Toward Methodological Guidelines for Process Theories and Taxonomies in Software Engineering

Paul Ralph, *Member, IEEE*

Abstract—Software engineering is increasingly concerned with theory because the foundational knowledge comprising theories provides a crucial counterpoint to the practical knowledge expressed through methods and techniques. Fortunately, much guidance is available for generating and evaluating theories for explaining *why* things happen (variance theories). Unfortunately, little guidance is available concerning theories for explaining how things happen (process theories), or theories for analyzing and understanding situations (taxonomies). This paper therefore attempts to clarify the nature and functions of process theories and taxonomies in software engineering research, and to synthesize methodological guidelines for their generation and evaluation. It further advances the key insight that most process theories *are* taxonomies with additional propositions, which helps inform their evaluation. The proposed methodological guidance has many benefits: it provides a concise summary of existing guidance from reference disciplines, it adapts techniques from reference disciplines to the software engineering context, and it promotes approaches that better facilitate scientific consensus.

Index Terms—Research methodology, process theory, taxonomy, theory for analysis, theory for understanding, model, framework, guidelines, case study, grounded theory, questionnaire, experiment, action research

1 Introduction

The software engineering (SE) community has recently witnessed increased interest in theories for explaining, predicting, analyzing and understanding diverse SE phenomena [1], [2], [3], [4], [5], [6]. These theories provide "basic concepts and underlying mechanisms, which constitute an important counterpart to knowledge of passing trends" [4].

A theory is a system of ideas for explaining some phenomenon [7], [8]. Theories can be divided into *variance theories* (which explain why something happens), *process theories* (which explain how something happens) and *theories for understanding* (which organize what is happening into useful categories) [7], [8].

A variance theory is a system of ideas that aims to explain and predict the *variance* in a dependent variable using independent variables. Variance theories typically posit probabilistic, causal relationships between constructs [9], [10].

A *process theory*, contrastingly, is a system of ideas intended to explain, understand (and sometimes predict) how an entity changes and develops (cf. [8], [11]). For example, the theory that organisms evolve through mutation and natural selection (Fig. 1) is a process theory because it explains how species change and develop over time. It explains a higher-level process (evolution) by dividing it into several lower-level component processes (mutation, selection and reproduction). Software engineering research has produced numerous theories of the software development process and other related processes (see below).

Meanwhile, *a theory for understanding* is a system of ideas for making sense of what exists or is happening in a domain. The term "theory for understanding" is common among interpretivist researchers [12], [13]. SE researchers sometimes call them "frameworks", "models" or "typologies," although these terms are used inconsistently. Information systems researchers sometimes call them "theories for analysis" [7]. However, this paper adopts the earlier and more precise term for these contributions: "taxonomy".

Taxonomy is the study of classification, and *a taxonomy* is a classification. For example, the tree of life is a taxonomy because it organizes instances (organisms) into a set of related classes (e.g., species, genus, family). Software engineering research often produces taxonomies, including classifications of roles, success factors and software quality dimensions (see below).

While the SE community has developed considerable methodological guidance for generating and evaluating variance theories (e.g., [14]), little SE-specific guidance is available for process theories or taxonomies. The purpose of this paper is therefore as follows.

> ***Purpose***: *to adapt existing guidance on developing and evaluating taxonomies and process theories for software engineering.*

• *The author is with the Department of Computer Science, University of Auckland 28 Princes St., Auckland 1010, New Zealand. E-mail: paul@paulralph.name.*

Manuscript received 6 Apr. 2017; revised 29 Dec. 2017; accepted 15 Jan. 2018. Date of publication 22 Jan. 2018; date of current version 24 July 2019. (Corresponding author: Paul Ralph.) Recommended for acceptance by T. Zimmermann. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TSE.2018.2796554

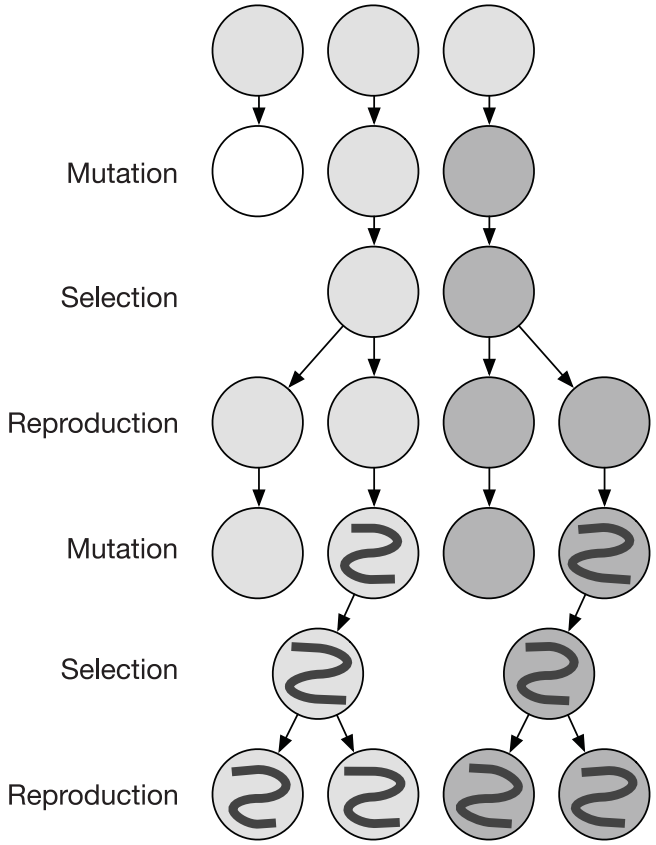

Fig. 1. Theory of evolution through natural selection.

1.1 Are Process Theories and Taxonomies Theories?

Some scholars do not believe that process theories or taxonomies are theories (cf. [7]). For example, defining a theory as a set of constructs and propositions [9], [15] marginalizes process theories and taxonomies. One might complain that process theories do not typically posit laws, or that taxonomies do not have quantitative constructs. However, there is simply no consensus across scientific communities as to the meaning of theory. The definition adopted in this paper ("a system of ideas for explaining some phenomenon"), is intentionally broad to capture the diversity of theories across scientific communities.

Practically speaking, to dismiss process theories as not real theories is to dismiss evolution by natural selection, plate tectonics and climate change as not real theories, which is patently absurd.

The case for taxonomies is subtler. Classification systems that do not explain real world phenomena are not theories. For example, if we randomly divide a class of 100 students into 20 groups of five students each to complete a course project, the resulting classification is not a theory. If, however, we study 100 students, divide them into six groups based on some principle or empirical finding, and most importantly *claim* that we have discovered six types of students, then it is a theory because it *explains* something.

1.2 Studying Process Theories and Taxonomies Together

This paper simultaneously considers taxonomies and process theories because many process theories *are* taxonomies (see below). Considering them together therefore helps us better understand each. Methodological guidelines specifically tailored for taxonomies and process theories are needed for at least three reasons:

1) Most research approaches for evaluating causal relationships are inappropriate for evaluating process theories or taxonomies.
2) Most guidance from reference disciplines needs to be adapted to the types of contexts and questions SE researchers investigate. Guidance from sociology and organization studies, for example, often lacks SE's focus on technical artifacts.
3) Reading a single introduction, tailored to one's field, is more efficient and less confusing than wading through a hundred papers and books from a dozen other fields trying to create and justify a reasonable methodology.

1.3 Outline

This article therefore proceeds as follows. Next, it reviews the importance of taxonomies and process theories for software engineering research. Then, it gives examples and discusses quality criteria for process theories (Section 3), followed by taxonomies (Section 4). Sections 5 and 6 provide methodological guidelines for theory generation and evaluation, respectively. This is followed by a discussion of the paper's limitations and applications (Section 7). Section 8 concludes the article concludes with a summary of its contributions.

2 Why Are Taxonomies and Process Theories Important?

Process theories and taxonomies are important in SE research for several reasons. They provide nomenclature that helps us to identify, describe and understand the entities and events in a domain. Scientists need nomenclature not only to communicate clearly and precisely but also to reason effectively about observations [16]. For example, role names like "developer" and "project manager" help us think about conflicts between developers and managers. SE in particular suffers from myriad ill-defined and overlapping terms [17].

Without taxonomies and process theories, some SE research is conceptualized using concepts from software development methods (SDMs). An SDM is a system of prescriptions for creating, evolving or maintaining software. Because SDMs aim to prescribe good ways of doing something rather than to explain all the ways of doing something, they inevitably oversimplify and over-rationalize reality [18], [19]. Researchers indoctrinated in SDMs may therefore struggle to recognize phenomena that are observable in real projects but downplayed in the SDM literature (e.g., problem framing [20], illusory requirements [21] and design coevolution [22], [23]). When surveys and qualitative coding schemes are based on the concepts and assumptions of SDMs (e.g., [24]), it introduces insidious, poorly-understood biases. For instance, a survey question might ask, *during which phase of the project were the requirements agreed—analysis, design, coding, testing, implementation or maintenance?* This presumes: 1) all projects have discernable phases; 2) all phases map to those listed; 3) all projects have requirements;

4) requirements are always agreed. In summary, by encapsulating empirical, explanatory knowledge, process theories and taxonomies counterbalance SDMs, which provide prescriptions.

Furthermore, educators need some basis for teaching SE. Taxonomies provide classes in meaningful structures (e.g., six roles on agile teams; seven dimensions of software quality). Process theories provide accounts of SE processes (e.g., development can be divided into sensemaking, coevolution and implementation). Without process theories and taxonomies, educators continue to base curricula on SDMs [18], [25]. This gives students idealistic, oversimplified, incorrect views of the great diversity of SE approaches and projects. It encourages conflating how SE should be done with how it is done. Contrasting theories with SDMs helps distinguish description and explanation from prescription.

Process theories and taxonomies moreover help identify gaps in our knowledge and direct future research. Taxonomies often reveal neglected areas. For example, if we had nine coupling metrics, eight size metrics and eleven cohesion metrics but only one complexity metric, we might need more research on complexity metrics. Similarly, process theories often reveal previously unknown or understudied activities, forces, conflicts or mechanisms. For instance, realizing that a code base and its test suite coevolve (rather than tests simply driving changes in code), suggests investigating antecedents of code-test coevolution effectiveness.

Process theories and taxonomies also inform causal theories. For example, we cannot have an independent variable such as *Test Case Understandability* without a *Test Case* class. Meanwhile, if process research showed that ecommerce website developers predominately design databases before drawing user interface mock-ups, we would not hypothesize that mock-up quality leads to database quality.

In addition, taxonomies increase cognitive efficiency by grouping similar instances together (see below). Taxonomies help us make inferences about instances based on their classification (e.g., if these are mammals then they should have fur; if these are unit tests then they should be automated). We gain great cognitive efficiency by reasoning about classes (e.g., quality assurance specialist, subroutine) rather than instances (e.g., Susan Wang across the hall, System.out.println()).

Process theories, meanwhile, often reveal counterintuitive processes or refute common misconceptions. For example, process theory research on decision-making shows that, rather than generating several alternatives and choosing the best one, software designers and programmers tend to implement the first plausible option that comes to mind [26] [27]. Without such empirical knowledge, pseudoscientific nonsense continues to infest SE discourse. For example, the Gartner Hype Cycle posits that information technologies exhibit a common pattern of visibility as they mature [28]. Without a rigorous theory of technology visibility, the hype cycle remains influential despite poor predictive validity [29].

In summary, process theories and taxonomies are needed in software engineering because they provide concepts and technical language needed for precise communication and education. Taxonomies also provide cognitive efficiency by facilitating reasoning about classes rather than instances, while process theories often expose counterintuitive truths and combat pseudoscience.

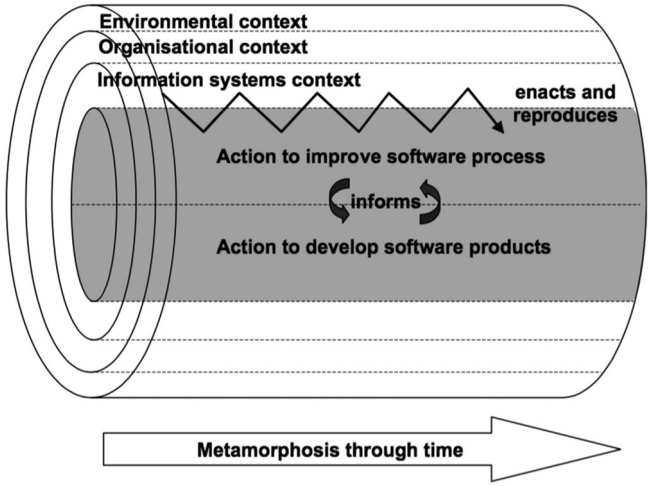

Fig. 2. An emergent view of software process improvement (from [31]).

3 A Brief Review of Process Theories

A *process theory* is a system of ideas intended to explain, understand (and sometimes predict) how an entity changes and develops (cf. [8], [11]). This section describes the types of process theories, distinguishes process theories from similar contributions, and discusses their main quality criteria.

3.1 Types of Process Theories

Process theories can be classified in several ways. For example, Hernes and Weik [30] distinguish between "endogenous" process theories, where the process occurs with a stable environment, and "exogenous" process theories, where the process creates stability.

Meanwhile, Van de Ven and Poole distinguish four process theory architypes—dialectic, teleological, evolutionary and lifecycle [11]. These architypes vary in their structure and driver of change, or "motor".

In a *dialectic process theory*, "stability and change are explained by reference to the balance of power between opposing entities" [11, p. 517]. Although rooted in Hegelian dialectics (where thesis begets antithesis begets synthesis), dialectical process theories more generally involve an interplay between two or three activities, actors, concerns, forces or organizational logics. One dialectic familiar to many software developers is the tension between project velocity and product quality. This tension shapes many decisions, for instance, whether to refactor or ignore mounting technical debt [27]. Dialectic process theories therefore posit two or more conflicting entities and explain how these conflicts shape ongoing change. These theories are often concerned with power imbalances.

For example, Allison and Merali [31] propose a dialectic theory of software process improvement (Fig. 2). It posits a dialectic interplay between software development and software process improvement, where each informs the other. Both process and product metamorphose over time, changing and being changed by their surrounding context.

In a *teleological process theory*, an agent "constructs an envisioned end state, takes action to reach it and monitors

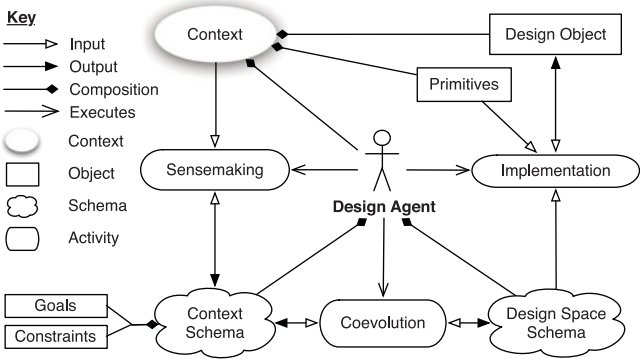

Fig. 3. Sensemaking-coevolution-implementation theory (adapted from [146]).

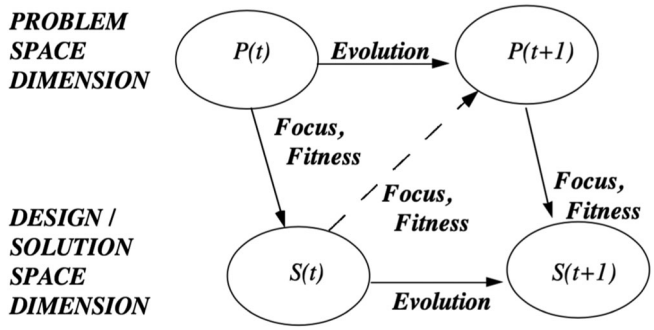

Fig. 4. Problem-design exploration model (from [23]).

the progress" [11, p. 518]. In other words, teleological theories explain the behavior of intelligent agents taking actions to reach goals, but the agent chooses its own sequence of actions. Teleological process theories are very common in the social sciences [11].

Sensemaking-Coevolution-Implementation Theory (SCI) is a teleological theory that explains how a development team builds complex software systems (Fig. 3). It posits that development teams engage in three basic activities: 1) making sense of an ambiguous, problematic context (*Sensemaking*); 2) rapidly oscillating between ideas about the context and ideas about the space of possible design artifacts (*Coevolution)* and building the system (*Implementation*) [22], [32]. These activities may occur in any order or in parallel.

In an *evolutionary process theory*, a population undergoes structural changes through *variation* (producing new entities), *selection* (preserving entities with higher fitness and eliminating those with lower fitness) and *retention* (beneficial properties of instances endure).

For example, the Problem-Design Exploration Model [23] is an evolutionary process theory that explains how genetic algorithms may be used to design systems (Fig. 4). The problem space (constraints) and solution space (designs) are modeled as coevolving populations—each affecting the other.

In a *lifecycle process theory*, an entity transitions between discrete states or events "which [are] cumulative (characteristics acquired in earlier stages are retained in later stages) and conjunctive (the stages are related such that they derive from a common underlying process)" [11, p. 515]. This progression occurs because "the trajectory to the final end state is prefigured and requires a particular historical sequence of events" [11, p. 515]. Despite their name, many lifecycle theories are linear rather than cyclical.

Lifecycle theories have their roots in biological life cycles (e.g., metamorphosis) and are common in natural sciences (e.g., the Carbon Cycle, the Nitrogen Cycle, Plate Tectonics, various theories of early childhood development). SE researchers, in contrast, often describe lifecycles without labeling them as theories (see below). If it were a theory, the Waterfall Model [33]—either the linear version Royce condemned or the more iterative version he proposed—would be a lifecycle theory.

Process theories can combine two or more archetypes. Poole and Van de Ven [34] call these "multiple motor processes" and emphasize the importance of clearly explaining how the motors interrelate. Sometimes the archetypes exist at different levels of abstraction. For example, the coevolution activity in SCI is a dialectical process. Other times the motors are entangled, or two or more subprocesses with different motors combine to shape the changing entity (see [34] for more details).

While process theories do not typically posit causal relationships between constructs, they may adopt a particular approach to causation [10]. Evolutionary process theories, for instance, adopt a probabilistic approach to causation since entities with higher fitness have a greater probability of surviving. Teleological process theories adopt teleological causation; that is, actions are caused by the choices of agents with free will. Lifecycle theories adopt counterfactual causation since later phases cannot occur with earlier phases.

3.2 Common Misconceptions About Process Theories

Some researchers conflate process theories with SDMs, process models, algorithms or variance theories. This leads to misunderstanding the kinds of claims process theories make (see below) and evaluating them inappropriately.

Process Theories are not SDMs. A software development method(ology) prescribes while a process theory explains. Scrum [35], Lean [36] and the V-model [37] are SDMs (or possibly project management frameworks) because they prescribe specific practices, techniques, tools or sequences, which are ostensibly better than their alternatives. A process theory contrastingly explains how something actually occurs, regardless of effectiveness. While an SDM claims "this way will work;" a process theory claims "it happens this way." The theory of natural selection posits that evolution follows a specific pattern, not that this pattern is efficient. Scrum contrastingly claims that organizing development into timeboxed sprints is wise—not that everyone does so [35].

Process Theory is not a Synonym for Process Model. Process theory research differs significantly from process model research. "A process model is an abstract description of an actual or proposed process that represents selected process elements that are considered important to the purpose of the model" [38, p. 76]. A process model represents a distinct collection of processes, while a process theory offers a universal explanation of a change process.

For example, Fig. 5 shows how a manufacturer might build custom components. It does not claim that *all* manufacturers act this way or that a specific manufacturer acts this way all of the time. The theory of natural selection, in contrast, posits that *all* complex life arises through variation,

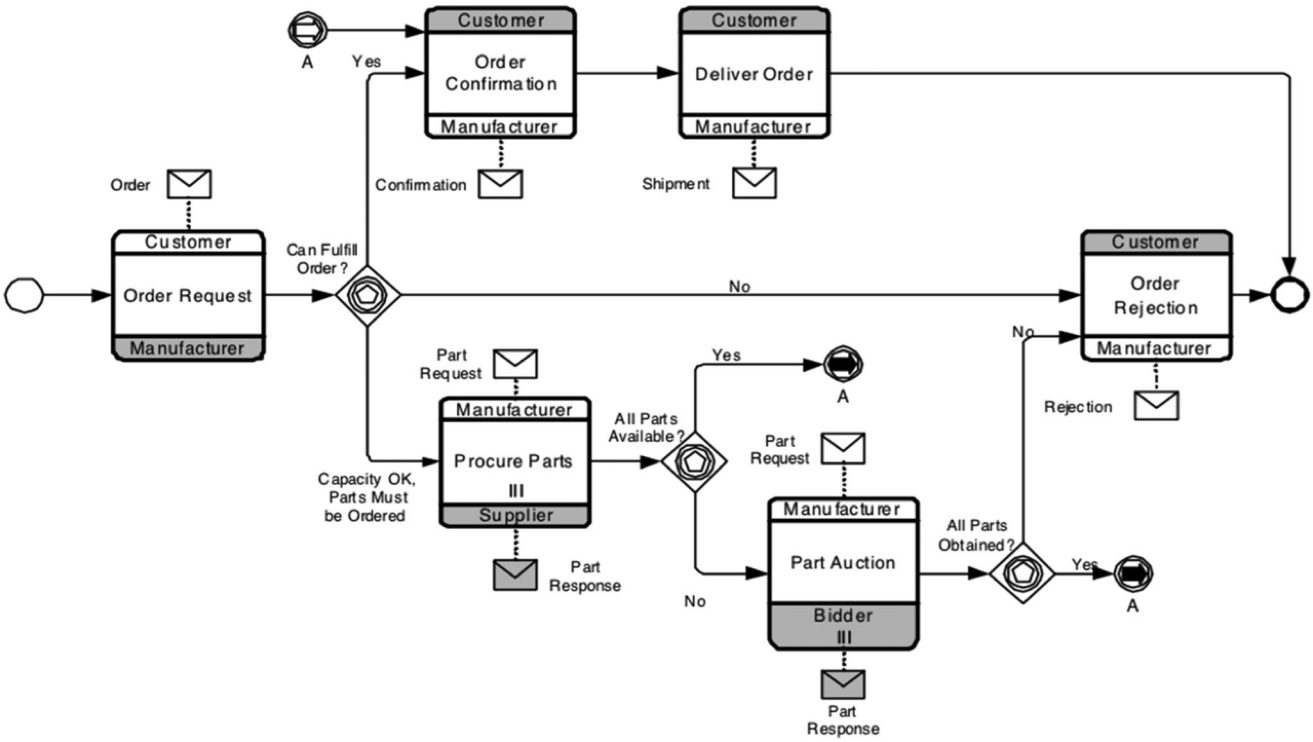

Fig. 5. Made-to-order process model (from [146]).

selection and retention. Process models (like SDMs) often describe one or a few sequences while process theories seek to encompass all of the ways an entity can change.

A Process Theory is not an Algorithm. While software engineers often think of a process as a set of steps in a specific order (an algorithm), process theories do not necessarily posit sequential orders or divide processes into steps. Thinking of variation, selection and retention as steps, for example, is deeply misleading. These activities occur in parallelized chaos rather than a disciplined cycle. Teleological or dialectical theories rarely posit steps. Even lifecycle theories divide a process into phases or stages rather than algorithm-like steps.

Process Theories do not Typically Posit Causal Relationships between Variables. Variance theories (e.g., The Unified Theory of the Adoption and Use of Technology [39] (Fig. 6) explain and predict the *variance* in a dependent variable (e.g., use behavior) using independent variables (e.g., performance expectancy), mediating variables (e.g., behavioral intention) and moderating variables (e.g., gender). While a variance theory comprises variables and causal relationships, a process theory may include activities, actors, objects, phases, steps, subprocesses and diverse non-causal interconnections. Most process theories only posit causality in the general sense of a causal motor (e.g., in a teleological theory, change emerges from the actions of an agent).

3.3 Quality Criteria for Process Theories

For an empiricist, the primary quality criteria for theories is the strength of evidence supporting them. While many other quality dimensions have been proposed, research quality cannot be reliably ascertained from a pre-ordained set of criteria [40], and attempting to standardize common quality criteria for theories is undesirable [41]. Rather, this section examines how the quality criteria laid out by Charmaz [12] (a constructivist) and Trochim [42] (a positivist) pertain to process theories. Related criteria are discussed together.

Usefulness/Relevance. Many have suggested that theories should be useful to someone for some purpose. However, when Kuhn [43] argues that more useful theories supersede less useful ones, he is talking about usefulness to scientists for solving scientific puzzles and problems, not usefulness to professionals for solving practical problems. Species do not *use* the theory of evolution; scientists use it to explain the evolution of species. Similarly, software developers do not use Allison and Merali's theory of software process improvement or SCI, they are for scientists to study developer behaviour.

Even in an applied science, theories are mostly useful to academics for understanding the domain and for creating tools, practices and pedagogy. If a theory is also directly

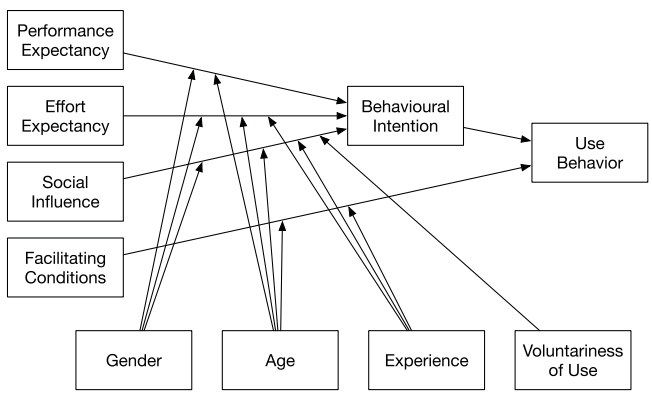

Fig. 6. The unified theory of the adoption and use of technology (from [39]).

useful to practitioners, that should be viewed as an added bonus in my opinion.

Originality. Good theories "offer new insights" and "challenge, extend or refine existing ideas, concepts and practices" [12, p. 182]. It is far too easy, however, to dismiss good research because 'we already know this' (translation: 'we already suspected this but had no cohesive explanation or empirical evidence'). When a new theory synthesizes existing ideas into a simpler, more integrated and more useful form, it is not appropriate to reject it for lack of novelty. The opposite—a theory completely divorced from existing research—is suspicious.

Resonance/Believability. In interpretivist research, resonance or believability refers to the degree to which findings make sense to the study's participants. Because interpretivist research focuses on the meaning ascribed to events by participants, if participants say the theory does not make sense to them or reflect their beliefs, we have good reasons to doubt the theory. Resonance may be a good criteria for interpretive (and grounded theory) research. However, suppose a social psychologist developed a theory of the psychological process by which people are manipulated through propaganda. Participants might deny ever having been manipulated and so reject accurate portrayals of their manipulation. Good theories that refute common misconceptions therefore may have poor resonance.

Parsimony/Simplicity. When we say theories should be parsimonious, we do not mean all theories should be simple. String theory and quantum mechanics are not simple. Rather, parsimonious theories are no more complicated than necessary to explain the phenomenon of interest. The key is to find a reasonable middle ground between oversimplifying reality and overcomplicating the theory. Scholars can reveal this to readers by discussing both concepts that were omitted from the theory for not adding enough explanatory power, and justifying some concepts that were included. For example, SCI includes the concept of *primitives* because developers so often build systems from existing components, but excludes the concept of *time* because it is extraordinarily complicated to model.

Internal and Conclusion Validity. Conclusion validity refers to whether a study supports a hypothesized numerical relationship between two variables; internal validity refers to whether a study supports a causal relationship between two variables [42]. Process theories do not typically posit causal or covariance relationships between numerical variables. None of the process theories discussed in this paper posit numerical variables or causal relationships between them. Internal validity and conclusion validity therefore usually do not apply to process theories.

Construct Validity. A *construct* is a variable that cannot be measured directly [42]. Software engineering success, software complexity, coupling, cohesion, program size, project risk and developer productivity are constructs. Social scientists operationalize constructs as the shared variance of several reflective indicators. For example, we might use lines of code, number of classes and number of function points as reflective indicators of program size. Variance (causal) theories largely posit relationships between constructs, and much of social science demonstrates causal relationships between constructs using observations of reflective indicators. Process theories, however, rarely have constructs. Actors, agents, activities, process, stages, phases, dialectic tensions, evolving populations, contexts, objects and fitness functions are not constructs. You cannot have a software process improvement of −17 or a coevolution of 0.13. Unless the process theory in question posits one or more quantitative variables that cannot be directly measured, construct validity does not apply.

Generalizability / Transferability / External Validity / Universality. In positivist, objectivist research—especially questionnaire studies—*external validity* usually refers to the degree to which findings can be statistically generalized from a sample to a population. Researchers are expected to demonstrate that their findings apply to a larger group; that is, the theory is "universal." Postmodernism rejects universal, general theories [44]. Many interpretivists therefore reject universality and external validity as quality criteria. In interpretivist research, transferability refers to the degree to which concepts can be transferred from one setting to another, and this is usually the responsibility of those applying the concepts to a new setting, rather than the researcher who generated the concepts. However, researchers actually engage in at least four types of generalization: 1) from data to descriptions, 2) from descriptions to concepts, 3) from concepts to theory, 4) from theory to descriptions [45].

In a theory development paper, we expect researchers to explain how they generalized from data to descriptions to concepts to theory. Statistical generalization, in contrast, is usually out of scope. Interpretivist researchers often discuss transferability; for example, researchers might say something like "While the dialectic tension between velocity and technical debt may not be universal, it resonates with our experience in many web development projects."

In a positivist theory evaluation paper, on the other hand, we would normally expect some discussion of statistical generalizability. This is problematic, however, because we do not have comprehensive population lists for many phenomena of interest. For instance, we have no list of all of the software developers, code libraries, unit tests, software projects or software companies in the world or in any particular country.

In my opinion, then, two avenues of external validity criticism are reasonable, and two are not. If a process theory is generated based on data from a single context and includes concepts or propositions that seem rare or difficult to apply to other, well-understood SE contexts, we can reasonably question their transferability. Furthermore, if a process theory is evaluated using a questionnaire in a situation where a reasonable sampling frame is available, we can reasonably expect random sampling and criticize the absence thereof. For example, if we have access to a population of projects hosted on GitHub, which supports random sampling, analyzing a convenience sample for no clear reason would be questionable. However, rejecting qualitative process theory research for having small, convenience samples while accepting experiments with small, convenience samples is prejudice against qualitative research, not "high standards." Moreover, cross-paradigm criticism is nearly always inappropriate because it applies quality criteria rooted in one set of philosophical assumptions to a study rooted in opposite assumptions. Criticizing an interpretivist theory development paper for poor external validity is like

criticizing an experiment for poor believability—both are unhelpful cross-paradigm criticism.

Testability. Most scholars would probably agree that theories should be testable, but testability is more complicated than it first appears. Testability comes from the verifiability criterion of meaning, the central thesis of the logical positivists in the 1920s. It held that a statement is meaningful if and only if it can be verified from sensory experience. Popper [46] disputed verificationism and argued instead for falsificationism, that is, the idea that scientists should attempt to disprove, rather than prove theories. Both verificationism and falsificationism were refuted by Quine [47] among others.

Briefly, when actual observations do not match expected observations, at least four possibilities are evident: 1) the theory is wrong; 2) the proposition was incorrectly deduced from the theory (i.e., the theory was misapplied to the domain); 3) the instrumentation or observer made a mistake; 4) the researcher made a mistake analysing the data.

More generally, one cannot prove a universal hypothesis (e.g., all swans are white) because it requires checking every member of the population and somehow knowing we have not missed any. One cannot falsify an existential hypothesis (e.g., there exists a black swan) for exactly the same reason. Meanwhile, social science is predominately concerned with probabilistic hypotheses, (e.g., smoking causes lung cancer), which can be neither proven nor falsified. No number of observations of smokers with lung cancer proves that smoking causes cancer and no number of observations of smokers without lung cancer disproves the hypothesis.

This led to the development of the strong inference model [48] and Bayesian epistemology [49]. In this view, a theory is only testable against another theory that makes contradictory claims [50]. Theories are tested by comparing them against explicit or implicit rival theories. Consistent with Kuhn [43], each study tips the epistemic scales toward or away from any particular theory, and knowledge is simply the best explanation we have at the time.

Should process theories be testable then? In established, theory-rich areas, it should be possible to identify a rival theory and give plausible examples of observations that would favour the proposed theory and alternatives that would favour an existing rival theory. For early theories in emerging areas, however, no plausible rival theory may be evident. However, researchers can still specify exactly what the proposed theory claims and identify plausible alternative propositions and potential supporting observations. If one cannot imagine plausible observations that would constitute evidence against a theory, the theory may be tautological.

Theoretical Saturation. In grounded theory, a theory is saturated when its concepts and relationship are robust and fully developed, with no obvious gaps or unanswered questions [12]. This makes sense for process theories whether or not they are generated using grounded theory. Saturation can be demonstrated by thoroughly discussing the theory's concepts, relationships and implications.

Summary. In summary, authors should discuss quality criteria that make sense for their contexts. Quality criteria should be applied critically rather than simply used to promote or disparage a theory. No single, universal set of quality criteria exists. Reviewers should be free to ask researchers to discuss additional relevant quality criteria. Reviewers should not expect authors to adopt a universal, pre-ordained set of quality criteria because no such set exists.

3.4 Limitations

Process theories are intrinsically limited in several ways. They normally do not make point estimate predictions; for example, how long a process will take. With the exception of evolutionary theories, process theories rarely make probabilistic predictions; for example, whether a project will succeed. Process theories usually address causation only in the vague sense of identifying the primary mechanism driving the change. Moreover, incorporating prescriptions into a process theory is intrinsically problematic. While a researcher might make prescriptions by interpreting a situation using a process theory, the theory itself should explain only what is, not what should be.

Most process theories do not address sequences of events, and those that do (lifecycle theories) tend to oversimplify and over-rationalize human behavior [18], [19]. A slight tangent is needed here to explain the problem with theorizing human behavior in terms of lifecycles.

Consider the lifecycle of a butterfly. The development of butterflies is conducive to a lifecycle theory for three reasons:

1. There are distinct, identifiable, cohesive phases: egg, caterpillar, cocoon, butterfly.
2. The phases are universal: all surviving butterflies go through the same phases.
3. The phases always occur in the same order; butterflies cannot hatch directly from eggs.

Consider, in contrast, software projects:

1. Many software projects have (at best) fuzzy phases with messy transitions; others have no identifiable phases at all.
2. Phases vary dramatically between projects. Some projects spend months developing a formal proposal, or funding application; others do not. Some projects involve months of requirements analysis, others skip requirements entirely. Some projects have a long "learning about the legacy system" phase while others begin with a blank slate. Some projects have an elaborate deployment phase while others employ continuous delivery. In some projects, phases map to dominant activities (e.g., requirements, design, programming); in others the phases are two-week iterations comprising everything from a planning meeting to shipping a product update.
3. The order of phases varies dramatically. Some projects begin with big-design-up front, others begin with programming. Some projects end with maintenance; others start out as maintenance initiatives and morph into development efforts.

Lifecycle theories are for predictable, sequential processes, like the life of a butterfly. Human behavior generally and software development specifically are not predictable and sequential. People can change their goals, change their activity sequence, invent new activities, give up part way through a process and generally take unexpected actions. In Glaser's terms, "forcing" an unpredictable, irregular,

Organization
- Commitment
  - Senior Management Commitment
  - Commitment of Stakeholders
- Organizational Culture
  - Organizational Change
  - Speed of Change
- Organizational Learning
  - Lessons Learned
  - Feedback
  - Knowledge Transfer

Fig. 7. Excerpt of software process deployment critical success factors taxonomy [54].

non-sequential process into a lifecycle blinds the researcher to what is actually happening. Other approaches, including dialectical and teleological theories, are better for understanding human behavior [22], [51].

4 Taxonomies and Classification Systems

Classification is a fundamental cognitive process for improving cognitive efficiency and supporting inferences [52]. Models, frameworks, theories for analysis, descriptive theories and theories for understanding often classify instances into classes. In classification theory, an *instance* is an individual thing, while "a *class* is an abstract description of a set of properties shared by a set of instances" [53, p. 841] (emphasis added). A classification or taxonomy is a collection of classes. For the remainder of this paper, *class* generally refers to taxonomic classes, not classes in object-oriented programming.

4.1 Examples of Taxonomies in Software Engineering

SE researchers often create taxonomies. For example, Bayona-Oré et al. [54] present a taxonomy of critical success factors for software process deployment (see Fig. 7 for excerpt). It organizes critical success factors (instances) into subclasses (e.g., Organizational Culture) and superclasses (Organization). It has two characteristics often associated with taxonomies—hierarchy and mutually exclusive classes.

Not all taxonomies have mutually exclusive classes. For example, Boehm's [55] taxonomy (Fig. 8) organizes software

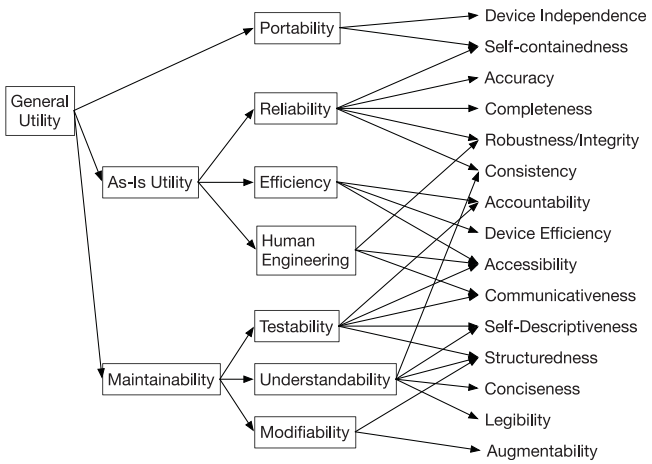

Fig. 8. Boehm's taxonomy of software quality dimensions (from [55]).

TABLE 1
Role Taxonomies for Software Project Managers [56] and Self-Organizing Teams [57]

Project Manager Roles	Agile Team Roles
Expert	Mentor
Strategist	Coordinator
Leader	Translator
Problem Solver	Champion
	Promotor
	Terminator

quality dimensions (e.g., accessibility, structuredness) into classes (e.g., testability, maintainability). Some dimensions are part of two or more classes.

Similarly, not all taxonomies have hierarchy. For example, Maglyeas et al. [56] generated a taxonomy of software project manager roles while Hoda et al. [57] generated a taxonomy of roles in self-organizing agile teams (Table 1). Here, the roles are classes and the instances are individual people or patterns of behavior.

Some taxonomies organize a single instance into two or more orthogonal classes. For example, Ralph and Tempero [27] propose a taxonomy of coding decisions (Fig. 9). Here, individual decisions are organized into two orthogonal superclasses: *kind* (e.g., architectural decision, design decision) and *choice architecture* (e.g., choosing between two alternatives, choosing between more than two alternatives).

More generally, systematic mapping studies (e.g., [58]), interview studies (e.g., [59]), interpretive case studies (e.g., [60]) and grounded theory studies (e.g., [61]) typically present taxonomies without necessarily labeling them as such. Qualitative coding [62] is a kind of classifying where instances including interviewee quotations and excerpts from documents, field notes and memos are organized into classes including themes, roles, objects, process, activities, agents and tensions. We can therefore conceptualize most products of qualitative research as taxonomies.

The key point of this section, however, is that a taxonomy is simply a set of classes defined over some instances. Taxonomies do not always have mutually exclusive categories or hierarchy. They do not always formally define the properties of and relationships between classes (as ontologies do).

4.2 Quality Criteria for Taxonomies

Recall that the purposes of classification are to increase cognitive efficiency and facilitate inferences. Better taxonomies

	Choice Architecture				
Kind	> 2	2	Action	Dialectic	Total
Scope	2%	0%	8%	3%	13%
Design	2%	6%	16%	14%	38%
Architecture	0%	4%	5%	2%	11%
Environment	1%	2%	8%	2%	13%
Implementation	2%	3%	13%	7%	25%
Total	7%	15%	50%	28%	100%

Fig. 9. Taxonomy of coding decisions, with frequency of observation [27].

therefore: 1) increase cognitive efficiency more, and 2) facilitate more inferences. Where a taxonomy is designed for a specific purpose (e.g., organizing a library, helping customers find products to purchase), the degree to which it achieves its objective forms a third quality criteria.

For example, suppose we are observing a team developing software. We observe Kim writing the CallBack() method at 10:49, Sean revising a unit test at 11:17, Kelly looking up syntax at 11:21 and James prioritizing user stories at 11:38. At this level of specificity, we observe too many instances to hold in working memory. We therefore group instances into classes such as writing comments, writing documentation, reading code, reading websites, searching, abstracting, defining, decomposing, planning, conversing or sketching. Depending on our aims, we may have to further organize these into higher level classes (e.g., programming, managing). We then reason about the classes rather than specific instances, facilitating cognitive economy.

However, abstracting instances into classes only facilitates reasoning when classes contain similar instances. Arbitrarily combining *writing software documentation*, *running* and *hats* into a *things in this sentence* class does not assist reasoning. While researchers do not usually produce such capricious classifications, at least five errors are possible:

1. An *inclusion error* occurs when an instance is included in a class despite being dissimilar from the other instances in a class; for example, including a black-and-white horse in the class *zebra*.
2. An *exclusion error* occurs when an instance is excluded from a class despite being similar to the other instances in a class; for example, not including a black-and-white horse in the class *horse*.
3. A *combination error* occurs when two or more dissimilar groups of instances are combined into a single class; for example, classifying pandas and black bears together.
4. A *differentiation error* occurs when similar instances are capriciously divided into two or more groups; for example, classifying brown bears and polar bears separately from black bears and grizzly bears.
5. An *excess* error [63, p. 92] occurs when a class has no instances; for example, positing a *blue bear* class despite there being no blue bears. (Excess errors are rare in evidence-based taxonomies.)

These errors (which follow from the definition and theory of classification) promote bad inferences. For example, misclassifying a black-and-white horse as a zebra might lead us to infer incorrectly that it lives in Africa.

Good classes not only avoid the above errors but also support inferences. Consider a class *red things*. To be a member of class *red things*, an instance must have colour, and that colour must be red. If we know X is a member of *red things*, we cannot infer much if anything further about X. In contrast, consider the class *planet*. To be a member of class *planet*, an astronomical body must orbit a star, be large enough to be round but not large enough to generate nuclear fusion and have cleared its orbit. However, if Y is a *planet*, we can infer many of Y's properties beyond those necessary for inclusion in the class *planet*. For example, Y will rotate on an axis, reflect light and have substantial mass and gravity. If we fly our spaceship too close to Y, we will crash into it. Inferring properties of an instance based on its membership in a class is how classification increases cognitive economy.

In summary, the same instances can often be organized in many ways. In a good taxonomy: 1) class structure reflects similarities and dissimilarities between instances; 2) we can infer many relevant properties of an instance based on its membership in a class, beyond the properties necessary for class membership; 3) the taxonomy is effective at the specific purpose for which it was designed.

4.3 Most Process Theories Are Taxonomies

Teleological, dialectic and lifecycle process theories include taxonomies. That is, teleological, dialectic and lifecycle theories all posit classes. I am not aware of any existing research or commentaries that have made this point, but it logically follows from the definitions provided above.

In a dialectical process theory, the conflicting entities are typically classes. Allison and Merali's theory of software process improvement (Fig. 2), for instance, posits a dialect relationship between "Action to improve software process" and "Action to develop software products". These are obviously classes. The former might include subclasses such as retrospective meetings, process modelling and process reviews. The retrospective meetings subclass might contain instances such as a specific meeting at the Wellington office on January 29 from 3 to 4 pm. The three concentric contexts in which this dialectic is situated are also obviously classes. A transaction database might be an instance of the information systems context, while a specific company might be an instance of the organizational context and a regulatory change might be an instance of the environmental context.

Similarly, the activities posited by a teleological theory are typically classes. In SCI (Fig. 3), for instance, sensemaking, coevolution and implementation each encompasses numerous subclasses. For example, coevolution activities include sketching, brainstorming, revising user stories and discussing. Each of these are, in turn, classes containing specific instances. For instance, the sketching class might include Veronica drawing mock-up 16A on April 9.

Meanwhile, the phases posited by a lifecycle theory are usually classes. For instance, Waterfall model phases (e.g., testing), might include numerous subclasses (e.g., static code analysis, unit testing) defined over even more instances (e.g., Kelly writes unit test HelloWorldTest on September 4).

However, evolutionary process theories are not taxonomies in any meaningful sense. The core processes of variation, selection and retention act directly on the instances. Developing an evolutionary theory is mainly about determining the fitness function, not organizing instances into classes.

This discussion suggests the following alternative definitions for some process theory archetypes.

- *Dialectical process theory*: a taxonomy of opposing entities and their conflicts
- *Teleological process theory*: a taxonomy of activities performed by a goal-oriented intelligent agent
- *Lifecycle process theory*: a taxonomy of phases (/ states / stages / events) through which an entity passes in one or more defined sequences.

Reconceptualising process theories as taxonomies is useful because it suggests using taxonomy quality criteria to evaluate process theories. Obviously, however, not all taxonomies are process theories. Quality criteria for process theories therefore may not apply to taxonomies.

4.4 Limitations

Taxonomies are more limited than process theories. Taxonomies simply define some classes over a population of instances. In addition to the limitations of process theories, taxonomies explain neither how nor why important events occur.

5 Recommended Steps for Generating Taxonomies and Process Theories

5.1 Step 1: Choose a Strategy

Process theories and taxonomies can be generated in many ways. The best ways, in my opinion, are using secondary studies, interpretive case studies or grounded theory studies.

5.1.1 Secondary Studies

Researchers can create taxonomies or process theories by synthesizing, combining, refining or adapting existing theories from SE or reference disciplines. Systematic literature reviews [64], mapping studies [65] and thematic syntheses [66] often organize studies or their results into taxonomies (e.g., [54], [67]). Meta-analysis [68] is fundamentally a positivist approach to theory-testing, so it is not recommended for generating theories. Meta-synthesis [69] (also known as meta-ethnography) is more appropriate for theory development.

Furthermore, some SE processes are special cases of more general processes. For example, making software design decisions is a special case of decision making. A process theory of software design decision making therefore might be adapted from a more general theory of decision making (e.g., [70]). General theories of negotiation, group mind, contracts, learning, communication, creativity, problem structuring and crime may be similarly applied to SE special cases. SCI, for example, was developed by refining an older model of "self-conscious" design [22], [71].

Benefits of the synthesizing approach include speed (compared to months or years of field work), natural linkages to existing theories and possibly greater scientific rigor (insofar as working with rigorous theories encourages us to produce more rigorous theories). It should produce results that are consistent with existing research.

However, existing literature will inevitably frame the new theory, possibly overemphasizing some instances, properties, activities or actors while ignoring others. Theories from reference disciplines do not always translate well to SE. Additionally, generating theory from poor research or nonempirical literature may produce a poor, nonempirical theory.

Generating a theory from a software development method (SDM) is also hazardous. An SDM is a system of prescriptions for doing something effectively. *Methods are inappropriate foundations for taxonomies and process theories* [72], [73]. Methods prescribe what *should* exist and *should* be done. Taxonomies and process theories describe what *does* exist and what *actually* happens. Adapting a method into a theory therefore tends to over-rationalize reality [72], [73].

Theorizing from methods can also blind the researcher to the facts of the case. For example, suppose we develop a taxonomy of artifacts from the Rational Unified Process [74]. It suggests classes such as business case, architecture document, use case and design model. Further suppose that an actual project has informal, natural language feature descriptions on index cards. Our method-based taxonomy encourages us to misinterpret and misclassify these instances as use cases, when they are really user stories.

A related pitfall is the tendency to blend explanation and prescription; that is, combine theory and method. Simultaneously modeling how something should happen and how it actually happens is difficult and confusing [73].

Researchers who generate theory from a literature review should consult established guidelines for a plain literature review [75], meta-synthesis [76], systematic literature review [64] or thematic synthesis [66] and skip to step five.

5.1.2 Grounded Theory and Interpretive Case Studies

Grounded theory and interpretivist case studies are well-suited to generating process theories and taxonomies.

"A case study is an empirical inquiry that investigates a contemporary phenomenon within its real-life context" and "relies on multiple sources of evidence, with data needing to converge in a triangulating fashion" [77, pp. 13–14]. Studies taking place entirely in a lab, using only a single source of evidence, or missing triangulation are not case studies. An *interpretivist* case study is a case study performed with the view that studying social phenomena requires different empirical methods and assumptions than studying physical phenomena (cf. [78]). Practically speaking, interpretivist case studies generate theories, taxonomies or other concepts from data, rather than testing a priori theories or artifacts.

Grounded theory encompasses a specific family of methodologies for generating theory from observations [79]. "Grounded Theory" is not a synonym for qualitative research and differs from interpretivist case studies both philosophically and practically. Grounded theory benefits from a rigorous and systematic analytical approach. Like interpretive case studies, it is appropriate for generating taxonomies and process theories because it encourages the researcher to ground findings in empirical data and inhibits confirmation bias [80]. Since classification is a core activity in grounded theory, grounded theory studies often produce taxonomies and can produce process theories. However, while a researcher can conduct a case study explicitly to generate a taxonomy or process theory, grounded theory interdicts any a priori notion of the form of the findings.

The primary advantage of grounded theory and interpretive case studies is that they encourage deep immersion in the data, which may protect researchers from missing instances or oversimplifying and over-rationalizing processes. However, these studies may require weeks or months of observations, interviewing, document collection and analysis.

Some people mistakenly differentiate between shorter "case studies" and longer "ethnographies." Ethnography is the study of culture [81], not a synonym for longitudinal field study. Long, situated empirical inquiries that do not specifically study culture are case studies.

5.1.3 Single-Source Primary Studies

Researchers *can* generate taxonomies and process theories from simpler, one-source primary studies, but I do not recommend it because mono-method bias is especially problematic here. Single-source studies include using one of: questionnaires, focus groups, interviews, documents, software repository mining or social network analysis.

It is tempting to assume that experienced software developers are experts, and that their classifications and descriptions of processes are therefore trustworthy. Real expertise, however, goes beyond first-hand experience and "incorporates sound relevant theoretical knowledge supported by secondary experience and second order observation" [82, p. 222]. Without observation and theory, we are not justified in assuming a participant has accurate perceptions of software processes, correct causal inferences (e.g., critical success factors for software projects) or good classifications.

Indeed, participants' accounts of events may be systematically biased [83], [84], [85], [86] or simply wrong. Verbal reports of cognitive processes are only reliable "when influential stimuli are salient and are plausible causes of the responses they produce," [87, p. 231]. Software professionals' post-hoc reconstructions of processes, in particular, are demonstrably unreliable. For example, developers claim to follow methods while practically ignoring them [88]. In my experience, interviewees regularly make indefensible statements, present speculation as fact and backfill reasoning they had not thought of during the events in question (cf. [27]). Experienced professionals' accounts of their processes often diverge from contemporaneous observations [32].

Simply asking participants to describe a process or classification is effectively asking the research question directly, which unconsciously forces the researcher's preconceptions onto the participant. This biases responses and prevents the participant and researcher from reconstructing what is actually happening [12], [89].

While outright lying seems rare, interviewees misunderstand, oversimplify and over-rationalize; that is, they present objects and events as we think they should be rather than as they are. Examples of over-rationalizing include presenting something messy as clean, something chaotic as well-behaved, or something unpredictable as predictable. When economists model humans as rational agents, they over-rationalize reality.

Documents suffer the same problems. Requirements documents, for example, are routinely filled with nonsense such as 'The system shall scale infinitely with no degradation in performance' (a real example from a large, reputable software company). Meanwhile, actual processes often depart dramatically from official processes [90], [91]. That is one reason why the methodological literature on case studies emphasizes data triangulation [92], [93].

Considering these problems with interviews, focus groups, document analysis and open-ended survey questions, it may be tempting to generate theory without consulting participants at all. In my opinion, however, it is very easy to generate theories of professional practice that do not make any sense to practicing professionals, and many theories in closely related disciplines (e.g., information systems) fall into this category. This is why qualitative researchers typically investigate whether findings resonate with participants, usually via interviews or focus groups.

In summary, all individual data sources are problematic: documents over-rationalize reality, interviewees give inaccurate accounts of events, researchers project inappropriate conceptualizations on direct observations, etc. The only practical way to generate good theory from flawed data is to triangulate across multiple data sources. I therefore do not recommend generating theory from a single data source. Moreover, when phenomena of interest are ambiguous or frequently misunderstood, direct observation by expert researchers may provide the most straightforward route to the truth.

5.1.4 Personal Experience

Similarly, researchers *can* generate process theories or taxonomies by reflecting on their personal experiences, without referring to any empirical data or existing theory. This is often dressed up as 'synthesizing years of experience through careful reflection.'

No matter how it is presented, though, *theorizing by reflecting on your own experience is hazardous because our experiences are misleading*. They are rarely a representative sample. Our beliefs are often not calibrated to available evidence [94] and our memories often diverge from actual events [95]. Our expertise does not inoculate us against cognitive and perceptual biases [96], [97]. For example, experts are susceptible to system justification—the tendency to irrationally defend the status quo [98], [99]. Researchers are also prone to over-rationalizing phenomena (cf. [100], [101], [102] for examples of over-rationalized process theory based on experience). The entire point of Grounded Theory is, arguably, to overcome these problems by holding the researcher close to the data (cf. [89]).

Of course, researchers often understand and interpret empirical observations through personal experience. The point of this discussion is not to dismiss personal experience altogether, but to emphasize the importance of primary and secondary research.

5.2 Step 2: Site Selection

Much has been written about choosing the site for a case study [103]. In most case study research, the "site" is an organization, culture, tribe or group of some sort. In SE research however, the site can also be an artifact (e.g., Apache), group of artifacts (e.g., The Qualitas Corpus [104]) or repository (e.g., GitHub).

However, the key considerations for site selection remain the same: the site should be relevant to the research questions, produce rich data and believable explanations, and be practically accessible [105]. The same criteria apply to grounded theory sites.

Sometimes the research question is tied to a particular site, so the site is effectively given. Other times the researchers select the site. Sometimes researchers have compelling reasons to study a particular site (e.g., the organization is doing something unique; the product has dominant market share). Other times we just choose the most convenient site. Because the goal here is to generate a theory, not to statistically generalize results, the most important criteria are that the site is accessible and produces rich data. Authors should not need a litany of justifications.

5.3 Step 3: Data Collection

Established guidance for data collection in case studies [78], [77], [106] and grounded theory [12], [79] generally apply here. Multiple sources of evidence is a defining trait of a case study and a key quality criterion for grounded theory. A typical investigation might include direct observation, interviewing participants and collecting documents and archival data. For process theories, longitudinal data is needed [51]. In SE, researchers might also collect technical artifacts including source code, test suites, test cases, user stories, benchmarks or performance data, data from project management software and interpersonal communications.

Often, one type of data is the dominant source of information, while the others play supporting roles. In my opinion, the *dominant* data collection strategy for generating a process theory or taxonomy should be one of the following.

1. *Direct observation* involves watching participants (e.g., watching pair programming sessions, observing a retrospective meeting). Researchers typically take detailed field notes during the observation and reflect on them afterward.
2. *Participant observation* involves taking part in events (e.g., pair programming with a participant, discussing issues during retrospective meetings). Researchers typically take very brief notes in situ with more post hoc elaboration.
3. *Technical observation* involves accessing, creating or copying digital artifacts such as source code, unit tests, server logs or database entries.

While time-consuming, the above three strategies can establish the basic facts of what exists in and what is happening at the site. They can reveal facts, patterns and insights that may not be available elsewhere, not to mention corroborate (or refute) statements from interviews, documents and other data sources. While participant observation introduces a specific threat to validity (the research alters the context by participating in it), this is often a reasonable tradeoff for an insider's perspective that is otherwise unavailable.

Additional data collection, including interviews, focus groups, questionnaires, and copying official documents, help us interpret and elaborate the facts established by firsthand observations.

5.3.1 Guiding Questions

Beyond standard approaches to data collection, some more specific questions may help guide data collection.

A taxonomy organizes instances based on their properties, to achieve goals. This suggests three questions, which may be useful when developing a taxonomy:

1. *What instances are here?* If we begin with classes, our pre-existing classes tend to shape our perceptions of instances. For example, professionals indoctrinated in traditional (i.e., 1970s) SDMs often mistake user stories for requirements. This is why interpretivist and grounded theory researchers allow classes to emerge from instances in the data.
2. *Which properties of these instances are important?* Instances such as *Brenda the database specialist* and *java.util.Hashtable* have extremely large (possibly uncountable) numbers of properties. We therefore (often implicitly) select a small number of properties with which to organize instances. Consciously questioning which properties to consider and why should produce more defensible classifications.
3. *What is this taxonomy for?* Some taxonomies seek only to facilitate understanding by organizing observations into a meaningful classification. Other taxonomies have a specific purpose; for example, we might classify software quality dimensions to build better code analysis tools. A clear purpose helps to determine which properties to focus on.

Similarly, a process theory describes a pattern of changing entities, in terms of specific concepts and relationships. This suggests four questions, which may be useful for developing a process theory:

1. *What entities are changing?* Process theories explain how an entity changes, so what is the entity whose change process we want to explain? Example entities include software artefacts, tests, documentation, projects, teams, individuals and organizations.
2. *Is there a pattern at all?* Some processes are too chaotic, lawless or context-dependent to support a process theory. Software development may even be such a process [107]. It is crucial to recognize our propensity for apophenia (seeing patterns in randomness).
3. *If there is a pattern, what type is it?* A person or cohesive team pursuing a few goals through numerous activities suggests a teleological approach. Several conflicting forces or a small number of actors engaged in a power struggle suggest a dialectic approach. A large set of competing entities suggests an evolutionary approach. A well-behaved sequence of phases suggests a lifecycle approach. Combinations of these suggest a hybrid approach. Contrastingly, causal relationships between constructs suggests a variance theory approach.
4. *What are the specific concepts and relationships of the theory?* For a teleological theory, what are the agent, its goals and activities? For a dialectic theory, what are the entities, what is the nature of their power structure and how do they conflict? For an evolutionary theory, what is the population, how do entities exhibit variation, selection and retention, and what is the fitness function? For a lifecycle theory, what are the stages or phases and what is the sequence? For a hybrid theory, which types are we hybridizing and how do the elements from the different types relate to each other?

5.4 Step 4: Data Analysis

(While presented here as a linear sequence, data collection and analysis are often cyclical. Grounded theory, in particular, demands iterating where preliminary findings drive further data collection, i.e., theoretical sampling.)

In situ SE research can produce an enormous amount of data including millions of lines of code, thousands of bug reports, hundreds of user stories and dozens of interviews. Making sense of all this data is challenging. In qualitative data analysis, labeling and categorizing observations is

TABLE 2
Strategies for Making Sense of Process Data

Strategy	Application	Description
Narrative	Generating	The researcher writes one or more detailed, descriptive stories from the data. Narratives convey realistic complexity, organized temporally. The narrative can be the product of the research or the subject of further analysis.
Quantification	Generating	"Researchers start with in-depth process data and then systematically list and code qualitative incidents according to predetermined characteristics, gradually reducing the complex mass of information to a set of quantitative time series that can be analyzed using statistical methods" [108, p. 697]. This produces more simple and general, but less rich, accounts of processes.
Alternate Templates	Evaluating	"The analyst proposes several alternative interpretations of the same events based on different but internally coherent sets of a priori theoretical premises. He or she then assesses the extent to which each theoretical template contributes to a satisfactory explanation" [108, p. 698].
Grounded Theory	Generating	Grounded theory is a comprehensive approach to collecting and analyzing data to generate theory inductively. It is characterized by theoretical sampling, line-by-line initial coding, memoing and constant comparison [79].
Visual Mapping	Either	The researcher diagrams specific processes and then abstracts common elements from the diagrams into a theory.
Temporal Bracketing	Either	The researcher divides the data into successive, adjacent periods "without presuming any progressive developmental logic" [108, p. 703]. Periods are therefore not phases in the lifecycle sense. Rather, the periods help the researcher investigate "how actions of one period lead to changes in the context that will affect action in subsequent periods" [108, p. 703]. This strategy enables replication within a single case and facilitates analyzing nonlinear processes and multidirectional causal relationships.
Synthetic	Generating (Variance Theories)	"The researcher takes the process as a whole as a unit of analysis and attempts to construct global measures from the detailed event data to describe it. The researcher then uses these measures to compare different processes and to identify regularities that will form the basis of a predictive theory relating holistic process characteristics to other variables (e.g., outcomes and contexts)" [108, p. 704].

called *coding* (not to be confused with programming). There are dozens of approaches to coding (cf. [62] for an extensive list and guidance) and qualitative researchers often apply several simultaneously, better to explore the data.

To generate a taxonomy, researchers can apply one or more qualitative data analysis strategies. In my opinion, *initial coding*, *in vivo* coding and *descriptive* coding [62] are useful for generating taxonomies, because they focus on entities or objects. Memoing and theoretical coding [12], which are used heavily in Grounded Theory, also support taxonomy generation. Meanwhile, *process coding* [62] is useful for developing teleological process theories because it focuses on activities, while *versus coding* [62] is useful for developing dialectical process theories because it focuses on conflict.

Langley [108] reviews several more advanced strategies for making sense of process data (Table 2). Of these, *temporal bracketing* is useful for developing evolutionary theories because it helps define generations, as is *quantification* because it is necessary to define the fitness function. *Visual Mapping* and *Grounded Theory* can be useful for developing any kind of process theory. Writing rich *Narratives* is often a useful intermediate step in abstracting raw data into a process theory.

The details of applying each of these data analysis procedures would consume many pages and are therefore beyond the scope of this paper. Rather, this section gives pointers to the relevant analytical strategies, which have been exhaustively described in existing literature. Furthermore, some special considerations for using these techniques specifically for process theories and taxonomies should be noted:

1. Although Langley's *quantification* strategy facilitates statistical analysis, the statistics in question are quite different from more common descriptive and inferential statistics (e.g., standard deviation, ANOVA). See Monge [109] for extensive guidance.
2. Most qualitative data analysis naturally produces taxonomies, not process theories. The researcher may have to focus consciously on process elements to arrive at a process theory.
3. Faced with semi-structured data (e.g., database records, source code) that resists the kind of line-by-line qualitative analysis used on interview transcripts, researchers can always write memos about their observations and then analyze the memos.

5.5 Step 5: Conceptual Evaluation

A theory development paper (that is, one with no empirical evaluation) should conceptually evaluate the proposed theory.

As explained above, research quality cannot be reliably ascertained from a pre-ordained set of criteria [40], and attempting to standardize common quality criteria for theories is undesirable [41]. The researcher should therefore choose evaluation criteria that make sense for the emerging theory. The extensive discussion (above) of quality criteria for process theories and taxonomies should help here. For

example, we might evaluate the *transferability* of a theory generated from an interpretivist case study, and the *statistical generalizability* of a theory generated from a positivist meta-analysis.

Practically, however, this is problematic. Reviewers will often complain that the choice of evaluation criteria is arbitrary. Researchers may encounter snarky comments like "unsurprisingly, the theory performs well on the criteria chosen by its creators."

In my opinion, the solution here is twofold. Reviewers are supposed to be experts and experts are supposed to know that there is no objectively correct set of quality criteria for scientific theories. Reviewers *should* identify specific weaknesses in a proposed theory. Reviewers *should* suggest additional quality criteria that make sense for the theory at hand. Reviewers *should not* simply complain that the quality criteria are arbitrary. Meanwhile, faced with the arbitrariness criticism, researchers have two options. They can essentially concede by adopting a well-known set of criteria, such as Guba and Lincoln's [13] credibility, transferability, dependability and confirmability. Alternatively, they can make a principled argument that no universal set of criteria exists and ask the reviewer to be more specific.

5.6 Step 6: Writing-Up

In my opinion, generating theory from an empirical study typically warrants a stand-alone theory development paper. Generating a literature review only warrants a stand-alone theory development paper when the theory is particularly novel, and its generation is unusually complicated.

In either case, writing up a theory development paper is challenging. SE suffers from a lack of theory-development papers so there are few examplars to emulate. Many SE academics are broadly skeptical of theories [5]. It is therefore crucial that:

1. the literature review or empirical study (from which the theory is generated) is well-executed and clearly described, and
2. the chain of evidence or reasoning from the data to the theory is explicit.

In other words, the paper should describe the emergence of the theory. While the theory generation process may involve non-replicable intuitively leaps, the reader should be able to recover and understand the researcher's logic and process. In my experience, the literature review or empirical study, not the theory itself, often carries the paper and convinces the reviewers.

Researchers should respect relevant established guidelines (e.g., Saldana's [62] guidelines for qualitative data analysis; Stol et al.'s [79] guidelines for grounded theory). Here, *respect* does not mean follow blindly; it means understand, follow under normal circumstances and deviate from for good reasons. The same is true for this article. While it recommends literature reviews and data triangulation, in some contexts it might make more sense to generate a process theory from a protocol study (cf. [110], [111], [112], [113]), or generate a taxonomy through discourse analysis (e.g., [111], [114]). Guidelines are supposed to help, not shackle.

Beyond following established guidance, I have several recommendations specific to process theory and taxonomy generation:

- Explain why the proposed theory is needed.
- Provide a compelling justification for strategies other than literature review or case study(/grounded theory).
- To generate a process theory from a case study, the site should produce rich longitudinal data.
- If an empirical strategy is used, the dominant approach to data collection should be first-hand (e.g., direct observation), supported by second-hand approaches (e.g., interviews).
- The primary data analysis strategy should make sense for the type of theory generated; for instance, versus coding for dialectical theories.
- Define the elements of the theory clearly and carefully. Terms including *construct*, *agent*, *instance*, *object*, *method* and *actor* are overloaded and can carry unintended theoretical baggage.
- Include a detailed conceptual evaluation of the proposed theory, which exceeds the usual limitations section of an empirical paper.
- Tie the new theory back into existing research. Identify prior theories or findings consistent with the new theory. Highlight where the new theory challenges prior work or conventional wisdom.

5.7 Step 7: Peer Review

Reviewing a theory development paper can be almost as challenging as writing one, not least because there exists widespread confusion regarding the purpose of scholarly peer review. Put simply, the goal of peer review is, and has always been, "to ensure that the valid article is accepted, the messy article cleaned up, and the invalid article rejected" [115]. The key word here is *valid*. For a theory development paper, *valid* means the manuscript adheres to established guidelines for its chosen strategy (or justifies deviations) and a clear chain of evidence connects the theory to the empirical data or prior research on which it is based. *Valid* does not mean that the proposed theory concurs with the reviewer's personal experience, prior findings, common sense or conventional wisdom. Evaluating validity is not about nitpicking style, tone, positioning, framing, grammar, spelling, image resolution, typographical conventions or adherence to publisher's templates.

Of course, reviewers can make all manner of helpful suggestions, but peer review should focus on validity in the broad sense of methodological correctness. Beyond, this, my recommendations for reviewers of taxonomy and process theory development papers include the following.

- *Do* demand justification for adopting a theory development strategy other than a secondary study, case study or grounded theory study.
- *Do not* expect detailed justification for every aspect of the methodology, e.g., systematic mapping versus meta-synthesis.
- *Do* read the guidelines the manuscript claims to follow.

- *Do* expect detailed description of the site of a case study.
- *Do not* rejected the site choice as arbitrary.
- *Do not* dismiss sites consisting of small companies, novice developers, open-source projects, non-English speaking participants or in developing countries. These are all just as worthy of study as Microsoft and Google.
- *Do not* complain that the site is "not a representative sample." There is no such thing as representative sampling in case studies or grounded theory.
- *Do* expect a detailed account of data collection, including what was collected, who collected it, and when, where and how was it collected.
- *Do* expect a focus on first-hand data collection strategies (e.g., direct observation) and question overuse of second-hand data collection (e.g., interviews, focus groups).
- *Do* identify areas where the manuscript's logic or process is not clear.
- *Do* make concrete suggestions for improving the veracity of the theory.
- *Do* expect explicit definitions for all elements of the proposed theory.
- *Do not* criticize a theory for lacking novelty unless you can cite an existing theory that explains the same phenomenon better than the proposed theory. Researchers may avoid emphasizing novelty because peer review is biased against new ideas [116].
- *Do* suggest relevant prior work the author has missed.
- *Do not* expect a manuscript to review *all* relevant prior work, because that could include thousands of studies.
- *Do* suggest relevant quality criteria not included in the manuscript.
- *Do not* reject the chosen quality criteria as arbitrary.

6 Recommended Steps for Evaluating Taxonomies and Process Theories

Once a process theory or taxonomy has been generated, further evaluation depends on many factors.

For a constructivist or grounded theorist, it may be sufficient to generate a theory from a case to understand that particular context. Postmodernism rejects general theories, so researchers with postmodern views might never attempt to evaluate a theory in a different context.

Positivists, in contrast, expect empirical evaluation beyond theory development. They may evaluate their own theories or someone else's, in similar or very different contexts, immediately or years after the theory's development.

From a (contemporary Bayesian) positivist perspective, then, this section explores how we can determine whether a taxonomy or process theory is empirically veracious. It extends and clarifies previous guidance for empirically evaluating process theories, which emphasizes observation, questionnaires, document analysis and simulation [117], [118], [119], [34]. These instructions are first for researchers seeking to evaluate taxonomies or process theories, and second for reviewers of such evaluations.

6.1 Step 1: Choose a Theory

The theory to evaluate is often given—it is the theory we have just devised or the one on which our research rests. Other times, we begin with an idea, question, problem, context, site or suggestion and look for a relevant theory. In either case, theories need evaluating; the main criterion is that the chosen theory and context are compatible.

6.2 Step 2: Choose One or More Rival Theories

In my opinion, researchers should evaluate a taxonomy or process theory against a reasonable alternative taxonomy or process—a rival. The rival may be an established theory in SE, an adaptation from a reference discipline, or simply a plausible alternative explanation generated by the researcher. The rival should seek to explain similar phenomena in similar contexts.

Evaluating rival theories is a crucial alternative to null hypothesis testing. Null hypothesis testing is not suited to evaluating taxonomies and process theories because it answers the wrong question. For example, SCI posits that designers engage in sensemaking [22]. The null hypothesis, *designers never make sense of anything*, is a straw man. The relevant question is whether sensemaking has more explanatory power than alternative concepts like analyzing and learning. Comparative questions like these demand a different approach: strong inference.

Strong inference is a variation of the hypothetico-deductive scientific method in which researchers compare two or more plausible alternative hypotheses, propositions or theories [48] instead of testing null hypotheses. Strong inference leads to faster scientific progress by mitigating confirmation bias [48]. Langley's [108] *alternate templates* strategy for making sense of process data (Table 2) implements strong inference. Comparing rival theories is now commonly recommended for case study research [92] and some argue that theory testing is essentially comparative [50]. The idea is analogous to engineering research, where studies often aim to show that a new system outperforms a legacy system on specific dimensions.

Sometimes, researchers will identify two or more logical rivals. While this paper assumes a single rival for simplicity, most of the techniques extend to multiple rivals. However, multiple rivals will complicate manuscripts and many reviewers (who do not appreciate strong inference) may react negatively to an 'overcomplicated research design.'

Comparisons are easier between theories of the same archetype; for instance, comparing two taxonomies is simpler than comparing a dialectical theory to an evolutionary theory. However, all rivals are imperfect and imperfect rivals are still better than straw-man null hypotheses.

In summary, comparing rivals is simply more tractable and defensible than null hypothesis testing for evaluating process theories and taxonomies. Rivals should be used with all of the evaluation approaches discussed below.

6.3 Step 3: Analyze Truth Claims

Different kinds of theories make different kinds of propositions or claims. Taxonomies posit an ostensibly good way to classify phenomena. The question, then, is *what does 'good' mean?* Recall that we create taxonomies to improve cognitive efficiency, facilitate inferences, and sometimes achieve a specific purpose (e.g., a taxonomy of game elements to

TABLE 3
Truth Claims

#	Truth Claim	Taxonomy	Dialectical	Teleological	Lifecycle	Evolutionary
1	The proposed taxonomy improves cognitive efficiency.	☑	☑	☑	☑	
2	The proposed taxonomy facilitates inferences.	☑	☑	☑	☑	
3	The proposed taxonomy/theory is useful for a specific purpose (optional).	☑	☑	☑	☑	
4	There exist one or more changing entities.		☑	☑	☑	☑
5	There exist one or more conflicts between proposed entities (classes).		☑			
6	There exists an intelligent, autonomous agent.			☑		
7	The changing entity passes through a series of states or events; only a limited subset of possible state transitions is permitted.				☑	
8	There exists a fluctuating population of entities; "As in biological evolution, change proceeds through a continuous cycle of variation, selection and retention" [11, p. 518].					☑
9	Change is driven by conflict between entities.		☑			
10	Change is driven by the goal-directed actions of an intelligent, autonomous agent.			☑		
11	"The trajectory to the final end state is prefigured and requires a specific sequence of historical events" [11, p. 515].				☑	
12	Change is driven by competition for scarce resources.					☑

teach video game design [120]). Taxonomies are therefore associated with three basic truth claims (Table 3).

Taxonomies do not literally posit a set of things that exist in the world. For example, when Hoda et al. [57] posited six roles for self-organizing agile teams, the six roles do not objectively exist in the physical world. People exist. People take actions. Different researchers observing the same people and actions may classify them differently. It would be absurd to claim that a particular taxonomy is the *only* way to classify the phenomena. *Good* means that the posited classes help with cognitive efficiency, facilitating inferences and achieving relevant specific purposes.

Since dialectical, teleological and lifecycle theories are taxonomies, they also make the three basic taxonomy truth claims, with diverse additions (Table 3). Dialectical theories claim that some entities (classes) conflict and this conflict drives change. Teleological theories posit an agent—an autonomous, intelligent being capable of observing its environment and taking goal-directed actions—and present a taxonomy of the agent's actions. Lifecycle theories present a taxonomy of states, phases, stages or events through which the changing entity passes, and define the allowable transitions. Evolutionary theories are not taxonomies. They posit an evolving population of *instances* competing over resources.

Researchers need to identify each theory's truth claims to create alternative templates and select an appropriate data collection strategy. This should not be complicated—the researcher simply lists each theory's claims. However, since some theories are not presented as clearly as others, specifying their truth claims can be challenging and controversial.

6.4 Step 4: Create Templates

Given a list of claims for each theory, researchers can organize these claims in at least two different ways: using seperate templates (Table 4) or using a conflict template (Table 5).

6.4.1 Separate Templates

One approach is to create separate coding schemes for each theory (Table 4). Each coding scheme lists all of its theory's claims. For a taxonomy, this would include claims that each of its classes is good (i.e., promotes cognitive efficiency and supports inferences). It might also include a claim that the taxonomy is useful for a specific purpose. Process theories would have additional claims as shown in Table 3. Each claim should have space for both supporting and contradictory evidence. Typically, the template will have at least one claim for every concept (e.g., activities, actors, objects, phases) and relationship (e.g., sequence, dependency, conflict) in the theory.

For example, there exist two different taxonomies of waste in software development (cf. [36], [121]). To test these, we can simply make a template for each theory, listing each waste type. Later we can assign supporting and contradictory evidence to each waste type, and see which theory has more supporting and less contradictory evidence (more on this below).

6.4.2 The Conflict Template

A different approach involves focusing on conflicting claims. The researcher identifies all of the conflicts between the two theories; that is, differences that lead to contrasting predictions. The researcher then creates a coding scheme around the conflicting propositions (Table 5). This approach should produce cleaner, simpler analysis. However, it only works when the researcher can draw numerous conflicting predictions from the theories.

For example, SCI posits that designers frame and reframe system goals throughout the design process [22]. The Function-Behavior-Structure Framework contrastingly posits that engineering design problems are simply given [102]. Because these claims unambiguously conflict, we can

TABLE 4
Coding Scheme (Separate Templates)

Theory A			Theory B		
Theory Component	Evidence For	Evidence Against	Theory Component	Evidence For	Evidence Against
Concept A.1			Concept B.1		
Concept A.2			Concept B.2		
...			...		
Concept A.n			Concept B.n		
Relationship A.1			Relationship B.1		
Relationship A.2			Relationship B.2		
...			...		
Relationship A.n			Relationship B.n		

combine them into a bipolar spectrum from 'problem given' to 'ongoing problem framing'. Later, evidence can be organized according to which end of the spectrum it supports, and researchers can judge which theory is supported by the preponderance of evidence.

6.4.3 Sanity Check

In my opinion, researchers should sanity-check their templates in several ways. For example, have a colleague review them or try to explain them to a non-expert. Most importantly, brainstorm examples for each evidence category. If you cannot imagine a plausible observation that would support or contradict each claim, the claim may be a straw man or tautology. These kinds of pre-pilot evaluations can be described in the instrument development section of a manuscript.

6.5 Step 5: Select a Research Method

Different research methods are appropriate for evaluating different kinds of propositions. For example, pair programming ostensibly increases software quality [122]. This suggests a simple between-subjects randomized control experiment where the treatment group pair-programs, the control group programs individually, and the researchers measure software quality.

The nature of the proposition to evaluate, not the type of theory, determines which research methods are appropriate. A single study may not be capable of evaluating all of the theory's claims, so the researcher may have to choose. In my opinion, however, the choice is simpler than it appears:

- A positivist case study [93] is appropriate for all claims except claim 3 (usefulness), and for either separate templates or the conflict template.
- A *questionnaire study* is appropriate when using the conflict template for claims 1-7 and 9-11. In other words, questionnaires are not appropriate for testing evolutionary theories.

TABLE 5
Coding Scheme (Conflict Template)

Proposition	Evidence for Theory A	Evidence for Theory B
1		
2		
...		
n		

- An experiment [123] is appropriate for evaluating claim 3 in a controlled, laboratory setting.
- Action research [124] is appropriate when evaluating claim 3 in a contemporary, real-life setting.

Combining one or more of these methods will produce richer and more compelling but more complicated evaluations. Please note that while this paper focuses on the most common and recommended methods used to evaluate process theories and taxonomies, they are not the only methods. Some less common approaches include lab-based simulation [125], think-aloud protocol studies (cf. [126], [127]), quasi-experiments [128] and archival data analysis.

6.6 Step 6: Collect and Analyze Data

This section will explore data collection and data analysis for each recommended research method. Again, researchers should consult established guidelines for each method (e.g., [92], [129], [130], [131]). Rather than repeating existing guidelines, this section will focus on considerations specific to taxonomies and process theories, as well as common misconceptions.

6.6.1 Positivist Case Study

While *interpretivist* case studies [78] were recommended for theory generation, *positivist* case studies are recommended for theory evaluation. Again, "a case study is an empirical inquiry that investigates a contemporary phenomenon within its real-life context" and "relies on multiple sources of evidence, with data needing to converge in a triangulating fashion" [77, pp. 13–14]. A positivist case study is one that begins with clear research questions, theories and coding schemes (e.g., the templates discussed above).

Data collected in software engineering case studies include direct observation (field notes), interview transcripts, copies of relevant documents, diagrams and messages, segments of source code, code commit logs, screen shots, photographs of the physical environment and video of meetings or activities (cf. [93]). Researchers should collect both quantitative data (e.g., bug counts, number of unit tests, number of code commits, time spent in meetings, word frequencies) and qualitative data (e.g., field notes, interview transcripts, software documentation) [93].

Indeed, not all case studies involve observing or speaking with human participants. For target phenomena such as

embedded systems, cybersecurity incidents or open source software projects, the sources of evidence might be technical logs, database entries, real-time performance data, online forum posts, etc. What matters is that researchers examine the phenomenon in its real-life context and triangulate across multiple data sources. When human participants are involved, researchers should avoid fixating on interviews and corroborate interviewee statements with first-hand observations for the reasons discussed in the previous section.

The researcher then organizes evidence into the template (s) using standard qualitative data analysis techniques (cf. [62], [93]). Evaluating claims using quantitative data really depends on the exact nature of the claims and the data. Various techniques can be used, for example, to estimate or evaluate the fitness function of an evolutionary process theory (cf. [132]). In general, researchers should consider statistical techniques for analyzing patterns, such as clustering, latent semantic analysis and multi-dimensional scaling (cf. [109]).

In my opinion, a longitudinal case study including direct observation of real-world events is the best single methodology for evaluating a process theory or taxonomy for two reasons. First, it can provide evidence concerning all but one of the relevant truth claims. Second, its observational richness, depth and realism inoculates researchers against oversimplified, over-rationalized accounts of messy realities. However, such studies are time-consuming and do not support statistical generalization, which brings us to questionnaire studies.

6.6.2 Questionnaire Studies

In my opinion, a questionnaire study is best used with a conflict template. The researcher first generates questionnaire items reflecting each conflicting proposition. Specifically, one can generate bipolar scales (including Likert and semantic differential scales) where one pole is consistent with Theory A and the other with Theory B; for example:

The goals of my current project were. . .

a) *. . . completely provided by the client (Theory A)*
b) *. . . mostly provided by the client*
c) *. . . determined equally by the client and development team*
d) *. . . mostly determined by the development team*
e) *. . . completely determined by the development team (Theory B)*

On a scale of 1, strongly disagree (Theory A), to 9, strongly agree (Theory B), to what extent do you agree with the following?

a) *our project goals change often*
b) *the project goals are clearer now than when the project began*
c) *developing the product changed its goals*

(Obviously, real questions do not have the "Theory A" and "Theory B" labels.)

Researcher typically create a questionnaire by combining these items with demographic questions. Questionnaires may also include open-ended questions about perceptions of the process or domain. Depending on the theory, the questionnaire could also include a scale for perceived usefulness (cf. [39]); however, perceived usefulness may not reflect actual usefulness.

Validating these kinds of items fundamentally differs from validating a variance theory testing questionnaire. Conflicting propositions, classes, agents, activities, dialectic tensions, evolving populations and phases are not constructs. The multitrait-multimethod matrix (MTMM) and similar approaches to construct validity therefore do not apply [133]. Instead, these questionnaires should be validated using pilot studies with both experts and members of the target population. Researchers should request item-by-item feedback from pilot participants—some familiar with the purpose of the study, others not (see [32] for a detailed example).

The questionnaire will produce a response distribution for each question; for example, a) 5; b) 10; c) 20; d) 30; e) 15. Visual inspection suggests that this distribution favors whichever theory or taxonomy is on the right pole (the 'e' side). While visual inspection is often sufficient, non-parametric tests including Chi-Square and Kolmogorov-Smirnov may be used to test significance. This requires an *expected distribution,* three options for which are:

1. uniform distribution, e.g., 20, 20, 20, 20, 20
2. pseudo-normal distribution, e.g., 6, 19, 30, 19, 6
3. reflected distribution, e.g., 15, 30, 20, 10, 5

In practice, the researcher can simply test all three distributions. However, the reflected distribution is the most defensible as it addresses the question, *is the observed distribution significantly different from an equally compelling distribution supporting the rival theory?* and reflecting a distribution does not obviously violate statistical norms (cf. [32] for a detailed example of this approach). The other distributions make less sense because we have no theoretical reason to expect a uniform or normal distribution, and treating discontinuous, ordinal data as normally distributed is statistically problematic.

This questionnaire approach is simpler, faster and supports statistical generalization if researchers can obtain a representative sample of respondents. However, where case studies often allow first-hand observation of the phenomena of interest, questionnaires are limited to respondents' perceptions. This creates problems analogous to over-reliance on interviews in theory generation. The researcher should take great care to craft questions that alleviate threats to validity including acquiescence bias, memory effects and social desirability bias. Meticulous word choice is critical because of widespread confusion about the meaning of common SE terms (see [17]).

6.6.3 Controlled Experiment

Randomized controlled experiments are used to investigate causal relationships between independent and dependent variables. Since most of the truth claims in Table 3 are not causal relationships between variables, experiments cannot be used to investigate them. However, experiments are ideal for investigating the usefulness of a theory for a specific purpose. This is much simpler than a case study or questionnaire. The researchers simply:

1. recruit appropriate participants
2. randomize participants into two groups
3. give Theory A (or materials based on it) to Group A
4. give Theory B (or materials based on it) to Group B
5. assign one or more tasks, based on the theories' aims
6. measure performance
7. compare the performance of the two groups using inferential statistics

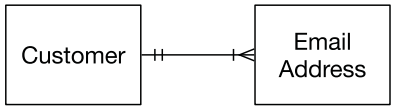

Fig. 10. Simple entity-relationship diagram.

For example, suppose we have three taxonomies that should be useful for teaching video game design: the mechanics-dynamics-aesthetics framework [134], Schell's elemental tetrad [135] and the unified theory of digital games [120]. We can divide students into three groups (one for each theory), give each group learning materials based on their assigned theory, then see if one group gets superior scores on a test.

While it may seem plausible to investigate claims 1 (cognitive efficiency) and 2 (facilitating inferences) using experiments, both are problematic. There are deep, unresolved problems with quantifying cognitive efficiency [136]. Task performance, for instance, is not necessarily a good operationalization. Empirically evaluating a taxonomy's ability to facilitate inference is similarly problematic because research participants may misunderstand it, draw incorrect inferences, or fail to draw available inferences. For example, Fig. 10 supports two inferences (each customer has one or more email addresses; each email address is associated with one and only one customer) regardless of how many undergraduate students can correctly draw these inferences on their database exam.

Claims 9-12 also appear similar to causal hypotheses for which experiments are well-suited. However, when process theory research claims that change is "driven by" a "causal motor" neither the change nor its cause is necessarily a quantifiable variable, and "driven by" does not imply covariance or probabilistic causation. For example, claiming that change emerges from the interaction between software development and software process improvement, means the nature of the changes is determined by the interaction [31], not that more interaction leads to more change.

6.6.4 Action Research

Action Research is a type of field study in which "the researcher enters a real-world situation and aims both to improve it and to acquire knowledge" [124]. Action research is similar to case study research except that action researchers are trying to improve the situation while (positivist) case study researchers usually try to minimize their impact on the site.

Briefly, action researchers enter the site and implement the theory in some way; for instance:

- providing instruction based on the theory to participants
- analysing the situation using the theory
- making suggestions based on the theory
- taking corrective actions based on the theory
- developing or deploying an artefact (e.g., tool, technique) based on the theory

The researcher then attempts to measure the impact of this activity. Objective quantitative success measures are rare in action research. Instead, the researcher typically identifies performance indicators including: 1) quotations from participants about the intervention; 2) events consistent with success (or failure); 3) quantitative performance indicators (e.g., sales figures). Alternatively, the study can include a repeated-measures quasi-experimental design [123] or pre-/post-intervention questionnaire to quantify success.

Action research works well in initial proof-of-concept studies because the theory is evaluated under ideal conditions: an expert researcher applies it to an appropriate site. Another benefit is the opportunity to collect data on claims besides usefulness, using case study techniques.

However, action research is criticized for researcher bias and lacking rigor because the researcher interferes with the site [137]. Applying rival theories in the same site is often impractical—the research cannot suggest and the site cannot adopt opposite courses of action based on conflicting propositions.

One idea for mitigating these problems is to begin as an independent observer (i.e., a case study). After collecting and analyzing data on the other relevant claims, researchers select the better-supported theory. They then transition into action research to evaluate the usefulness of the better-supported theory.

6.6.5 Cross-Cutting Concerns

Multimethodological approaches are often recommended but rarely executed. On one hand, combining a case study with a questionnaire, experiment or action research may be the only way to investigate all of the truth claims, and the benefits of one methodology may overcome the limitations of another. On the other hand, multi-methodological research is much more difficult to present. The longer the article and more complicated the methodology, the more reviewers can find fault. In SE at least, one study per article is the de facto standard; more than one is commendable.

Similarly, many methodologists recommend multiple-case studies [92], and many reviewers expect them because several cases are ostensibly "more representative." This misapplies "the statistical, sampling-based conception of generalizability to nonstatistical, nonsampling research" [45, p. 223]. No set of 3, 5 or even 10 case studies can possibly represent the breath, complexity and distribution of real-world software engineering when there are dozens of programming languages and industries, hundreds of cultures and languages, thousands of SDMs, techniques, practices, tools, and libraries, and millions of unique development contexts. Again, one case per article is the de facto standard; more than one is commendable. Researchers can simplify multiple case articles by omitting individual case analyses and providing only the cross case analysis [93].

Meanwhile, many research methodologists also recommend team approaches to research [92]. On one hand, having two researchers simultaneously on site, observing the same phenomena, one organizing evidence concerning Theory A while the other organizes evidence concerning Theory B, may ameliorate confirmation bias. The process of reconciling conflicting interpretations through discussion may furthermore spur more nuanced analysis and reflection. On the other hand, team-based approaches are uncommon and impractical in many circumstances (e.g., traditional PhD dissertation projects). Therefore, like multi-methodology and multiple cases, team-based approaches are unnecessary but commendable.

6.7 Step 7: Writing Up

Writing up is simpler for theory-testing studies than for theory-generation studies. Researchers can largely follow established guidelines for presenting the kind of study that was done. A few recommendations specific to writing up studies that evaluate process theories or taxonomies are as follows.

- Clearly explain both the importance of having a rival theory and why the chosen rival is appropriate.
- Give a complete account of the specific truth claims of both theories, then explain which claims will be evaluated.
- Provide the data analysis template(s) as an appendix.
- If conducting a case study or action research, use tables based on the data analysis template(s) to provide numerous examples of observations supporting (or contradicting) truth claims. In other words, use the template(s) to convey the chain of evidence from observations to theory.
- If conducting a questionnaire study, provide the questionnaire as an appendix and detail its validation.

6.8 Step 8: Reviewing

Reviewing, again, should focus on methodological correctness including respecting established guidelines for the specific research method. Some further suggestions for reviewing evaluations of process theories and taxonomies are as follows.

- *Do* expect comprehensive descriptions of the theories.
- *Do not* be overly pernickety about the choice of theory.
- *Do* expect a rival theory.
- *Do not* complain that rivals overcomplicate the research design.
- *Do* expect a discussion of why the rival theory is appropriate.
- *Do not* criticize the choice of rival unless you can cite a better one.
- *Do* expect a clear discussion of the theories' claims.
- *Do not* expect the study to cover all claims.
- *Do* expect clear, strong evidence concerning the studied claims.
- *Do not* expect a multimethodological study, a multiple-case study, or a team-based approach to research, but if one is used, *do* commend it and *do not* complain that it overcomplicates the study.

7 Discussion

To review, a taxonomy is a set of classes. A process theory is an explanation of how something changes. A dialectical process theory explains how change emerges from conflicting entities or forces; entities, forces and conflicts are classes. A teleological process theory explains how change emerges from the activities of an intelligent agent; the activities are classes. A lifecycle process theory explains how change follows an immutable series of phases; the phases are classes. Evolutionary process theories explain how change emerges from resource competition in populations; they are not taxonomies.

Taxonomies and process theories can be generated from empirical data or literature reviews. If empirical data is used, data triangulation is critical and I recommend including first hand observations wherever possible. Generating a taxonomy or process theory from expert opinion, personal experience or software development methods (or prescriptive literature more generally) is not recommended because these sources are often unfounded or biased, and encourage oversimplification and overrationalization.

Since taxonomies and process theories do not typically posit causal relationships between variables, recommended evaluation methods include positivist case studies and questionnaire studies for most claims. Rigor can be improved by comparing a proposed theory or taxonomy to an existing rival and employing multi-methodological or team-based approaches. Experiments and action research may be appropriate for evaluating usefulness.

With these highlights in mind, this section discusses the limitations of the proposed empirical guidelines. It then elaborates on some applications of process theories and taxonomies in SE, and suggests avenues for future research.

7.1 Limitations

The guidelines presented in this paper are limited in several ways. They are based on my understanding of taxonomies, process theories, research methods and philosophy of science. My understanding is incomplete, and likely flawed or biased in ways I cannot perceive. Since we have no objective basis for methodological guidelines, and no study can tell us which way is best, some subjectivity is unavoidable. We do not have strong empirical evidence that the proposed guidelines are effective, and other guidelines could be better.

The proposed guidance for generating and evaluating taxonomies and process theories is, moreover, rooted in a contemporary pragmatic philosophy. That is, it regards contemporary positivism (i.e., Bayesian epistemology and the strong inference model) and interpretivism as views of the world to be adopted situationally, rather than enduring dogmas. My understanding of truth claims and knowledge is heavily influenced by Quine [138] and critical realism [139]. A reasonable scholar with a different philosophical perspective might arrive at different guidelines. For instance, an uncompromising antipositivist might reject the entire notion of evaluating a theory.

More concretely, the paper does not discuss using quantitative simulations to develop or test process theories (cf. [119, pp. 402–403]) or estimating or evaluating the fitness function of an evolutionary process theory (cf. [132]). Mathematical simulation and evolutionary computation are rich research areas, with numerous techniques and considerations. A reasonable summary would consume many pages and depart sharply from not only the research methods and techniques described here but also the author's expertise.

In summary, the guidelines presented by this paper comprise suggestions that would have saved me a great deal of time and frustration. They are intended to provide a starting point for others to elaborate and critique.

7.2 SE Applications

Despite the above limitations, taxonomies and process theories are crucial for scientific advancement. As explained in Section 2, process theories and taxonomies provide concepts and technical language needed for precise communication and education. Taxonomies also provide cognitive efficiency by facilitating reasoning about classes rather than instances. Meanwhile, the lack of SE process theory research obstructs scientific consensus by focusing the academic community on SDMs, which inevitably oversimplify and over-rationalize complex and unpredictable human phenomena.

SE already benefits from taxonomies of software quality dimensions, critical success factors, coding decisions, manager and developer roles, and many other important phenomena. However, many existing taxonomies are not based on rigorous research. SE also lacks a taxonomy of development activities or practices (analagous to Sim and Duffy's taxonomy for engineering design [140]).

Meanwhile, SE research is fundamentally concerned with the process of developing software. In my opinion, SE needs more evolutionary, dialectic and teleological process theories to overcome the tendency to oversimplify and over-rationalize human processes into SDMs and lifecycles. SE-related processes that may benefit from a process theory approach include the following.

- The process of forming SE teams.
- The process through which agility arises.
- The process of evaluating the coverage of a test suite.
- The process of integrating a new developer into an existing team.
- The process of communicating a software design to a new developer.
- The process of selecting an architectural pattern (e.g., tiered, service-oriented, model-view controller).
- The process by which open source projects gain momentum, contributors and market share.
- The process of finding a bug.
- The process of finding and eliminating security vulnerabilities.
- The process of problem/solution coevolution.
- The software development process in general.

For example, we know that designers simultaneously reframe problems and design artifacts to address them (i.e., coevolution [141]). However, we do not know exactly how the coevolution process unfolds in SE. Because coevolution involves the interaction between entities (e.g., the developers, solution conjectures, problem frames), a dialectical process theory may be appropriate.

Similarly, while many believe that agility is important, we lack a clear understanding what agility is and how agility, the team property, differs from ostensibly Agile methods ([142]). Since agility involves the interaction between the need for change and responses for change, a dialectical process theory of how developers experience change might help explain the role of agility in SE projects [143].

Open source projects, meanwhile depend on networks of contributors to implement features, improve quality and gain market share [144]. A process theory may help explain how some projects succeed while others stagnate. Because so many open source projects compete for attention among developers and users, an evolutionary approach may be appropriate.

7.3 Future Work

Methodological guidelines should evolve with our growing understanding of research methods and theory. Evaluating the proposed methodological guidelines requires other experienced researchers to apply them in diverse contexts. Future work may therefore include reflections, refinements or refutations from other scholars who adopt, adapt or abandon the above guidance.

Further work is also clearly needed on at least four methodological fronts:

1. Sampling problems, that is, the lack of population lists from which to randomly sample and the tendency for reviewers to capriciously reject some papers for convenience sampling despite the absence of defensible representative sampling throughout the SE literature.
2. Construct validity problems, that is, widespread confusion regarding the meaning of 'construct' and appropriate techniques for establishing construct validity.
3. Evolutionary process theories, that is, providing more detailed guidelines for generating and evaluating them, especially applying techniques from evolutionary computation to theorize SE phenomena.
4. Quantitative approaches, that is, most process theory research is predominately qualitative; however, there is great potential for quantitative analysis in both generating and evaluating process theories [51].

8 Conclusion

In my experience, understanding process theories—their relationship to taxonomies and variance theories, their epistemic basis, their differing claims, how to generate them, how to evaluate them, how to explain them to others—has been especially challenging because there was no comprehensive, straightforward guidance, and no guidance tailored to SE. Simultaneously defending new theory and new research methods was an enduring obstacle. I often did not know how to proceed and others did not know how to advise me or evaluate my work.

This paper attempts to overcome these challenges so that other scholars can work with process theories and taxonomies with less initial legwork, and reviewers and editors would not have to consult a dozen books from a handful of fields to effectively evaluate a process theory study.

This paper therefore makes three main contributions:

1. It defines and describes taxonomies and process theories, with specific examples from SE.
2. It argues that most process theories are taxonomies with additional truth claims; therefore, taxonomy quality criteria apply to most process theories.
3. It presents specific guidelines for applying numerous research methodologies to generate and evaluate taxonomies and process theories, and clearly explains why other methodologies are not recommended.

While some guidance is available in other fields for generating and evaluating taxonomies and process theories, no other

work: a) approaches the level of specifics given here, particularly regarding the coding schemes and instrument development for evaluation; b) explains the relationship between taxonomies and process theories; or c) specifically adapts methodological guidance to SE contexts and phenomena.

Two main barriers to taxonomy and process theory research in SE appear to endure. First, many scholars do not realize that their "theory," "model" or "framework" is a taxonomy; therefore, they do not think to apply the evaluation techniques or criteria applicable to taxonomies. Second, many SE academics are unfamiliar with process theory and little SE-specific methodological guidance is available; therefore, scholars apply less rigorous research methods and reviewers incorrectly evaluate process theories and taxonomies using criteria and norms of variance theory research. This article attempts to overcome these two barriers.

Acknowledgments

Thanks are due to Jeffrey Parsons, Marshall Scott Poole and the participants of the 2014 General Theory of Software Engineering Workshop for their constructive feedback.

References

[1] T. B. Bollinger, "The interplay of art and science in software," *IEEE Comput.*, vol. 30, no. 10, pp. 125–128, 1997.
[2] E. M. Gray and W. L. Smith, "On the limitations of software process assessment and the recognition of a required re-orientation for global process improvement," *Softw. Quality J.*, vol. 7, no. 1, pp. 21–34, 1998.
[3] M. M. Lehman and J. F. Ramil, "An approach to a theory of software evolution," in *Proc. 4th Int. Workshop Principles Softw. Evol.*, 2001, pp. 70–74.
[4] J. Hannay, D. Sjøberg, and T. Dyba, "A systematic review of theory use in software engineering experiments," *IEEE Trans. Softw. Eng.*, vol. 33, no. 2, pp. 87–107, Feb. 2007.
[5] P. Johnson, M. Ekstedt, and I. Jacobson, "Where's the theory for software engineering?," *IEEE Softw.*, vol. 29, no. 5, pp. 94–96, Sep./Oct. 2012.
[6] P. Ralph, P. Johnson, and H. Jordan, "Report on the first SEMAT workshop on a general theory of software engineering (GTSE 2012)," *ACM SIGSOFT Softw. Eng. Notes*, vol. 38, no. 2, pp. 26–28, 2013.
[7] S. Gregor, "The nature of theory in information systems," *MIS Quarterly*, vol. 30, no. 3, pp. 611–642, 2006.
[8] A. H. Van de Ven, *Engaged Scholarship: A Guide for Organizational and Social Research*. Oxford, U.K.: Oxford Univ. Press, 2007.
[9] K.-J. Stol and B. Fitzgerald, "Uncovering theories in software engineering," presented at the 2013 2nd SEMAT Workshop on a General Theory of Softw. Eng. (GTSE), 2013, pp. 5–14.
[10] J. Kim, "Causation," in *The Cambridge Dictionary of Philosophy*, 2nd ed., R. Audi, Ed. Cambridge, U.K.: Cambridge Univ. Press, 1999, pp. 125–127.
[11] A. H. Van de Ven and M. S. Poole, "Explaining development and change in organizations," *Academy Manage. Rev.*, vol. 20, no. 3, pp. 510–540, 1995.
[12] K. Charmaz, *Constructing Grounded Theory*. London, U.K.: Sage, 2006.
[13] Y. S. Lincoln and E. G. Guba, *Naturalistic Inquiry*. Newbury Park, CA, USA: Sage, 1985.
[14] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering: An Introduction*. Berlin, Germany: Springer, 2012.
[15] D. Sjøberg, T. Dyba, B. C. D. Anda, and J. E. Hannay, "Building theories in software engineering," in *Guide to Advanced Empirical Software Engineering*, no. 12, F. Shull, J. Singer, and D. Sjøberg, Eds. London, U.K.: Springer, 2008, pp. 312–336.
[16] W. Nash, *Jargon: Its Uses and Abuses*. Hoboken, NJ, USA:Wiley, 1993.
[17] P. Clarke, et al., "An investigation of software development process terminology," in *Software Process Improvement and Capability Determination*, vol. 609, no. 10, P. Clarke, R. O'Connor, T. Rout, and A. Dorling, Eds. Cham, Switzerland: Springer, 2016, pp. 351–361.
[18] F. P. Brooks, *The Design of Design: Essays from a Computer Scientist*. Boston, MA, USA: Addison-Wesley, 2010.
[19] M. W. Lewis, "Exploring paradox: Toward a more comprehensive guide," *The Academy Manag. Rev.*, vol. 25, no. 4, pp. 760–776, Jan. 2000.
[20] D. A. Schön, *The Reflective Practitioner: How Professionals Think in Action*. New York, NY, USA: Basic Books, 1983.
[21] P. Ralph, "The illusion of requirements in software development," *Requirements Eng*, vol. 18, no. 3, pp. 293–296, 2013.
[22] P. Ralph, "The Sensemaking-coevolution-implementation theory of software design," *Sci. Comput. Program.*, vol. 101, pp. 21–41, 2015.
[23] M. Maher, J. Poon, and S. Boulanger, "Formalising design exploration as co-evolution: A combined gene approach," in *Proc. Preprints 2nd IFIP WG5.2 Workshop Advances Formal Design Methods CAD*, 1995, pp. 1–28.
[24] P. Palvia and J. Nosek, "An empirical evaluation of system development methodologies," *Inf. Resource Manage. J.*, vol. 3, no. 3, pp. 23–32, 1990.
[25] P. Ralph, "Introducing an empirical model of design," in *Proc. 6th Mediterranean Conf. Inf. Syst.*, Art. no. 135, 2011.
[26] C. Zannier, M. Chiasson, and F. Maurer, "A model of design decision making based on empirical results of interviews with software designers," *Inf. Softw. Technol.*, vol. 49, no. 6, pp. 637–653, Jun. 2007.
[27] P. Ralph and E. Tempero, "Characteristics of decision-making during coding," in *Proc. Int. Conf. Eval. Assessment Softw. Eng.*, 2016, Art. no. 34.
[28] J. Fenn and M. Raskino, *Mastering the Hype Cycle: How to Choose the Right Innovation at the Right Time*. Boston, MA, USA: Harvard Business, 2008.
[29] M. Steinert and L. Leifer, "Scrutinizing Gartner's hype cycle approach," in *Proc. Technol. Manage. Global Economic Growth*, 2010, pp. 1–13.
[30] T. Hernes and E. Weik, "Organization as process: Drawing a line between endogenous and exogenous views," *Scandinavian J. Manage.*, vol. 23, no. 3, pp. 251–264, Sep. 2007.
[31] I. Allison and Y. Merali, "Software process improvement as emergent change: A structurational analysis," *Inf. Softw. Technol.*, vol. 49, no. 6, pp. 668–681, Jun. 2007.
[32] P. Ralph, "Software engineering process theory: A multi-method comparison of sensemaking-coevolution-implementation theory and function-behavior-structure theory," *Inf. Softw. Technol.*, vol. 70, pp. 232–250, 2016.
[33] W. Royce, "Managing the development of large software systems," in *Proce. IEEE WESCON*, 1970, pp. 1–9.
[34] M. S. Poole and A. H. Van de Ven, "Empirical methods for research on organizational decision-making processes," in *The Blackwell Handbook of Decision Making*, P. C. Nutt, and D. Wilson, Eds. Oxford, U.K.: Blackwell, 2010, pp. 543–580.
[35] K. Schwaber and J. Sutherland, "The scrum guide," 2010. [Online]. Available: http://www.scrum.org/scrumguides/, Accessed on: Jul 02, 2012.
[36] M. Poppendieck and T. Poppendieck, *Lean Software Development: An Agile Toolkit*. Boston, MA, USA: Addison-Wesley, 2003.
[37] K. Forsberg and H. Mooz, "The relationship of system engineering to the project cycle," *Eng. Manag. J.*, vol. 4, no. 3, pp. 36–43, 1992.
[38] B. Curtis, M. I. Kellner, and J. Over, "Process modeling," *Commun. ACM*, vol. 35, no. 9, pp. 75–90, 1992.
[39] V. Venkatesh, M. G. Morris, G. B. Davis, and F. D. Davis, "User acceptance of information technology: Toward a unified view," *MIS Quarterly*, vol. 27, no. 3, pp. 425–478, 2003.
[40] G. Rolfe, "Validity, trustworthiness and rigour: Quality and the idea of qualitative research," *J Advanced Nursing*, vol. 53, no. 3, pp. 304–310, Feb. 2006.
[41] S. Kvale, "The social construction of validity," *Qualitative Inquiry*, vol. 1, no. 1, pp. 19–40, Jun. 2016.
[42] W. Trochim, *Research Methods Knowledge Base*. Cincinnati, OH, USA: Atomic Dog, 2001.
[43] T. S. Kuhn, *The Structure of Scientific Revolutions*, 2nd ed. Chicago, IL, USA: Univ. Chicago Press, 1996.
[44] B. Duignan, "Postmodernism," in *Encyclopædia Britannica*, Chicago, IL, USA, 2017.

[45] A. S. Lee and R. L. Baskerville, "Generalizing generalizability in information systems research," *Inf. Syst. Res.*, vol. 14, no. 3, pp. 221–243, 2003.

[46] K. Popper, *The Logic of Scientific Discovery*. New York, NY, USA: Basic Books, 1959.

[47] W. V. Quine, "Two dogmas of empiricism," *Philosoph. Rev.*, vol. 60, no. 1, pp. 20–43, Jan. 1951.

[48] J. R. Platt, "Strong inference," *Sci.*, vol. 146, no. 3642, pp. 347–353, 1964.

[49] W. Talbott, "Bayesian epistemology," in *Stanford Encyclopedia of Philosophy*, Stanford, CA, USA: Stanford Univ., 2008.

[50] E. Sober, "Testability," in *Proceedings and Addresses of the American Philosophical Association*, Newark, DE, USA: American Philosophical Association, vol. 73, no. 2, pp. 47–76, 1999.

[51] A. Langley, C. Smallman, H. Tsoukas, and A. H. Van de Ven, "Process studies of change in organization and management: Unveiling temporality, activity, and flow," *Academy Manage. J.*, vol. 56, no. 1, pp. 1–13, Feb. 2013.

[52] E. Rosch, "Principles of categorization," in *Cognition and Categorization*, E. Rosch and B. Lloyd, Eds. Hillsdale, NJ, USA: Lawrence Erlbaum, 1978, pp. 27–48.

[53] J. Parsons and Y. Wand, "Using cognitive principles to guide classification in information systems modeling," *MIS Quarterly*, vol. 32, no. 4, pp. 839–868, 2008.

[54] S. Bayona-Oré, J. A. Calvo-Manzano, G. Cuevas, and T. San-Feliu, "Critical success factors taxonomy for software process deployment," *Softw. Quality J.*, vol. 22, no. 1, pp. 21–48, Dec. 2012.

[55] B. W. Boehm, J. R. Brown, and M. Lipow, "Quantitative evaluation of software quality," in *Proc. IEEE 2nd Int. Conf. Softw. Eng.*, 1976, pp. 592–605.

[56] A. Maglyas, U. Nikula, and K. Smolander, "What are the roles of software product managers? An empirical investigation," *J. Syst. Softw.*, vol. 86, no. 12, pp. 3071–3090, Dec. 2013.

[57] R. Hoda, J. Noble, and S. Marshall, "Self-organizing roles on agile software development teams," *IEEE Trans. Softw. Eng.*, vol. 39, no. 3, pp. 422–444, Mar. 2013.

[58] N. Paternoster, C. Giardino, M. Unterkalmsteiner, T. Gorschek, and P. Abrahamsson, "Software development in startup companies: A systematic mapping study," *Inf. Softw. Technol.*, vol. 56, no. 10, pp. 1200–1218, Oct. 2014.

[59] P. Ralph and P. Kelly, "The dimensions of software engineering success," in *Proc. Int. Conf. Softw. Eng.*, 2014, pp. 24–35.

[60] V. G. Stray, Y. Lindsjørn, and D. Sjøberg, "Obstacles to efficient daily meetings in agile development projects: A case study," in *Proc. ACM/IEEE Int. Symp. Empirical Softw. Eng. Meas.*, 2013, pp. 95–102.

[61] G. Coleman and R. O'Connor, "Using grounded theory to understand software process improvement: A study of Irish software product companies," *Inf. Softw. Technol.*, vol. 49, no. 6, pp. 654–667, Jun. 2007.

[62] J. Saldana, *The Coding Manual for Qualitative Researchers*. London, U.K.: SAGE Publications, 2012.

[63] R. Weber, *Ontological Foundations of Information Systems*. London, U.K.: Coopers & Lybrand, 1997.

[64] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele University and University of Durham Version 2.3, 2007, https://pdfs.semanticscholar.org/e62d/bbbbe70cabcde3335765009e94ed2b9883d5.pdf

[65] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in *Proc. Int. Conf. Eval. Assessment Softw. Eng.*, 2008, pp. 68–77.

[66] D. S. Cruzes and T. Dyba, "Recommended steps for thematic synthesis in software engineering," in *Proc. Int. Symp. Empirical Softw. Eng. Meas.*, 2011, pp. 275–284.

[67] N. Condori-Fernandez, M. Daneva, K. Sikkel, R. Wieringa, O. Dieste, and O. Pastor, "A systematic mapping study on empirical evaluation of software requirements specifications techniques," in *Proc. 3rd Int. Symp. Empirical Softw. Eng. Measurement*, 2009, pp. 502–505.

[68] L. Pickard, B. Kitchenham, and P. Jones, "Combining empirical results in software engineering," *Inf. Softw. Technol.*, vol. 40, no. 14, pp. 811–821, Nov. 1998.

[69] D. Finfgeld-Connett, "The future of theory-generating meta-synthesis research," *Qualitative Health Res.*, vol. 26, no. 3, pp. 291–293, Nov. 2015.

[70] J. S. Dyer, P. C. Fishburn, R. E. Steuer, J. Wallenius, and S. Zionts, "Multiple criteria decision making, multiattribute utility theory: The next ten years," *Manage. Sci.*, vol. 38, no. 5, pp. 645–654, May 1992.

[71] C. W. Alexander, *Notes on the Synthesis of Form*. Cambridge, MA, USA: Harvard Univ. Press, 1964.

[72] P. Ralph, "Fundamentals of software design science," PhD Dissertation, Univ. British Columbia, Vancouver, Canada, p. 117, Oct. 2010, https://circle.ubc.ca/handle/2429/29536

[73] P. E. Vermaas and K. Dorst, "On the conceptual framework of John Gero's FBS-model and the prescriptive aims of design methodology," *Des. Studies*, vol. 28, no. 2, pp. 133–157, 2007.

[74] P. Kruchten, *The Rational Unified Process: An Introduction*, 3rd ed. Reading, MA, USA: Addison-Wesley, 2004.

[75] J. Webster and R. Watson, "Analyzing the past to prepare for the future: Writing a literature review," *MIS Quarterly*, vol. 26, no. 2, pp. xiii–xxiii, 2002.

[76] G. W. Noblit and R. D. Hare, *Meta-Ethnography: Synthesizing Qualitative Studies*. Thousand Oaks, CA, USA: Sage Publications, 1988.

[77] R. Yin, *Case Study Research: Design and Methods*, 3rd ed., Thousand Oaks, CA, USA: Sage Publications, 2003.

[78] K. M. Eisenhardt, "Building theories from case study research," *Academy Manage. Rev.*, vol. 14, no. 4, pp. 532–550, 1989.

[79] K.-J. Stol, P. Ralph, and B. Fitzgerald, "Grounded theory in software engineering research: A critical review and guidelines," in *Proc. IEEE Int. Conf. Softw. Eng.*, 2016, pp. 120–131.

[80] J. Grudin, "Bias," *Interactions. ACM,* Aug. 2013, http://interactions.acm.org/blog/view/bias

[81] OED, *OED Online*. Oxford, UK: Oxford Univ. Press, 2017.

[82] A. Bogner, B. Littig, and W. Menz, *Interviewing Experts*. London, U.K.: Palgrave Macmillan, 2009.

[83] W. Stacy and J. MacMillan, "Cognitive bias in software engineering," *Commun. ACM*, vol. 38, no. 6, pp. 57–63, 1995.

[84] P. Ralph, "Toward a theory of debiasing software development," in *Proc. 4th SIGSAND/PLAIS EuroSymp. Res. Syst. Anal. Design*, 2011, pp. 92–105.

[85] D. Arnott, "Cognitive biases and decision support systems development: A design science approach," *Inf. Syst. J.*, vol. 16, no. 1, pp. 55–78, 2006.

[86] J. Parsons and C. Saunders, "Cognitive heuristics in software engineering: Applying and extending anchoring and adjustment to artifact reuse," *IEEE Trans. Softw. Eng.*, vol. 30, no. 12, pp. 873–888, Dec. 2004.

[87] R. E. Nisbett and T. D. Wilson, "Telling more than we can know: Verbal reports on mental processes," *Psychological Rev.*, vol. 84, no. 3, pp. 231–259, 1977.

[88] J. Bansler and K. Bødker, "A reappraisal of structured analysis: Design in an organizational context," *ACM Trans. Inf. Syst.*, vol. 11, no. 2, pp. 165–193, 1993.

[89] B. Glaser, *Basics of Grounded Theory Analysis: Emergence vs Forcing*. Mill Valley, CA, USA: Sociology Press, 1992, pp. 1–134.

[90] L. Mathiassen and S. Purao, "Educating reflective systems developers," *Inf. Syst. J.*, vol. 12, no. 2, pp. 81–102, 2002.

[91] E. H. Ferneley and P. Sobreperez, "Resist, comply or workaround? An examination of different facets of user engagement with information systems," *Eur. J. Inf. Syst.*, vol. 15, pp. 345–356, 2006.

[92] L. Dube and G. Pare, "Rigor in information systems positivist case research: Current practices, trends and recommendations," *MIS Quarterly*, vol. 27, no. 4, pp. 597–635, Dec. 2003.

[93] R. K. Yin, *Case Study Research: Design and Methods*, 4 ed., Thousand Oaks, CA, USA: Sage, 2008.

[94] M. Shermer, *Why People Believe Weird Things*. New York, NY, USA: Holt Paperbacks, 2002.

[95] A. Koriat, M. Goldsmith, and A. Pansky, "Toward a psychology of memory accuracy," *Annu. Rev. Psychology.*, vol. 51, no. 1, pp. 481–537, Feb. 2000.

[96] B. Fischoff, "Debiasing," in *Judgment Under Uncertainty: Heuristics and Biases*, no. 31, D. Kahneman, P. Slovic, and A. Tversky, Eds. Cambridge, MA, USA: Cambridge Univ. Press, 1982.

[97] R. F. Pohl, Ed., *Cognitive Illusions*. East Sussex, U.K.: Psychology Press, 2004.

[98] J. T. Jost and R. Andrews, "System justification theory," in *The Encyclopedia of Peace Psychology*, Hoboken, NJ, USA: Blackwell Publishing, 2011.

[99] J. T. Jost, M. R. Banaji, and B. A. Nosek, "A decade of system justification theory: Accumulated evidence of conscious and unconscious bolstering of the status quo," *Political Psychology*, vol. 25, no. 6, pp. 881–919, 2004.

[100] J. S. Gero and U. Kannengiesser, "A function-behavior-structure ontology of processes," *Artif. Intell. Eng. Des. Anal. Manuf.*, vol. 21, no. 4, pp. 379–391, 2007.

[101] J. S. Gero and U. Kannengiesser, "The situated function-behaviour-structure framework," *Design Studies*, vol. 25, no. 4, pp. 373–391, 2004.
[102] J. S. Gero, "Design prototypes: A knowledge representation schema for design," *AI Mag.*, vol. 11, no. 4, pp. 26–36, 1990.
[103] S. Curtis, W. Gesler, G. Smith, and S. Washburn, "Approaches to sampling and case selection in qualitative research: Examples in the geography of health," *Social Sci. Med.*, vol. 50, no. 7, pp. 1001–1014, Apr. 2000.
[104] E. Tempero, et al., "The qualitas corpus: A curated collection of java code for empirical studies," in *Proc. IEEE 17th Asia Pacific Softw. Eng. Conf.*, 2010, pp. 336–345.
[105] M. B. Miles and A. M. Huberman, *Qualitative Data Analysis: An Expanded Sourcebook*. Thousand Oaks, CA, USA: Sage Publications, 1994.
[106] P. Runeson, M. Host, A. Rainer, and B. Regnell, *Case Study Research in Software Engineering: Guidelines and Examples*. Hoboken, NJ, USA: Wiley, 2012.
[107] D. P. Truex, R. Baskerville, and J. Travis, "Amethodical systems development: The deferred meaning of systems development methods," *Accounting, Manage. Inf. Technol.*, vol. 10, no. 1, pp. 53–79, 2000.
[108] A. Langley, "Strategies for theorizing from process data," *Academy Manage. Rev.*, vol. 24, no. 4, pp. 691–710, Oct. 1999.
[109] P. R. Monge, "Theoretical and analytical issues in studying organizational processes," *Org. Sci.*, vol. 1, no. 4, pp. 406–430, Jan. 1990.
[110] K. A. Ericsson and H. A. Simon, *Protocol Analysis: Verbal Reports as Data*. Cambridge, MA, USA: MIT Press, 1993.
[111] K. Dorst, "Analysing design activity: New directions in protocol analysis," *Design Studies*, vol. 16, no. 2, pp. 139–142, Apr. 1995.
[112] J. S. Gero and T. McNeill, "An approach to the analysis of design protocols," *Design Studies*, vol. 19, no. 1, pp. 21–61, 1998.
[113] K. A. Ericsson and H. A. Simon, "How to study thinking in everyday life: Contrasting think-aloud protocols with descriptions and explanations of thinking," *Mind Culture Activity*, vol. 5, no. 3, pp. 178–186, Jul. 1998.
[114] H. Sharp, H. Robinson, and M. Woodman, "Software engineering: Community and culture," *IEEE Softw.*, vol. 17, no. 1, pp. 40–47, Jan./Feb. 2000.
[115] A. C. Weller, *Editorial Peer Review: Its Strengths and Weaknesses*. Medford, NJ, USA: Information Today, 2001.
[116] J. M. Campanario, "Commentary on influential books and journal articles initially rejected because of negative referees' evaluations," *Sci. Commun.*, vol. 16, no. 3, pp. 304–325, Mar. 1995.
[117] R. A. Wolfe, "Organizational innovation: Review, critique and suggested research directions," *J. Manage. Studies*, vol. 31, no. 3, pp. 405–431, 1994.
[118] M. Poole, A. H. Van de Ven, K. Dooley, and M. E. Holmes, *Organizational Change and Innovation Processes Theory and Methods for Research*. New York, NY, USA: Oxford Univ. Press, 2000.
[119] M. S. Poole, "On the study of process in communication research," in *Communication Yearbook 36*, C. T. Salmon, Ed., New York, NY, USA: Routledge, 2012, pp. 371–409.
[120] P. Ralph and K. Monu, "Toward a unified theory of digital games," *Comput. Game J.*, vol. 4, no. 1, pp. 81–100, 2015.
[121] T. Sedano, P. Ralph, and C. Péraire, "Software development waste," in *Proc. Int. Conf. Softw. Eng.*, 2017, pp. 130–140.
[122] K. Beck, *Extreme Programming eXplained: Embrace Change*, 2nd ed., Boston, MA, USA: Addison Wesley, 2005.
[123] D. T. Campbell and J. Stanley, *Experimental and Quasi-experimental Designs for Research*. Boston, MA, USA: Houghton Mifflin, 1963.
[124] P. Checkland and S. Holwell, "Action research: Its nature and validity," *Systemic Practice Action Res.*, vol. 11, no. 1, pp. 9–21, 1998.
[125] P. Ralph, "Lab-based action design research," in *Companion Proc. Int. Conf. Softw. Eng.*, 2014, pp. 528–531.
[126] M. E. Fonteyn, B. Kuipers, and S. J. Grobe, "A description of think aloud method and protocol analysis," *Qualitative Health Res.*, vol. 3, no. 4, pp. 430–441, Nov. 1993.
[127] K. Dorst and J. Dijkhuis, "Comparing paradigms for describing design activity," *Design Studies*, vol. 16, no. 2, pp. 261–274, 1995.
[128] V. B. Kampenes, T. Dyba, J. E. Hannay, and D. Sjøberg, "A systematic review of quasi-experiments in software engineering," *Inf. Softw. Technol.*, vol. 51, no. 1, pp. 71–82, Jan. 2009.
[129] M. J. Baker, "Data collection – questionnaire design," *Mark. Rev.*, vol. 3, no. 3, pp. 343–370, Jun. 2003.
[130] D. W. Straub, "Validating instruments in MIS research," *MIS Quarterly*, vol. 13, no. 2, pp. 147–169, 1989.
[131] M.-C. Boudreau, D. Gefen, and D. W. Straub, "Validation in information systems research: A state-of-the-art assessment," *MIS Quarterly*, vol. 25, no. 1, pp. 1–16, 2001.
[132] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft Comput.*, vol. 9, no. 1, pp. 3–12, Oct. 2003.
[133] D. T. Campbell and D. W. Fiske, "Convergent and discriminant validation by the multitrait-multimethod matrix," *Psychological Bulletin*, vol. 56, no. 2, pp. 81–105, 1959.
[134] R. Hunicke, M. LeBlanc, and R. Zubek, "MDA: A formal approach to game design and game research," in *Proc. AAAI Workshop Challenges Game AI*, pp. 1–5, 2004.
[135] J. Schell, *The Art of Game Design: A Book of Lenses*. San Mateo, CA, USA: Morgan Kaufmann, 2008.
[136] B. Hoffman, "Cognitive efficiency: A conceptual and methodological comparison," *Learn. Instruction*, vol. 22, no. 2, pp. 133–144, Apr. 2012.
[137] J. McKay and P. Marshall, "The dual imperatives of action research," *Inf. Technol. People*, vol. 14, no. 1, pp. 46–59, 2001.
[138] W. Van Orman Quine, *Pursuit of Truth*. Cambridge, MA, USA: Harvard Univ. Press, 1990.
[139] R. Bhaskar, *A Realist Theory of Science*. Abingdon, U.K.: Routledge, 2008.
[140] S. K. Sim and A. H. B. Duffy, "Towards an ontology of generic engineering design activities," *Res. Eng. Des.*, vol. 14, no. 4, pp. 200–223, 2003.
[141] K. Dorst and N. Cross, "Creativity in the design process: Co-evolution of problem–solution," *Des. Studies*, vol. 22, no. 5, pp. 425–437, Sep. 2001.
[142] K. Conboy, "Agility from first principles: Reconstructing the concept of agility in information systems development," *Inf. Syst. Res.*, vol. 20, no. 3, pp. 329–354, 2009.
[143] M. Wufka and P. Ralph, "Explaining agility with a process theory of change," in *Proc. Agile Conf.*, 2015, pp. 60–64.
[144] W. Scacchi, J. Feller, B. Fitzgerald, S. Hissam, and K. Lakhani, "Understanding free/open source software development processes," *Softw. Process: Improvement Practice*, vol. 11, no. 2, pp. 95–105, 2006.
[145] P. Ralph, "Comparing two software design process theories," in *Proc. Int. Conf. Des. Sci. Res. Inf. Syst. Technol.*, 2010, pp. 139–153.
[146] T. Allweyer, "Business process model and notation," OMG Standard, formal/2011-01-03, 2011, http://www.omg.org/spec/BPMN/2.0

Paul Ralph received a PhD in management from the University of British Columbia. He is an award-winning scientist, author and consultant, a senior lecturer in computer science with the University of Auckland and a visiting assistant professor of management with the University of British Columbia. His research centers on the empirical study of software engineering, game development, and the ethics of technology. His research has been published in premier software engineering and information systems outlets including the *International Conference on Software Engineering*, *IEEE Transactions on Software Engineering*, the *International Conference on Information Systems*, the *Journal of the Association for Information Systems* and *Information and Software Technology*. He has received funding from Google and The National Sciences and Engineering Research Council of Canada. Additionally, he has written editorials on technology, education and design for influential outlets including Business Insider, Lifehacker and The Conversation. He is the founding director of the Auckland Game Lab, co-founder of the AIS Special Interest Group for Game Design and Research (SIGGAME) and a member of the IEEE Technical Council on Software Engineering and ACM Special Interest Group on Software Engineering. Previously, he was a lecturer with the Lancaster University Management School, the highest rated management research institution in the United Kingdom.