Science of Computer Programming 101 (2015) 1–5

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Editorial

Towards general theories of software engineering

In 2013, as a part of the SEMAT (Software Engineering Method and Theory) initiative, we organized a workshop devoted to the theme of general theories of software engineering (GTSE). The workshop, in total involving around a hundred authors, participants and program committee members, was held in conjunction with the International Conference on Software Engineering (ICSE 2013). The interest in the workshop, both during ICSE 2013 and afterwards, prompted us to follow up on the theme. The current special issue of Science of Computer Programming is the result.

The GTSE concept may be parsed into two parts, namely *general theory* and *software engineering*. Regarding the complicated concept general theory, it seems easier and more enlightening to provide an extensional definition, i.e. definition by examples, than an intensional one.

Well-known general theories in other disciplines include Maxwell's equations [1], the Standard Model of particle physics [2], the Big Bang theory [3], the theory of the cell [4], probability theory [5], general equilibrium theory in economics [6], John Maynard Keynes' general theory of employment, interest and money [7], Sigmund Freud's theory of the psyche [8], and Einstein's general theory of relativity [9]. Additionally, there are many general theories that are well-established within their respective disciplines, but have not reached the general public, including Henry Mintzberg's general theory of organizational structure [10], Richard Easterlin's unified theory of income and happiness [11], Michael Gottfredson and Travis Hirschi's general theory of crime [12], Nobel Prize winner Gary Becker's theory of marriage [13], and many more. In fact, the business of proposing general theories is in full bloom. Google Scholar reports over 200 hits on the term "general theory" in the titles of scientific articles only in 2013, including contributions such as a general theory of acute and chronic heart failure [14], a general theory of business marketing [15], a more general theory of commodity bundling [16], life history theory and the general theory of crime [17], a general theory of environment-assisted entanglement distillation [18], a general theory of implementation [19], a general theory of behavior and brain coordination [20], and "an almost general theory" of mean size perception [21].

A general theory is in contrast to a specific theory. Typically, the prefix "general" is either attached to a theory that addresses a greater empirical domain than its predecessors, or to a theory that encompasses a whole field (which begs the question of how "fields" are defined). Typically, a general theory aims at answering the big questions in its field. Ideally, it should be able to explain and predict important phenomena in the field. A general theory of electromagnetism should help us understand why one transformer design is superior to another. A general theory of marriage [13] should be able to explain divorce rates. A general theory of economic equilibrium [6] should be able to explain the pricing of goods. A general theory of international relations [22] should predict the effects of foreign policy decisions.

If the concept of *general theory* constitutes one part of the theme of this special issue, then the second part, *software engineering*, may be considered by pondering the big questions that general theories of software engineering would address. Arguably, the issues that have most concerned software engineering since its inception are related to productivity and quality. Why do some software engineering projects produce a fine product at a low cost, while others do not? What tools, processes, methods and competencies will increase productivity and quality? Ideally, a general theory of software engineering will be able to predict the effects of new methods and tools before they are deployed.

1. Why strive for a general theory?

It is not self-evident why it is desirable to strive for a general theory of software engineering. As a first question, we may ask what the problem is with today's situation? It is obviously possible to get by without general theories of software engineering. In brief, there are two problems with today's state of affairs: (a) theory is implicit, and (b) theory is disconnected.

At ICSE 2013, we asked some 60 participants of the 2013 International Software Engineering Conference to produce diagrams describing their personal understanding of causal relationships between core software engineering constructs. No two respondents produced identical diagrams, indicating that consensus on the basic cause-and-effect of software engineering is very low. Thus, there exist general theories of software engineering, but they are subjective. The obvious effect is that two

http://dx.doi.org/10.1016/j.scico.2014.11.005

decision makers facing the same problem will make different choices. If the decision makers are on different projects, then one of them will be fortuitous while other will not. If they are on the same project, chance will determine whether they choose the decision with more or less desirable consequences. There will be no way to rationally compare their beliefs. By making theory explicit, it can be subjected to the critical gaze of science, a gaze that has been remarkably successful in producing credible decision support in many other disciplines.

In addition to personal general theories, the software engineering discipline also features a multitude of well-known specific theories. For instance, the theory of languages and automata [23] is highly mature, and it is a theory of major practical relevance for traditional software engineering problems, such as programming language design. Less formalized, yet well-established, are theories such as Brooks's Law [24], Conway's Law [25], and Boehm's Laws [26]. There are hundreds of such "laws" [26], principles [27–29] and hypotheses [24]. However, these small theories are disconnected from each other. They form no coherent knowledge structure. In fact, based on this mass of principles, an infinite number of contradictory statements can be derived. Each individual must therefore make their own selection of which laws, principles and hypotheses to believe in. Under these circumstances, it is no wonder that every software engineer seems to have their own general theory.

In the absence of a common theory, scientific progress is painstakingly slow, claims philosopher of science Thomas Kuhn [30]. Without a theory to guide information gathering, "different men confronting the same range of phenomena... describe and interpret them in different ways." Rather than jointly scrutinizing and developing common theories, scientists talk past each other and thus fail to focus. This inhibits the development of a cumulative body of knowledge. With a common theory, however, the scope of the field narrows, as joint investigations are concentrated to the topics designated by the theory. The ensuing stage of normal science, characterized by the development of idiosyncratic vocabularies, specialized equipment and advanced skills accelerates progress in terms of scientific depth and detail. Such focused investigations may highlight regularities and anomalies that would have been impossible to detect through the tinkering of individual researchers. Additionally, as is also the case when developing software, joint effort often leads to more advanced results than those attainable for the solitary explorer. Perhaps most importantly of all, a general theory, scrutinized and debated by a whole scientific community, is arguably more likely to reflect the real world than the untested, many times unquestioned, idiosyncratic mix of ideas harbored in the mind of any given community member.

So far, we have discussed the advantages of a common, general theory. It is, however, not certain that a single theory will come to dominate the field, especially from the start. Many academic disciplines feature two or more rival theories. In the field of international relations, there has been a longstanding debate between the realists and the liberalists. In psychology, the cognitive-behavioral theories have confronted the psychoanalytical ones. Although these fields do not harvest all of the benefits of Kuhn's normal science, they represent a significant increase in scientific maturity as compared to disciplines without any general theories. The underlying assumption in the few-theory fields is typically that one of the theories eventually will win the day, thus unifying the scientific community around a common world view.

2. How are general theories created?

We argue above that the endeavor of creating one or a small set of competing general theories of software engineering is a worthwhile endeavor. But how can general theories come into being? The topic of scientific theory generation is a part of the philosophical field of epistemology, and as such has a long history. One of the big epistemological debates has been the one between rationalists and empiricists, between those who believe that things can be known before experience, a priori, and those who believe that knowledge requires observations of the world, a posteriori. We can divide approaches to theory generation into two camps loosely based on this dichotomy. In the first, deductive, a priori case, the theory is derived from other, pre-existing theories. In the second, inductive, a posteriori case, the theory is based (more) directly on empirical observation.

Considering the rationalistic, deductive approach, the adaptation of existing theory to new phenomena seems to be inherent in our way of thinking. The concepts of metaphors and analogical thinking are embodiments of this strategy. In science, analogical thinking is a common approach. For instance, the wave theory of light, proposed by Hooke in the 17th century, applied the theory of fluid waves to the phenomenon of light [31]. Another way to derive new theories from preexisting ones is through integration. A well-known example is the Standard Model of particle physics, which describes the fundamental particles of the universe and how they interact. In 1961, Sheldon Glashow discovered a way to combine the electromagnetic and weak interactions. A few years later, Steven Weinberg and Abdus Salam were able to integrate the Higgs mechanism with Glashow's theory, thus proposing the Standard Model [2]. A long-standing ambition for the future is to also incorporate the strong interaction into the Standard Model, an ambition oftentimes referred to as Grand Unification [2]. The sociologist Robert Merton also championed "consolidating special theories into more general sets of concepts and mutually consistent propositions" [32].

When it comes to the empiricist, inductive, a posteriori, approach, one well-known example is Carl Linnaeus' biological taxonomy [33]. In Linnaeus taxonomy, all living organisms are classified in a tree structure of species and subspecies. Fundamentally descriptive, the taxonomy can be used to predict properties of individuals from knowledge of other properties. For instance, observation of the plumage of a bird may allow us to predict feeding habits, clutch size, migratory habits, life expectancy, and many other aspects. Another kind of data-driven theory development is common within the healthcare field, where large empirical studies are used to identify risk factors. When a factor, such as smoking, is found, the theory