Requirements Engineering as Science in the Small

Munindar P. Singh
North Carolina State University
Raleigh, NC 27695, USA
singh@ncsu.edu

Amit K. Chopra
Lancaster University
Lancaster LA1 4WA, UK
amit.chopra@lancaster.ac.uk

***Abstract*—This paper identifies similarities between requirements engineering (RE) and the scientific method. It argues that RE is concerned with the development of models of small universes and thus relates to natural science despite RE having a prescriptive rather than a descriptive nature. An explicitly philosophical stance sheds light on RE processes and brings up criteria for judging RE processes.**

***Index Terms*—Philosophy of science; Foundations of requirements engineering; Requirements engineering process**

I. Introduction

Requirements engineering (RE) is a science of the "artificial" [1]: it is concerned with engineering artifacts according to stakeholder requirements. RE involves the efforts of eliciting requirements from stakeholders and building effective specifications that enable software development. The question is not simply whether a software solution is correct (i.e., whether it meets requirements), but also whether we have a correct model of the requirements themselves.

We contrast RE, as a science of the artificial, with the "natural" sciences, which we take to include not just physics and biology, but also the humanities (psychology and sociology). Models and their correctness constitute the fundamental concerns of the natural sciences. Well-known models include of the solar system, atom, cell, and electricity. Models can be shown to be incorrect by empirical evidence. Famous discredited models include the Ptolemaic model of the cosmos, the miasma ("bad air") model of disease, and the four-humor model of health and temperament.

Whereas the natural sciences are concerned with building descriptive models that explain the environment, RE is concerned with prescriptive models that would alter the environment. Whereas the validation criteria in the natural sciences originate in nature, those in RE originate with stakeholders, whose assessments moreover are inherently subjective. Despite these differences, we posit that RE can be usefully understood as *science in the small*, since it studies small, contingent "universes" as constructed from stakeholder requirements.

This viewpoint suggests how key ideas from the philosophy of science may be fruitfully adapted for RE and thereby help improve RE research and practice. In particular, the core ideas in the philosophy of science either apply directly or have clear correlates in RE. These include validation; dealing with assumptions, especially tacit ones; and understanding the social processes involved in developing theories of the (in case of RE, *a*) universe.

II. Software Development and Science

Figure 1 shows broadly the methodology a natural scientist follows. A scientist instruments and observes the universe, and theorizes to produce a model that (purportedly) explains the observations. The scientist evaluates the model against observations and, in doing so, may uncover tacit assumptions. Making the assumptions explicit serves to delimit the scope of applicability of the model in that the model explains observations only in those universes where the assumptions hold.

Scientists use assumptions as a pragmatic separation of concerns device—to limit the scope of the phenomena addressed.

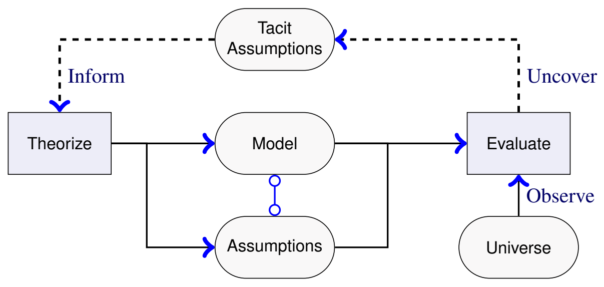

Fig. 1. Natural science. The scientist controls the model but the universe, as instrumented and observed, is fixed. The assumptions scope out the observations and the model.

Figure 2 shows that the methodology a software developer follows is analogous. The developer's job is to design a solution that meets the stated stakeholder requirements. In evaluating whether a solution meets the requirements, the developer may uncover tacit assumptions and make them explicit. Let us consider the example of building an elevator control solution with the requirement that the elevator must safely carry at least eight passengers. An assumption is introduced that any eight passengers would weigh at most 680 kg in total. Another assumption is that the physical construction of the elevator enables it to safely move loads of up to 680 kg. The solution (software) consists of an elevator controller that blocks elevator movement at the current floor if its load exceeds 680 kg. To this end, the software reads the load from a sensor installed in the elevator.

The elevator could violate the requirements if the sensors malfunction. Therefore, the assumption that the sensors were working reliably is now made explicit. The validity of this assumption would, in practice, be supported by a maintenance

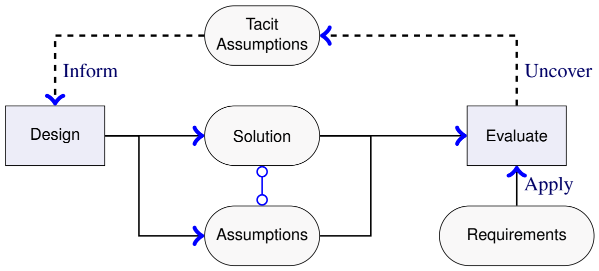

Fig. 2. Software development. The developer controls the solution artifact but not the requirements. The assumptions carry contractual force between developers and stakeholders

contract between the elevator manufacturer and its client. The requirement of safely carrying at least eight people is satisfied by the software solution in those universes where no group of eight passengers who can crowd into the elevator weighs more than 680 kg; the elevator has a load bearing capacity of 680 kg; and the load sensors undergo regular maintenance.

The foregoing account highlights that both software development and science are empirical activities that involve the iterative creation of suitable artifacts (model and solution, respectively) and give key importance to the evaluation of these artifacts. The above-mentioned similarity holds even though the artifacts are differently understood. Scientific models are descriptive whereas engineering solutions do not describe, but necessarily alter, their operating environments.

III. Requirements Engineering

We understand the RE enterprise as a way of factoring Figure 2 into two activities as shown in Figures 3 and 4, each of which follows the same structure as Figure 1.

Adopting well-known terminology [2], Figure 3 shows how RE involves the production of a specification that refines and formalizes stated requirements. In essence, a specification is realizable by a solution artifact. The requirements engineer evaluates whether the specification meets stakeholder requirements. This evaluation is nontrivial and may involve humanistic methods such as ethnographic analysis or user surveys. In doing so, the requirements engineer may uncover tacit assumptions that help refine the specification.

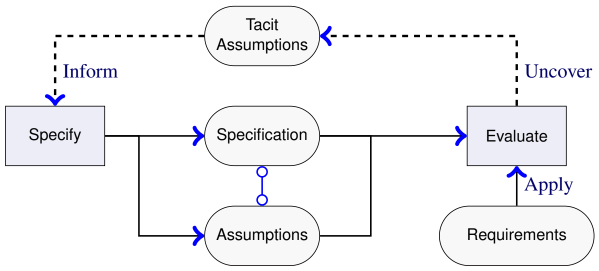

Fig. 3. Refine requirements into specifications and assumptions. Requirements can be informal and lack clear criteria for evaluating specifications.

Specifications of software normally take the form of formal descriptions of actions that the software takes in particular conditions. For example, (and eliding the formal language aspects), the specification of the elevator controller might say that the action $\mathsf{moveTo}(\mathit{floor})$ is a noop in the condition that the variable load has a value greater than 680 (kg). Let's refer to this specification as S_e. Actions and variables in the specification designate actions taken by the software at the software-universe interface and states of the universe, respectively. The advantage of formal specifications is that they lend themselves to automated verification. Thus, defects in specifications can potentially be detected at this stage. In common practice, however, the specifications are often informal descriptions.

The specification serves as a basis for contractual software development. The developer controls the solution but not the specification. In Figure 4, the developer designs a solution that meets the specification. The solution would normally be a program that interfaces with input-output devices. The evaluation could involve rigorous computational methods, such as testing and verification.

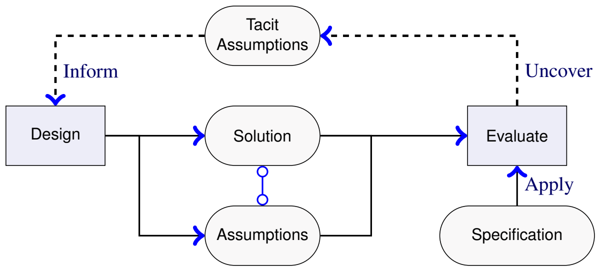

Fig. 4. Solutions implement specifications. The assumptions delineate the limitations of the solution artifact, and should be more general for a solution than for the specification.

The assumptions in Figure 4 capture the conditions under which the solution would satisfy the specification. In essence, these assumptions suggest that the solution is imperfect and characterize its limitations. Stating assumptions may be appropriate as a way to capture whether an existing product satisfies a specification, but not when we seek to implement a specification. The need for assumptions on a solution indicates a failure in creating an appropriate specification.

In contrast, Figure 5 describes the setting where the assumptions are paired with a specification. The assumptions associated with a specification would be part of the contract that the specification identifies. Relying on additional assumptions on a solution, as in Figure 4, risks breaching that contract.

IV. Adopting Ideas from the Philosophy of Science

A scientific theory stakes claims about the way the universe is. For RE, we motivate criteria on evaluability of models from criteria on processes for their construction.

A. Specification Quality

The *verifiability* of a theory means that it makes claims that address an observable part of reality, not metaphysical (e.g., religious) [3]. The *falsifiability* of a theory means that there

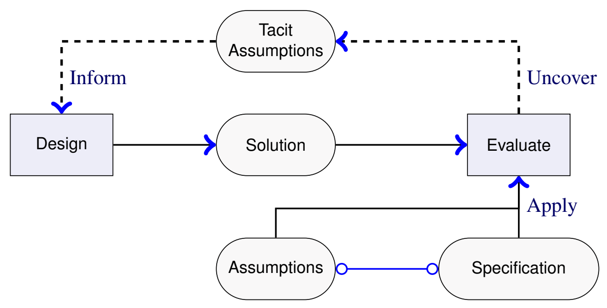

Fig. 5. Shifting the assumptions to the specification is crucial, else the contract with the stakeholders is meaningless; the solution may fail; and different solutions are incomparable.

is a phenomenon observing which would disprove the theory. Otherwise, the theory does not discriminate between possible states of affairs [4]. Both verifiability and falsifiability have been proposed as essential properties of "meaningful" theories.

Although both proposals are controversial, the underlying ideas are applicable to requirements. We posit the above criteria apply to specifications, but because specifications mediate requirements and solutions, the criteria apply in two forms. First, a specification must be such that it should be possible to judge whether a solution meets it: i.e., the specification must be verifiable and falsifiable. For example, S_e is verified by any execution of the elevator system where the elevator is blocked when the load is greater than 680 kg. S_e is falsified by any execution in which the elevator moves even when the load is higher than 680 kg.

Second, a specification must be such that we can determine whether it would meet stated requirements. So just as we asked for specifications, we can ask if the requirements themselves are verifiable and falsifiable. In general, requirements can be understood only through empirical analysis of stakeholders' needs, e.g., via ethnographic studies [5]. Hence, requirements are not conducive to the above criteria. For example, the requirement to "safely" carry at least eight persons is informal. Whether S_e and any solution that implements S_e meets this requirement relies on an interpretation of "safely."

But how then are specifications related to requirements? Specifically, how can we establish the validity of specifications? The validity of specifications is established statically by systematically refining stakeholder requirements into specifications [2]. For example, the requirement to safely carry at least eight passengers was refined into the requirement that the elevator should block when overloaded. An engineer can construct a formal refinement argument to show that the specification meets the requirement. The crux of the argument is that the requirement to carry passengers safely would be met if the elevator does not move when overloaded—which is itself an assumption. The same requirement could be refined differently, e.g., instead of blocking, inform passengers of the reason—under the different assumption that once the passengers are informed, enough of them would step off.

B. *Process Quality*

The *generality* of a theory means it must capture more than the phenomena that are explicitly observed. The *parsimony* of a theory means that it must not introduce concepts and features unnecessarily. *Strong inference* refers to developing theories (specifically, hypotheses) in a manner that their evaluations help discriminate between potential states of affairs [6], [7]. We associate these properties with process since they characterize how a model is constructed.

A specification can be judged for generality based on how the requirements morph and how it tackles them. For example, suppose a new requirement were that the elevator safely ferry passengers *and* crates. Then, an elevator controller built to S_e could be used for that purpose. However, an alternative specification S_c that is based on counting the number of bodies in the elevator would not be able to handle the changed requirement: the controller would not block the elevator when of number of passengers were low enough, but the crates took the weight to above 680 kg. Thus, for the set containing the original and morphed requirements, S_e is more general than S_c. However, for some other requirements, the situation may be reversed. Since we cannot anticipate how requirements may change, we cannot tell if a specification is more general than another unless both support the same requirements but one calls for weaker assumptions than the other. Additional knowledge, such as about what uses an elevator may potentially be put to, would be crucial in determining generality.

A specification can be judged for parsimony based on how few concepts and relationship it involves. For example, an elevator specification that involves assessing the heights of passengers in addition to their weights is not parsimonious. Parsimony involves identifying assumptions so that the requirements are captured succinctly and no component of the specification may be excised without violating a requirement. Achieving parsimony therefore presumes a systematic process of refinement from requirements to specifications. A specification can be general or parsimonious, both, or neither.

Strong inference applies in two respects. First, it corresponds to refining a specification where each small set of components of the specification may be comparatively evaluated with respect to a competing set of components. Doing so facilitates producing a specification that is general and parsimonious. Second, closer to its original sense, strong inference involves ensuring that a specification provides clarity on whether a solution satisfies it.

V. RE AS SCIENCE IN THE SMALL

Both software engineering and science are social activities, occurring in their respective communities of practice who (seek to) establish the correctness of their representations via communication among their members. Unlike natural science, software engineering is a more explicitly sociopolitical activity, wherein the correctness of the representations (requirements, specification, and solution) is negotiable and wherein the specification not only separates requirements from solutions but also provides a basis for a contract.

The foregoing discussion brings out the intuition that RE has to do with the study and creation of small artificial universes. An artificial universe comprises three main artifacts: requirements (the universe to be), assumptions, and specifications (a way to realize the universe to be). Just as natural science mediates between observations and theoretical constructs, requirements engineering mediates between requirements and solutions. However, natural science has little loyalty to commonsense, whereas RE has primary loyalty to stakeholder requirements.

The preceding discussion makes key simplifications. We usually can't evaluate a requirement (or specification) but in terms of the solutions that emerge from it. Moreover, the available solutions often influence the requirements. For example, before Otis, people may not have realized they required an elevator, and skyscrapers may never have been built. Here, we treat requirements as known to the stakeholders. In practice, requirements may be elicited from stakeholders or possibly even sold to stakeholders.

Evaluation is often difficult and expensive. It is imperfect and delayed, thereby loosening or breaking the feedback cycle. People tend to become entrenched in thought and practice—for example, QWERTY keyboards remain the most dominant even though superior text input methods exist.

VI. Conclusions and Directions

The RE lifecycle involves continual negotiation—over requirements and assumptions, specifications, and solutions. Specifically, a negotiator would identify tacit assumptions and ask to make them or their negation explicit. As requirements evolve, so do the associated specifications, usually incrementally. Because stakeholders would become entrenched in certain practices, the process necessarily sacrifices generality and parsimony, leading to suboptimal specifications. But sometimes incremental change to specifications is impossible. In such cases, a rethinking of the solution is necessary, analogous to a paradigm shift in science [8].

Requirements can conflict. The hardest cases of conflict arise when requirements from different universes must be consolidated because the universes abut each other. Coherence across these universes is important even though in general consistency may not be obtained between them. Current RE generally doesn't support building systems of multiple autonomous parties, each as a separate locus of computation [9], [10]. Such an ability is essential in achieving coherence with different universes being modeled differently.

RE needs a renewed focus on verifiable and falsifiable specifications. RE has embraced ideas such as the testability of specifications (by testing solutions) and runtime monitoring of solutions. These ideas intersect with verifiability. However, RE has largely overlooked falsifiability. For example, some efforts on adaptive RE describe adaptive requirements in such a way that they can never be violated. And, strong inference can guide both testing methodologies and work prioritization in a project to uncover specifications with increasing confidence.

Achieving validity and generality of specifications calls for a more complete understanding of the application domain and stakeholder requirements. To this end, RE should leverage various sources of information, e.g., user discussions on the Web and interviews with stakeholders. Semantic approaches for aggregating and querying such information to gain insight into the domain are largely absent. More generally, we need tool-assisted methodologies for mapping information sources to plausible requirements. Then we could ask questions of completeness (do the requirements capture the ideas implicit in the information source?) and validity (are the requirements supported by the information source?)

Systematically refining requirements into specifications is crucial to establishing the validity and parsimony of a specification. Recognizing a refinement as an assumption motivates the question of its validity, which is crucial because a refinement is usually context sensitive and not universal. Existing approaches embed refinements in the process, whereas reifying and representing a refinement is a precursor to expressing it as an assumption.

The sciences seek to understand external phenomena: the phenomena are in broad terms regular, although their regularity is not fixed—consider the continuum from physics to biology to the humanities (psychology and sociology). In contrast, RE must deal not only with the subtleties of the social phenomena in understanding where a solution would fit given stakeholder requirements but must also create a (prescriptive) solution that would realize the required outcome. To rise to this challenge requires cultivating more rigorous methodologies in RE, some facets of which we have outlined above.

Acknowledgments

Singh thanks the US Department of Defense for partial support under the Science of Security Lablet. Chopra thanks the EPSRC for grant EP/N027965/1 (Turtles).

References

[1] H. Simon, *The Sciences of the Artificial*, 3rd ed. Cambridge, Massachusetts: MIT Press, 1996.

[2] P. Zave and M. Jackson, "Four dark corners of requirements engineering," *ACM Transactions on Software Engineering and Methodology*, vol. 6, no. 1, pp. 1–30, Jan. 1997.

[3] A. J. Ayer, *Language, Truth and Logic*, 2nd ed. Mineola, New York: Dover Publications, 1936, reprinted 1972.

[4] K. Popper, *Conjectures and Refutations*. London: Routledge & Kegan Paul, 1963.

[5] H. Sharp, Y. Dittrich, and C. R. B. de Souza, "The role of ethnographic studies in empirical software engineering," *IEEE Transactions on Software Engineering*, vol. 42, no. 8, pp. 786–804, 2016.

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

[7] R. H. Davis, "Strong inference: Rationale or inspiration?" *Perspectives in Biology and Medicine*, vol. 49, no. 2, pp. 238–250, Apr. 2006.

[8] T. S. Kuhn, *The Structure of Scientific Revolutions*. Chicago: University of Chicago Press, 1962.

[9] A. K. Chopra, F. Dalpiaz, F. B. Aydemir, P. Giorgini, J. Mylopoulos, and M. P. Singh, "Protos: Foundations for engineering innovative sociotechnical systems," in *Proceedings of the 22nd IEEE International Requirements Engineering Conference (RE)*. Karlskrona, Sweden: IEEE Computer Society, Aug. 2014, pp. 53–62.

[10] M. P. Singh, "Norms as a basis for governing sociotechnical systems," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 5, no. 1, pp. 21:1–21:23, Dec. 2013.