How Do You Know What You Know: Epistemology in Software Engineering

Oluwatosin Ogundare

Industrial and Systems Engineering, Texas Tech University, Lubbock, USA
Email: tosinogundare@hotmail.com

Abstract

Ubiquitous computing emphasizes the notion of automation in the daily human experience. With the ease, comes the responsibility of knowing, the knowledge of the intrinsic nature of the machine and the evolution of human-computer interaction (HCI). The quest for knowledge is engrained in the act of questioning itself. “How?” “What? “Why?” dominates the vocabulary of every scientist and models the endlessness of our natural inquisitiveness. For example, an interaction of software systems in the case of a user who withdraws money from the ATM and automatically gets a text message and an e-mail containing notification of the transaction, engenders questions about how it all works; in technical terms, the nature of the special science that enables wireless communications. Knowledge that derives from aphorisms or other self-evident truths is easier to acknowledge, for example, the knowledge of “multiplication” is justified by the truthfulness of “addition”—the Apriori. However, in Software Engineering (SE), the Apriori is more obscure. The investigation of the nature of knowledge in SE requires an expansion of the general idea of the Apriori in establishing knowledge.

Keywords

Software Engineering, Epistemology, Ontology

1. Introduction

The delineation of engineering knowledge to emphasize its distinction from scientific knowledge forms the premise of its understanding and highlights its own importance. According to Antonio Dias de Figueiredo, engineering knowledge is multidimensional and every notion it portrays can only be completely understood when analyzed as such [1].

So how is engineering knowledge constructed and what is its nature? Diana Forsythe “an anthropologist studying a scientific community engaged in formu-
“Lating knowledge descriptions” asserts that knowledge in the applied sciences (engineering) is established through acquisition and formalization [2].

“Knowledge Acquisition” according to R. Studer et al. can be realized in one of two ways:

1) Transfer Process
2) Modeling Process

The notion of knowledge construction is more emphasized in the study of Artificial Intelligence (AI) and its implications in complex computer software such as expert or knowledge based systems and systems developed with the goal of reasoning for problem solving. Therefore, epistemology as it relates to the study of nature and validity of knowledge in applied sciences is increasingly more important to engineer more intelligent software to control machines.

2. Examining the Difference in Scientific & Engineering Paradigms

There is a marked distinction between knowledge realized through scientific methods and knowledge established from an engineering process. The question is how are they different? The difference is in the approach. Scientific theories or paradigms are largely based on controlled empirical processes while engineering methods are derived predominantly from the experience of what works. Figure 1 below suggests that while both paradigms begin as a search to the solution to a problem, the steps taken to find an answer is very different [3].

The 10 steps in the engineering paradigm will in fact be an example of knowledge acquired through the “modeling” process.

<table>
<thead>
<tr>
<th>The Scientific Paradigm</th>
<th>The Engineering Paradigm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Problem Definition</td>
<td>1. Problem Definition</td>
</tr>
<tr>
<td>2. Identify Variables</td>
<td>2. Fact Finding</td>
</tr>
<tr>
<td>3. Formulate Hypothesis</td>
<td>3. Establish Parameters</td>
</tr>
<tr>
<td>4. Define Methodology</td>
<td>4. Define Acceptance Criteria(s)</td>
</tr>
<tr>
<td>5. Design Experiment</td>
<td>5. Define System Boundaries</td>
</tr>
<tr>
<td>6. Perform Experiment</td>
<td>6. Prototype Design</td>
</tr>
<tr>
<td>7. Test Hypothesis</td>
<td>7. Build Prototype</td>
</tr>
<tr>
<td>8. Analyze Results</td>
<td>8. Test</td>
</tr>
<tr>
<td>10. Present Results</td>
<td>10. Present Results</td>
</tr>
</tbody>
</table>

Figure 1. Knowledge realized through scientific paradigm vs knowledge established from the engineering paradigm.
Background

“Software engineering is one of the most knowledge-intensive professions. Knowledge and its management are relevant to several aspects of software engineering at different levels, from the strategic or organizational to the technical” [4].

If engineering is largely viewed or broadly considered to be the application of pure “hard” science(s), then it follows that “Software Engineering” can be simply viewed as the application of principles or theories from the discipline of computer science to real world problems. However, IEEE gives the formal definition of “Software Engineering” as follows:

“…It is the application of a systematic, disciplined, quantifiable approach to development, operation, and maintenance of software, that is, the application of engineering to software” [5].

From the definition, the adjectives “systematic” and “quantifiable” implies there must be a set of agreed upon rules by which measurement is carried out. Cuadrado-Gallego et al. suggest that the development of such global quantifiers will be impossible in SE without ontology. In fact, he says that without these formal ontologies, there will be no “shared, consensual conceptualization” in Software Engineering (SE) [6].

3. Ontologies in Software Engineering

Overview

There are different types of Ontologies in Software Engineering, each of them serving different purposes. For example, Reference ontologies, whose main purpose is to eliminate ambiguities in terminology and mitigate the occurrence of what Thomas Kuhn refers as “local incommensurability” [7].

The role of philosophy in creating such standard terminologies is elegantly stated by Nicola Guarino “…philosophy and linguistics play a fundamental role in analyzing the structure of a given reality at a high level of generality and in formulating a clear and rigorous vocabulary” [8].

In contrast however, Application ontologies are not as dense as its counterpart and thus do not require the detail and structure that would involve either Philosophy or Linguistics (at least in a formal sense). Application ontologies can be generally viewed as taxonomies of domain specific vocabulary.

Apart from establishing commensurability, why are ontologies relevant to epistemology in Software Engineering (SE)? According to Plato, knowledge is justifiable true belief, and according to Zelkowitz and Wallace, at least 30% of papers published in SE are not based on empirical science.

Therefore, if knowledge in SE is presented without experimentation, then there are inherent and valid questions about its assumptions and the nature of its truths because it lacks the justification that experimentation provides.

The rationalist might suppose that knowledge presented in this manner (without experimentation) might derive from first principles (self evident truths) thus its validity is in deductive logic; if the premise is true then the derived con-
clusions may be considered such. However, this notion is challenged by Forsythe in her research on the nature of knowledge in Artificial Intelligence (AI), she states that

“Talking with AI professionals, I had been struck by the apparent parallels between the process that they call ‘knowledge acquisition’ … and what anthropologist do in the course of field research … Asked how they went about the task of gathering knowledge for their expert systems, the knowledge engineers I met tended to look surprised and say, ‘We just do it’…” [2].

So how does developing ontologies tie into the quest for the foundation of knowledge? Ontologies in SE formalize the taxonomies which leads into asking the right questions in establishing knowledge based on experience in line with the empiricists. Furthermore, ontologies help establish aphorisms which constitute Apriori truths in SE.

3.1. Reference Ontologies

Reference Ontologies (ROs) focuses on presenting generalized expressions (uses quantified variables in statements in line with First-Order logic) in favor of orthodox propositions (propositional logic) and is grounded in both metaphysical and epistemological philosophical realism with establishing Truth as its theme. However, ROs can be wrong [9].

3.2. Application Ontologies

Application Ontologies (AOs) focuses on reasoning and is localized to a specific computational application and is philosophically grounded in pragmatism (essentially, what we know is whatever works) and assumes metaphysical realism is at least irrelevant because of its basis on the abstract, which lacks merit in real world computational applicability. AOs have a strong methodological emphasis on fidelity i.e. maintaining consistent expressions within the specific computational application [9].

So what then?

It follows from the preceding arguments that the foundation of establishing knowledge in engineering and specifically SE begins with the definition of a formal Ontology. In the case of SE, we must define a Reference Ontology (RO).

4. “Knowing” in Software Engineering

Overview

Having established the perquisite to establishing knowledge as the formalization of ontologies, it is important to understand what can be potentially qualified as “known” and how they can be constructed and validated in SE.

4.1. The Nature of What Can Be Known in SE

Knowledge in SE is primarily acquired through modeling; the directed abstraction of “real world” problem solving without the interference of computational or software constructs or artifacts [10].
Constructing knowledge from modeling Guus seems to be saying, is done at a very conceptual level. This abstraction is grounded in the metaphysical nature of a Reference Ontology (RO), the goal here is not applicability but foundational Truth. Along these lines, the model must maintain internal consistency [11].

4.2. Knowing Models in SE

Knowledge models in SE can be realized as a group of mathematical transformations composing an algorithm mimicking a problem solving “use case” in a specific problem domain; it could also be a logical combination of rules like the “if-else-then” method popular in expert systems. However way the model is realized, it is important that it is grounded in ontology and maintains internal consistency.

4.3. Validating Knowledge Models in SE

Establishing knowledge requires validation according to the operational definition of knowledge as “justified true belief”. Validating knowledge models can be viewed as the means of justifying the model in reality. According to Kathleen Carly, validating these computational models can be assessed at different levels targeting distinct aspects of the model.

She identified at least eight aspects of the model that requires validation, and summarized below is the subset that should apply to all “knowledge models” [12].

“Face Value” validation: This measures how closely the model resembles the actual problem solving method in real world [12].

Parameter Validation: This measures how closely the model parameters fit real world parameters [12].

Process Validation: This measures how closely the process described by the model resembles the actual real world process [12].

Pattern Validation: This measures if the pattern of outputs generated by the model exhibit the same pattern as in the real world [12].

Theoretical Validity: This checks the model for adherence to established theoretical constructs and procedures [12].

5. Conclusions

Guus Schreiber claims that attempts at describing the nature of knowledge are irrelevant to the actual application of knowledge to “problem solving” or inventing new technologies. His justification lies in the perceived inarticulate nature of engineers or physicists when asked to expound a topic they have learned, compared to other scientists, they fall short in providing detail yet they constantly apply these concepts to building faster and more efficient cars etc.

Essentially, he seems to be saying that we do not need to be able to explain knowledge to apply it [13].

In spite of the truth of the aforementioned, without the ability to learn the nature of knowledge, construct and validate it, the limits of artificial intelligence
have been sadly reached; it is only in sound epistemology that the future opens up to endless engineering possibilities.

References


Submit or recommend next manuscript to SCIRP and we will provide best service for you:

Accepting pre-submission inquiries through Email, Facebook, LinkedIn, Twitter, etc.
A wide selection of journals (inclusive of 9 subjects, more than 200 journals)
Providing 24-hour high-quality service
User-friendly online submission system
Fair and swift peer-review system
Efficient typesetting and proofreading procedure
Display of the result of downloads and visits, as well as the number of cited articles
Maximum dissemination of your research work

Submit your manuscript at: http://papersubmission.scirp.org/
Or contact jsea@scirp.org