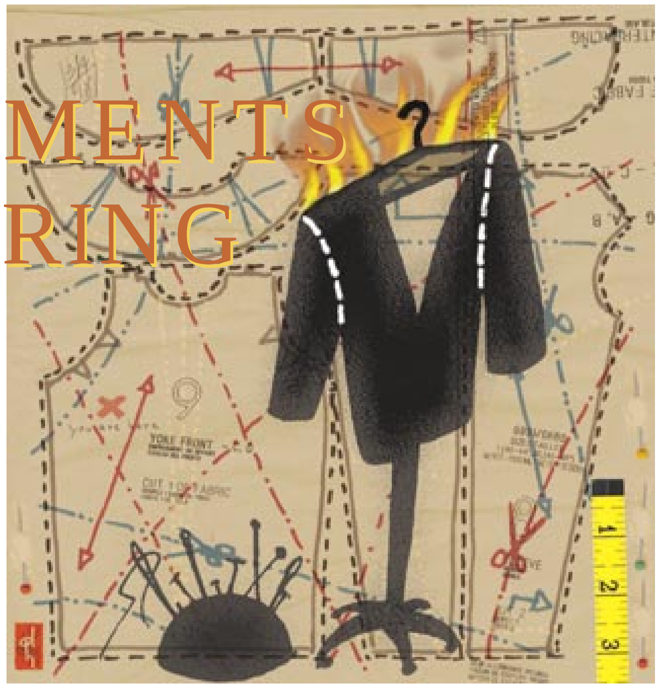
Slowly, we are bridging the gap between requirements engineering research and practice. The gap is still large, but we have a few more practice-validated methods and tools in our pockets, and the bridge building continues.

Requirements Engineering



Daniel M. Berry, Technion–Israel Institute of Technology
Brian Lawrence, Coyote Valley Software Consulting

More than a decade ago Fred Brooks wrote, "The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is so difficult as establishing the detailed technical requirements.… No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later."[1] Throughout our industry's history, we have struggled with this truth. Defining and applying good, complete requirements is hard work, and success in this endeavor has eluded many of us. Yet, we continue to make progress. As with many fields, researchers propose new methods to deal with the problem, and practitioners eventually adopt some of them. This lack of complete adoption is known as the research–practice gap.

Closing the Gap

In their March 1994 *IEEE Software* guest editors' introduction, Alan Davis and Pei Hsia—the general chair and one of the program cochairs for the 1994 International Conference on Requirements Engineering—reported that the gap between software engineering research and practice is no more evident than in the field of requirements engineering.[2] We are happy to report now that in SE research generally, and in requirements engineering research particularly, there appears to be a fairly universal recognition of this gap. An increasing percentage of the research done over the last two years has addressed this gap and focused on immediately applicable ideas or on understanding actual practice. For example, in ICSE '97,[3] five papers validated or debunked popular conceptions about the inspection process, and one of us (Berry) walked away from that conference with a firmer understanding of the inspection process.

Practitioners have long known that requirements and design affect each other.

Today, more software engineers recognize that any description of a new approach to solving a hard problem in SE must come from practical experience or be accompanied by a description of at least one actual application of the approach to an industrial-strength problem and an assessment of that effort's success. Fewer published papers present "yet another method" that hasn't been tried in industry; fewer authors beg readers to accept their unproved conviction that their method will *obviously* work, when in fact, their method is as obscure as any other.

In this spirit, the ICRE '98 program committee established an important criterion for paper acceptance: bridging the gap between research and practice.[4]

Conventional and Nonconventional Wisdom

In the March '96 issue of this magazine, ICRE '96 program cochairs Jawed Siddiqi and M. Chandra Shekaran wrote the guest editors' introduction, updating the current status of requirements engineering.[5]

In taking stock of the field, they observed that requirements engineering attempts to bring an engineering orientation to traditional information system analysis and apply the result to all software engineering efforts from the beginning. As with other attempts to be more engineering-like, this has led to a variety of requirements engineering methods and tools, developed primarily by researchers. The major drawback of these methods and tools has been insufficient attention to how the context of the proposed system affects its requirements, development, and evolution. The *context* of a computer-based system is the real-world environment in which the system operates, including the social structure and the people therein. Practitioners are used to focusing on context and do so as a matter of course. So they find the methods and tools wanting. Thus, the editors concluded, the gap between practice and research remained wide.

However, Siddiqi and Shekaran saw some hope when they said, "the conventional wisdom about requirements engineering is rapidly evolving, …and the latest research is taking context into account." They reviewed the history of developments in requirements engineering and showed that it has followed the same trends as in systems development. The first wave focused on writing code. The second wave focused on the development life cycle, of which requirements analysis is the first phase. The third wave focused on evolutionary development and the implication that requirements are always incomplete; each stage involves identifying new requirements based on experiences from previous stages. As a result, today we recognize that requirements engineering has its own life cycle, although we still debate which activities are part of it.

Another longstanding debate arises out of the not universally supported conventional wisdom that requirements are a statement of *what* a system will do, with no reference to *how* the system does what it does. Researchers generally stick to this point of view, while practitioners have long been comfortable with the notion that requirements and design affect each other.

The articles in the '96 special issue of *IEEE Software* focused on the identification of nonfunctional or quality requirements, management of multiple requirements perspectives, and traceability of requirements, particularly to the context in which they are situated.

Signposts and Landmarks

Overview Books

Gerald Weinberg, *Rethinking Analysis and Design*, Dorset House, New York, 1988.

Donald Gause and Gerald Weinberg, *Exploring Requirements*, Dorset House, New York, 1989.

Donald Gause and Gerald Weinberg, *Are Your Lights On?* Dorset House, New York, 1990.

Alan M. Davis, *Software Requirements: Objects, Functions, and States*, Prentice Hall, Upper Saddle River, N.J., 1993.

Michael Jackson, *Software Requirements and Specifications: A Lexicon of Practice, Principles and Prejudices*, Addison Wesley Longman, Reading, Mass., 1995.

Pericles Loucopoulos and Vassilis Karakostas, *System Requirements Engineering*, McGraw-Hill, New York, 1995.

Roel J. Wieringa, *Requirements Engineering: Frameworks for Understanding*, John Wiley & Sons, New York, 1996.

Hugh Beyer and Karen Holtzblatt, *Contextual Design: Defining Customer-Centered Systems*, Morgan Kaufmann, San Francisco, 1998.

Periodicals

This special issue represents the seventh that the IEEE Computer Society has devoted to requirements engineering. The previous ones were *IEEE Transactions on Software Engineering*, Jan. 1975; *Computer*, May 1982; *Computer*, Apr. 1985; *IEEE Transactions on Software Engineering*, Mar. 1991; *IEEE Software*, Mar. 1994; *IEEE Software*, Mar. 1996 (annual archive available online to electronic and combo subscribers).

In 1996, Springer-Verlag started a new journal, *Requirements Engineering*, and *The Annals of Software Engineering* devoted Volume 3, 1997, to the topic.

Proceedings

The International Symposium on Requirements Engineering (ISRE) convenes in odd-numbered years.

The International Conference on Requirements Engineering (ICRE) takes place in April of even-numbered years in Colorado Springs, Colorado. The IEEE Computer Society Press publishes their proceedings.

Requirements engineering sessions are scheduled increasingly at the various software engineering meetings, such as the International Conference on Software Engineering (ICSE), the European Software Engineering Conference (ESEC), and the Brazilian Symposium on Software Engineering (SBES).

Other conferences deal largely or in part with requirements engineering, for example, the International Workshop on Software Specification and Design (IWSSD) and the International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ).

Anthologies

Richard H. Thayer and Merlin Dorfman, *Software Requirements Engineering*, second edition, IEEE Computer Society Press, Los Alamitos, Calif., 1997.

Web and FTP sites

Renoir, the European Union Requirements Engineering Network of Excellence: http://www.cs.ucl.ac.uk/research/renoir/

Requirements Engineering Newsletter: the files named renl* at ftp://ftp.cs.city.ac.uk/pub/requirements/

The CSIRO–Macquarie University Joint Research Centre for Advanced Systems Engineering (JRCASE) Requirements Engineering Web page: http://www.jrcase.mq.edu.au/seweb/requirements/requirements.html

Information about a software requirements engineering mailing list: http://www.jrcase.mq.edu.au/~didar/seweb/groups.html

The Debates Continue

Still other debates continue to this day. One concerns the distinction between functional and nonfunctional requirements. Here too, the research insists on this distinction, whereas practice accepts that in many cases the distinction is unclear.

Another debate concerns the nature of the systems being developed and their requirements. Heretofore, most requirements engineering work focused on large, one-of-a-kind systems, developed for a specific customer under a contract that includes requirements, with the idea that implementing less than the full set of requirements amounts to failure to implement the contracted system. However, practitioners face several realities:

- ♦ Most software development today is market driven, developed to satisfy anonymous customers and to keep them coming back to buy upgrades.
- ♦ There are not enough resources to build software that can do everything the customer wants. It is essential to rank requirements so that, in the face of pressure to release quickly, the most important requirements can be implemented first.
- ♦ Requirements are inherently incomplete, mainly because they are never fully understood and because they typically change as a result of system deployment.

Finally, practitioners have pointed to two particular needs that methods and tools must fulfill: First, the portion of the architectural design necessary for effective requirements engineering must be integrated into the requirements. Second, our methods and tools must become more accessible.

In This Issue

This issue contains two of the best articles submitted and accepted for presentation at ICRE '98, scheduled for 6–10 April in Colorado Springs, Colorado. In the spirit of ICRE '94 and the '94 *IEEE Software* special issue, the articles selected for this special issue, "Acquiring COTS Software Selection Requirements" by Neil A. Maiden and Cornelius Ncube and "Scenarios in System Development: Current Practice" by Klaus Weidenhaupt, Klaus Pohl, Mathias Jarke , and Peter Haumer, deal with the translation of previously developed research ideas into practice and report what has been learned from real-life application. See page 4 for summaries of these and the other articles in this issue. Both articles make compelling reading in that what was learned often surprised not only the authors but also us.

Also, the March 1994 guest editors provided a requirements engineering reading list. We have updated it in the boxed text "Signposts and Landmarks" on page 28.

This issue's Point–Counterpoint deals with who should own the requirements of a project. In the Point essay Carl Clavadetscher argues that the users need to own and manage them; only users know enough about a system's needs to be able to make the proper decisions and trade-offs.

Brian Lawrence, wearing the hat of author, argues in his Counterpoint essay that a project's designers should own and manage the requirements or they won't understand enough about the requirements to design a good implementation. Here, the term "good" means all desired properties, including correctness, completeness, efficiency, and so on.

We, wearing the hat of guest editors, take a rabbinical point of view. Both Carl and Brian are right for precisely the reasons they state. A specification owned and written by someone else is just as boring to users as it is to designers. Both will say, "Do I have to read all this? Let's just get on with it."

Users must know the requirements specification intimately to do their job: to make "need" and "want" decisions and to validate the specification. Designers must know the specification intimately to do their job: to design an implementation for it. Neither group can do its job without owning, managing, and participating fully in the writing of the specification.

We hope you enjoy this issue. ❖

References

1. F.P. Brooks Jr., "No Silver Bullet," *The Mythical Man-Month: Essays on Software Engineering* (second ed.), Addison Wesley Longman, Reading, Mass., 1995, pp. 179-203.
2. A. Davis and P. Hsia, "Giving Voice to Requirements Engineering," *IEEE Software*, Mar. 1994, pp. 12-16.
3. *Pulling Together: Proc. 19th Int'l Conf. Software Eng. (ICSE '97)*, ACM Press, New York, 1997, pp. 17-23.
4. *Proc. Third Int'l Conf. Requirements Eng. (ICRE '98)*, IEEE Computer Soc. Press, Los Alamitos, Calif., to be published in Apr. 1998.
5. J. Siddiqi and M.C. Shekaran, "Requirements Engineering: The Emerging Wisdom," *IEEE Software*, Mar. 1996, pp. 15-19.

About the Authors

Daniel M. Berry has been a computer science professor at Technion since 1987. Before that he taught for 15 years at the University of California, Los Angeles. From 1990 until 1994, he worked for half of each year at the Software Engineering Institute at Carnegie Mellon University, where he helped build CMU's Master of Software Engineering program. Berry's current research interests are software engineering in general, and document processing and requirements engineering in particular. He is a member of the IEEE Computer Society and ACM.

Berry received a PhD in computer science from Brown University.

Brian Lawrence is a principal at Coyote Valley Software Consulting, where he consults on software requirements practices. He is also currently an instructor at the University of California, Santa Cruz Extension in software engineering. He is a member of the IEEE Computer Society and ACM.

Lawrence received a BA in psychology from The College of William and Mary and an MS in computer science from Virginia Commonwealth University.

Address questions about this article to Berry at dberry@cs.technion.ac.il or to Lawrence at lawrence@acm.org.