Evaluating Process Theories in Software Engineering

Paul Ralph
Lancaster University
Lancaster, UK
paul@paulralph.name

ABSTRACT

A process theory is an explanation of how and why an entity changes and develops. While software engineering is fundamentally concerned with how and why software artifacts change and develop, little research explicitly develops and empirically evaluates software engineering process theories. This lack of theory obstructs scientific consensus by focusing the academic community on methods. Methods inevitably oversimplify and over-rationalize reality, obfuscating crucial phenomenon including uncertainty, problem framing and illusory requirements. Better process theories are therefore needed to ground software engineering in empirical reality. However, poor understanding of process theory issues impedes research and publication. This paper therefore attempts to clarify the nature of process theories, address some common misconceptions and elucidate the unique issues of process theory evaluation.

Categories and Subject Descriptors

D.2.0 [**Software Engineering General**]: Standards

General Terms

Experimentation, Standardization

Keywords

Research Methodology, Process Theory, Reviewing, Checklist

1. What is a Process Theory?

Some research distinguishes between two broad categories of theories – *variance theories* focus on causal relationships between constructs while *process theories* focus on how something changes over time [16]. The theory that test-first development reduces bugs is a variance theory. The theory that organisms evolve through mutation and natural selection is a process theory. More formally, *process theory* may be defined as follows.

Process Theory: *a system of ideas intended to explain (and possibly to describe, to predict or to analyze) how an entity changes and develops* (cf. [9, 16]).

Process theories **are not** methods. A method(ology) *prescribes* while a process theory *explains*. A method focuses on one (or a few) effective ways of doing something. A process theory explains all of the ways something occurs, both effective and ineffective. A method claims "this way will work"; a process theory claims "it is done this way". Furthermore, process theories **are not** process models. "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" [3]. A process model *describes*; a process theory *explains*. Like methods, process models describe one or a few sequences while process theories seek to encompass all of the ways an entity changes.

Despite numerous calls (e.g. [5, 6, 12, 14]) for more theory in Software Engineering (SE), theories of SE processes remain elusive. This lack of theory obstructs scientific consensus by focusing the academic community on methods, which inevitably oversimplify and over-rationalize reality, obfuscating crucial phenomena including problem framing [15] and illusory requirements [13].

Software Engineering needs process theories, which encapsulate empirical, explanatory knowledge, to counterbalance methods and process models, which encapsulate more practical prescriptions. Without process theories, SE academics conceptualize their research using concepts from methods. For example, Waterfall, Spiral, the Unified Process, the V-Model and eXtreme Programming all present analysis and design as mutually exclusive activities, while empirical research on expert designers consistently finds that analysis and design are inseparable [11, 15]. Researchers indoctrinated in methods may therefore have trouble recognizing observable phenomenon (e.g. Coevolution [11, 15]), which violate the preconception that analysis and design are separate activities.

Additionally, since SE is a process, a General Theory of Software Engineering (GTSE) should attempt to explain how this process occurs by including process theory elements [6, 14]. However, the GTSE initiative is impeded by the lack of guidance for process theory research. This paper therefore provides crucial guidance for developing, evaluating and reviewing process theories, with an eye toward developing a general theory of the SE process.

2. Types of Process Theories

At least four types of process theories are evident in the literature – lifecycle, dialectic, evolutionary and teleological [16]

A lifecycle theory "is a unitary sequence (it follows a single sequence of stages or phases), which is 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)" [16]. This progression occurs because "the trajectory to the final end state is prefigured and requires a particular historical sequence of events" [16]. Lifecycle theories have their roots in biological life cycles. If it were a theory, the Waterfall Model would be a lifecycle theory.

In a teleological process theory, an agent "constructs an envisioned end state, takes action to reach it and monitors the progress" [16]. In other words, teleological theories explain the

ICSE'14, May 31 – June 7, 2014, Hyderabad, India

behavior of agents taking steps to reach goals, but the agent chooses its own sequence of steps. Teleological process theories are very common in the social sciences [16]. Sensemaking-Coevolution-Implementation Theory (SCI) is a teleological theory that explains how a cohesive development team build a complex software system (Figure 1). It posits that development teams engage in three basic activities: making sense of an ambiguous, problematic context, *coevolution* and actually building the system. Coevolution refers to a rapid oscillation between ideas about the context and ideas about the space of possible design artifacts.

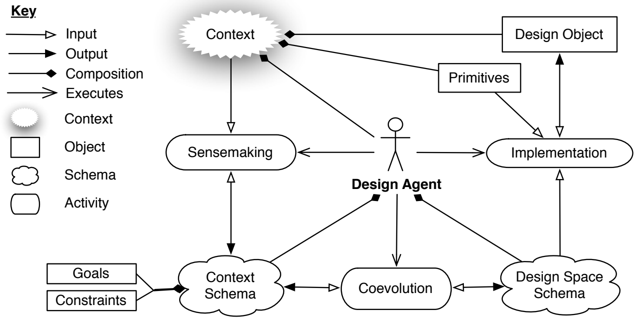

Figure 1. Sensemaking-Coevolution-Implementation Theory (adapted from [11])

In a dialectic process theory, "stability and change are explained by reference to the balance of power between opposing entities" [16]. This is rooted in the argumentative methods of classical philosophy. Dialectic process theories, therefore, posit two or more conflicting entities and model change with respect to inter-entity power. For example, Allison and Merali [1] proposed a dialectic theory of software process improvement (Figure 2). It proposed a dialectical interplay between software development and software process improvement, where each informs the other.

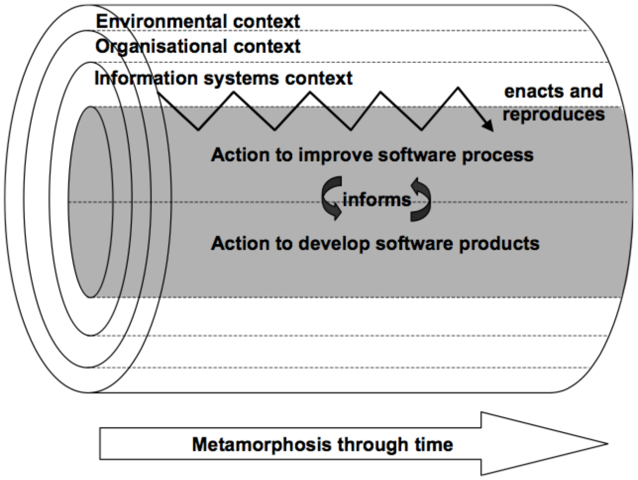
Environmental context
Organisational context
Information systems context
enacts and reproduces
Action to improve software process
informs
Action to develop software products
Metamorphosis through time

Figure 2. An Emergent View of Software Process Improvement (from [1])

In an evolutionary process theory, a population of "organizational entities" undergoes structural changes through "variation, selection and retention" [16]. Variation involves producing new entities through chance occurrences. Selection is the preservation of entities with higher fitness and elimination of those with lower fitness. "Retention involves forces (including inertia and persistence) that perpetuate and maintain certain organizational forms" [16]. For example, the Problem-Design Exploration Model [8] is an evolutionary process theory that explains how genetic algorithms may be used to design systems (Figure 3).

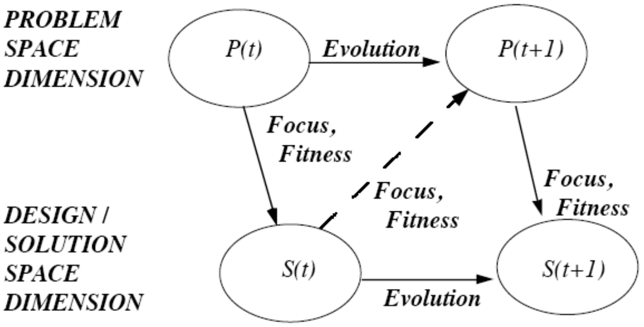

Figure 3. Problem-Design Exploration Model (from [8])[1]

[1] *P = Problem Space; S = Solution Space; t = Time.*

While process theories do not posit causal relationships between constructs, they may adopt a particular approach to causality [7]. The Theory of Evolution, for instance, adopts a probabilistic approach to causality in that evolution is driven by fitter organisms' greater propensity to reproduce. Teleological process theories adopt teleological causality, i.e., actions are caused by the choices of agents with free will.

3. How to Evaluate Process Theories

Process and variance theories make different kinds of truth claims. Therefore, they need different kinds of evaluation – both empirically and conceptually. This section extends and clarifies previous guidance for empirically evaluating process theories, which emphasizes questionnaires and field studies [10, 17].

Randomized controlled trials are often used to test *variance* theories. The investigator manipulates the independent variable(s) and observes the dependent variable(s) while controlling everything else to establish a causal relationship. As process theories do not have independent or dependent variables, randomized controlled trials are inappropriate for testing them.

Moreover, in a randomized controlled trial, a hypothesis (e.g. X causes Y) is often tested against a null hypothesis (e.g. X and Y are unrelated). Null hypothesis testing is less helpful for process theories. For example, SCI posits that designers engage in sensemaking, i.e., giving meaning to an ambiguous situation. The null hypothesis, situations are never ambiguous and designers do not make sense of them, is a straw man. The relevant question is not whether we can view some design activities as sensemaking, the question is whether *sensemaking* has more explanatory power than an alternative concept, e.g., *analysis*. More generally, evaluating a process theory on its own is deeply problematic. In predominately qualitative field studies, one will usually make some observations consistent with the theory (unless it is a straw man) and some observations incongruous with the theory. In more quantitative, questionnaire-based approaches it is too easy to unconsciously exploit response bias for positive results. Rather, process theories should be evaluated against a rival theory [18]. Asking *which of these two theories is more consistent with these observations?* is simply more tractable and defensible.

Selecting an appropriate rival may be challenging. If several potential rivals are evident, reviewers may legitimately complain that the choice is subjective arbitrary. While authors should justify their choice of rival theory; reviewers should not criticize this choice unless they can name and cite a different rival and give a definitive reason why it is superior. Imperfect rivals are often better than no rivals. Process theories can be tested against rivals in two main ways – predominately quantitative questionnaire studies or predominately qualitative field studies.

3.1 The Questionnaire Approach

A good process theory should suggest numerous testable propositions. However, some propositions are more interesting than others. For example, the proposition that software developers create software artifacts is not interesting because no one disputes that. Interesting propositions contradict rival theories. For example, SCI suggests that the goals of a system are constructed by the designer rather than given by other stakeholders.

After identifying several propositions that distinguish the proposed theory from its rival, one can generate numerous questions that reflect these differences. Specifically, one can generate bipolar scales where one pole reflects the proposed theory and the other pole reflects the rival theory; for example:

The goals of my current project were...
a) ... completely provided by the client (Rival Theory)
b) ... primarily provided by the client (Rival Theory)
c) ... determined equally by the client and the development team
d) ... primarily determined by the development team (SCI)
e) ... completely determined by the development team (SCI)

Running the survey will produce a response distribution for each question; e.g., a) 5; b) 10; c) 20; d) 30; e) 15. In this example, visual inspection suggest the distribution favors SCI. To determine statistical significance we can use non-parametric tests including Chi-Square Test and Kolmogorov-Smirnov test. These tests require an *expected distribution,* three options for which are:

1. Uniform Distribution, e.g., a) 20; b) 20; c) 20.; d) 20; e) 20.
2. Pseudo-Normal Distribution, e.g., a) 6; b) 19; c) 30; d) 19; e) 6.
3. Reflected distribution (the inverse of the observed distribution), e.g., a) 15; b) 30; c) 20; d) 10; e) 5.

Using a uniform distribution makes little sense as it does not answer the question at hand. Using the pseudo-normal distribution addresses the question, *is the extent of skew in the observed distribution significant?* However, treating discontinuous, ordinal data as normally distributed is statistically problematic. 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. The chi-square goodness of fit test (with significance via the sign test) may be used with few assumptions.

Meanwhile, sampling is hindered by the lack of comprehensive population lists. While consumer marketing researchers can randomly select numbers from a phone book, SE researchers have no comprehensive lists of software developers. Two strategies are evident: 1) randomly sample from a specific community that does have a population list (e.g. Source Forge); 2) try to maximize the diversity of a convenience sample. Neither approach supports statistical generalization *to the population of interest.* Both approaches have benefits and drawbacks; reviewers should accept both as no superior option is available in the absence of a defensible population list. A more ambitious approach might utilize respondent-driven sampling [4] – a method of addressing bias in referral-chain (snowball) samples.

Additionally, validating this kind of questionnaire fundamentally differs from validating a variance theory testing questionnaire. Variance theories often posit causal relationships between constructs. A construct is a postulated (often psychometric) variable that cannot be measured directly, e.g., trust, extroversion, team cohesion, software quality. A variable is a quantity that may have different values. Therefore, the researcher generates several questionnaire *items* (questions) intended to *reflect* each construct. To analyze validity, the researcher pilots the questionnaire and analyzes the correlations between the items. Items that reflect the the same construct should be highly correlated with each other (convergent validity) and less correlated to items reflecting other constructs (discriminate validity) [2].

Process theories do not necessarily have constructs, e.g., sensemaking and coevolution are activity-categories, not constructs – they cannot take on quantities. Moreover, questionnaire items that reflect differing predictions of two process theories, e.g., *goals are given by stakeholders / constructed by developers*, are definitely not constructs. When a researcher devises several questions about a process theory proposition, the questions do become variables but these variables are not reflective indicators of the same underlying construct. As there is no *a priori* reason to believe that items associated with one theory difference will be more closely related to each other than to items associated with other differences, convergent and discriminate validity analysis does not apply. Instead, process theory testing questionnaires may be validated by piloting the questionnaire with both experts and members of the target population, and inviting detailed, qualitative feedback on their interpretations of each item.

3.2 The Many Qualitative Approaches

Alternatively, a researcher may take a variety of qualitative approaches to evaluate a process theory. To be clear, as this discussion concerns evaluating pre-existing theory, we are considering broadly *positivist* qualitative research.

The researcher develops separate coding schemes for the proposed and rival theories. Each coding scheme lists all of the components (e.g. phases, activities, actors) and relationships (e.g. sequence, dependency) of the theory. The coding scheme provides space for evidence both for and against each component and relationship.

Data collection depends on the specific approach. In an observational field study, data may include field notes, interview transcripts, documents, diagrams, emails, segments of source code, screen shots, photographs of the physical environment, code commit logs and video of meetings or other activities. In a lab-based simulation, data may include video of the simulation, copies of artifacts produced and the written or oral reflections of participants. In a think-aloud protocol study (where participants are asked to think aloud during a task) data would include the protocol transcripts. In a retrospective analysis, data would consist of previous case data (e.g. transcripts, documents) or case narratives (i.e. descriptions of cases written by researchers). In a discourse analysis, the data would consist of a corpus of relevant texts (e.g. books, articles, blog posts, wiki entries).

Regardless of the combination of data, the researcher organizes the data, identifying evidence for and against each theory component. The researcher then weighs the evidence to reach a conclusion about which theory is better supported. While the evidence may better support the proposed theory or the rival theory, other conclusions are possible, e.g., that both theories are deficient or both are partly correct and can be merged.

Both questionnaire and qualitative approaches are valid on their own. However, combining one or more qualitative studies with a questionnaire study facilitates significant data triangulation, mitigates mono-method bias and generally encourages more nuanced reflection. That said, multi-methodological studies are significantly more complicated to write up and to review. Paradoxically, multi-method studies often appear to have more limitations and problems despite being fundamentally more sound than mono-method studies. Authors must be extra clear while reviewers should give due credit for these difficult studies.

4. Publishing and Reviewing Process Theories

A GTSE should consider not only the causes of SE success but also the process of developing software. The latter necessitates a process theory perspective. This paper therefore provides guidelines for presenting and reviewing process theories in software engineering research.

Reviewers should not reject process theories as intrinsically inferior to variance theories. *Theory* has never denoted a set of mathematical expressions relating a dependent variable to one or more independent variables. The Theory of Evolution, the Nitrogen Cycle, Plate Tectonics, the Constructivist theory of learning, Attribution Theory and Structuration Theory are all process theories. To marginalize these and other process theories has no basis in philosophy of science and contradicts decades of physical- and social-scientific consensus. Process theories **are** theories and should be treated as such by authors and reviewers.

In conclusion, while developing and testing process theories is crucial for SE's legitimacy, it may be complex and difficult. Reviewers should recognize this difficulty and favor a developmental approach. Authors and reviewers might make use of the following checklist for conducting, communicating and reviewing process theory evaluations (Table 1).

Table 1. Key Questions for Process Theory Evaluation Papers

Does the process theory *explain* how something happens?	
Are components, relationships and assumptions clearly defined?	
Is theory type (e.g. lifecycle, teleological) discussed?	
Does the theory have components specific to its type (e.g. teleological theories should propose an agent)?	
Is scope clear? Are specific boundary conditions provided?	
Has a rival theory been identified and justified?	
Have specific disagreements between the proposed and rival theories been identified?	
Has the theory been evaluated using a questionnaire, field study or other reasonable approach?	
Is the conclusion supported by the evidence and does it address the research question?	
Questionnaire Study	
Focus on theory differences?	Instrument validation pilot?
Reasonable sampling strategy?	Reflective distribution used?
Qualitative Study	
Context (e.g. the organization studied) described?	
Data collection and analysis processes described?	
Coding scheme reflects theory components?	
Coding examples given and reasonable?	
Conclusions based on balance of evidence?	
Multi-method Approach	
Triangulation between data from different approaches?	
Reviewers	
Am I being too hard on this paper, given the intrinsic difficulties of this kind of research?	

5. REFERENCES

[1] Allison, I. and Merali, Y. 2007. Software process improvement as emergent change: A structurational analysis. *Information and Software Technology*. 49, 6, 668–681.

[2] Campbell, D.T. and Fiske, D.W. 1959. Convergent and discriminant validation by the multitrait-multimethod matrix. *Psychological Bulletin*. 56, 2, 81–105.

[3] Curtis, B., Kellner, M.I. and Over, J. 1992. Process Modeling. *Communications of the ACM*. 35, 9, 75–90.

[4] Heckathorn, D.D. 1997. Respondent-Driven Sampling: A New Approach to the Study of Hidden Populations. *Social Problems*. 44, 2, 174–199.

[5] Johnson, P., Ekstedt, M. and Jacobson, I. 2012. Where's the Theory for Software Engineering? *IEEE Software*. 29, 5, 94–96.

[6] Johnson, P., Ralph, P., Goedicke, M., Ng, P.-W., Stol, K.-J., Smolander, K., Exman, I. and Perry, D.E. 2013. Report on the Second SEMAT Workshop on General Theory of Software Engineering (GTSE 2013). *SIGSOFT Software Engineering Notes*. 38, 5.

[7] Kim, J. 1999. Causation. *The Cambridge Dictionary of Philosophy*. R. Audi, ed. Cambridge University Press. 125–127.

[8] Maher, M., Poon, J. and Boulanger, S. 1995. Formalising design exploration as co-evolution: A combined gene approach. *Preprints of the Second IFIP WG5.2 Workshop on Advances in Formal Design Methods for CAD*, 1–28.

[9] Oxford English Dictionary 2013. Oxford University Press.

[10] Poole, M., Van de Ven, A.H., Dooley, K. and Holmes, M.E. 2000. *Organizational change and innovation processes theory and methods for research*. Oxford University Press.

[11] Ralph, P. 2010. Comparing Two Software Design Process Theories. *Global Perspectives on Design Science Research: Proceedings of the 5th International Conference, DESRIST 2010*. R. Winter, J.L. Zhao, and S. Aier, eds. Springer. 139–153.

[12] Ralph, P. 2013. Possible Core Theories for Software Engineering. *Proceedings of the 2nd SEMAT Workshop on a General Theory of Software Engineering, International Conference on Software Engineering*. SEMAT.org.

[13] Ralph, P. 2013. The Illusion of Requirements in Software Development. *Requirements Engineering*. 18, 3, 293–296.

[14] Ralph, P., Johnson, P. and Jordan, H. 2013. Report on the First SEMAT Workshop on a General Theory of Software Engineering (GTSE 2012). *SIGSOFT Software Engineering Notes*. 38, 2, 26–28.

[15] Schön, D.A. 1983. *The reflective practitioner: how professionals think in action*. Basic Books.

[16] Van de Ven, A.H. and Poole, M.S. 1995. Explaining development and change in organizations. *The Academy of Management Review*. 20, 3, 510–540.

[17] Wolfe, R.A. 1994. Organizational innovation: review, critique and suggested research directions. *Journal of Management Studies*. 31, 3, 405–431.

[18] Yin, R.K. 2008. *Case study research: Design and methods*. Sage.