Is the European Industry Moving toward Solving Requirements Engineering Problems?

Natalia Juristo, Ana M. Moreno, and Andrés Silva,
Universidad Politécnica de Madrid

Years ago, several surveys raised concerns about problems in requirements engineering practice. Has the industry progressed since?

Requirements engineering is critical for successful software development. Nowadays, software development organizations are not likely to question the importance of issues related to requirements management (RM) and specification. However, despite its importance, the requirements process has traditionally been connected with a host of problems. Frederick Brooks used the two Aristotelian categories, *essential* and *accidental*, to classify these problems.[1] As systems become more complex, software developers can do little to overcome essential difficulties such as software invisibility or pressure for change. However, several surveys have highlighted principal flaws in the requirements process that can be linked to accidental difficulties such as tool integration or bad documentation.[2,3] Not only are these problems solvable, they're also often ones that researchers have already addressed. For years, researchers have conducted requirements engineering-related surveys, revealing problems and identifying potential solutions. Yet according to our own survey, RE problems persist.

We contacted RE practitioners from European organizations to analyze how much progress European software development organizations have made in RE. Unlike other surveys, we don't just point out RE problems. Our results call attention to the gap between current RE practice and published solutions and to the poor communication between researchers and practitioners.

Problems identified in previous RE surveys

In 1998, Bill Curtis, Herb Krasner, and Neil Iscoe conducted one of the first RE surveys,[4] providing information on critical development issues (see Table 1 for brief descriptions of the surveys discussed here). Their field study of 10 organizations suggested that information on project functionality and the ease with which people could change this information were key to application success.

Many surveys have also identified incorrect tool use as an important issue. In 1993, Mitch Lubars, Colin Potts, and Charlie Richter surveyed 10 US software develop-

0740-7459/02/$17.00 © 2002 IEEE

ment organizations and found that "the most obvious documentation tools are word processing packages," which is a poor choice for a documentation tool.[2] Two years later, Khaled El Emam and Nazim Madhavji identified the same problem at several Canadian organizations, saying proper tool use is one of the seven key issues for RE success.[5] This was still an issue in 2000, when Uolevi Nikula, Jorma Sajaniemi, and Heikki Kälviäinen surveyed 12 Finnish software development organizations and found that "no company used ... RM tools, and even tools such as templates, checklists, and metrics were in standard use in one or two companies only."[6] Furthermore, just last year, Humber Hofmann and Franz Lehner's field study found that "the most common tool used during RE was an internal Web site," used to post and maintain the requirements.[7]

Another RE problem has been the lack of proper documentation in software requirements specifications (SRS). Lubars, Potts, and Richter found that documentation was excessively formal and detailed in customer-specific projects but specifications were informally expressed in market-driven projects.[2] Similarly, Nikula, Sajaniemi, and Kälviäinen found that "the decision whether the requirements document is created or not depends on many factors."[6] Erik Kamsties, Klaus Hormann, and Maud Schlich also detected this shortcoming in their survey of 10 small- to medium-sized European software organizations: "Only when subcontracted are SRSs done seriously."[3] Similarly, some findings made in the context of the REAIMS (*RE a*daptation and *im*provement for *s*afety and dependability) Esprit Project concern the importance of proper requirements documentation. This project led to *Requirements Engineering: A Good Practice Guide*, which identifies guidelines for successful RE.[8] The importance of properly performing an SRS underpins most of the guidelines.

User involvement during the requirements process is paramount, but this is yet another stumbling block in software development. The Chaos Report series, from 1994 to 2001 (see, for example, www.standishgroup.com/sample_research/chaos_1994_1.php), revealed that user involvement is one of the two main success factors in software development (the other is executive support). It consistently found low user involvement in failed or challenged projects. Other surveys identified similar results.[5,7]

Another problem detected in several surveys is traceability. In general, we might distinguish between prespecification traceability and postspecification traceability—the first links requirements to their sources (users, documents, and so forth), and the latter links requirements to development artifacts.[9] Developers traditionally address only postspecification traceability. Balasubramanian Ramesh analyzed 26 organizations and concluded that postspecification traceability—but not prespecification traceability—is a general attribute.[10] Stakeholders see this lack of traceability as hurting the project.[7]

Our survey

We applied a method similar to that used in many other surveys. We contacted more than 150 practitioners from European organizations to provide an overview of the current situation, without emphasizing statistical data. The size of the organizations and of their products varied, but we considered all of them as representative developers of the applications that are shaping the information society, from embedded systems to Internet applications. We chose to contact practitioners on the basis of their involvement in RE. Most had a medium-to-high level of responsibility in the RE process at their companies, from a software development viewpoint (our set of respondents did not include marketing or sales personnel).

We secured responses from 11 organizations in seven European countries. Seven of the organizations were small to medium-sized (five to 100 employees), and four were larger. The 11 organizations developed software for these domains: consumer electronics with embedded software, IT products for the health-care-systems market, software and sys-

Table 1
Requirements engineering surveys

Researchers	Purpose[a]	Mechanism used	Analysis
Curtis, Krasner, and Iscoe[4]	Prescriptive	Case studies	Qualitative
Gotel and Finkelstein[9]	Prospective	Multipronged	Qualitative
El Emam and Madhavji[5]	Prescriptive	Case studies	Quantitative
Hofmann and Lehner[7]	Prescriptive	Questionnaire	Quantitative
Kamsties, Hormann, and Schlich[3]	Prescriptive	Multipronged	Qualitative
Lubars, Potts, and Richter[2]	Prescriptive	Case studies	Qualitative
Nikula, Sajaniemi, and Kälviäinen[6]	Descriptive	Questionnaire	Quantitative
Ramesh[10]	Prescriptive	Multipronged	Quantitative

[a] Descriptive surveys try to determine current practices; prescriptive surveys identify good, or bad, practices; and prospective surveys ascertain future needs that would further research.

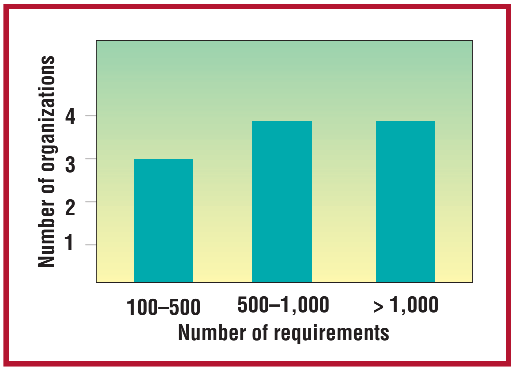

Figure 1. The organizations, sorted by their number of requirements.

Table 2

Questions used in the survey

Question type	Question
Current profile	Briefly describe the methods and tools used.
	Describe the advantages of current methods and tools.
	Describe the disadvantages of current methods and tools.
	Are current methods and tools well suited for dealing with current and typical applications?
	Describe the RE life cycle.
	Describe how you integrate the RE process with other business processes.
	Who is involved in RE tasks (system engineers, marketing personnel, requirements engineers, and so on)?
Adoption	Have people or processes affected by changes been correctly identified?
	Has any guide or translation package been used?
	Have people been properly trained?
	Is there still some remaining resistance? Why?
	Stability of current practice: When was the last change made? Why?
Sources of requirements	Are there problems identifying users and/or stakeholders?
	What impact do standards, certification, and COTS have on the RE process and products?
Dependability requirements	How do you manage your dependability requirements?
	How do you manage the trade-off between dependability needs and available dependability?

tems targeting the industrial and public sector, Web multimedia applications, aircraft systems, virtual consultancy for e-commerce, smartcards, software tools for client-server application development, cryptography, intellectual-property-rights-handling software, and software for electrical-network maintainability. (We can't, however, reveal the identity of the respondent organizations.)

Figure 1 illustrates the approximate size of the systems these organizations built in terms of the number of requirements. Although this measure is not precise, it assures that we are considering organizations that cover a broad spectrum of software size and complexity.

Table 2 shows the questions we used to gather information, which we designed to address the key issues the earlier surveys detected: tool misuse, improper requirements documentation, low user involvement, and nontraceability. We also included two more points: adopting new techniques and dependability.

Adopting new requirements techniques and tools is essential for process improvement and technology transfer.[11] However, industrial uptake of RE technology has rarely lived up to expectations.[6,12] So we wanted to determine how conscious organizations were of its importance.

We also asked about dependability because numerous services and products, based on both the Internet and the massive ubiquitous deployment of embedded systems, are used in many areas, including health care, transport, finance, commerce, and public administration. These areas have significant dependability implications, embracing security, safety, reliability, availability, and survivability.[13]

Current practice

Our survey confirmed that immaturity still defines current practices.

Although the questionnaire distinguished between methods and tools, the responses clearly indicate that the two concepts are used indistinctly. Some respondents described their "method" as "tool X," which is not surprising because tools drive or enhance methods. However, our results indicated that organizations are more informed about requirements tools than previously.[2,3,5]

Most organizations reported using tools such as word processors for specifying and managing requirements, and more than 30 percent used only these tools. Organizations reported that, for applications with relatively few requirements, these tools had the advantage of simplicity. However, for those with many requirements (approximately 1,000) and where word processors were the main tool, the disadvantages outweighed the advantages: lack of scalability, no baseline, and so forth.

Not surprisingly, the organizations using either in-house or commercial requirements tools (approximately 70 percent) worked on larger applications (over 1,000 requirements). This shows that industry realizes that requirements tools are useful for large projects. However, as expected (and as identified elsewhere[8]), these organizations pointed out that no single tool is valid for the whole process, and tool integration poses a considerable obstacle to efficiency.

Our results are similar to other surveys concerning the lack of proper SRSs, especially in market-oriented applications.[2,3,6] We attribute this to the difficulty in finding specific users for this sort of application. Also, although most companies reported having no problems identifying their systems' users and stakeholders, some reported that this didn't mean they were involved in the process or had clearly defined roles and responsibilities. Furthermore, our survey also confirmed general problems related to a lack of traceability and supported Ramesh's findings about postspecification and prespecification traceability.[10] Only one organization mentioned how difficult it is to deal with prespecification traceability.

Adoption

At least six organizations had recently introduced new practices, although the scope of these changes varied among organizations. The introduction of tools and traceability-related issues were prominent, but nobody reported using transition packages to introduce new technologies.

The organizations did not report significant problems in identifying the people and process affected by the introduced practices. However, there was some discrepancy among organizations concerning the changes' effect. Half the organizations that had introduced improvements stated that reorganization did not have dramatic effects. In fact, one organization (in the health-care industry) reported that it was "always ready for change," so the effects of reorganizing tasks were never dramatic.

Other organizations, however, indicated that they met with resistance from project managers, who thought that changes could destroy their schedule. For example, the respondent from one multinational (consumer electronics) company said that even though current practices in different business units evolve at a different pace, this is not without problems: "People are so used to their existing ways of working that new RE practices must be fitted to them." Consistently, Hofmann and Lehner's study also found that, regarding tool adoption, users perceived an interference, not a support, with current activities.[7]

Requirements sources

The survey results indicated that organizations must consider a multiplicity of requirements sources, including internal sources such as the marketing department, product managers, and sales personnel, and customers and related sources, such as users and help desks. They must also consider constraints, such as

- *Standards*: Three-quarters of the surveyed organizations considered the impact of standards to be important. Standards are useful mainly for organizing various parts of the process, because they provide a list of things to remember, guides for doing tasks, documentation, and so forth.
- *Laws*: Organizations should anticipate future changes in laws and adapt their products accordingly.
- *Certification*: Certification has less impact on requirements. One-third of the respondents seemed concerned about certification issues (such as ISO9000).
- *Commercial-off-the-shelf components*: Respondents reported that using COTS components changed the requirements process, because the focus shifted from needs that the developer had to satisfy to needs that available COTS components should satisfy.

Identifying all requirements sources has been a successful practice.[7] However, the multiplicity of requirements sources increases RE complexity. Managing multiple documents and sources of sometimes-conflicting information can become overwhelming.

Dependability requirements

Our respondents were interested in dependability issues for their products. We can summarize the main dependability-related finding as the difficulty in quantitatively establishing dependability. The reason for this is that dependability is subordinate to a variety of elements:

- *Architectural components*: These components' final characteristics are often unknown early in the project. Particular difficulties arise when an external company develops these components.
- *Available technology*: It is essential to

Our survey confirmed that immaturity still defines current practices.

understand what you can achieve with current technology at a reasonable cost. However, as technology quickly evolves, you cannot know all of its characteristics at the start of development.

- *System interaction*: A medical application might operate perfectly, but another system with which it operates might supply inaccurate information. Developers should clearly establish responsibilities in case something fails.

So, unless a contractor imposes dependability levels, you cannot clearly express a dependability level when a project starts. Also, developers cannot always create a fixed prioritization of dependability requirements: priorities might change as you gather new information. Some of these practices for dealing with dependability requirements demonstrate that it is difficult to follow the advice given in much of the RE literature, which recommends expressing nonfunctional requirements quantitatively.[8]

Organizations need processes for selecting appropriate methods, which should consider, among other things, tool availability.

Survey implications

Here, we present guides for solving some of the problems found. These might not be the best solutions, and we can't guarantee what improvements a particular solution will provide, but many people in the RE community advocate these well-known solutions. The fact that they have not been adopted clearly indicates the need to improve both technology transfer and industrial uptake.

Requirements techniques

Our survey revealed that requirements techniques are not used enough for either elicitation or negotiation. Elicitation techniques have been available for some years, but organizations seem unfamiliar with them, which means that knowledge is still not being transmitted effectively. Possible solutions would be to use transition packages,[11] promote training within organizations,[10] or use outside consultants.[6]

Our survey and others also detected that formal SRS documents are not defined, which generates much extra work. However, the definition of an SRS raises many critical questions, including how detailed the SRS should be. Ian Sommerville and Pete Sawyer provide guides for addressing this problem, taking into account that the detail level depends on whether the project is in-house or subcontracted.[8] Requirements are often written in natural language, but this should not impede high-quality documentation or using tools to help analyze this documentation. Quality can be achieved in natural-language requirements documents by using style guides.[14] Also, there are semiautomated techniques for analyzing natural-language requirements, sometimes requiring a strict syntax aimed at easing the analysis.[15]

One method suggested for providing some degree of formality in traceability is Quality Function Deployment (QFD).[2,16] Orlena Gotel and Anthony Finkelstein found that prespecification traceability is a significant issue[9] and proposed using contribution structures that help capture the network of people who participate in RE.[17] Ramesh has discovered more specific factors that improve, and impede, traceability.[10] Any organization should be able to improve traceability practices by supporting the improvement factors and trying to correct impeding factors.

Requirements tools

The main problem with multiple-tool approaches is the lack of tool integration. There does not appear to be any definite solution to this problem, apart from tool developers considering these issues to develop tools with more potential. Ideally, requirements tools should incorporate features such as capacity for large volumes of documentation, multiple levels of formality, traceability, configuration management, and model simulation.[2]

Another problem is how to select the tools best suited for the process and class of applications. There is public information on the characteristics and potential of commercial tools (see www.incose.org/tools/tooltax.html), but this is not enough. Organizations should use this information merely as a basis for conducting more specific studies using in-house criteria. (Merlin Dorfman offers advice on how to introduce tools into the organization.[18]) Organizations need processes for selecting appropriate methods, which should consider, among other things, tool availability. Only then should the organization purchase tools. Transition packages, including training and consultancy services, can be a great aid for RM tool adoption.[11]

Support from tool manufacturers during tool deployment is helpful, but other solutions exist. In our survey, one big organization reported having a group of people who provided internal consultancy services to different company units. These people were well acquainted with the characteristics of the tools on the market and had a sound knowledge of the organization. This assured informed decisions regarding the suitability of available tools for the organization's projects. However, this approach is cost-effective only in large organizations.

Requirements sources

Donald Gause and Gerald Weinberg proposed requirements techniques that raise user involvement in the process,[19] and approaches such as usage-centered design can help.[20] Although suited for the development of bespoke systems, these techniques do not completely solve the problem for market-driven or Web-based applications. These situations require marketing-related techniques, such as portfolio-based techniques[21] or QFD.[16]

Another source to consider originates from using COTS components. This involves relating the requirements of the applications to be built to the characteristics of available COTS components. Some solutions model both customer requirements and software products (including a model of complex interdependencies) and use multicriteria decision-making techniques for COTS selection.[22] Other solutions propose weighted averages for calculating a particular COTS requirements coverage ratio, further refined by user evaluation of the product in particular scenarios.[23] To overcome the lack of compatibility between packages and the lack of control over COTS evolution, Barry Boehm and Chris Abts have offered several maxims:

- Do not prematurely commit to a combination of COTS packages.
- Try to achieve COTS substitutability.
- Avoid tightly coupled, independently evolving COTS components.
- Try to establish strategic partnerships with COTS vendors to secure their continuous support.[24]

Dependability requirements

Precisely quantifying dependability attributes faces two main obstacles:

- Determining what you can achieve with current technology and using that information as input for the requirements process. A risk is never acceptable if it can be easily reduced with available technology.
- Specifying the dependability level. The desired dependability level can be unrealistically specified and lead to long delays and increased costs.

A criterion such as Alarp (as low as reasonably possible) could replace the strict quantification of risks,[25] considering the state of the art, particularly for safety and security issues. A solution used in environmental contexts for some time under the acronym Batneec (best available technology not entailing excessive cost),[26] now included in many standards for environmental protection, means that product developers should define a tolerable region of risk on the basis of the available technology's capabilities and costs. This process involves extensive negotiation with certification authorities and with technology developers. Safety-critical software has borrowed many ideas from other engineering fields;[27] to our knowledge, however, there is no guidance yet on applying the Batneec principle.

Some suggest using risk analysis techniques to deal with potential risks in safety-critical systems,[27] with the aim of establishing an adequate protection level. However, these approaches implicitly assume that once the protection level is fixed, it will remain fixed. Developers need some flexibility to deal with possible variances in cost or technology availability. For example, one surveyed organization identified gaps between highly desirable and probably achievable dependability levels at the project's start. The organization took further actions to narrow this gap until it reached an adequate balance. Rather than establishing a fixed protection level, it moved between flexible upper and lower bounds that defined the cost-effective and risk-tolerable region. However, it carried out this process ad hoc.

Another problem identified is how to set priorities for requirements related to de-

Developers need some flexibility to deal with possible variances in cost or technology availability.

Requirements Engineering Resources

The Atlantic Systems Guild: This site provides a collection of requirements and software engineering resources. www.systemsguild.com/GuildSite/Guild/resources.html

The Chaos Report series: These widely cited reports demonstrate that RE problems are the major causes of failure in software development projects. www.standishgroup.com

The Good Practice Guide on RE: The GPG Web site at Lancaster University offers guidelines for RE. www.comp.lancs.ac.uk/computing/resources/re-gpg

ICRE and RE conferences: The International Conference on Requirements Engineering and the International Symposium on Requirements Engineering are traditional forums. These two conference series on RE have been united as the IEEE Joint International Requirements Engineering Conference in 2002. www.re02.org

INCOSE: The International Council on Systems Engineering provides good information on commercial RE tool features. www.incose.org

Karl Wiegers' site: This site provides a good collection of downloadable RE tools and templates. www.processimpact.com

pendability issues. By their very nature, these are outstanding issues, but their relative importance could change as we learn more about costs, limitations, and the technical risk of available technologies. In this case, we need flexibility to set priorities. Karl Wiegers offers a useful schema for trade-offs between requirements, taking into account multiple factors.[28]

Dependability also includes security issues that greatly influence the requirements process, particularly for e-business software development. The lack of information about what is really needed when the project kicks off could be counteracted through a framework of some sort that could be used to discover security-related requirements. For example, Sarah Jones and her colleagues' proposal[29] considers this trade-off between cost and acceptable risk and, simultaneously, helps to achieve a complete set of high-level requirements related to trust and dependability in e-business systems. Again, the suitability of the decisions made will depend on a thorough knowledge of available technologies.

In the last year, awareness of requirements problems has increased, new books on RE have been published, more tools are available, and it seems that RE is getting more resources than before (see the "Requirements Engineering Resources" sidebar).[7] As the Chaos Report series demonstrates, the percentage of successful projects increases each year, but 45 to 65 percent of projects are still over budget and schedule. Improvements are occurring, but we are still far from achieving the level of performance that would be desirable in an engineering discipline. This immaturity has two potential solutions. On the industry side, technology uptake could be more proactive; on the research side, there could be more elaborate and practical guidelines for packaging and transferring recommendations to industry and including real evaluations of the advantage that these technologies provide.

Acknowledgments

We performed our survey at the Joint Research Centre of the European Commission, and the Requirements Engineering Network of International Cooperating Research Groups (ESPRIT project 20800) funded it. We are grateful to RENOIR for its support and to Philip Morris at the JRC.

References

1. F.P. Brooks, "Essence and Accidents of Software Engineering," *Computer*, vol. 20, no. 4, Apr. 1987, pp. 10–19.
2. M. Lubars, C. Potts, and C. Richter, "A Review of the State of the Practice in Requirements Modeling," *IEEE 1st Int'l Symp. Requirements Eng.* (RE 93), IEEE CS Press, Los Alamitos, Calif., 1993, pp. 2–14.
3. E. Kamsties, K. Hormann, and M. Schlich, "Requirements Engineering in Small and Medium Enterprises: State-of-the-Practice, Problems, Solutions and Technology Transfer," *Proc. Conf. European Industrial Requirements Eng.* (CEIRE 98), British Computer Soc., London, 1998, pp. 40–50.
4. B. Curtis, H. Krasner, and N. Iscoe, "A Field Study of the Software Design Process for Large Systems," *Comm. ACM*, vol. 31, no. 11, Nov. 1988, pp. 1268–1287.
5. K. El Emam and N.H. Madhavji, "A Field Study of Requirements Engineering Practices in Information Systems Development," *IEEE 2nd Int'l Symp. Requirements Eng.* (RE 95), IEEE CS Press, Los Alamitos, Calif., 1995, pp. 68–80.
6. U. Nikula, J. Sajaniemi, and H. Kälviäinen, *A State-of-the-Practice Survey on Requirements Engineering in Small-and-Medium-Sized Enterprises*, tech. report, Lappeenranta Univ. of Technology, Lappeenranta, Finland, 2000; www.cs.ucl.ac.uk/research/renoir/TBRC_RR01.pdf
7. H.F. Hofmann and F. Lehner, "Requirements Engineering as a Success Factor in Software Projects," *IEEE*

Software, vol. 18, no. 4, July/Aug. 2001, pp. 58–66.

8. I. Sommerville and P. Sawyer, *Requirements Engineering: A Good Practice Guide*, John Wiley & Sons, New York, 1998.
9. O. Gotel and A. Finkelstein, "An Analysis of the Requirements Traceability Problem," *Proc. 1st Int'l Conf. Requirements Eng.* (ICRE 94), IEEE CS Press, Los Alamitos, Calif., 1994, pp. 94–101.
10. B. Ramesh, "Factors Influencing Requirements Traceability Practice," *Comm. ACM*, vol. 41, no. 12, Dec. 1998, pp. 37–44.
11. P. Fowler et al., "Transition Packages: An Experiment in Expediting the Introduction of Requirements Management," *Proc. 3rd IEEE Int'l Conf. Requirements Eng.* (ICRE 98), IEEE CS Press, Los Alamitos, Calif., 1998, pp. 138–147.
12. P. Morris, M. Masera, and M. Wilikens, "Requirements Engineering and Industrial Uptake," *Proc. Int'l Conf. Requirements Eng.* (ICRE 98), IEEE CS Press, Los Alamitos, Calif., 1998, pp. 130–137.
13. *Defining the European Dependability Initiative*, Joint Research Centre of the European Commission, 1999; http://deppy.jrc.it.
14. B.L. Kovitz, *Practical Software Requirements: A Manual of Content and Style*, Manning Publishing, Greenwich, Conn., 1999.
15. B. Macias and S.G. Pulman, "Natural Language Processing for Requirements Specification," *Safety Critical Systems*, Chapman and Hall, London, 1993.
16. S. Haag, M.K. Raja, and L.L Schkade, "Quality Function Deployment: Usage in Software Development," *Comm. ACM*, vol. 39, no. 1, Jan. 1996, pp. 41–49.
17. O. Gotel and A. Finkelstein, "Extended Requirements Traceability: Results of an Industrial Case Study," *Proc. IEEE 3rd Int'l Symp. Requirements Eng.* (RE 97), IEEE CS Press, Los Alamitos, Calif., 1997, pp. 169–178.
18. M. Dorfman, "Requirements Engineering," *Software Requirements Eng.*, 2nd ed., R. Thayer and M. Dorfman, eds., IEEE CS Press, Los Alamitos, Calif., 1997, pp. 7–22.
19. D.C. Gause and G.M. Weinberg, *Exploring Requirements: Quality before Design*, Dorset House Publishing, New York, 1989.
20. L.L. Constantine and L.A.D. Lockwood, *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design*, Addison-Wesley, Boston, 1999.
21. S. Sivzattian and B. Nuseibeh, "Linking the Selection of Requirements to Market Value: A Portfolio-Based Approach," *Proc. 7th Int'l Requirements Eng.: Foundation for Software Quality* (REFSQ 01), Essener Informatik Beitragë, Essen, Germany, 2001, pp. 202–213.
22. N.A.M. Maiden and C. Ncube, "Acquiring COTS Software Selection Requirements," *IEEE Software*, vol. 15, no. 2, Mar./Apr. 1998, pp. 46–56.
23. P.K. Lawlis et al., "A Formal Process for Evaluating COTS Software Products," *Computer*, vol. 34, no. 5, May 2001, pp. 58–63.
24. B. Boehm and C. Abts, "COTS Integration: Plug and Pray?" *Computer*, vol. 32, no. 1, Jan. 1999, pp. 135–138.
25. N. Storey, *Safety-Critical Computer Systems*, Addison-Wesley, Boston, 1996.
26. *Batneec Guidance Notes*, Environmental Protection Agency of Ireland, Johnstown Castle Estate, Wexford, Ireland, 2001; www.epa.ie/licences/batneec.htm.
27. N. Leveson, *Safeware: System Safety and Computers*, Addison-Wesley, Boston, 1995.
28. K. Wiegers, *Software Requirements*, Microsoft Press, Buffalo, N.Y., 1999.
29. S. Jones et al., "Trust Requirements in E-Business," *Comm. ACM*, vol. 43, no. 12, Dec. 2000, pp. 81–87.

For more information on this or any other computing topic, please visit our Digital Library at http://computer.org/publications/dlib.

About the Authors

Natalia Juristo is a full professor of computer science at the Universidad Politécnica de Madrid. Her research interests include requirements engineering, software engineering, the intersection of software engineering and knowledge engineering, and software process evaluation. She received her PhD in computer science from the Universidad Politécnica de Madrid. She is a senior member of the IEEE. Contact her at natalia@fi.upm.es; www.ls.fi.upm.es/UDIS/miembros/natalia.

Ana M. Moreno is associate professor at the Universidad Politécnica de Madrid. Her research interests include defining a conceptual model independent of the development approach, transforming textual requirements in object-oriented conceptual modeling, requirements engineering, and the intersection of software engineering and knowledge engineering. She received her PhD in computer science from the Universidad Politécnica de Madrid. Contact her at ammoreno@fi.upm.es; www.ls.fi.upm.es/UDIS/miembros/amoreno.

Andrés Silva is an assistant professor at the Universidad Politécnica De Madrid. His research interests include requirements engineering for emergent applications, viewpoint-based requirements engineering, and knowledge management. He received his PhD in computer science from the Universidad Politécnica de Madrid. Contact him at asilva@fi.upm.es; www.ls.fi.upm.es/UDIS/miembros/asilva.