management, there may be quite a lot of behavior that does not fit a recognized pattern. If no actions conform to recognized patterns, that behavior is not routinized.

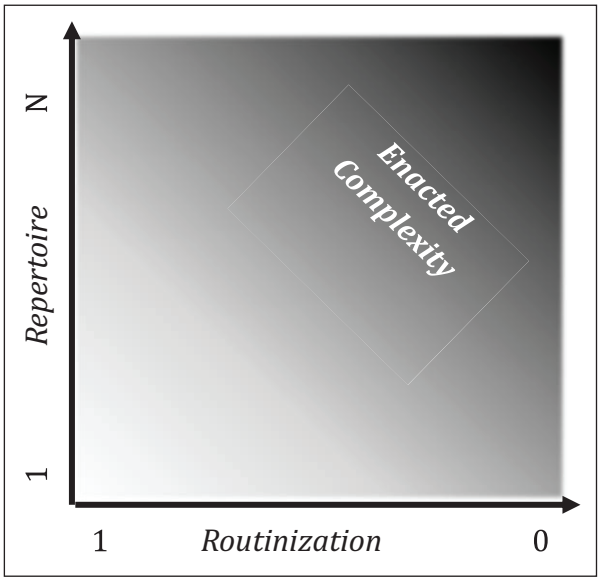

Figure 1. Repertoire, Routinization and Enacted Complexity.

Enacted complexity: How many enacted paths are there to the task resolution?

Conceptually, the enacted complexity is the number of possible paths one can take to perform the task, given the enacted network of actions (Hærem et al., 2015). The enacted network is determined by the sequence of the performed actions. In tasks where there are few possible paths to the desired end, the enacted complexity is low. When there are many possible paths, the enacted complexity is high. In practice, possible paths can form and dissolve during each enactment of a task (Danner-Schröder & Ostermann, 2020).

Relation of repertoire, routinization, and enacted complexity

A key point is that repertoire, routinization and enacted complexity are distinct constructs. *Routine* often has the connotation of *simple*, but as we explain here, it is possible to have high routinization and high enacted complexity. Figure 1 provides an overview of how repertoire, routinization, and enacted complexity are related.

First, consider the influence of routinization. Let's start from the idealized case where routinization is perfect, and the size of the repertoire is one, there is only one path. In this imaginary setting, enacted complexity is minimized. In Figure 1, this occurs at the lower-left corner. From there, we can move along the horizontal axis by keeping the repertoire constant and varying the percentage of behavior that matches the repertoire (the degree of routinization). When non-conforming behavior occurs (e.g., exceptions or workarounds), it adds to the number of possible pathways and increases the enacted complexity (Pentland, Liu, Kremser, & Hærem, 2020). Increasing routinization will tend to decrease enacted complexity; decreasing routinization will tend to increase enacted complexity.

Now, while keeping the degree of routinization constant, consider the influence of the repertoire of action patterns (the vertical axis). As the repertoire of patterns grows, there will tend to be more possible paths in the network, so enacted complexity increases. This will be true even when

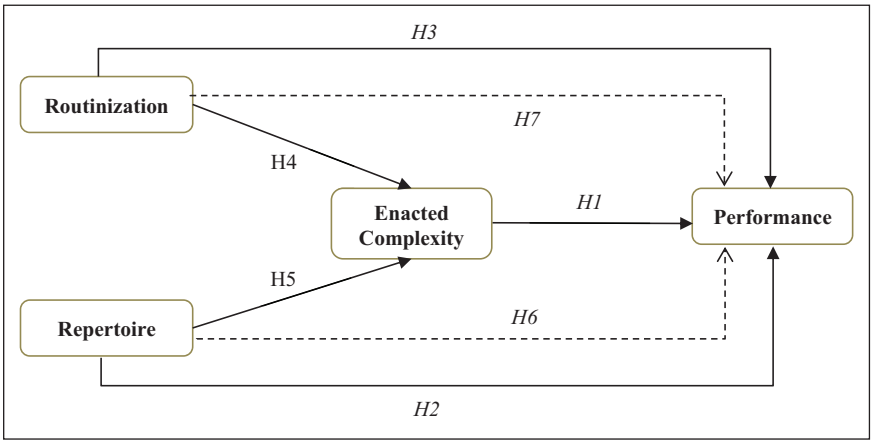

Figure 2. The Research Model.

routinization is 100%. This is because a small number of patterns can generate a large number of paths. In general, increasing the size of the repertoire will have a combinatorial effect on the possible paths and therefore on the enacted complexity.

Hypothesis Development

With these constructs clearly defined, we are ready to hypothesize about their effect on task performance. The research model is summarized in Figure 2. We discuss the rationale for each hypothesized relation in the following sections. Across all of these hypotheses, we are connecting the performative aspect of the routines (Feldman & Pentland, 2003) with the outcomes of the task. In the following section, we discuss how each of these variables is operationalized in each of our two studies.

Enacted complexity and task performance

The central idea is that tasks can be enacted in many different ways. A simple enactment means fewer possible paths. A more complex enactment means more possible paths (Goh & Pentland, 2019).

Performing a task in an unfamiliar environment can be seen as a form of information seeking or an action-based inquiry, where more actions may lead to more insights and improved outcomes (Rudolph et al., 2009; Waller, 1999; Weick & Sutcliffe, 2015). A complex enactment may provide a richer picture of the setting. Complexification is associated with search, learning, and action-based inquiry. Simplification can obscure fine-grained information that may foreshadow unexpected events (Weick & Sutcliffe, 2015). For example, in research on crew resource management, Kanki and Palmer (1993) show that oversimplification is negatively related to team performance. This is in line with Weick and Sutcliffe's (2015) warning against premature categorization and simplification.

However, simplification can be a good thing. In a more programmable setting, doing and saying only the right things is better than saying and doing redundant and erroneous things (Stachowski et al., 2009). However, when the task is less well-understood, enacting it in a more complex manner may help to identify viable solution paths and avoid the disastrous paths (Rudolph et al., 2009). This thinking also aligns with Pentland, Hærem, and Hillison's (2011) finding that programming

leads to the enactment of more atypical sequences for novices, but less atypical sequences for experts. More atypical sequences would result in higher enacted complexity.

Based on this reasoning, we hypothesize that the relation between enacted complexity and task performance is contingent on the programmability of the task. Following March and Simon (1958), less search is required in a more programmable task. More search is required when a task is less programmable. Greater enacted complexity can be interpreted as an indicator of more search. Therefore:

Hypothesis 1: The relation between enacted complexity and task performance depends on the programmability of the task.

H1a: In a less programmable task, the higher the enacted complexity, the higher the task performance.

H1b: In a more programmable task, the higher the enacted complexity, the lower the task performance.

Repertoire and task performance

Westrum (1993) argues that when systems expand their repertoire to act on something, it improves their alertness to problems associated with this repertoire. Weick and Sutcliffe (2006) expanded on this argument and suggested that a larger repertoire improves people's alertness to weak signals (p. 1724). Weick and Sutcliffe (2015) summarize these points and hold that in less programmable settings, a larger repertoire of actions is needed. In more programmable settings, however, a smaller repertoire is sufficient because the number of exceptions is limited and the analyzability of those exceptions tends to be high (Lillrank, 2003; Perrow, 1967). This logic also aligns with Ashby's Law of requisite variety (Ashby, 1961). Furthermore, in highly programmable tasks, a larger repertoire may increase search costs and opportunities for errors (Perrow, 2011). This leads us to a pair of related hypotheses:

Hypothesis 2: The relation between the size of the repertoire of routines and task performance depends on the programmability of the task.

H2a: In a less programmable task, the larger the size of the repertoire, the higher the task performance.

H2b: In a more programmable task, the larger the size of the repertoire, the lower the task performance.

Routinization and task performance

Routinization is theorized to have a contingent effect on outcomes. There are both positive and negative effects of routinization. On the one hand, routinization tends to reduce vigilance and makes teams less likely to respond adequately to weak signals of changes in situational demands, as present in less programmable tasks (Hollenbeck, Ilgen, Tuttle, & Sego, 1995; Pentland & Hærem, 2015; Rudolph et al., 2009). On the other hand, routinization helps to reduce ambiguity in role responsibilities, determines the prioritization and distribution of tasks, and contributes to effectiveness (Waller, 1999), which all are easier to achieve in a more programmable task. Economic efficiency and institutional legitimacy also require responses to be routinized, such that a specific stimulus generates a predictable, well-tuned response (Hannan & Freeman, 1984; March & Simon, 1958). This reasoning leads to the following hypothesis:

Hypothesis 3: The relation between routinization and task performance depends on the programmability of the task.

H3a: In a less programmable task, the higher the degree of routinization, the lower the task performance.

H3b: In a more programmable task, the higher the degree of routinization, the higher the task performance.

Repertoire, routinization, and enacted complexity

The relations between repertoire, routinization, and enacted complexity are suggested by the shading in Figure 1. As discussed above, greater routinization is associated with reduced enacted complexity, while a greater repertoire is associated with increased enacted complexity. We can formalize these relations as hypotheses that do not depend on the programmability of the task environment:

Hypothesis 4: The higher the degree of routinization, the lower the enacted complexity.

Hypothesis 5: The larger the size of the repertoire, the higher the enacted complexity.

Indirect effect of repertoire

In addition to the direct effects hypothesized above, Figure 2 contains two indirect effects. First, enacted complexity serves as a mediator between the size of the repertoire of action patterns and outcomes. In a less programmable task, the repertoire is positively related to enacted complexity (H5), and enacted complexity is positively related to performance (H1a), so repertoire should have a positive indirect effect on performance. By the same reasoning, we expect the indirect effect (H5*H1b) to be negative in the more programmable task.

Hypothesis 6: The indirect effect of the size of repertoire on task performance through enacted complexity depends on the programmability of the task.

H6a: In a less programmable task, the indirect effect of the size of repertoire on task performance through enacted complexity is positive.

H6b: In a more programmable task, the indirect effect of the size of repertoire on task performance through enacted complexity is negative.

Indirect effect of routinization

Likewise, enacted complexity mediates the relation between routinization and performance. In the less programmable task, routinization is negatively related to enacted complexity (H4), and since enacted complexity is hypothesized to have a positive effect on the outcome (H1a), routinization should have a negative indirect effect. By the same reasoning, we expect the indirect effect (H4*H1b) to be positive in the more programmable task.

Hypothesis 7: The indirect effect of the routinization on task performance through enacted complexity depends on the programmability of the task.

H7a: In a less programmable task, the indirect effect of the routinization on task performance through enacted complexity is negative.

H7b: In a more programmable task, the indirect effect of the routinization on task performance through enacted complexity is positive.

Table 2. Settings for study 1 (crisis management) and study 2 (invoice management).

	More programmable task	Less programmable task
Low task experience		Study 1
High task experience	Study 2	

Methods

Our two task settings—crisis management and invoice management—exemplify extremes of programmability. Since the programmability of a task setting can be confounded with the experience level of the task doers, as discussed above, we amplified the distinction by matching a less programmable setting to low experience level (i.e., crisis management done by novices, study 1), and a more programmable setting to high experience level (i.e., invoice management done by full-time employees, study 2), as illustrated in Table 2 (Shadish, Cook, & Campbell, 2002).

Description of tasks in study 1 and study 2

In study 1, the data are collected from an experimental task simulation called MindLab. The simulation takes place in a controlled laboratory environment at business school, but it provides participants with the classic elements of a crisis (Pearson & Clair, 1998):

(1) Threat: The terrorists may attack one or several oil platforms in the North Sea.
(2) Surprise and uncertainty: The participants don't know who is going to attack, nor when nor how.
(3) Time pressure: There are several vessels in the area that need to be detected, searched and, if a terrorist, eliminated and the time is limited.

Key aspects of this simulation resemble the team effectiveness lab developed by Hollenbeck and Ilgen (e.g., Hollenbeck et al., 2002). Specifically, the primary subtasks of both games are to detect the presence of unidentified objects, identify whether the objects are friendly or hostile, and intercept hostile objects before they enter a critical area. However, the search process undertaken to identify and intercept terrorists in MindLab is more complex (especially with respect to communications and coordination challenges) and involves more steps.

Each player has a role with a unique and complementary set of capabilities and resources. The roles are: Orion, which is a surveillance airplane detecting vessels; Patrol, which is a coastal motorboat searching vessels; Frigate, which is a military vessel able to attack and eliminate threats. The players are separated from each other and the only form of communication is through the built-in chat/email functionality. The participants are randomly assigned to different roles and teams and participate in three different scenarios. The simulated scenarios present individuals with challenges related to team monitoring, information exchange, coordination, and collective team strategy.

MindLab provides a log of behavior in which each action is logged as one integer in a sequence. The data consist of both performed actions and written communications, which are sorted into a library of actions. The data are captured automatically and do not require coding, providing a precise and objective record of action categories and outcome measures. The library of actions is presented in Appendix A.

An example of a recognized action pattern is: (1) Patrol Receives Message about an unidentified object (friend or foe), (2) Patrol Searches unidentified object (friend or foe), (3) Patrol Sends

Message to Frigate with order to attack or not. However, these patterns are not enforced by the game. The performed actions do not have to follow any particular rules and there are many possible solutions.

For studying a programmable task (study 2), we wanted a generic and highly regulated process present in most organizations. A good example is the invoice management process, where there are many rules and only a few possible solutions. In our case, all invoices ended up with the final state "Posted." We identified an organization that used off-the-shelf software to further program the process. Both these factors promote generalizability. We extracted the data from the invoice management system in a major European business school with approximately 1000 employees and more than 20,000 invoices per year. The invoice management system provides a log of all actions performed during the approval process for invoices.

The data look like an array of integers, where each integer represents one action performed by a certain actor. Vendors present different environmental stimuli that require different responses. The invoice management task is to generate an adequate response to the invoices from each vendor. An example of a recognized pattern for the invoice management task in study 2 is the following: *Distribute—Approve invoice—Posted*, another is *Distribute—Send reminder—Approve invoice—Check procedure—Posted*. The library of actions is presented in Appendix A.

Sample

The participants in study 1 were undergraduate students in an organizational behavior class at a business school. Participation in the experiment is voluntary and follows the applicable local regulations about data protection. The sample in study 1 includes 62 teams, where each team consisted of three players who participated in three different scenarios of the crisis management simulation, each with slightly different details. Participants were randomly assigned to teams and roles.

In study 1, there were 11 missing data points. These were related to technical problems outside the control of the participants and the lab coordinators. We believe that these problems were random so that the missing data did not bias our sample (Tremblay, Dutta, & Vandermeer, 2010). After deleting incomplete runs, we have 175 observations across all three scenarios on the team level (N = 175).

In study 2, there were no missing data. The data consist of workflow sequences from one organization and includes 1961 vendors with a total of 14,775 invoices. We exclude all vendors with two or fewer invoices and end up with an effective sample size of 673 vendors and 6006 invoices (N = 673). The level of analysis is on vendor level and, accordingly, the computer measures repertoire, routinization, and complexity for different vendors.

Operationalizations

Performance. We wanted the performance variables across the two studies to be as similar as possible. Both studies operationalize performance in terms of speed and accuracy. In study 1 accuracy was scored as a dichotomous variable (completed/not completed), while speed was measured as the time spent completing the tasks. The performance measure reflects both the accuracy and the speed of the actions performed by the team. The performance variable ranged from -60 to 225. The negative scores are due to the negative points that are assigned for attacking the wrong vessel or attacking without obtaining the necessary information.

The measure of performance in study 2 also reflects the speed and, indirectly, accuracy. All invoices in study 2 were completed (posted) only when accuracy was achieved. Efforts to achieve accuracy caused more workflow iterations and processing time so that these efforts were translated into a longer time to complete the approval process. Speed was measured as the time spent

completing the approval process measured as the number of days. The higher the number, the longer it took. To make the scale comparable to study 1 we reversed the scale so that higher numbers reflect better performance.

Enacted complexity (EC). Conceptually, the enacted complexity is the number of possible ways a task can be performed. We measure it by counting the number of possible paths (i.e. simple paths) in the network of actions used to perform the task. An action network is a directed graph that traces the sequential relations between actions (Pentland & Feldman, 2007; Pentland, Recker, & Wyner, 2017). The more complex the task enactment is, the more interconnected elements there are in the action network, and thus the higher the number of possible paths to complete a task. We use the *log10* transformation because the number of paths is exponentially related to the size of the action network. In study 1, EC reflects the action network based on actions performed by all team players throughout a given task scenario. In study 2, EC reflects the network of actions involved in the processing and handling of invoices for one particular vendor, where the task can be executed by one or several actors. We count the number of possible paths (without loops) from start to finish using the *iGraph* package in R. For details see Appendix B.

Repertoires of action patterns (RP). We operationalize the repertoire of action patterns as the number of distinct, recognizable patterns within a stream of actions used to perform the task. For pattern recognition, we use the sequential pattern mining framework SPMF (Fournier-Viger et al., 2014a), which is an open-source data mining library. From this library, we chose the VMSP algorithm (vertical mining of maximal sequential patterns: Fournier-Viger et al., 2014b). The goal of maximal pattern mining is to recognize patterns with maximal sequential information content. The maximal information content is selected by the longest pattern that is repeated often enough to be recognized given the chosen support level. At the same time, the algorithm ignores patterns that are embedded in larger patterns. Fournier-Viger et al.'s algorithm identifies patterns with different lengths according to a consistent set of criteria across different action networks (see Fournier-Viger et al. 2014b for a further elaboration of the process of recognizing patterns). Maximal pattern mining results in more comprehensible, efficient, and readable results than other techniques (Gupta & Han, 2012). In the analysis, we set the support level to 33%. For example, in study 1, we ran the pattern recognition algorithm (VMSP) on all series of enactments (in total 175 sequences) across all scenarios played by the teams. The algorithm recognized all patterns that occurred 58 times or more (i.e., 33% support level). We then calculated the team's repertoire by counting how many of the recognized patterns were used in each scenario. For study 1, the average size of the repertoire was 14.6 with SD 5.3 (min 3, max 27). We investigated the robustness of the results by also using 20% and 40% support levels, and found that the directions and the correlation with performance remain the same - at the same level of significance.

Routinization (RT). We operationalize routinization as the percentage of actions that fit the recognized patterns within a stream of actions used to perform the task. To measure this percentage, we sum the identified patterns times the pattern length and divide it by the total number of actions in the task sequence:

$$RT = \frac{\sum_{i=1}^{RP} \left(Number\ of\ occurences\ of\ recognized\ pattern_i\ *length\ of\ pattern_i \right)}{Total\ number\ of\ actions\ in\ the\ task\ sequence}$$

where RP is the total number of unique patterns identified in the sample; *i* is the observed instance of a particular pattern; *length* is the number of actions in an identified pattern *i;* and *Total number*

of actions in the task resolution sequence is the sum of all actions used to perform the task for given unit of analysis. RT ranges from 0 (indicating an absence of routinization) to 1 (indicating complete routinization). RT is based on the same set of patterns recognized for RP and is calculated for each team across all three scenarios. See Appendix B for details on how this measure is calculated.

Results

Study 1

On average, a group of three players performs 110 actions in one scenario. The average length of recognized patterns is 3.1 actions. The descriptive statistics of our measures are provided in Table 3.

The average size of the repertoire per group is 14.57 (SD 5.29), with the min and max values equal to 3 and 27 respectively. Both the size of the repertoire of action patterns and enacted complexity are significantly and positively correlated to team performance (i.e., teams with larger repertoire and higher enacted complexity on average performed better). The measures of the size of repertoire and routinization are highly correlated (r = .73), however, the variance inflation factor (VIF) ranged from 1.09 to 2.36 and was within the acceptable range (Bowerman & O'Connell, 1990; Myers, 1990).

To test our hypotheses, we used an ordinary least squares regression analysis with robust variance estimation based on a bootstrap procedure (Efron & Tibshirani, 1994). The effects of enacted complexity, repertoire, and degree of routinization on performance are reported in model 3 (see Table 4). The model shows that enacted complexity and the size of repertoire had a positive effect on performance, while the degree of routinization has a negative effect. All of these effects are statistically significant and provide support for our hypotheses 1a, 2a, and 3a.

The effects of routinization and repertoire of action patterns on enacted complexity are presented in model 1. A higher degree of routinization is shown to have a negative effect on enacted complexity, whereas a larger repertoire of action patterns is shown to have a positive effect on enacted complexity. Given the statistical significance, we conclude with support for hypotheses 4 and 5.

The hypotheses related to indirect effects were conducted based on the recommendations of Hayes and Preacher (2014), who suggest that if there are several predictors and indirect effects, they should be analyzed individually. Mathematically, all resulting paths will be the same as those that would have been estimated in structural equation modeling (Hayes & Preacher, 2014). We used a bootstrap procedure (MacKinnon, Lockwood, Hoffman, West, & Sheets, 2002) with a confidence interval of 95% and a bootstrap sample equal to 10,000. The results are presented in Figure 3. We find that both the size of the repertoire of action patterns and the degree of routinization influence team performance through their effects on enacted complexity, which is positively related to team performance. The bias-corrected bootstrap confidence intervals for the indirect effects of the repertoire of action patterns ($a_1b = .06$) and routinization ($a_2b = -.05$) does not straddle zero, indicating that the indirect effects are significant at $p < .05$. We find support

Table 3. Descriptive Statistics, Study 1 (N = 175).

	Mean	SD	RP	RT	EC
Repertoire (RP)	14.57	5.29			
Routinization (RT)	.19	.07	.73***		
Enacted complexity (EC)	1.35	.57	.19*	–.01	
Performance	74.72	46.90	.23**	–.08	.25***

*$p < .05$, **$p < .01$, ***$p < .001$.

Table 4. Regression Results, Study 1 (N = 175).

Variables	Enacted complexity		Performance			
	Model 1		Model 2		Model 3	
	b	*p*-values	b	*p*-value	b	*p*-value
(Constant)						
Routinization (RT)	–.32***	.00	–.54***	.00	–.49**	.00
Repertoire (RP)	.42***	.00	.62***	.00	.56***	.00
Enacted complexity (EC)					.14*	.04
Adj. R2	.07		.17		.19	
F Δ	7.72***	.00	19.37***	.00	3.82*	.05

*$p < .05$, **$p < .01$, ***$p < .001$. The reported betas are standardized.

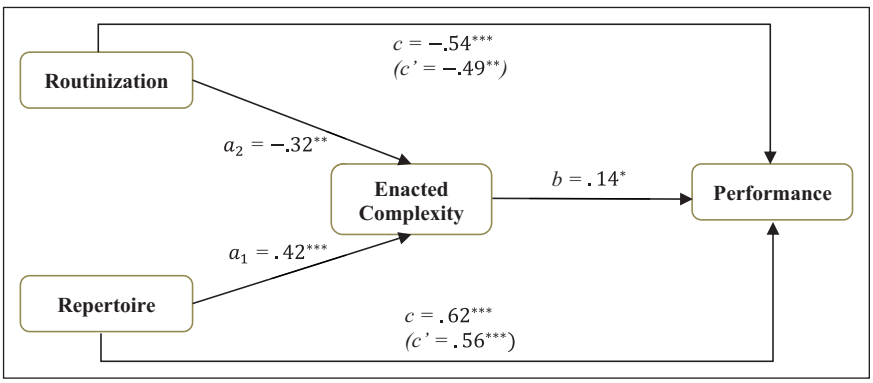

Figure 3. The Mediation Model for Study 1, Crisis Management.
*$p < .05$, **$p < .01$, ***$p < .001$. The reported betas are standardized.

for hypotheses 6a and 7a, i.e., enacted complexity is a mediator for routinization and repertoire of action patterns when explaining team performance in crisis management setting (i.e., less programmable task).

Study 2

The average number of actions in study 2 is 16.5 actions per vendor. This means that it takes 16.5 actions to process all invoices from the average vendor. The average length of the recognized patterns in the invoice management task is 4.3 actions, which is 39% longer than in the crisis management task. However, these numbers should be seen in relation to the average number of actions needed to execute a given task (110 for crisis management and 16.5 for invoice processing).

The descriptive statistics and correlations for study 2 are provided in Table 5. The results show that each vendor, on average, could be handled with two response patterns (RP=2.07). For some vendors, it was impossible to recognize a pattern with our required support level, while others generated a repertoire of three response patterns. In general, the invoice management task, where the IT systems set constraints on what actions are possible to perform at the different stages of the processes, results in simpler enactments with longer patterns. However, we still see a lot of deviations and many different performed patterns.

Table 5. Descriptive Statistics, Study 2 (N = 673).

	Mean	SD	RP	RT	EC
Repertoire (RP)	2.07	.90			
Routinization (RT)	.58	.23	.46***		
Enacted complexity (EC)	.55	.55	.42***	–.14**	
Performance	–51.10	63.41	–.33***	.16**	–.71***

*$p < .05$, **$p < .01$.

Table 6. Regression Results, Study 2 (N = 673).

	Enacted complexity		Performance			
	Model 1		Model 2		Model 3	
Variables	b	p-values	b	*p*-value	b	p-value
(Constant)						
Routinization (RT)	–.46***	.00	.44***	.00	.14***	.00
Repertoire (RP)	.63***	.00	–.58***	.00	–.17***	.00
Enacted complexity (EC)					–.65***	.00
Adj. R2	.28		.24		.54	
F Δ	153.68***	.00	99.73***	.00	398.91***	.00

***$p < .001$. The reported betas are standardized.

As in study 1, the size of the repertoire of action patterns, degree of routinization, and enacted complexity are all significantly correlated to performance. However, in this setting, repertoire and enacted complexity are negatively correlated whereas routinization has a positive correlation. This is the opposite of what we observed in study 1. The average level of routinization is much higher for the invoice processing task (.57) than for the crisis management task in study 1 (.19) (see Tables 2 and 4). The size of the repertoire of action patterns and the enacted complexity are significantly smaller (2.11 and .66, respectively, in study 2 versus 14.57 and 1.35, respectively, in study 1). This result provides some face validity to the measures: We would expect that in more programmable task setting (invoice management), enacted complexity is lower, the level of routinization is higher, and the size of the repertoire is smaller relative to those in less programmable task setting (like crisis management).

We tested our hypotheses in this setting using the same method as in study 1. The results are presented in Table 6. Despite relatively high correlations between constructs, the variance inflation factors remained within an acceptable range (from 1.39 to 2.00), indicating no multicollinearity problems (Bowerman & O'Connell, 1990; Myers, 1990).

The effects of enacted complexity, repertoire, and degree of routinization on performance are reported in model 3 (Table 6), which shows that enacted complexity and the size of repertoire had a negative effect on performance, while the degree of routinization had a positive effect. These are the opposite effects as the less programmable task setting. All of these effects are statistically significant and provide support for hypotheses 1b, 2b, and 3b.

The effects of routinization and the size of the repertoire of action patterns on enacted complexity are presented in model 1. These are the same relations as we found in study 1 and we conclude with support of hypotheses 4 and 5. Note that these effects are not contingent on the programmability of the task environment.

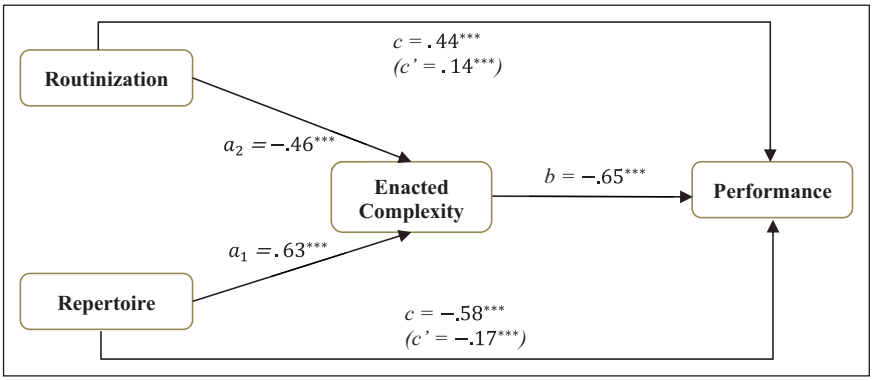

Figure 4. The Mediation Model for Study 2, Invoice Management.
***$p < .001$. The reported betas are standardized.

We test the hypotheses related to the indirect effects using the same method as in study 1 (i.e., Hayes & Preacher, 2014). The results of the analyses are presented in Figure 4. We find that both the size of the repertoire of action patterns and the degree of routinization influence performance through their effects on enacted complexity. The bias-corrected bootstrap confidence intervals for the indirect effects of the repertoire of action patterns (a_1b=-.41) and routinization (a_2b=.30) does not straddle zero, indicating that the indirect effects are significant at $p < .05$. These are the opposite effects as compared to less programmable task setting. Given the statistical significance of indirect effects, we conclude with support of hypotheses 6b and 7b.

Discussion

In this article, we have defined and operationalized three key constructs that allow us to clarify how routines influence task performance. We provide a simple, general model based on how tasks are enacted, and demonstrate that this model is contingent on the programmability of the task. In doing so, we bring empirical evidence and a fresh conceptual vocabulary to the ongoing debate about programs and routines (Anderson & Lemken, 2019; Kay, 2018). While March and Simon (1958) were unquestionably ahead of their time in many respects, their work predates the existence of viable pattern mining technology. They were theorizing about repetitive action patterns without adequate tools to measure those patterns.

Taking patterns seriously

Pattern mining is a methodological innovation that creates new possibilities for empirical research. Pattern mining focuses on recognizability and repetitiveness of behavior, which are defining properties of organizational routines (Feldman & Pentland, 2003). By inductively recognizing patterns of action, we were able to unpack mechanisms that connect patterns to outcomes without needing to measure individual traits, incentives, or other constructs that are often connected to task performance.

By looking closely at enacted behavior, we find that participants enact fewer and longer patterns in a more programmable setting. In a less programmable setting, they enact a larger number of shorter patterns. This finding adds to March and Simon's insight that a highly complex and organized set of responses can be assembled from a repertoire of performance programs. Theoretically, the prevalence of short and long patterns has relevance for the enacted complexity, as well. All else being equal, shorter patterns can generate a larger set of possible paths than longer patterns. Shorter

action patterns provide more opportunities to take different paths, so a more complex response can be enacted. Longer patterns will tend to limit the response to a smaller set of possible paths. Although not studied specifically in this article, the length of the action pattern is an interesting issue to investigate in future research.

Understanding task complexity

Focusing on behavioral patterns also has implications for the concept of task complexity. Our study speaks to two distinct issues.

First, our analysis demonstrates a new role for enacted complexity in empirical research. As Hærem et al. (2015) point out, task complexity is usually treated as an independent variable. This is because tasks are usually conceptualized as separate from their enactment (Hackman, 1969; Wood, 1986), so task complexity is a constant for all enactments of a task. Further, it is estimated as high or low based on perceptual measures (Hærem et al., 2015), so its role in empirical research is very limited. However, when we focus on observed behavior, we see that *enacted* complexity can vary each time a task is performed (Danner-Schröder & Ostermann, 2020). Our analysis demonstrates how this variation contributes to our understanding of task performance.

Second, measuring the complexity of an idealized task, apart from its enactment, implicitly assumes that the task is highly programmable (repetitive, minimal search, few exceptions, well-understood, etc.) When a task is less programmable, enacted complexity can vary as each performance unfolds (Danner-Schröder & Ostermann, 2020). In this situation, we cannot readily identify the required acts and information cues needed to complete it, or their required sequence, as required by standard definitions of task complexity (e.g., Wood, 1986). Thus, for less programmable tasks, it is practically meaningless to attempt an *ex-ante* measure of task complexity using conventional metrics such as component complexity or coordinative complexity (Wood, 1986). However, we can readily measure *enacted* complexity for any kind of task based on observable behavioral data with the methods we use here.

Disentangling routinization and complexity

It is easy to confuse routinization with lack of complexity because the terms routine and routinization connote simplicity. However, as we demonstrate here, routinization and enacted complexity are distinct constructs, conceptually and empirically. Routinization does not necessarily lead to simplification. You can have a lot of routinization and still have complex enactments. This is because enacted complexity also depends on the repertoire of patterns and how those patterns are used to enact task performance. As we define it, routinization means that a greater percentage of the observed behavior follows recognized patterns. If the routinization is high (closer to one), then the network that represents the task will tend to have fewer paths. Therefore, enacted complexity will be lower.

Programming a repertoire

March and Simon introduced the idea of *programming* a decision or a process at a time when computer technology was starting to become increasingly influential in organizations. The term was partly metaphorical ("organizations are like computers") and partly literal, as organizations began to automate tasks with computers. In the intervening decades, the use of digital technology to support or automate organizational processes has advanced dramatically (Dumas et al., 2013). Typical enterprise software systems offer complete, end-to-end "solutions," but these monolithic systems are not always well-suited to the needs of workers (Berente et al, 2016). Our findings suggest that a repertoire of patterns that can be (re)assembled as needed may be preferable. The size of the repertoire

and the length of the patterns will depend on the characteristics of the task environment, as outlined in Table 1. The basic notion of fit is an old idea, but it seems particularly relevant today.

Limitations

Support level for patterns. Despite the advantages of inductive pattern mining as a tool for analyzing organizational routines and processes, support level needs to be carefully considered. The support level sets the threshold of evidence that determines whether a pattern will be accounted for or not. Thus, it is directly linked to the measures of the size of the repertoire of action patterns and the degree of routinization. The results we report here are based on a support level of 33%. We investigated a range of values from 20% to 40% to check the robustness of our findings. However, there are no clear rules regarding the frequency with which a certain pattern should occur in the data to be considered a routine (Becker, 2004). Like many other choices, these types of consideration will be a matter for researcher judgment.

Levels of granularity. The apparent complexity of a task depends on how it is described (Hærem et al., 2015). A finer-grained description results in more variety (Pentland, Hærem, & Hillison, 2010). How a researcher defines a task is often influenced by their subjective judgment. A task can usually be broken down into smaller sub-tasks or be seen as a part of a more overarching task. We took a pragmatic approach and, in both studies, we used the actions as defined by the IT system. For both systems, we mapped the library of possible actions (see Appendix A). In the invoice management process examples of actions available to the users are "approve," "forward," "post," "distribute," and "reject." In the crisis management process example actions are search, move, send a message, attack, and patrol. Using the IT system's labels for actions makes the coding reliable and actions easily observable.

Broader range of settings. The two studies presented here provide evidence for the relations shown in Figures 2, 3, and 4. While each study has sufficient power to provide statistically significant results, they represent two specific cases. It would be desirable to test this model over a broader range of settings. Fortunately, this study can be replicated in any setting where there is a trace of task enactments and a measurable outcome for each enactment or set of enactments (such as speed, accuracy, cost, or quality).

Concluding Remarks

As organizations continue to push the bounds of programming, we need concepts and theories that can help us understand the limits and trade-offs involved. More programming is not necessarily better, but the way a process is programmed is important. A larger repertoire of smaller patterns can generate a larger space of possible paths, and that can be valuable in less programmable settings. As we have demonstrated, taking patterns of action seriously requires a novel approach to organizational research. It requires new sources of data that reflect streams of activity from multiple actors. It requires new tools, such as pattern mining, to identify recognizable and repetitive patterns. It requires new concepts, such as enacted complexity, to help us grasp the implications of the patterns we observe. It requires pattern-aware operationalizations of foundational concepts, such as routinization and repertoire.

Funding

The author(s) received no financial support for the research, authorship, and/or publication of this article.

ORCID iD

Mathias Hansson https://orcid.org/0000-0002-9221-9696

Notes

1. To be clear, as we use the terms here, "routine" is a noun that has a performative and ostensive aspect, as defined by Feldman and Pentland (2003). The performative aspect reflects enacted behavior. When we use "program" as a noun, it reflects idealized or prescribed behavior, not enacted behavior. When we use "program" as a verb, we refer to the process of creating a program.
2. Quotes about the fragility of battle plans have been attributed to everyone from Sun Tzu to Dwight Eisenhower.
3. https://github.com/fandu/maximal-sequential-patterns-mining.

References

Anderson, Marc H., & Lemken, Russell K. (2019). An empirical assessment of the influence of March and Simon's *Organizations*: The realized contribution and unfulfilled promise of a masterpiece. *Journal of Management Studies*, *56*, 1537–1569.

Ashby, W. Ross (1961). *An introduction to cybernetics*. London: Chapman & Hall.

Becker, Markus C. (2004). Organizational routines: a review of the literature. *Industrial and Corporate Change*, *13*, 643–678.

Benner, Mary J., & Tushman, Michael L. (2003). Exploitation, exploration, and process management: The productivity dilemma revisited. *Academy of Management Review*, *28*, 238–256.

Berente, Nicholas, Lyytinen, Kalle, Yoo, Youngjin, & King, John L. (2016). Routines as shock absorbers during organizational transformation: Integration, control, and NASA's enterprise information system. *Organization Science*, *27*, 551–572.

Bowerman, Bruce L., & O'Connell, Richard T. (1990). *Linear statistical models: An applied approach*. Boston: PWS-Kent Publishing Company.

Buijs, Joos C., van Dongen, Boudewijn F., & van der Aalst, Wil M. (2014). Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity. *International Journal of Cooperative Information Systems*, *23*.

Cohen, Michael D., & Bacdayan, Paul (1994). Organizational routines are stored as procedural memory: Evidence from a laboratory study. *Organization Science*, *5*, 554–568.

Csardi, Gabor, & Nepusz, Tamas (2006). The igraph software package for complex network research. *InterJournal, Complex Systems*, *1695*(5), 1–9.

Cyert, Richard M., & March, James G. (1963). *A behavioral theory of the firm*. Englewood Cliffs, NJ: Prentice Hall.

Daft, Richard L., & Weick, Karl E. (1984). Toward a model of organizations as interpretation systems. *Academy of Management Review*, *9*, 284–295.

Danner-Schröder, Anja, & Ostermann, Simone M. (2020). Towards a processual understanding of task complexity: Constructing task complexity in practice. *Organization Studies*, *41*, 1055–1078.

Dumas, Marlon, La Rosa, Marcello, Mendling, Jan, & Reijers, Hajo A. (2013). *Fundamentals of business process management*. Heidelberg: Springer.

Efron, Bradley, & Tibshirani, Robert J. (1994). *An introduction to the bootstrap*. Boca Raton, FL: CRC Press.

Feldman, Martha S. (2000). Organizational routines as a source of continuous change. *Organization Science*, *11*, 611–629.

Feldman, Martha S., & Pentland, Brian T. (2003). Reconceptualizing organizational routines as a source of flexibility and change. *Administrative Science Quarterly*, *48*, 94–118.

Feldman, Martha S., Pentland, Brian T., D'Adderio, Luciana, Dittrich, Katharina, Rerup, Claus, & Seidl, David (Eds.) (2022). *Cambridge Handbook of Routine Dynamics*. Cambridge, UK: Cambridge University Press.

Fournier-Viger, Philippe, Lin, Jerry C. W., Kiran, Rage U., Koh, Yun S., & Thomas, Rincy (2017). A survey of sequential pattern mining. *Data Science and Pattern Recognition*, *1*, 54–77.

Fournier-Viger, Philippe, Gomariz, Antonio, Gueniche, Ted, Soltani, Azadeh, Wu, Cheng-Wei, & Tseng, Vincent S. (2014a). SPMF: A Java open-source pattern mining library. *Journal of Machine Learning Research*, *15*, 3569–3573.

Fournier-Viger, Philippe, Wu, Cheng-Wei, Gomariz, Antonio, & Tseng, Vincent S. (2014b). VMSP: Efficient vertical mining of maximal sequential patterns. In Marina Sokolova & Peter van Beek (Eds.), *Advances in artificial intelligence. 27th Canadian conference on artificial intelligence* (pp. 83–94). Cham, Switzerland: Springer.

Srivastava, Tushar, Jain, Akash, & Rashid, Naved (2020). *Market guide for intelligent business process management suites*. Gartner Inc.

Goh, Kenneth T., & Pentland, Brian T. (2019). From actions to paths to patterning: Toward a dynamic theory of patterning in routines. *Academy of Management Journal*, *62*, 1901–1929.

Gupta, Manish, & Han, Jiawei (2012). Approaches for pattern discovery using sequential data mining. In Pradeep Kumar, P. Eadha Krishna, & S. Bapi Raju (Eds.), *Pattern discovery using sequence data mining: Applications and studies* (pp. 137–154). Hershey, PA: IGI Global.

Hackman, J. Richard (1969). Effects of task factors on job attitudes and behavior: A symposium: IV. Nature of the task as a determiner of job behavior. *Personnel Psychology*, *22*, 435–444.

Haerem, Thorvald, & Rau, Devaki (2007). The influence of degree of expertise and objective task complexity on perceived task complexity and performance. *Journal of Applied Psychology*, *92*, 1320–1331.

Hærem, Thorvald, Pentland, Brian T., & Miller, Kent D. (2015). Task complexity: Extending a core concept. *Academy of Management Review*, *40*, 446–460.

Hannan, Michael T., & Freeman, John (1984). Structural inertia and organizational change. *American Sociological Review*, *49*, 149–164.

Hayes, Andrew F., & Preacher, Kristopher J. (2014). Statistical mediation analysis with a multicategorical independent variable. *British Journal of Mathematical and Statistical Psychology*, *67*, 451–470.

Hogarth, Robin M. (2001). *Educating intuition*. Chicago: University of Chicago Press.

Hollenbeck, John R., Ilgen, Daniel R., Tuttle, Dale B., & Sego, Douglas J. (1995). Team performance on monitoring tasks: An examination of decision errors in contexts requiring sustained attention. *Journal of Applied Psychology*, *80*, 685–696.

Hollenbeck, John R., Moon, Henry, Ellis, Aleksander P. J., West, Bradley J., Ilgen, Daniel R., Sheppard, Lori, Porter, Christopher O. L. H., & Wagner, John A. III (2002). Structural contingency theory and individual differences: Examination of external and internal person–team fit. *Journal of Applied Psychology*, *87*, 599–606.

Howard-Grenville, Jennifer, & Rerup, Claus (2016). A process perspective on organizational routines. In Ann Langley & Haridimos Tsoukas (Eds.), *The Sage handbook of process organization studies* (pp. 323–337). Newbury, CA: SAGE Publications.

Kanki, Barbara G., & Palmer, Mark T. (1993). Communication and crew resource management. In Earl L. Wiener, Barbara G. Kanki, & Robert L. Helmreich (Eds.), *Cockpit resource management* (pp. 99–136). San Diego, CA: Academic Press.

Kay, Neil M. (2018). We need to talk: Opposing narratives and conflicting perspectives in the conversation on routines. *Industrial and Corporate Change*, *27*, 943–956.

Lillrank, Paul (2003). The quality of standard, routine and nonroutine processes. *Organization Studies*, *24*, 215–233.

MacKinnon, David P., Lockwood, Chondra M., Hoffman, Jeanne M., West, Stephen G., & Sheets, Virgil (2002). A comparison of methods to test mediation and other intervening variable effects. *Psychological Methods*, *7*, 83–104.

March, James G., & Simon, Herbert A. (1958). *Organizations*. New York: John Wiley & Sons.

Myers, Raymond H. (1990). *Classical and modern regression with applications* (Vol. 2). Belmont, CA: Duxbury Press.

Newell, Allen, & Simon, Herbert A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall.

Pearson, Christine M., & Clair, Judith A. (1998). Reframing crisis management. *Academy of Management Review*, *23*, 59–76.

Pentland, Brian T., & Feldman, Martha S. (2007). Narrative networks: Patterns of technology and organization. *Organization Science*, *18*, 781–795.

Pentland, Brian T., & Hærem, Thorvald (2015). Organizational routines as patterns of action: Implications for organizational behavior. *Annual Review of Organizational Psychology and Organizational Behavior*, *2*, 465–487.

Pentland, Brian T., Hærem, Thorvald, & Hillison, Derek (2010). Comparing organizational routines as recurrent patterns of action. *Organization Studies*, *31*, 917–940.

Pentland, Brian T., Hærem, Thorvald, & Hillison, Derek (2011). The (n)ever-changing world: Stability and change in organizational routines. *Organization Science*, *22*, 1369–1383.

Pentland, Brian T., Liu, Peng, Kremser, Waldemar, & Hærem, Thorvald (2020). The dynamics of drift in digitized processes. *MIS Quarterly*, *44*, 19–47.

Pentland, Brian T., Recker, Jan, & Wyner, George (2017). Rediscovering handoffs. *Academy of Management Discoveries*, *3*, 284–301.

Perrow, Charles (1967). A framework for the comparative analysis of organizations. *American Sociological Review*, *32*, 194–208.

Perrow, Charles (2011). *Normal accidents: Living with high risk technologies*. Princeton, NJ: Princeton University Press.

Rice, Ronald E., & Cooper, Stephen D. (2010). *Organizations and unusual routines: A systems analysis of dysfunctional feedback processes*. Cambridge: Cambridge University Press.

Rudolph, Jenny W., Morrison, J. Bradley, & Carroll, John S. (2009). The dynamics of action-oriented problem solving: Linking interpretation and choice. *Academy of Management Review*, *34*, 733–756.

Shadish, William R., Cook, Thomas D., & Campbell, Donald T. (2002). *Experimental and quasi-experimental designs for generalized causal inference*. Boston, MA: Houghton Mifflin.

Simon, Herbert A. (1977 [1947]). *Administrative behavior: A study of decision-making processes in administrative organization*. New York: Macmillan.

Stachowski, Alicia A., Kaplan, Seth A., & Waller, Mary J. (2009). The benefits of flexible team interaction during crises. *Journal of Applied Psychology*, *94*, 1536–1543.

Su, Norman M., Brdiczka, Oliver, & Begole, Bo (2013). The routineness of routines: Measuring rhythms of media interaction. *Human–Computer Interaction*, *28*, 287–334.

Thompson, James D. (1967). *Organizations in action: Social science bases of administrative theory*. New York: McGraw-Hill.

Tremblay, Monica C., Dutta, Kaushik, & Vandermeer, Debra (2010). Using data mining techniques to discover bias patterns in missing data. *Journal of Data and Information Quality*, *2*(1), 1–19.

Turner, Scott F., & Rindova, Violina P. (2018). Watching the clock: Action timing, patterning, and routine performance. *Academy of Management Journal*, *61*, 1253–1280.

Waller, Mary J. (1999). The timing of adaptive group responses to nonroutine events. *Academy of Management Journal*, *42*, 127–137.

Weick, Karl E., & Sutcliffe, Kathleen M. (2006). Mindfulness and the quality of organizational attention. *Organization Science*, *17*, 514–524.

Weick, Karl E., & Sutcliffe, Kathleen M. (2015). *Managing the unexpected: Sustained performance in a complex world*. Hoboken, NJ: John Wiley & Sons.

Weick, Karl E., Sutcliffe, Kathleen M., & Obstfeld, David (2008). Organizing for high reliability: Processes of collective mindfulness. *Crisis Management*, *3*, 81–123.

Westrum, Ron (1993) Cultures with requisite imagination. In John A. Wise, V. David Hopkin, & Paul Stager (Eds.), *Verification and validation of complex systems: Human factors issues* (pp. 401–416). Heidelberg: Springer.

Wood, Robert E. (1986). Task complexity: Definition of the construct. *Organizational Behavior and Human Decision Processes*, *37*, 60–82.

Author biographies

Mathias Hansson is an Associate Professor at Inland Norway University of Applied Sciences and BI Norwegian Business School. His research is focused on organizational routines, pattern mining, business analytics, and organizational theory. He received his PhD from BI Norwegian Business School in 2018.

Thorvald Hærem is Professor at BI Norwegian Business School. His research interests include organizational routines, behavioral decision-making, and theories of information processing. He has published research in

journals including *Journal of Applied Psychology, Journal of Behavioral Decision Making, Organization Science, Leadership Quarterly, MISQ*, and *Academy of Management Review.*

Brian T. Pentland is the Main Street Capital Partners Endowed Professor in the Department of Accounting and Information Systems at Michigan State University. His research is focused on the analysis of repetitive patterns of action, such as organizational routines. He received his PhD in management from the Massachusetts Institute of Technology in 1991.

Appendix A

Table A-1. Library of Actions, Study 1.

	Orion	Patrol	Frigate
Move	11	21	31
Patrol	12	22	32
Follow	13	23	33
Search objects		24	
Attack			35
Received mission order	16	26	36
Received intelligence report	17	27	37
Received message	18	28	38
Sent message	19	29	39

Table A-2. Library of Actions, Study 2.

Action	Code
Approve	1
Check procedure	2
Distributed	3
Forward	5
Posted	7
Reminder	8
Rejected	9

Appendix B

This appendix describes how the measures of the independent variables are calculated before they are included in the analysis. Consider the following example involving a set of sequences from the invoice management setting. Each sequence represents a set of actions performed when handling a given invoice. For simplicity, we limit our example to nine sequences and three vendors (A, B, and C), as shown in Table B-1.

Table B-1. Example data from invoice management.

Vendor	Invoice	Action sequence	Number of actions
A	A-1	3 1 1 7	4
A	A-2	3 1 7	3
A	A-3	3 3 1 7	4

(Continued)

Table B-1. (Continued)

Vendor	Invoice	Action sequence	Number of actions
B	B-1	3 1 7	3
B	B-2	3 5 1 2 7	5
B	B-3	3 1 2 7	4
C	C-1	5 8 3 1 1 7 7	7
C	C-2	5 8 5 1 1 7	6
C	C-3	8 5 9 5 1 7	6

The size of the repertoire of routines (RP)

We operationalize the size of the repertoire as the number of recognized patterns within a stream of actions used to perform the task. To recognize patterns in the example, we use the VMSP algorithm (vertical mining of maximal sequential patterns; Fournier-Viger et al., 2014b). This algorithm recognizes patterns with maximal information (i.e., the longest patterns that meet the chosen recognition criteria) but at the same time ignores patterns that are embedded into longer ones. We use the nine sequences in Table B-1 as input data and set the support level to 33%. We also set the minimum pattern length to three actions. The choice of these parameters will influence the number of recognized patterns. The Python code for VMSP pattern mining algorithm with examples can be found on Gitub.[3] The output of the pattern mining is presented in Table B-2.

Table B-2. Four patterns are recognized in the data.

Pattern ID	Recognized patterns	Length of pattern	Support	Used by vendor
1	8 1 7	3	3	C
2	5 1 7	3	4	B, C
3	3 1 7	3	7	A, B, C
4	1 1 7	3	3	A, C

There were four recognized patterns. The most common pattern was *3 1 7* (Distributed—Approve—Posted). The frequency of this pattern is 7 (i.e., 7/9 = 78% support level). Counting the number of recognized patterns used for handling the invoices from each vendor results in the size of the repertoire for that vendor. This number varies from vendor to vendor, as shown in Table B-2. For vendors A and B, the repertoire size is equal to two, while for vendor C it is equal to four, implying that all patterns were used to handle the invoices from that vendor.

The degree of routinization (RT)

Having recognized the patterns and size of repertoire, we can now compute the degree of routinization, i.e., the percentage of actions that fit the recognized patterns. We used the formula from the method section to calculate the routinization for handling each vendor, as shown in Table B-3.

Table B-3. Routinization is computed for each vendor.

Vendor	RT
A	$\frac{I_3 * 3_3 + I_4 * 3_4}{11} = .55$

(Continued)

Table B-3. (Continued)

Vendor	RT
B	$\frac{I_2 * 3_2 + I_3 * 3_3}{12} = .50$
C	$\frac{I_1 * 3_1 + I_2 * 3_2 + I_3 * 3_3 + I_4 * 3_4}{19} = .63$

Note: The subscript in the numerator denotes the ID of the recognized pattern (from Table B-2).

In this example, all identified patterns had the same length (three actions), but in general, the output of pattern mining provides a set of patterns that vary in length. Although vendors A and B had the same size of repertoire (RP=2) the degree of routinization differs. The reason for this is the different number of total actions in the denominator (11 vs. 12).

Enacted complexity (EC)

To calculate the enacted complexity, we use the action sequences to construct the corresponding action network for each vendor. The result is a directed network, where nodes represent actions and edges the sequential relations between these actions. The enacted complexity for each vendor is the number of paths from the start (*Source*) to the end (*Sink*). We use the *igraph* R package (Csardi & Nepusz, 2006) for this calculation. Figure B-1 illustrates the three action networks with the corresponding number of simple paths and the enacted complexity.

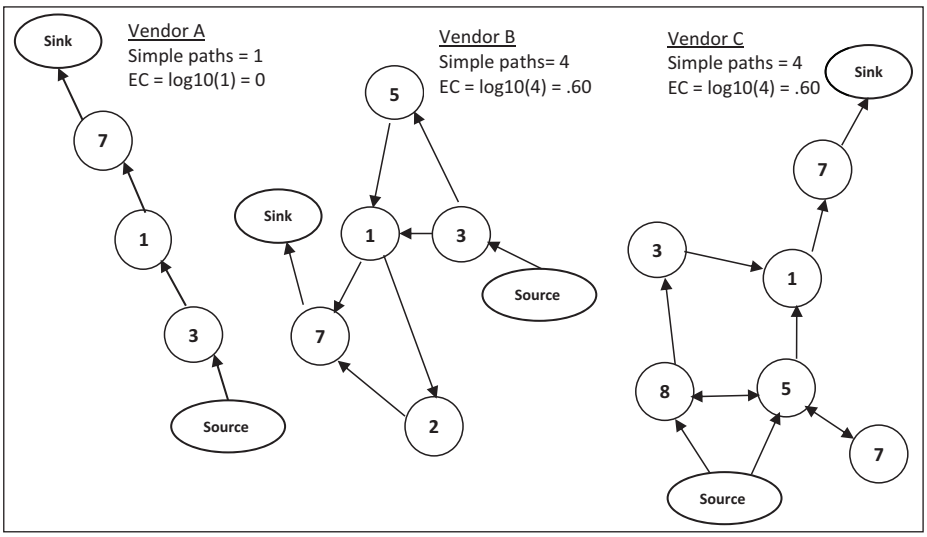

Figure B-1. Enacted complexity is based on the action network for each vendor.

For vendor A, despite three different sequences, the pairwise relations between actions remained the same, and the resulting action network produces a straight line. The number of possible paths is equal to 1, and the enacted complexity is equal to 0. For vendors B and C, the action networks are more complex because there are four different paths from *Source* to *Sink*. In general, the higher the number of nodes and distinct edges, the more complex the action network and the higher the enacted complexity. Because enacted complexity grows exponentially with the size of the network, we use the $\log_{10}$ transformation to normalize our variable.