Requirements Eng (2005) 10: 304–306
DOI 10.1007/s00766-005-0013-6

VIEWPOINTS

Roel Wieringa

Requirements researchers: are we really doing research?

Received: 20 September 2004 / Accepted: 03 March 2005 / Published online: 5 August 2005
© Springer-Verlag London Limited

1 Introduction

A few years ago, Davis and Hickey (2002) suggested that a reason why the results of requirements engineering research are not used in practice is that requirements engineering researchers do not practice what they preach: they do not analyze the problems of requirements engineering practice, and therefore their solutions do not address these problems. Although I agree with this diagnosis, I think it must be taken one step further in order to achieve a vision of a solution. The additional step that I propose here is to realize that currently, most requirements engineering researchers do not do research. Many of us create unvalidated designs, and move on from one unvalidated design to the other. The remedy that I propose is that we should learn to distinguish design from research, and start doing research. What we should do research about is the engineering process, so let me first explain what I mean by "engineering".

2 Requirements engineering is not engineering

What is engineering? The most succinct definition is given by Auyang (2004): "engineering is the art and science of production." Slightly more elaborate is the *Encyclopædia Britannica*, which defines engineering as "the application of science to the optimum conversion of the resources of nature to the uses of humankind." Let me adopt these definitions in their widest possible interpretation: engineering is the production of useful things, whether these things are material, symbolic (e.g. software), or social (e.g. organizations).

If engineering is the production of useful things, then it is the process of solving *practical problems*, which I define as problems in which we want to improve something in the world. A practical problem may exist if there is a difference between the way the world currently is and the way we would like it to be, and we are motivated to reduce this difference (Bossworth 1995, pp. 33–49). A practical problem can be solved in a problem-driven way, when stakeholders experience the problem and commission an engineer to help them solve it. It can also be solved in a goal-driven way, when no one experiences a problem but someone sees an opportunity to improve the situation nevertheless, by means of new technology. Innovative technology often does not solve a problem anyone experienced, but once the technology is available, many people want to have it anyway. Either way, the improvement process consists of the following logical tasks, called the *engineering cycle*.

- *Problem analysis*. Investigation of the problem to be solved, and of the goals to be reached.
- *Solution specification*. Description of one or more possible solutions.
- *Solution analysis*. Investigation of the properties of the specified solutions, and evaluation of the power of these solutions to solve the problems and achieve the desired goals.
- *Solution implementation*. Implementation of a selected solution.
- *Implementation analysis*. Investigation of the properties of the solution as implemented. This might be the problem analysis task in another engineering cycle.

There is nothing new about this list of engineering tasks. Variations of this cycle have been identified by various engineering methodologists, including Roozenburg and Eekels (1995), Pahl and Beitz (1986), and Cross (1994). The first important point to remark here is that this is not necessarily a temporal sequence. The tasks of the engineering cycle may all be performed concurrently, or interleaved in any order. A second remark to make is that there are three analysis tasks in the engineering cycle: problem analysis, solution analysis, and implementation analysis. In each of these tasks, the engineer must acquire knowledge. Note in particular that the investigation of proposed solutions is research, and can be done using scientific research methods. Engineers investigate structural properties of matter, dynamic properties of fluid flows, properties of chemical pro-

R. Wieringa
Department of Computer Science, University of Twente,
The Netherlands
E-mail: r.j.wieringa@ewi.utwente.nl

cesses, and properties of many other solution techniques, by the same scientific methods as those used by natural scientists. The other two tasks in the engineering cycle are not research: solution specification is a creative task in which a solution to a practical problem is specified, and solution implementation is an activity in which the selected solution is realized.

There are two schools of thought about requirements. According to one school, requirements are solution specifications. For example, the IEEE 830 standard for requirements specification prescribes a structure for the description of software solutions. According to another school, represented by Jackson (1995) for example, and Van Lamsweerde (2004), requirements are problem descriptions. Where solution-oriented requirements consist of functions and quality attributes of a desired solution, problem-oriented requirements consist of problematic phenomena and about goals to be achieved. I do not want to enter a discussion here about the "true" meaning of the word "requirement". Instead, let me point out that in either view, requirements engineering is not engineering:

– In the problem-oriented view, requirements are a theory about a practical problem. A theory is a set of validated propositions about reality; a problem theory consists of a set of propositions describing, for example, what the relevant phenomena are, what their causal relationships are, what the relevant goals and norms are, etc. Whatever the precise contents of a problem theory, problem-oriented requirements "engineering" is actually theory-building, not engineering. Problem-oriented requirements "engineering" is not engineering because it does not attempt to change reality but to understand the problem.
– In the solution-oriented view, requirements are a specification of a solution. A solution specification is a description of what we are going to implement, and creating such a specification is a design activity, not in the narrow sense of creating the internal decomposition of a software system, but in the general sense of inventing a solution to a practical problem. A solution specification is designed, in the same sense as a law can be designed, an organization can be designed, and a symphony can be composed. If requirements "engineering" is the creation of a solution specification, then it is a design activity, which is only one activity in the complete engineering cycle.

3 Requirements engineering research

If requirements are either problem theories or solution specifications, then what is requirements engineering research? Requirements engineering researchers should be engineers of the engineering cycle. Their aim should be to improve the problem analysis and solution specification tasks of the engineering cycle—the two requirements engineering tasks of the engineer. Consequently, the engineering cycle of requirements engineering researchers themselves is this:

– *Problem analysis*. Investigate the requirements problems that engineers have, and the goals to be achieved by requirements engineers.
– *Solution specification*. Describe one or more possible solutions to the identified problems. This is a creative task.
– *Solution analysis*. Investigate the properties of the specified solutions, and evaluate their power to solve the identified problems and achieve the desired goals.
– *Solution implementation*. Implement a solution.
– *Implementation analysis*. Investigate the properties of the implemented requirements solution.

This gives us three kinds of research problems to be investigated by requirements researchers: problem analysis, solution analysis, and implementation analysis. Since every practical problem will already involve the use of solutions implemented earlier, it will be difficult in practice to distinguish problem analysis and implementation analysis. But the issue here is not how many types of requirements research there are, but whether there is any requirements research to do at all. For example, to investigate the problems of requirements engineering, requirements research must investigate the problems as they exist in the real world, as Davis and Hickey (2002) pointed out. I add to this requirements researchers should also investigate the properties of the solutions they propose, and of the implementations of those solutions in practice. Inventing the solutions is not research but design. Requirements research consists of investigating problems, investigating solution proposals, and investigating implementations in requirements practice. There seems to be an identity crisis in which we refer to our design activity as "research" and are then left without a word for real research.

In an analysis of submissions to the International Requirements Engineering Conference (2003), my colleague Hans Heerkens and I (2004) found that about 30% of the submissions (accepted or not) describe problems or implementations. This is encouraging, although many of these studies are narrative, first-person reports that do not follow an accepted methodologically sound research design. It is revealing what the remaining 70% of the submissions (accepted or not) were about: they consist of solution proposals. 18% of these had some form of validation other than that the author illustrated his or her solution by means of an example, or asked students to use the technique. However, among these few validations, thorough investigations of solution proposals using accepted research methods are very rare. Apparently, many of us prefer creating designs to doing research.

4 There is nothing as practical as a good theory

Nineteenth-century anthropologists "studied" cultures from their armchair by imagining what it would be like to live in the cultures they had read about. Validation consisted of writing bulky books about these exotic cultures (Frazer 1992). Similarly, some of us practice "armchair engineering'' where we propose solutions for practical problems that we never encountered and do not bother to validate those solutions—I am not excluding myself from this judgment.

What we should do to make our results applicable is, first of all, stop calling design activities ''research''. Designing is a fascinating and potentially useful activity, and we need to publish our designs in order for ourselves and others to validate them. But designing is not the only thing we should do. We should start doing research: investigate problems in the requirements process, investigate solution properties, and investigate implementations of these solutions. This will yield us theories about problems in requirements practice, about techniques that could solve those problems, and about implementations of those techniques in requirements practice, that will be useful to practitioners. We should heed the words of Abraham Kaplan (1998, p. 295), who said the following.

> ''The criticism that a plan of action is 'all-right in theory but it won't work in practice' may well be a just one, but it must be properly understood. The theory may specify conditions which are not fulfilled in the particular case before us; the criticism then amounts to saying that the proposal is a good solution, but to another problem.''

Scientific theories of problems, solutions and implementations of requirements practice will be the means by which we can make clear when and where our solution designs are applicable in practice. We do not have to look far to find advice on how to conduct our research. Software engineering research has been observed to lack in validation as well (Tichy et al. 1997; Zelkowitzh and Wallace 1997), and some recent papers give good advice about how to do software engineering research (Zelkowitz 1998; Kitchenham et al. 2002). I see no reason why requirements researchers could not use this advice to their advantage as well.

References

Auyang SY (2004) Engineering—an endless frontier. Harvard University Press, Cambridge

Bossworth MI (1995) Solution selling: creating buyers in difficult markets. Irwin, IL

Cross N (1994) Engineering design methods: strategies for product design, 2nd edn. Wiley, New York

Dardenne A, van Lamsweerde A, Fickas S (1993) Goal-directed requirements acquisition. Sci Comput Program 20:3–50

Davis AM, Hickey AM (2002) requirements researchers: do we practice what we preach? Require Eng J 7(2):107–111

Frazer JG (1922) The golden bough. MacMillan, New York

Jackson MA (1995) Software requirements and specifications: a lexicon of practice, principles and prejudices. Addison-Wesley, Reading

Kaplan A (1998) The conduct of inquiry. Methodology for behavioral science. Transaction publisher, 1998. First edition 1964 by Chandler Publishers, San Francisco

B.A. Kitchenham, S.L. Pfleeger, D.C. Hoaglin, K.E. Emam, J. Rosenberg. ''Preliminary guidelines for empirical research in software engineering''. IEEE Transactions on Software Engineering. 28(8), August 2002. Pages 721–733.

van Lamsweerde A (2004) Goal-oriented requirements engineering: a roundtrip from research to practice. In: Proceedings of 12th IEEE joint international requirements engineering conference, IEEE Computer Science Press, pp 4–8

Pahl G, Beitz W (1986) Konstruktionslehre: Handbuch für Studium und Praxis. Springer, Berlin Heidelberg New York

Proceedings of the 11th IEEE International Requirements Engineering Conference (2003) IEEE Computer Science Press

Roozenburg NFM, Eekels J (1995) Product design: fundamentals and methods. Wiley, New York

Tichy WF, Lukowicz P, Prechelt L, Heinz EA (1997) Experimental evaluation in computer science: a quantitative study. J Syst Softw 28:9–18

Wieringa RJ, Heerkens H (2004) Evaluating the structure of research papers: a case study. In: Gervasi V, Zowghi D, Sim SE (eds) Second international workshop in comparative evaluation of requirements engineering

Zelkowitz MV (1998) Experimental models for validating technology. Computer 31(5):23–31

Zelkowitz MV, Wallace D (1997) Experimental validation in software engineering. Inform Softw Technol 39:735–743