Epistemological and Ontological Representation in Software Engineering

J. Cuadrado-Gallego[1], D. Rodríguez[1], M. Garre[1], and R. Rejas[2]

[1] Department of Computer Science
University of Alcalá, Madrid, Spain
[2] Department of Computer Science
Francisco de Vitoria University, Madrid, Spain
{jjcg,daniel.rodriguezg,miguel.garre}@uah.es, r.rejas.prof@ufv.es

Abstract. This paper provides an overview of how empirical research can be a valid approach to improve epistemological foundations and ontological representations in Software Engineering (SE). Despite of all the research done in SE, most of the results have not been yet been stated as laws, theories, hypothesis or conjectures, i.e., from an epistemological point of view. This paper explores such facts and advocates that the use of empirical methods can help to improve this situation. Furthermore, it is also imperative for SE experiments to be planned and executed properly in order to be valid epistemologically. Finally, this paper presents some epistemological and ontological results obtained from empirical research in SE.

Keywords: epistemological foundation; ontological representation; SE.

1 Introduction

This paper presents the empirical software engineering (SE) research from a epistemological and ontological point of view, where relevant concepts could be defined as follows:

– Epistemology is the branch of philosophy that studies the origin, nature, and limits of human knowledge [28]. From an epistemological point of view, the problem is *how* knowledge is acquired in the SE domain. Holloway [15] has described how knowledge can be acquired by the following epistemological approaches:
  - Authority-based epistemology states that truth is given to us by an authority, in the case of SE, an human expert [15].
  - Reason claims that what is true is that which can be proven using deductive logic. Reason dictates conditional absolute truth; if the premises on which a valid deductive argument are known to be true, then the conclusion of the argument must also be true [15].

  - Experience claims that what is true is that which can be encountered through the senses. The two variations relevant to this discussion are: (i) *anecdotal experience* yields possible truth; if something happened for one person, it is possible it may happen to others also; (ii) empirical evidence states that truth is that which can be verified. Empirical evidence provides probable truth; if controlled experiments are designed properly and replicated, then it is highly probable that the results accurately describe reality [15].
- Empirical Research is any activity that uses direct or indirect observation as its test of reality. In theoretical research, it is a form of inductive reasoning. It may also be conducted according to hypothetic-deductive procedures [28].
- SE is defined by the IEEE [16] as the application of a systematic, disciplined, quantifiable approach to development, operation, and maintenance of software; that is, the application of engineering to software.
- Formal ontologies are engineered artifacts aimed at representing a shared, consensual conceptualization of the knowledge of a given domain [14].

Therefore, epistemology in SE aims to establish the origins, nature and limits of knowledge in SE. Such knowledge should also be represented in a consensual, shared and optimal way, i.e., an ontology of the SE domain [14]. Taking into account the epistemological approach indicated above, i.e., authority-based, reasoning and experience, when applied to SE techniques, it is possible to claim that:

- SE has advanced for many years by mainly following authority-based epistemologies, i.e., imposed by *expert* opinions or organisations promoting a set of technologies. For example, in [6] there are 25 hypotheses such as *Object model reduces communication problems between analysts and users.* As Holloway states [15], this is a weak epistemological foundation on which to base an entire discipline.
- Deductive logic in SE could be the application of traditional engineering techniques in SE.
- Experimental research has received in recent years many favorable announcements as an important epistemological way for SE.

Another research work in the area includes the one by Aaby [1], which explores the foundation of Software Engineering from the perspectives of ontologies, epistemology and axiology.

The organisation of the paper is as follows. Section 2 provides an overview of empirical research in SE. Section 3 discusses epistemology in in the context of empirical SE research. Section 4 presents the use of ontologies in SE. Finally, Section 5 concludes the paper.

2 Empirical Research in Software Engineering

The use of empirical methods, closer to the scientific research method, consists of observing the world, proposing a model or theory of behaviour, measuring and analysing, validating hypotheses of the model or theory, and if possible, repeating the experiment [13]. The scientific method of research resides in opposition to the *advocacy* research where a researcher "*conceive an idea, analyse the idea, advocate the idea*". Furthermore, researchers highlight the positives aspects of empirical research methods (Fenton et al. [9], etc.).

The use of empirical research methods in SE is small if compared with other disciplines. For example, Zelkowitz and Wallace [29] analysed over 600 papers in the computer science literature and over one hundred papers from other scientific disciplines in order to (i) analyse if the computer science disciple validates theories and (ii) to compare with other disciplines. They found that around 30% of papers did not include experimentation although it was needed it and only the 10% of papers that included experimentation have controlled experimentation methods. Tichy et al [27] also analysed over 400 paper in the computer science literature concluding that 40% of papers did not include the required experimentation and they required empirical validation (compared with only only 10% to 15% of papers in other engineering disciplines).

In the last decades, however, the importance of empirical research in SE has grown considerably as noticed by Zelkowitz and Wallace [29]. As a consequence there are also numerous books, journals and conferences dedicated totally or partially to disseminate the results of empirical research in software engineering.

3 Epistemology in Empirical Software Engineering Research

Important issues in epistemology include the nature of knowledge, its presuppositions and foundations, its extent and validity. This describes what things are known and how that knowledge is acquired. Plato's Theaetetus already defined knowledge as justified true belief. The problem is how to define what we know is true and to what extent that is true. Modern philosophers such as Gettier [12] state that belief does not need to be *fully* justified to be true. Furthermore, Popper states that a law is a hypothesis that has not yet been falsified. For him, a scientific idea can never be proven true, because because no matter how many observations seem to agree with it, it may still be wrong. On the other hand, a single contrary experiment can prove a theory forever false. Theorists have had consensus defining the meaning of laws, theories, hypotheses and conjectures:

- Laws are generalizations on how things happens. From observations we can generalize about what it is expected to happen but do not need to have an explanation of why things happen.
- Theories are explanations of laws, i.e., why things happen.
- Hypothesis is tentative explanation that accounts for a set of facts that needs to be confirmed by observation.
- Conjectures are statements, opinions, or conclusion based on inconclusive or incomplete evidence.

3.1 Epistemology in Software Engineering

Despite the amount of research performed in empirical software engineering, most of the results have not been yet been stated as laws, theories, hypothesis or conjectures, i.e., an epistemological point of view. An exception could be the works of Lehman and Ramil [20]. They discuss why there is a need to develop a theory of software evolution on a large scale instead of analysing microcosms software evolution.

Following this approach, recently, Endres and Rombach [6] compiled a set of laws, hypotheses and conjectures related to Engineering and Information Systems. Although many of these reported laws are validated by *experience* or lessons learned, such formulation can help other researchers to design and create experiments in such a way that proper research questions are formulated. Experiments can then be designed properly to validate or refute hypotheses or theories.

3.2 Problems with Empirical Software Engineering Research

Not all empirical research in SE is been carried out properly and as Lehman states: *"our laws are certainly weaker than those formulated in biological sciences..."*.

In order for Empirical research to improve the epistemological value of SE, hypotheses need to be stated properly. In an initial attempt, Kitchenham et al [18] discuss a set of SE empirical guidelines highlighting the fact that SE empirical research is still pretty poor. In relation to hypotheses, they comment that many hypotheses are *shallow* hypotheses because they do not reflect an explanatory theory, and as a result, those experiments do not increase SE knowledge, i.e, those do not provide value from an epistemological point of view. Also, Fenton et al [9] state 5 questions should be asked for empirical research studies:

– Is it based on empirical evaluation?
– Was the experiment designed correctly?
– Is it based on a toy or real situation?
– Were the measurements used appropriate to the goals of the experiment?
– Was the experiment run for a long enough time?

Problems with empirical research can include the experimental design itself, samples and scope (toy vs. real, just as *software-development-in-the-small* differs from *software-development-in-the-large*, *research-in-the-small* may differ form *research-in-the-large* [9], the use of appropriate measures. An example of the postulates of Kitchenham et al [17] is the necessity to apply the representational theory of measurement to software measurements. Fenton and Pfleeger [10] provide guidelines to define and apply metrics to measure the process or product characteristics. Also, the validity of the metrics from an empirical and theorical point of view has been discussed in the literature. The representation condition [10] asserts that a measurement mapping M must map entities into numbers and empirical relations into numerical relations in such a way that the empirical relations preserve and are preserved by the numerical relations. For example, *if A is taller than B if and only if M (A) > M(B)* [10]. It is a well-known fact that there are many defined SE metrics that do not follow these principles. Examples include the McCalls' complexity metric, which has no unit.

A correct way to establish epistemological foundations of software measurement is that a set of metrics must make clear what aspects of quality they are trying to measure and who they are directed at because there are different points of view about what quality means (developers, managers, users). In the SE domain, metrics should be based on a range of properly defined quality model from fixed hierarchical models (Boehm, McCall's FCM) to more flexible approaches such as the *Goal-Question-Metric* (GQM) [4].

Finally, the empirical evaluation is needed to help evaluate, predict, understand, control and improve the software development process or product [3]. Furthermore, Fenton and Pfleeger [10] argue that conducting empirical evaluations is the only way to improve software processes and products.

3.3 Some Epistemological Results of the Empirical Research Approach to SE

Epistemology can help us define assertions and how those assertions in SE can be verified. Also the concepts of ontologies in Software Engineering are aimed at defining factors and relationships that help to clarify what needs to be reported in empirical studies.Epistemology and ontology foundations can help in conducting research and advancing the SE domain without antinomies that we have at the moment. For example, Lehman and Ramil [20] discuss why there is a need to develop a theory of software evolution, its practical impact, underlying strategies and outlines a strategy for the development of the theory. So far, Lehman is one of the few authors that have defined their work as hypothesis and empirically evaluate them to create laws.

There are also dichotomies in SE, e.g., in Formal Methods domain, Floyd [11] suggests a discussion from a philosophical point of view that can be extrapolated to other SE domains. Formal methods have focused quite a lot in the mathematical aspect. Floyd argues that a change from a product oriented perspective (which regards software as a product) to a process oriented perspective, (which views software in connection with human learning) is needed. In SE, the same dichotomy exits in the same context and others. For example, metrics and quality are usually related either to the products or processes but not much research has be carried out linking both aspects. Other examples where a long term view of empirical research has contradicted previous research include:

– Regarding to estimation Kitchenham states Function Points are not better than Lines of Code (LOC). Also, Dolado [5] has shown that some modern data mining estimation methods are not better than classical regression.
– Shneiderman et al experiment [24] on the same analysed whether an algorithm is easier to comprehend if presented as a flow chart or as a pseudo code. Authors concluded that there is no difference between both. Years

later, Scalan [23] highlights some experimental flaws of the original experiment (not taking time into account, questions that could only be answered from the pseudocode and too simple program). Scalan proved that flowcharts outperform pseudocode with a proper experimental design.
– Etc.

Finally, perhaps the most interesting point for the future of SE from the Popper theses regarding epistemology is that every solution of a problem raises new unsolved problems. Current research in empirical SE is the adoption of evidence-based approach following the success of the evidence-based paradigm in medicine during late 80's and early 90's. By analogy with evidence-based medicine, Kitchenham et al [19] define that the Evidence-based Software Engineering (EBSE) should be "*to provide the means by which current best evidence from research can be integrated with practical experiment and human values in the decision making process regarding the development and maintenance of software*".

4 Ontologies in Software Engineering

In SE, the use of ontologies has been used as a resource to integrate the information, to communicate what people have achieved, to adapt the goals of the organization and to support the efficiency of the processes. Ontologies can also be used by applications require a higher level of formality of definition. For example, the cataloguing of learning resources or the mapping of vocabularies from different information sources require precise definitions, or at least significant characterizations that help in deciding which terms to use in practical situations. Althoff et al [2] describe an architecture oriented to reuse the experience in SE that use ontologies as the underlying formalism.

From an ontological point of view, empirical research has different considerations. On the one hand, the use empirical research with the objective of improving the epistemological foundations in SE could help in defining more precise definitions, which in turn will provide more precise and useful ontologies. The "IEEE Standard Glossary of Software Engineering Terminology" [16] is a well-known attempt to provide precise characterizations of the main terms in the field. It uses natural language prose which is useful for an efficient communication, but it does not provide a clear demarcation for each of the concepts.

On the other hand, current works creating ontologies based on standards will help in defining entities, parameters and the relationships between them. This is turn will help to define the experiments stating which variables could be considered as dependant or independent variables in the design of an experiment and how those are related. An ISO standard, the Guide to the Software Engineering Body of Knowledge (SWEBOK) provides an agreement on the content of what compose the SE discipline. The SWEBOK project opened new possibilities to ontology engineering in the field of SE, since it represents a shared consensus on the contents of the discipline and provide pointers to relevant literature on each of its concepts, both are important elements in ontology engineering [14,25].

5 Conclusions

This paper provided an overview of empirical research from an epistemological and ontological point of view, linking such concepts in the context of SE. This paper also highlighted the fact that there are few laws, theories, hypothesis or conjectures in SE. This fact is not only related to the lack research in this area which has grown considerably in the last decade, but it may be also related to epistemological approximations. The use of empirical methods following the scientific method of research can help to improve this situation.

Empirical research must be properly defined to be valid from the epistemological point of view. There are many antinomies still operating that need to be solved taking into account the established empirical and research methods and the initial guidelines [18] that are appearing in the SE domain. Some epistemological and ontological results were presented highlighting these facts.

Acknowledgement

This work has been supported by the project MCYT TIN 2004-06689-C03.

References

1. Aaby, A.A.: The Philosophical Foundations of Software Engineering, Draft available at: http://cs.wwc.edu/∼aabyan/Articles/SE/.
2. Althoff, K.-D., Birk, A., Hartkopf, S., Muller, W., Nick, M., Surmann, D. and Tautz, C.: Systematic Population, Utilization, and Maintenance of a Repository for Comprehensive Reuse. In Learning Software Organizations - Methodology and Applications, Springer Verlag, LNCS 1756, (2000) 25–50
3. Basili, V., Selby, R.W. and Hutchens, D.H., Experimentation in Software Engineering, *IEEE Trans. on Soft Eng* **12** 7 (1986) 733–743
4. Basili, V. R., Caldiera, G. and Rombach, H. D.: 'The Goal Question Metric Paradigm', in *Encyclopedia of Software Engineering*, John Wiley & Sons, Inc., (1994) 528–532
5. Dolado, J.J. and Fernandez, L.: Genetic Programming, Neural Networks and Linear Regression in Software Project Estimation, in *International Conference on Software Process Improvement, Research, Education and Training* (INSPIRE98) (1998) 157–171
6. Endres, A., Rombach, H.D.: *A Handbook of Software and Systems Engineering: Empirical Observations, Laws and Theories*. Pearson Addison Wesley, (2003)
7. Briand, L., Bunse, C., Daly, J., Differding, C.: An Experimental Comparison of the Maintainability of Object-Oriented and Structured Design Documents, *Empirical Software Engineering* **2**(3), (1997) 291-312
8. Chidamber, S.R. and Kemerer, C.F.: A metric suite for object oriented design, *IEEE Trans. on Soft Eng*, **20**(6) (1994) 476-493
9. Fenton, N.E., Pfleeger, S.L. and Glass, R.L.: Science and Substance: A challenge to Software Engineers. *IEEE Software* **11**(4)(1994) 86–95
10. Fenton, N.E. and Pfleeger, S.L.: *Software Metrics: A Rigorous and Practical Approach, 2nd Edition*, PWS (1997)

11. Floyd, C.: Theory and Practice of Software Development, Stages in a Dialoge. LNCS Vol. 915, 6th International Joint Conference CAAP/FASE on Theory and Practice of Software Development, (1995) 25–41
12. Gettier, E.: Is Justified True Belief Knowledge?, Analysis **23**. Available at: http://www.ditext.com/gettier/gettier.html
13. Glass, R.L.: The Software-Research Crisis. IEEE Software **11**(6) (1994) 42–47
14. Gruber, T.: Towards principles for the design of ontologies used for knowledge sharing. Intl Journal of Human-Computer Studies **43**(5/6) (1995) 907–928
15. Holloway, C.M., Epistemology, Software Engineering, and Formal Methods Abstract of Presentation The Role of Computers in LaRC R&D, June 15-16 (1994). Available at: http://shemesh.larc.nasa.gov/people/cmh/epsefm-tcabst.html
16. IEEE, IEEE Standard Glossary of Software Engineering Terminology, IEEE Std 610.12-1990, (1990)
17. Kitchenham, B., Pfleeger L., Fenton, N.: Towards a Framework for Software Measurement Validation. IEEE Trans on Soft Eng **21** (12) (1995) 929–944
18. Kitchenham, B.A., Pfleeger, S.L., Pickard, L.M., Jones, P.W., Hoaglin, D.C., El Emam, K., Rosenberg, J.: Preliminary Guidelines for Empirical Research in Software Engineering. IEEE Trans. on Soft Eng **28**(8) (2002) 721-734,
19. Kitchenham, B.A., Dyba, T., Jorgensen, M.: Evidence-Based Software Engineering. 26th IEEE International Conference on Software Engineering (ICSE'04) (2004) 273–281
20. Lehman, M., Ramil, J.F.: Towards a Theory of Software Evolution and its Practical Impact, Proceedings Intl. Symposium on Principles of Software Evolution, ISPSE 2000, 1-2 Nov, Kanazawa, Japan (2000) 2–11
21. Lenat, D.B.: Cyc: A Large-Scale Investment in Knowledge Infrastructure. Communications of the ACM **38**(11) (1995) 33–38
22. Popper, K.R.: Conjectures and Refutations, Routledge and Kegan Paul, (1963)
23. Scanlan, D.A.: Structured Flowcharts Outperform Pseudocode: An Experimental Comparison. IEEE Software **6**(5) (1989) 28–36
24. Shneiderman, B.B., Mayer, R., McKay, D., Heller, P.: Experimental Investigations of the Utility of Derailed Flow charts in Programming. Communications of the ACM **20**(6) (1977) 373–38
25. Sicilia, M.A., Garcia, E., Aedo, I., Diaz, P.: A literature-based approach to annotation and browsing of Web resources. Information Research **8**(2), (2003) 1–10
26. Tautz, C. and von Wangenheim, C.G.: REFSENO: A Representation Formalism for Software Engineering Ontologies, Fraunhofer IESE IESE-015.98 (1998)
27. Tichy, W.F., Lucowiicz, L.,Prechelt, L., Heinz, E.A.: Experimental evaluation in computer science: a quantitative study, Journal of Systems and Software **28**(1) (1995) 9–18
28. Wikipedia: The Free Encyclopaedia. Available: http://en.wikipedia.org/wiki/Epistemology
29. Zelkowitz M.V., Wallace D.: Experimental Validation in Software Engineering. Information and Software Technology **39**(11) (1997) 735–743