Requirements on General Theories of Software Engineering
An Unusually Dense Position Paper

Pontus Johnson
KTH Royal Institute of Technology
Stockholm, Sweden
e-mail: pontus@ics.kth.se

Iaakov Exman
The Jerusalem College of Engineering
Jerusalem, Israel
e-mail: iaakov@jce.ac.il

Abstract—In this paper, we propose a set of quality criteria of general theories of software engineering: The quality of a general theory of software engineering depends on (i) the universality and precision with which it predicts the influence of the software decision makers’ actions on the software development goals, (ii) its degree of corroboration, (iii) its degree of formalization, (iv) the unambiguousness of its development goals, (v) its degree of corroboration, (vi) its degree of formalization, (vii) its unambiguousness of its measurement procedures. The argument for these quality criteria is based on (a) the oft-quoted adage by Kurt Lewin, "There is nothing so practical as a good theory", (b) Karl Popper's The Logic of Scientific Discovery, and (c) the Essence of the SEMAT Initiative.

I. INTRODUCTION
Quality criteria for scientific theories have been extensively examined within the field of philosophy of science [1]. In this paper, we propose a slightly modified set of criteria suitable for general theories of software engineering. The purpose of the proposal is to guide the search for a general theory of software engineering, as well as to contribute to an increased awareness of the importance in practice of software theories.

II. PREDICTING DECISIONS’ EFFECTS ON GOALS
Kurt Lewin famously proposed that “There is nothing so practical as a good theory” [2]. In the context of software engineering, the word “practical” arguably means that the theory is of benefit to the engineering effort, i.e. that it somehow aids the practitioner attaining her design goals. How then can a theory provide such aid? A theory can itself not perform actions in the real world. Its only manner of influencing the world is through the actions of decision makers; a person’s behavior is contingent on the theory she subscribes to. The benefits of theory thus must come through its decision-guiding capacity.

Given that decision-makers strive for certain goals with a limited set of means at their disposal, their problem is to determine which of the available actions achieve what results; which knobs influence what gauges. This is a problem that a theory can solve; a theory can be used to predict the effects of actions on goals. This leads to a first version of quality criteria (i): The quality of a general theory of software engineering depends on the extent to which it predicts the influence of the software decision makers’ actions on the software development goals.

III. UNIVERSALITY AND PRECISION
In a subsequent section, the software decision makers’ actions and goals are elaborated on. Another critical part of the formulation of the first criteria is “the extent to which [the theory] predicts […].” The iconic philosopher of science Karl Popper [3] employs the term empirical content to denote that “extent”. According to Popper, empirical content is increased either by increasing the universality of the theory or by increasing its precision. To explain these concepts, Popper exemplifies with the following four theories:

p: All heavenly bodies which move in closed orbits move in circles: or more briefly: All orbits of heavenly bodies are circles.
q: All orbits of planets are circles.
r: All orbits of heavenly bodies are ellipses.
s: All orbits of planets are ellipses.

Moving from p to q, the degree of universality decreases; and q says less than p because the orbits of planets form a proper subclass of the orbits of heavenly bodies. Moving from p to r, the degree of precision decreases: circles are a proper subclass of ellipses. Corresponding remarks apply to the other moves. To a higher degree of universality or precision corresponds a greater empirical content. In summary, higher universality means that more things can be predicted, and higher precision means that those predictions become more exact.

Universality and precision are implied in quality criteria (i). A theory that only predicts the effect of one kind of action, e.g. choice of development method, on one kind of goal, e.g. development effort, is less universal than a theory that also can predict the effect on software quality. A theory that only vaguely predicts that a big application will be more costly to develop than a small one is less precise than a theory that predicts the cost in dollars and cents based on function points. A reformulation of criteria (i) is thus: The quality of a general theory of software engineering depends on the universality and precision with which it predicts the influence of the software decision makers’ actions on the software development goals.

IV. DEGREE OF CORROBORATION
According to Popper, “Theories are not verifiable, but they can be ‘corroborated’. The attempt has often been made
to describe theories as being neither true nor false, but instead more or less probable. […] Instead of discussing the ‘probability’ of a hypothesis we should try to assess what tests, what trials, it has withstood; that is, we should try to assess how far it has been able to prove its fitness to survive by standing up to tests. In brief, we should try to assess how far it has been ‘corroborated’.

Software engineering theories do not differ from other theories in this respect; corroboration is a quality criteria. A theory that has passed many difficult empirical trials is better than one that has not. This leads to quality criteria (ii): The quality of a general theory of software engineering depends on its degree of corroboration.

V. DEGREE OF FORMALIZATION

Popper continues: ”The requirement of consistency […] can be regarded as the first of the requirements to be satisfied by every theoretical system, be it empirical or non-empirical. In order to show the fundamental importance of this requirement it is not enough to mention the obvious fact that a self-contradictory system must be rejected because it is ‘false’. […] But the importance of the requirement of consistency will be appreciated if one realizes that a self-contradictory system is uninformative. It is so because any conclusion we please can be derived from it. Thus no statement is singled out, either as incompatible or as derivable, since all are derivable.”

The best way to avoid inconsistency is by formalization. A sufficiently formalized system can be subjected to automated methods to detect inconsistencies. But benefits of formalization appear before a system has been mathematically formalized. A theory presented in a structured form is often less ambiguous than a theory presented casually.

This leads to quality criteria (iii): The quality of a general theory of software engineering depends on its degree of formalization.

VI. AMBIGUITY OF MEASUREMENT PROCEDURES

As stated in the second section, a precise theory is preferable to an imprecise theory. Imprecision can appear in two relations. The first relation, discussed in Section II, is between the constructs of the theory. For instance, the proposition “System X is larger than System Y” is less precise than “System X is twice as large as System Y”.

The second relation that can cause imprecision is between the constructs of the theory on the one hand and the real world on the other. In other words, the theory’s measurement procedures may be imprecise, or stated differently, the theory’s definitions may refer to the real world in imprecise ways. A theory that represents system response time on an ordinal scale of “high”, “medium” and “low” is less precise than a theory that represents response time in seconds.

This leads to quality criteria (iv): The quality of a general theory of software engineering depends on the ambiguity of its measurement procedures.

VII. SOFTWARE ENGINEERING ACTIONS AND GOALS

Having formulated the four quality criteria, we may ask: -Which specific features distinguish a software engineering theory from other theories?

To this end we elaborate additionally on the first quality criteria by detailing the actions and goals of the software decision maker. We believe that it is fruitful to relate these to the Essence developed by the SEMAT initiative [4]. In the Essence, a key concept is the Alphas. Alphas are “representations of the essential things to work with” [4]. Examples of alphas are Software System, Requirements, Work, Team, and Way of working. We propose that these alphas constitute the base for the actions of the software decision maker in quality criteria (i). For instance, an action might be to change the requirements, or to modify the way of working by replacing one practice with another.

Considering the software development goals, SEMAT describes these as better software, faster and with happier customers. Thus, criteria (i) can be further elaborated to the following: The quality of a general theory of software engineering depends on the universality and precision with which it predicts how actions on the SEMAT alphas lead to (or away from) better software faster and with happier customers.

VIII. SUMMARY

We have presented four quality criteria for general theories of software engineering that we believe should guide theory development within the field: (i) the universality and precision with which it predicts the influence of the software decision makers’ actions on the software development goals, (ii) its degree of corroboration, (iii) its degree of formalization, (iv) the unambiguousness of its measurement procedures.

We believe that good SE theories – obeying the above quality criteria – will not appear like fashionable meteors just to fall in disregard a short time after. Each good theory will be tested in the laboratory and in the field against real software systems and endeavors – its predictions gradually gaining confidence by corroboration. This will impart relatively long-term resilience, until a better theory appears and replaces it, like in any other field of science and engineering.

REFERENCES