Requirements Eng (2002) 7:107–111

Requirements Engineering

Viewpoints

'Viewpoints' is a regular section in *Requirements Engineeering* for airing readers' views on requirements engineering research and practice. Contributions that describe results, experiences, biases and research agendas in requirements engineering are particularly welcome. 'Viewpoints' is an opportunity for presenting technical correspondence or subjective arguments. So, whether you are a student, teacher, researcher or practitioner, get on your soapbox today and let us know what's on your mind...

Please submit contributions electronically to Viewpoints Editor, Didar Zowghi (didar@it.uts.edu.au). Contributions less than 2000 words in length are preferred.

Requirements Researchers: Do We Practice What We Preach?

Alan M. Davis and Ann M. Hickey
Department of Information Systems, University of Colorado at Colorado Springs, Colorado Springs, Colorado, USA

1. Introduction

All researchers have a responsibility to thoroughly understand the current research literature *and* the current state-of-the-practice. If they fail to understand the former, they risk repeating work that has already been done and their research will be for naught. If they fail to understand the latter, they risk creating new knowledge that has no practical value. Researchers in the field of requirements engineering are no exception. They must understand both how requirements engineering is practised and what research has been performed in the past. This paper argues that many requirements engineering researchers fail to understand current practices. The paper also argues that requirements engineering researchers, more than any other type of researcher, have little excuse for failing in this regard. This is true because by its very nature, requirements engineering is the investigation of how people do things currently. Thus, these individuals are not just irresponsible researchers, they are also poor requirements engineers. They seem to not understand the first rule of requirements engineering: *Know thy customer* This begs the question: Do we as requirements researchers practise what we preach?

Correspondence and offprint requests to: Ann Hickey, Department of Information Systems, University of Colorado at Colorado Springs, PO Box 7150, Colorado Springs, CO 80933-7150, USA. Email: ahickey@uccs.edu or adavis@uccs.edu

2. What Do We Preach?

As researchers and as educators we preach that requirements engineering practitioners should thoroughly understand the problem domain in which they are trying to solve a problem. After all, practising requirements engineers[1] have five primary purposes: (1) determining what needs exist, (2) determining which needs are to be addressed,[2] (3) selecting an appropriate solution[3] from a variety of possible solutions, (4) documenting the intended external behaviour of the system[4] to be offered (built or purchased), and (5) managing the ongoing evolution of the needs and the solution's intended external behaviour. Thus, requirements engineering maps problems from a problem domain into a proposed solution from the solution domain, as shown in Fig. 1.

To be effective as a requirements engineer, the individual must be capable of utilising knowledge to synthesise effective solutions. In the traditional view of

[1]We are *not* trying to debate the subtle differences between requirements engineering, requirements management, and requirements analysis. For the purposes of this article, we are considering them to be identical.
[2]Often termed triage [1].
[3]We intend to imply that one task of a requirements engineer is to propose alternative external behaviours to customers (e.g., via prototyping), and eventually select one. We are *not* implying that the requirements engineer selects data structures, algorithms, and the like.
[4]We use the term *system* in the broadest possible manner. It could be any subset of software, hardware, people, and procedures.

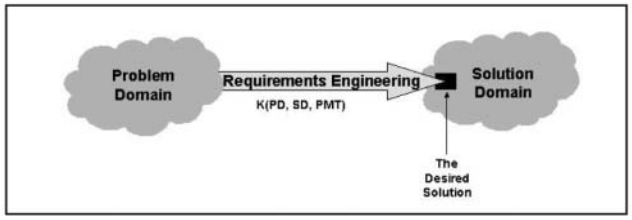

Fig. 1. The practice of requirements engineering.

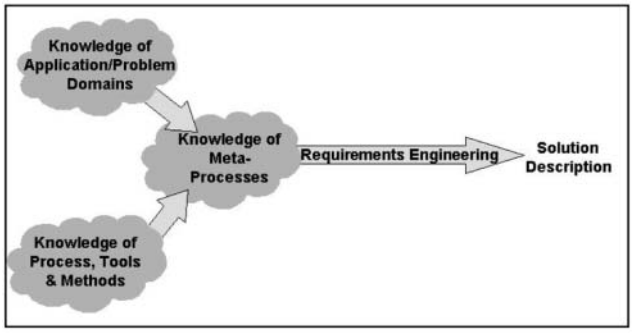

Fig. 2. Areas of knowledge for a requirements engineer.

requirements engineering, the knowledge required includes the items appearing just below the requirements engineering arrow, i.e., (1) knowledge of the problem domain [K(PD)], (2) knowledge of existing solutions within the solution domain [K(SD)], and (3) knowledge of processes, methods and tools of the practice of requirements engineering [K(PMT)]. And only recently, we have begun to recognise the need for a fourth area of knowledge: (4) knowledge of how to decide which processes, methods and tools make most sense as a function of certain aspects of the problem domain, the specific problem being addressed, the people involved, and so on. Figure 2 shows how the knowledge of these domains interact. Note that requirements engineers cannot do their jobs without understanding both the problem/application domains *and* the world of available process, tools and methods. And they need to understand the meta-processes necessary to determine which processes, methods and tools are appropriate for which applications and problems.

3. What Do Practitioners Do?

The final test of whether a requirements engineer has been successful is to determine how successful the resulting system is in the eyes of intended customers. If the intended customers are basically satisfied with the resulting system, then we could conclude the requirements engineering process was successful. If the intended customers are unsatisfied with the resulting system, then the requirements engineering process may have been at fault. According to the Standish Group[2], 31% of systems built today are never delivered, and another 15% had fewer than half of the intended customers' needs satisfied. This clearly demonstrates the failure of requirements engineers.

Many researchers blame practitioners for being unwilling to accept new ways of doing requirements engineering, that they insist on doing things the same old way. Our opinion is somewhat different. The most problematic requirements-related difficulties that companies face are related to the inherent difficulty of right-brained activities [3]. For example, here are scenarios we have seen often in industry:

1. Engineering/development has its view of what the customers' needs are and proceeds to satisfy those needs in direct opposition to the views of marketing (or the customer).
2. A development organisation thinks it knows the problems of the customer better than the customer does [4].
3. Multiple customer groups have different, and in some cases, conflicting needs.
4. Marketing has no idea how the inclusion of certain requirements relates to likelihood of successful use.
5. Needs of customers are in constant flux.
6. The customers really have little idea of what they need. What they do have is a fuzzy perception of a need for a solution to some ill-defined problem.
7. Market windows demand product delivery in a timeframe in which development/engineering cannot possibly deliver.
8. Development/engineering think that the solution to all requirements problems is to document requirements using a notation that the customers cannot possibly understand.
9. No respect exists between development/engineering and marketing (or customer). Insults regularly fly in both directions, e.g., 'You have no idea of what the needs are. If you did, you wouldn't be changing them all the time.' And 'You couldn't meet a delivery schedule no matter when it was.'
10. Naïve practitioners grasp repeatedly at left-brained panaceas to solve their right-brained problems. Examples include Structured Analysis, Object-Oriented Analysis, Use Cases and CASE tools.

4. What Do Researchers Do?

In most fields, the purpose of performing research is to synthesise new knowledge that can be either (1) put into practice, or (2) used by other researchers to help them

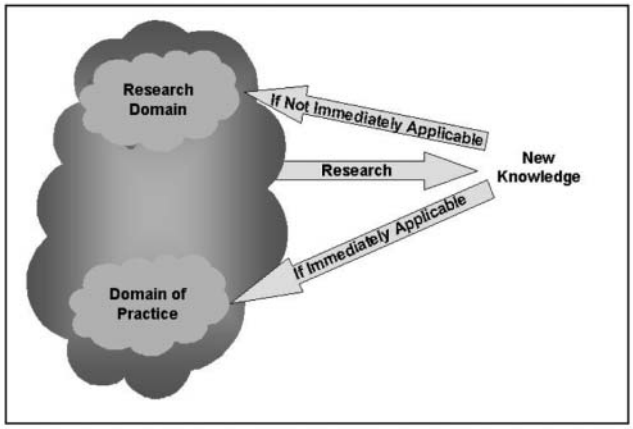

Fig. 3. The role of research.

perform research. As shown in Fig. 3, the responsible researcher needs to thoroughly understand two domains: the domain of practice (so the research results – or the results of subsequent research based on the current research results – can be applied to that practice), and previous research (so the research results do not simply duplicate previous research … resulting in no new knowledge). Researchers who fail to understand this basic concept are doomed to producing results that are never used by anybody in the 'real world'.

Every responsible requirements engineering researcher must at least identify (a) how their research compares and contrasts with existing research (in our opinion, this is not generally a problem), and (b) how it can be used practically in the real world. Part of this might be for the researcher to recognise to which of the three clouds in Fig. 2 the new research applies. Another would be for the researcher to understand the technical, political and social barriers to effective technology transfer and change the research to make it more assimilable. Since Fig. 2 defines the requisite knowledge of requirements engineering, we can embed Fig. 2 inside the 'Domain of Practice' cloud of Fig. 3 to see how and where in the real world any practical requirements research can be applied.

We believe that the process by which requirements engineering researchers perform research is broken.

Many requirements researchers complain that something is wrong with practicing requirements engineers because they don't adopt the 'latest and most terrific' research results. Some papers [5,6] have suggested that we need to improve communication between researchers and practitioners so that the *practitioners* will better understand the research. We believe that we need to improve the communication between researchers and practitioners so that the *researchers* will better understand the practice.

If a company produces a product and nobody buys it, is it the market's fault or the company's? We contend that only the company can be blamed. If a researcher performs research with the intent for it to be adopted into practice and nobody chooses to use it, is it the practitioner's fault or the researcher's? We contend that only the researcher can be blamed.

5. The Challenge

Most requirements research to date has not been used by its intended customers, i.e., the requirements engineers – and we *cannot* blame the customers. It is time to stop talking about how ill-informed the requirements engineers are, and start talking about how ill-informed requirements researchers are. If requirements engineering researchers would follow the first rule of any requirements engineer, i.e., 'Know thy customer', more of the research would prove to be helpful in practice. The research that needs to be done may not be fun or popular, but it is what needs to be done if we as researchers want to help our customers. Some of the more fruitful areas may include:

- *Scenarios*, as one of many approaches to talking in terms of the customer. As stated so well by the Robertsons [7], 'scenarios work well because they look like the stakeholders' world', not because there is some technical magic in some arbitrary notation to represent scenarios.
- *Situational research* [8], so we better learn when various requirements approaches work most effectively.
- *Conflict resolution*, *mutual respect* and *collaboration* [9]. Analysis of how to improve the ways that humans communicate.
- *Flux-sensitive architecture*. Requirements do change. Instead of talking about how requirements creep adversely effects our development, we should talk about how our development needs to change to accommodate requirements creep.
- Practical requirements *organisation* techniques, including both the use of lists and the use of word-processed documents [10,11].
- Techniques that increase alignment between marketing and requirements engineering.

The challenge to researchers in requirements engineering is to either acknowledge that their research is purely theoretical (and stop lamenting its lack of use), or to study current practice carefully. Studying current practice is tantamount to performing requirements analysis of the current practice – something we should be very good at! Note that studying current practice is

not just doing a survey, any more than performing requirements analysis is just doing a survey. Analysis is all about thinking, experiencing, observing, listening, feeling, understanding pains, and so on. Also, a practicing requirements engineer has a responsibility to produce feasible solutions, where feasibility includes sensitivity to political, operational, economic, schedule and technical issues. If the practitioner will use a new system only if it reflects a small change from current practice, the requirements engineer must make only small changes … or they will fail. Researchers have a similar challenge: understand not only how requirements engineers do their job, but also understand all the political, operational, economic, schedule and technical issues affecting their ability to adopt new ideas.

This discussion would not be complete without acknowledging the presence of systemic issues that interfere with our ability to produce research results ready for common practice. These include (1) emphasis on publishing, rather than technology transfer, in universities, (2) disincentives to leave the university for periods of time to work in industry, (3) emphasis on more theoretical journals for 'credit' toward tenure and/or promotion, (4) disincentives (e.g., low academic salaries) for individuals with extensive real-world experience from entering academia, (5) lack of reading by practitioners of journals in which researchers publish[5] (unlike physicians who do read the *Journal of the American Medical Association*), and (6) a learning environment in which (unlike medical school) students are taught by professors with little or no experience in the 'real world', and who have learned everything they know from yet other professors who have learned everything they know from yet other professors, and so on [12].

6. Conclusion

When we as requirements researchers lament that technology transfer takes a whopping 15 years [13], perhaps we should look no farther than ourselves.

As requirements engineering researchers we should be *experts* (not just 'knowledgeable') in how requirements engineers ply their trade. But even more importantly, we must be able to practice sound requirements engineering principles while we are doing research. Practicing requirements engineers use the acceptance of the resulting system by its intended users as the primary judge of whether they did a good job. We must use the same notion for our research: requirements engineer researchers should use the acceptance of the resulting research by its intended users, practicing requirements engineers, as the primary judge of whether they did a good job. We also need to collectively work on diminishing the systemic problems described at the end of Section 5, not use the systemic problems as an excuse for our own inadequacy. Let's start practising what we preach.

[5]Of course this will only come to pass when researchers learn to write for practitioners, and not just for other researchers.

References

1. Yourdon E. Death march. Prentice-Hall, Englewood Cliffs, NJ, 1997
2. Standish Group. The chaos report. www.standishgroup.com, 1995
3. Davis A. The harmony of rechoirments. IEEE Software 1998;15(2):6–8
4. Gause D, Weinberg G. Are your lights on? Dorset House, New York, 1990
5. Debou C, Fuchs N, Saria H. Selling believable technology. IEEE Software 1993;10(6):22–27
6. Potts C. Software engineering revisited. IEEE Software 1993;10(5):19–28
7. Robertson J, Robertson S. Requirements management: a Cinderella story. Requirements Eng 2000;5:134–136
8. Jackson M. Problem frames. Addison-Wesley, Reading, MA, 2001
9. Dean D, Lee J, Pendergast M, Hickey A, Nunamaker J. Enabling the effective involvement of multiple users: methods and tools for collaborative software engineering. J Manage Inform Syst 1997–1998;14(3):179–222
10. Davis A. Achieving quality in software requirements. Software Quality Professional 1999;1:3
11. Robertson J, Robertson S. Mastering the requirements process. Addison-Wesley, Reading, MA, 1999
12. Davis A. Practitioner, heal thyself. IEEE Software 1996;13(3):4–5
13. Redwine S, Riddle W. Software technology maturity. In IEEE eighth international conference on software engineering, 1985, pp 189–200

About the Authors

Al Davis is a professor of information systems at the University of Colorado at Colorado Springs. He is also chairman and CEO of Omni-Vista, Inc. He has consulted for many corporations over the past 20 years, including Boeing, Cigna Insurance, Federal Express, Fujitsu, General Electric, IBM, Loral, McDonald's, MCI, Mitsubishi Electric, Rockwell, Schlumberger and Storage Tek. Previously, he was a Vice President at BTG, Inc., a Director of R&D at GTE Communication Systems and Director of the Software Technology Center at GTE Laboratories. He is Editor-in-Chief Emeritus of *IEEE Software* after serving as Editor-in-Chief from 1994 to 1998. He is an editor for the *Journal of Systems and Software* (1987–present) and was an editor for *Communications of the ACM* (1981–1991). He is the author of *Software Requirements: Objects, Functions and States* (Prentice-Hall, 1990 and 1993) and *201 Principles of*

Software Development (McGraw-Hill, 1995). He is the founder of the IEEE International Conferences of Requirements Engineering, and served as general chair of its first conference in 1994. He has been a fellow of the IEEE since 1994, and earned his Ph.D. in Computer Science from the University of Illinois in 1975.

Ann Hickey is an assistant professor of information systems at the University of Colorado at Colorado Springs. She worked for 17 years as a program manager and senior systems analyst for the Department of Defense before beginning her academic career. She is currently teaching graduate and undergraduate courses in systems analysis and design and information systems project management. Her research interests include collaborative requirements elicitation, systems analysis, and scenario and process modeling. Her work has been published in the *Journal of Management Information Systems*, the *Database for Advances in Information Systems*, and several national and international conference proceedings. She received her B.A. in mathematics from Dartmouth College and her M.S. and Ph.D. in MIS from the University of Arizona.