The Epistemology of Validation and Verification Testing

T S E Maibaum

Department of Computing and Software
McMaster University
tom@maibaum.org
(original version co-authored with the late AM Haeberer and with the assistance of MV Cengarle, then of Institut für Informatik
Ludwig-Maximilians-Universität München)

Abstract. We wish to be able to give formal definitions (in the sense of science or engineering) for concepts like requirements validation and for the relationship between a requirements specification and an abstract design of the intended system. Ditto validation of designs and the final executable application with respect to the original "application concept", on the one hand, and the requirement specification, on the other. We have been developing a framework based on the work of the logical empiricists and other analytic philosophers over the last 80 years to support our understanding of software engineering concepts. Recent developments (dating from the 80s)in the area of "confirmation" (of a hypothesis concerning a theory by some (experimental) evidence) promises to illuminate some of these problematic concepts. In this talk we address the problem of establishing the very relation between requirement specifications and scenarios, as used, for example, in UML. The same framework can also be applied to the problem of testing implementations against designs, so called verification testing.

1. Introduction

Requirements engineering (RE) is a black art! We are forever confronted by the assertion that, whilst requirements specifications may be a formal entity, analysable even in a mathematical sense, it is informally related to an informal "entity", the so-called *application concept.* If we cannot define precisely (and meaningfully) the statement "this scenario confirms (or discomfirms) this behaviour specification", then how can we pretend we know what a behaviour specification (and therefore a requirements specification) specifies? Suppose further that we are interested in questions such as the following: Is requirements language X better than Language Y for defining the requirements of applications of class W? On what basis can we justify the fact that we like the work reported in [21,32,,22,23] and that it says something important about requirements engineering?

On what basis can we answer these questions so that the answers can be justified on a "scientific" or "engineering" basis? If we cannot answer the first question, how can we begin to address the others? If some entities and relationships are informal, what is there left aside from anecdote to support requirements "meta-analysis"? The purpose of this talk is to demonstrate that a *framework* can be defined, turning the "informal" entities and relationships of the above discussion into well defined concepts that are amenable to formal analysis.

2. Gedanken experiments, requirement specifications and confirmation

In former papers [16,17, 6,7, 8] we have endeavoured to lay the basis for the epistemological analysis of software engineering. In [17], we analyse superficially the relationships among the various objects in a metamodel of the software process we posited (called W) and which is reproduced in Figure 1.

At the leftmost lower corner of this figure we see the factual[1] relation *HPS*+*CTXP* ☞ *RSP*, where *HPS* is what we called the hypothetical posit of the intended software artifact *EA* in [17], *CTXP* its context, and *RSP* the requirements specification for *EA*. In [17] we claim that the relation ☞, whose analysis is the purpose of this talk, is what we called there a *quasi merotic explanation*.

To be able to study formally the leftmost lower hollow arrow in Figure 1, which is nothing but the notorious process of *ab initio* requirements elicitation, we need a framework in which we can reason about the nature of this process, about the objects *HPS*, *CTXP*, and *RSP*, and about the relation ☞. It is the purpose oft his talk to analyse the nature of this relation and, in order to do that, to establish an adequate framework for reasoning about it and the objects involved. It is very important to bear in mind that, in its present state, the Ω meta-model is idealised in various ways; one of them is that we are considering *ab initio* development, meaning that we are not considering legacy artifacts. This means that we consider requirement specifications as being elicited from (hypothetical) scenarios in which there are no legacy software artifacts or systems, and therefore, this eliciation process does not involve design recovery. Moreover, we will assume that there is no existing software artifact or system from which we can glean a single clue about decomposition; thus, merotic explanations[2] are inhabitants of our post-requirements world.

[1] We classify the relationships among the objects in the Ω meta-model into factual and logical, as we have done in [H&M00]. We give an exact definition of these clasificatory terms below.

[2] An exact definition of what a merotic explanation is can be found in [17]. Informally speaking, we can consider that the structure of the resulting software artifact *EA* in a software process satisfying the Ω meta-model (or any of the series of increasingly reified design specifications SP_i) provides an

The process we have in mind for devising a new engineering artifact is as follows. See the figure at the end of this extended abstract. We have a somewhat vague requirement (called *protoRSP*) for a new artifact to be, *EA*, which at this point is merely a hypothetical posit *HPS*. This vague requirement *protoRSP* is actually a set of properties we know (or we desire) the artifact to be (represented here by the hypothetical posit) should exhibit. These properties are of two kinds, i.e., abstract (theoretical) ones, such as, for instance, behaviours, and their observable counterparts, i.e., sets of observable instances of them, which we will call evidence *E* (e.g., scenarios). In our engineering setting, evidence is produced both by the operation of an engineering artifact *EA* (after its construction!), or by the operation of an hypothetical posit *HPS,* as in a gedanken experiment. The sets of such evidence are part of what we are calling here the context *CTXA* of the engineering artifact *EA*, or the context *CTXP* of the hypothetical posit, respectively. Then, we construct an extension of the language belonging to our underlying science/technology and to the already existing engineering discipline with the necessary symbols, etc., to enable us to state precisely the requirements specification *RSP*, and to show how evidence produced by the operation of the hypothetical posit *HPS confirms RSP*, and that (if the status of our current technology makes its construction viable) the resulting engineering artifact *EA* will be in a certain relation with *HPS* that enables us to expect that *EA* will also produce evidence that *confirms RSP*. (As we argue in [17], this relation is a positive analogy, i.e., *HPS* 👍 *EA*. See the figure.) Then, we construct *EA*, through a process of design and reification, which adds design and realisation detail to the above extension.

If, on the one hand, *protoRSP* is a description in everyday language, or in a previous stage in the development of our scientific/technological language, of the evidence produced by *HPS* and, on the other hand, *RSP* is the exact description of the behaviour confirmed by evidence produced by *HPS* or *EA*, then we may be tempted to characterise *protoRSP* as what Carnap [4] calls an *explicandum* and the corresponding requirement specification *RSP* as its *explicatum*, both related by an *explication*[3]. As Carnap stated [4] *"the task of explication consists in transforming a given more or less inexact concept into an exact one or, rather, in replacing the first by the second. We call the given concept (or the term used for it) the explicandum, and the exact concept proposed to take the place of the first (or the term proposed for it) the explicatum."*

The analysis of the reason why we said that this is a simplistic viewpoint will introduce the core points of this talk. Notice that we had distinguished between, on the one hand, evidence, which is observable (perhaps with the aid of certain apparata), such as scenarios, and, on the other hand, certain abstract (mathematical) objects, such as

---

explanation of why the requirements specification predicts (and retrodicts) correctly its operation. Since this explanation is composed from parts following *EA*'s (or SP_i's) structure, it was called "merotic".

[3] Notice that in this context (i.e., that of the Philosophy of Science) *explication* and *explanation* are not synonymous; we are using *explication* in the particular sense we are discussing, and *explanation* in the sense of *scientific explanation*.

behaviours. These abstract properties are of a very dangerous kind, because if we become overenthusiastic in their introduction, we can obtain a complete zoo of scientifically useless abstractions, such as, for instance, phlogiston, vital force, or entelechy. (Software engineering, as all the novel disciplines whose corpus is not well defined, is especially prone to accept such useless abstractions.) However, *mass* in physics is one of these concepts (as is *force*); *mass* is needed to state Newton's principle for relating force with acceleration. Otherwise, Newtonian mechanics cannot be developed, or even stated. If we look to current scientific language, even that familiar to laymen, we find many abstract terms denoting abstract objects or properties, such as, for instance, *gene*, *electron*, *magnetic field*, *preservation of the angular momentum*, or *esprit de corps*. For instance, some of Kepler's laws can be stated in a language the designata of whose nouns would be accepted by everyone as observables. However, this is not the case with Newtoninan Dynamics; terms such as *angular momentum*, *gravitational field*, and *universal gravitational constant* have non observable designata. Notwithstanding, nobody will say that these terms are useless; without them Newtonian Mechanics is unthinkable. The difference between Kepler's laws and Newtonian mechanics is the difference between empirical generalisations and scientific theories. This difference resides in their respective predictive powers; from Kepler's laws we can infer the movements and positions of the planets, whilst from Newtonian Mechanics we can infer the same but also particular laws, such as Kepler's laws. Unfortunately, it seems that the existence of such terms (nouns) with non-observable designata, is a must if we want an expressive scientific theory, or a statement belonging to a scientific theory, and not an empirical generalisation.

However, an exaggerated use of theoretical terms leads us down the path to metaphysics, so Occam's razor comes into play. In our case, we have evidence, as for instance the collection of behavioural data *hypothetically* generated by an *hypothetical posit*, which can be stated in a language the designata of whose nouns are observable, and we have abstract objects, such as behaviours, which do not designate observable things, but from which we can infer hypotheses potentially confirmable by evidence (e.g., scenarios in UML). Thus, the vocabulary of the language whose nouns designate observable things and properties is smaller than the vocabulary of the language whose nouns designate representatives of these observable things plus abstract things and properties. Furthermore, the restriction of *observability*[4] of the former language makes wider the difference between the two languages, for it is obvious that universal quantifiers in the former must be finite, i.e., equivalent to generalised finite conjunctions (neither our senses nor any physical instrument enables us to observe a whole from infinitely many parts), whilst those of the latter language can be, and are usually, infinite. Moreover, we

[4] We are using *observable*, *observability*, and *abstract*, without giving a precise definition of what we are referring to. We will give precise definitions for them, actually for their exact counterparts, which will have the same spelling but which will actually be different terms with exact meanings, i.e., *designata*.

can have in the latter language modalities, such as *permission* and *obligation*, and *temporal quantifiers*, such as *forever*, *once*, and *sometime in the future*.

The principal problem is, in Clark Glymour's words [12], "How can evidence stated in one language confirm hypotheses stated in a language that outstrips the first? How can one make an inference from statements in the narrower language to statements in the broader language? The hypotheses of the broader language cannot be confirmed by their instances, for the evidence, if framed in the narrower tongue, provides none. Consistency with the evidence is insufficient, for an infinity of incompatible hypotheses may obviously be consistent with the evidence, and the same is true if it is required that the hypotheses logically entail the evidence. The structure of the problem is: what relations between [...] observation statements, on the one hand, and statements [...] about unobservable things or unobservable properties, on the other hand, permit statements of the former kind to confirm statements of the latter kind?".

From what we have said above, it seems plausible to say that the relation ☞ is one of confirmation between the evidence produced by *HPS,* on the one hand, and *RSP,* on the other. As a first approximation we can state the following:

Definition. *Evidence E confirms RSP iff we can use some hypotheses deduced from RSP to deduce from E other hypotheses deducible from RSP*.

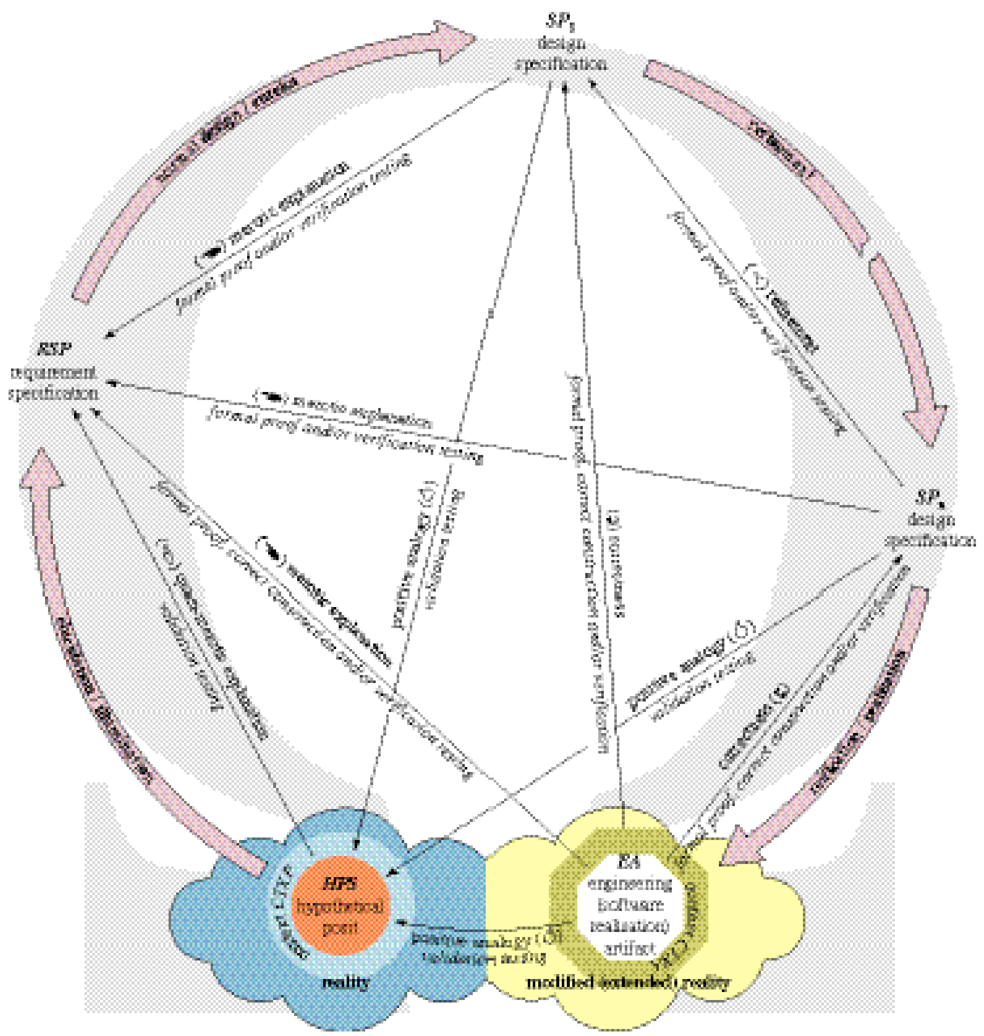
This idea about the mechanism by which we can decide if a theory agrees or disagrees with a piece of evidence (observable) was first conceived by Carnap [5] and later explored and developed by Clark Glymour [12]. Let us call the former hypotheses in the above definition, *bootstrap hypotheses*; thus, our definition can be re-stated as: *evidence E confirms RSP iff we can deduce from RSP a set of bootstrap subtheories of RSP which enable the deduction from E of other hypotheses deducible from RSP*. It is exactly in the conditions established for the deduction of bootstrap *subtheories* where, for instance, the necessary application of Occam's razor we had talked about above must be embedded. Such requirements are the source of the complexity of the confirmation procedure (the so-called bootstrap strategy of confirmation) we introduce below in the talk.

In discussing confirmation, we must here make something very clear. We need to separate carefully two different issues. The frst is the mechanism by means of which we can decide that a certain piece of evidence "agrees" or "disagrees" with a given theory. We will talk below of two of them: one is that succinctly presented in the discussion that led to the definition above and the other is the notorious and flawed *hypothetico-deductive method* (of Newton and others). The second is the criterion of confirmation. We can informally explain this issue by contrasting some of the proposed criteria. One, which we will call Popperian falsifiability (also used in the hypothetico-deductive method), is: if the evidence disagrees with the theory (we need some mechanism for deciding this, i.e., the first issue), then the theory should be discarded; conversely, if the evidence agrees with the theory, then we do not have any new information about the appropriateness of the theory for describing the phenomenon producing the evidence. Another criterion, advanced by Lakatos, says that a theory is something resulting from a difficult and expensive process and, therefore, nobody is willing to discard it because of a mere disagreement with a piece of evidence; so an auxilliary hypothesis is created to explain the disagreement. Finally, the Carnapian logical measure function [4] presents a criterion of confirmation based on degrees of confirmation: if the evidence disagrees with the theory, one can blame the theory or certain auxilliary hypotheses about the experimental method producing the evidence, the measurement instruments, etc. But actually, as in the Popperian case, we blame something, often the theory itself. The main difference between the Carnapian criterion of confirmation and Popperian falsifiability is about what we do when the theory agrees with the evidence. Here, instead of saying that we do not have more information about the appropriateness of the theory, we will say that the degree of confirmation of this theory is greater than the degree of confirmation of a theory not agreeing with this piece of evidence. Carnap associates with this degree of confirmation a logical function (which he calls Logical Probability [4]). This logical function is strongly related with Carnap's inductive logic (and today with theories about belief revision).

In this talk we will deal only with the first issue, i.e., how we can decide that a requirements specification *RSP* agrees or disagrees with a piece of evidence hypothetically produced by the hypothetical posit *HPS*. The second issue will be treated in a forthcoming paper, since if we adopt the Carnapian logical measure function, we should inspect also Carnap's inductive logic and his "continuum of inductive methods", which will bring us closer to the issue of requirements elicitation, and, therefore, to the leftmost lower hollow arrow in Figure 1. However, to be able to produce an effective setting for this talk, we need to append to the so-called bootstrap mechanism, which deals with the first issue above, some kind of confirmation criterion. We will use a not very complicated one, which is a modification of one put forward by Hempel.

3. References

1. Rudolf Carnap, *On Inductive Logic*. Philosophy of Science, Vol. 12, 72-97. 1945.
2. Rudolf Carnap, *Continuum of Inductive Methods*. Univ. of Chicago Press. 1952
3. Rudolf Carnap, *Meaning and Necessity*. Supplement A, *Empiricism, Semantics, and Ontology.* Midway Reprint Edition. 1988.
4. Rudolf Carnap, Logical Foundations of Probability. The University of Chicago Press. Second Edition. 1962.
5.Rudolf Carnap, *An Introduction to the Philosophy of Science* (re-edited from Philosophical Foundations of Physics, Basic Books, 1966). Ed. Martin Gardner, Dover Publications, Inc. 1995.
6. Maria V. Cengarle and Armando Haeberer, *Towards an epistemology-based methodology for verification and validation testing*. See http://www. informatik.uni-muenchen.de under M.V. Cengarle. 1999.
7. María V. Cengarle and Armando M. Haeberer, *Specifications, programs, and confirmation*. Proceedings of the Workshop on Requirements, Design, Correct Construction, And Verification:


SP_1
design specification
RSP
requirement specification
SP_n
design specification
formal proof and/or verification testing
refinement
positive analogy
validation testing
HPS
hypothetical point
EA
engineering (software realisation) artifact
reality
modified (extended) reality

Mind The Gaps! F.A.S.T. Gesellschaft für angewandte Softwaretechnologie mbH- Munich April 2000. http://www.fast.de

8. Cengarle, M V., Haeberer, A. M.. *A formal approach to specification-based black-box testing*. Proceedings of the Workshop on Modelling Software System Structures in a fastly moving scenario. June 13-16, 2000. Santa Margherita Ligure, Italia. www.disi.unige.it/person/FerrandoE/MSSSworkshop.
9. David Christensen, *Glymour on evidential relevance*, Philosophy of Science. Vol 50. 471-481, 1983.
10. John Earman and Clark Glymour, What Revisions does Bootstrap Testing Need? A Reply. Philosophy of Science. Vol. 55. 261-264, 1988.

11. Clark Glymour, *Hypothetico-deductivism is Hopeless*. Philosophy of Science. Vol. 47. 322-325, 1980.
12. Clark Glymour, *Theory and Evidence*. Princeton Univ. Press. 1980.
13. Clark Glymour, *On testing and evidence. In John Earman ed. Testing Scientific Theories*, Minnesota Studies in the Philosophy of Science, Vol. X. Univesity of Minnesota Press. 1983.
14. Clark Glymour, *Revisions of bootstrap testing*, Philosophy of Science. Vol 50. 626-629, 1983.
15. Carl G. Hempel, International Encyclopedia of Unified Science, Vol. 2, No. 7: *Fundamentals of Concept Formation in Empirical Science*. University of Chicago Press. 23-38, 1952.
16. Armando M. Haeberer and Tom S. E. Maibaum, *The very idea of software development environments: a conceptual architecture for the arts environment.* In B. Nuseibeh and D. Redmiles, eds. Proc. of 13th IEEE Int. Conf. on Automated Software Engineering (ASE-98), IEEE CS Press, 260–269. 1998.
17. Armando M. Haeberer and Tom S. E. Maibaum, *Scientific rigour, an answer to a pragmatic question: a linguistic framework for software engineering*, to appear in Proc. of ICSE2001, 23rd Int. Conf. on Software Engineering. Toronto 2001.
18. Mary Hesse. *The Structure of Scientific Inference*. University of California Press. 1974.
19. Jako Hintikka, *Towards a Theory of Inductive Generalization*. In Y. Bar-Hillel, Proc. of the 1964 Congress for Logic, Methodology, and the Philosophy of Science. 274-288. Stanford University Press. 1962.
20. Michael Jackson, *Formal Methods and Traditional Engineering*. Journal of Systems and Software special issue on Formal Methods Technology Transfer. Vol. 40. 191-194. 1998.
21. Michael A. Jackson and Pamela Zave, *Deriving Specifications from requirements: an Example*. Proc. ICSE'95 - 17th International Conference on SE. IEEE Computer Society Press, 15-24. 1995.
22. Axel van Lamsweerde and Emmanuel Letier, *Handling Obstacles in Goal-driven Requirements Engineering*. IEEE Transactions on SE, Vol. 26, September 2000.
23. Axel van Lamsweerde and L. Willemet, *Inferring Declarative Requirements Specifications from Operational Scenarios*. IEEE Transactions on SE, Vol. 24, No. 12, 1089-1114. December 1998.
24. Tom S.E. Maibaum, *Mathematical Foundations of Software Engineering: a roadmap*. In Eds. A. Finkelstein and J. Kramer, Future of Software Engineering, ICSE 2000. IEEE C.S. Press. 2000.
25. E. Nagel, *The Structure of Science*. Harcourt, Brace. 1961.
26. Bryan G. Norton, *Linguistic Frameworks and Ontology*: A Re-Examination of Carnap's Metaphilosophy. Mouton Publishers. 1977.
27. Karl R Popper, *The Logic of Scientific Discovery*. Hutchinson, London 1968.
28. Herbert Simon, *The axiomatization of physical theories*. Philosophy of Science. Vol 37. 16-26, 1970.
29. Wlad M. Turski, *An Essay on Software Engineering at the Turn of Century*. In Ed. Tom Maibaum, Proc.of Fundamental Approaches to Software Engineering 2000, LNCS 1783, Springer. 2000.
30. Wlad M. Turski and Tom S.E. Maibaum, *The Specification of Computer Programs*. Addison-Wesley, 1987.
31. Walter G. Vincenti, What Engineers Know and How They Know It : *Analytical Studies from Aeronautical History*. Johns Hopkins U. Press. 1993.
32. Pamela Zave and Michael A. Jackson, *Four Dark Corners of Requirements Engineering*. ACM Tansactions on SE and Methodology, Vol. 6, No. 1. 1-30. 1997.
33. Jan M. Zytkow, *What revisions does bootstrap testing need?*. Philosophy of Science. Vol 53. 101-109, 1986.