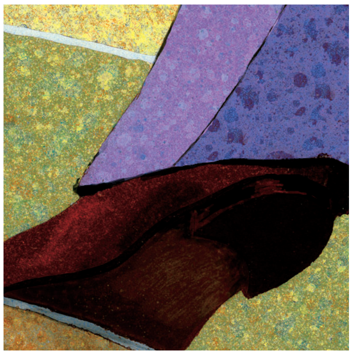
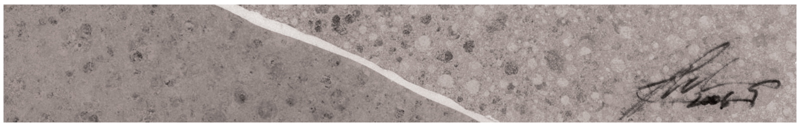
focus

guest editors' introduction



Why Are Small Software Organizations Different?



Ita Richardson, *University of Limerick*

Christiane Gresse von Wangenheim, *Universidade do Vale do Itajaí*

Small software organizations—independently financed and organized companies with fewer than 50 employees—are fundamental to many national economies' growth. In the US, Brazil, Canada, China, India, Finland, Ireland, Hungary, and many other countries, small companies represent up to 85 percent of all software organizations.[1,2] However, to persist and grow, small software companies need efficient, effective software engineering solutions. People often believe that good practices and solutions are expensive, time consuming, and targeted more toward large organizations, and therefore difficult to apply in small companies.

Considering the large percentage of small software organizations across the globe, relatively few publications present software engineering solutions focusing specifically on small software companies. In this special issue, we hope to address this gap. Considering their typical characteristics and limitations, how can small organizations apply software engineering methods, techniques, best practices, and tools to improve software quality and productivity without introducing unacceptable overhead?

Organizational challenges

Large and small software development companies face similar software engineering challenges. They need to manage and improve their software processes, deal with rapid technology advances, maintain their products, operate in a global software environment, and sustain their organizations through growth. However, they often require different approaches because of specific business models and goals, market niche, size, availability of (financial and human) resources, process and management capability, and organizational differences, among other things.

Small companies aren't just scaled-down versions of large firms.[3] Generally, they're extremely responsive and flexible, because that's their advertised competitive advantage. As Richard Daft noted, they typically "have a flat structure, with an organic and free-flowing management style that encourages entrepreneurship and innovation."[4] They usually focus on a market niche that large companies have disregarded, build components for other companies' products, develop innovative products, or offer services or maintenance for products that they or others produce. Unlike large companies, small companies don't have enough staff to develop functional specialties that would enable them to perform complex tasks secondary to their products. However, they might network with other small companies. Tight finances also constrain many small businesses, so they can't always afford to buy required expertise.

The special issue

We received articles for this issue from many countries, reflecting this topic's importance worldwide. The large number of submissions focusing on companies with fewer than 20 employees was an interesting revelation to us; often, researchers consider small organizations together with medium enterprises, not differentiating their specific characteristics. This can affect research results and ultimately the development agencies that are promoting small companies within their own national economic structure. Many submissions we received dealt with software process assessment and improvement, indicating ongoing work in developing national or tailored approaches for small organizations (see the sidebar "Process Initiatives for Small Organizations"). Other topics covered included agile methods, configuration management, reuse, and knowledge management. The four articles finally chosen for this special issue deal with process assessment and improvement, requirements prioritization, version control, and software tool support.

Process Initiatives for Small Organizations

In recent years, several initiatives have focused on small organizations' software processes.

The International Organization for Standardization set up a working group (ISO/IEC JTC1/SC7/WG24) on SE Life-Cycle Profiles for Very Small Enterprises (VSEs; companies with fewer than 25 employees). This group is establishing a common framework for describing assessable life-cycle profiles used in VSEs. It aims to let VSEs be recognized as producing quality software systems without the initial expense of implementing and maintaining an entire suite of systems and software engineering standards or performing comprehensive assessments.

The Software Engineering Institute initiated a CMMI in Small Settings project to provide approaches, tools, techniques, and guidance in small settings (defined in this project as organizations or companies of fewer than approximately 100 people and projects of fewer than approximately 20 people).[1] The SEI hosted the First International Research Workshop for Process Improvement in Small Settings in 2005.

Other national initiatives include

- In Brazil, the Associação para promoção de Exçelência do Software Brasileiro (Assoc. for Promoting Brazilian Software Excellence) developed the MPS.BR model (Melhoria de Processo de Software Brasileiro, or Brazilian Process Improvement Model)[2] based on the ISO/IEC 15504 standard and aligned it to the CMMI framework focusing on small and medium-sized software organizations.
- The Mexican Ministry of the Economy developed a standard[3] based on ISO/IEC 12207, including practices from ISO 9000:2000, CMMI, PMBOK, SWEBOK, and ISO/IEC 15504.
- In Ireland during 2006, Enterprise Ireland (the development agency with responsibility for small companies) initiated a three-year plan for developing and promoting software quality through improved processes.

Initiatives such as these demonstrate, among other things, the requirements of improved software processes in small software organizations.

References

1. *CMMI in Small Settings Toolkit Repository*, Software Eng. Inst., 2004; www.sei.cmu.edu/cmmi/acss.
2. *MPS.BR—Melhoria de Processo de Software Brasileiro* [Brazilian Process Improvement Model], SOFTEX, 2006; www.softex.br/mpsbr.
3. *Information Technology—Software—Models of Processes and Assessment for Software Development and Maintenance*, tech. report NMX-059-NYCE-2005, Ministry of the Economy, Mexico, 2005.

Process assessment and improvement

Several models or standards for software process assessment and improvement are available, including CMM and CMMI, and ISO/IEC 15504 (also known as SPICE). However, many small organizations remain unaware of these methodologies.[1] They also perceive these meth-

You might argue that in relatively small projects, we don't need structured techniques—so, what interest can they hold for small software organizations?

ods as expensive and time consuming and, therefore, difficult to perform.[5,6]

In relation to the most prominent models, new assessment methods tailored to the context of small software companies have been developed, such as, for example, in conformance with ISO/IEC 15504: Rapid (Rapid Assessment for Process Improvement for Software Development),[7] Spini (Software Process Improvement Initiation),[8] Mares (Método de Avaliação de Processo de Software),[9] and Spire (Software Process Improvement in Regions of Europe),[10] among others. Basically, these methods focus on process improvement assessments. They either assess a preselected set of processes or provide an initial step for choosing the processes to assess using a continuous representation, focusing generally on assessments up to Capability Level 3.

To stay within the CMMI framework while minimizing assessment time and cost, some small companies run a class B or C assessment[11] instead of a Scampi (Standard CMMI Appraisal Method for Process Improvement) class A assessment (www.sei.cmu.edu/cmmi/appraisals/appraisals.html). Class B and C appraisals are scaled-down assessments that focus, for example, only on a single project or process area and don't produce any formal ratings.

In this context, the article "Adept: A Unified Assessment Method for Small Software Companies" by Fergal Mc Caffery, Philip S. Taylor, and Gerry Coleman presents a cohesive model covering both plan-driven and agile development approaches. It enables a focused, tailored improvement path driven by the company's operational context and business goals.

Requirements prioritization

When developing products, companies must deal with the difficulties of eliciting, managing, and prioritizing requirements. They must select the most important functional and nonfunctional requirements from what might be a very large set of requirements. Project managers face questions such as

- What requirements give the customer the greatest benefit?
- Can we exclude any requirements from our product without sacrificing market share?
- What requirements should we include in initial and subsequent versions of our product?

They must prioritize the requirements to ensure that they develop the correct product for the customer, but they must choose them in such a way so that they can develop them incrementally if needed. Several authors proposed solutions to prioritization, most of which let groups of stakeholders determine, through a structured method, each requirement's "value." Such techniques aim to eliminate the development team's personal preferences from the decision-making process and to ensure that the team hears the customer's voice and includes it in requirements prioritization. These techniques include Quality Function Deployment[12,13] and the Analytic Hierarchical Process.[14,15] Joachim Karlsson, Claes Wohlin, and Björn Regnell evaluated methods for prioritizing software requirements.[16]

You might argue that in relatively small projects, we don't need structured techniques—so, what interest can they hold for small software organizations? However, in this issue, Jim Azar, Randy K. Smith, and David Cordes present a value-oriented prioritization framework based on Karl Wiegers' approach. The authors' case study demonstrates how a very small organization implemented the process successfully and how it contributed to that organization's growth.

Version control

When a small organization starts to grow, so does its need to maintain versions of its software. Configuration management ensures that the organization manages its versions of software products and their associated artifacts. Mary Beth Chrissis, Mike Konrad, and Sandy Shrum point out that "for some work products it may be appropriate to maintain version control (i.e. the version of the work product in use at a given time, past or present, is known and changes are incorporated in a controlled manner)."[17] Small organizations might see implementing version control as an investment with little return: they can't do any productive development during initial implementation. However, some companies lose a lot of time by not implementing it—for instance, when they have to retrieve old software or remove changes when maintenance issues arise. In fact, configuration management, the process area in which version control falls, is a CMMI Level 1 process, indicating its importance in supporting other processes in the organization.

But how can a small organization implement manageable version control? The case study presented by Jan Ploski, Wilhelm Hasselbring, Jochen Rehwinkel, and Stefan Schwierz in their article "Introducing Version Control to Database-Centric Applications in a Small Enterprise" deals with this topic. Having successfully implemented the system, they now plan to automate it. In their conclusion, they mention another issue that interests us: they point out the importance of small organizations linking up with local universities to benefit from research carried out there.

Software tools and environments

Today, developers can't carry out software engineering tasks without reasonable tool support.[18] SE tools and environments aim to assist in software development and maintenance or even to automate repetitive, well-defined actions. A large number and variety of such tools and environments exist, providing either support for individual tasks or integrated support in software engineering environments that encompass the complete life cycle. For example, they might include tools for requirements development and management, design, software construction, testing, documentation, configuration management, reengineering, reuse, and project management and measurement.

A step ahead are process-centered software engineering environments (PSEEs), which explicitly incorporate information on software processes and guide and monitor the user according to a defined process model. They can provide flexible support by considering the defined process itself as a changeable parameter. However, most PSEEs today are academic prototypes, which researchers have applied thus far only in a few industrial settings.

On the other side, industry has recognized that integrated environments that you can't adapt to a company-specific process don't enhance productivity or software quality.[19] And, in parallel, the software process itself has gained more attention. So, companies are increasingly choosing or even developing specific tools and working on their integration or adaptation to support a company-specific software process.

Rather than focusing on automation, PSEEs should facilitate the execution of such complex processes, often even in distributed environments. One current trend supports cooperative work and communication, so multiple users can work together in a controlled way.[20]

Today, lots of SE tools, both commercial and open source ones, either provide specific support or operate in an integrated manner. However, in general, we observed a lack of systematic evaluations or comparisons. A difficulty here might be the large number of tools and their high rate of change, which makes concrete, up-to-date evaluations difficult.

Information on SE tools or their usage specifically in small organizations is even more scarce. In this context, the article "An Open Source Approach to Developing Software in a Small Organization" by Ken Martin and Bill Hoffman presents their long-term experience in integrating tool support in a small software company. They focus principally on communication, revision control, build and release management, and testing.

Industry has recognized that integrated environments that you can't adapt to a company-specific process don't enhance productivity or quality.

The Point/Counterpoint debate

The question we chose for the point-counterpoint discussion is this: can small and large organizations apply the same software engineering techniques, given their specific characteristics and limitations? Larry Lumsden, chief technical officer of Cúram Software in Ireland, argues yes, and Wolfgang Strigel, president of QA Labs in Canada, argues no. Both authors have considerable experience dealing with small software organizations' software engineering challenges, so their differing views are worth reading.

What will the future bring? As this special issue shows, developers around the world are working on adapting software engineering solutions for small organizations, and the number of experience reports on such applications is increasing. Customized approaches will likely become more available. Furthermore, interest in research in small software companies seems to be increasing, so researchers' skills and experience are becoming more available in those settings. These factors will contribute to supporting small organizations as they apply software engineering solutions and help them operate more effectively and efficiently.

References

1. Qualidade no Setor de Software Brasileiro–Pesquisa 2001 [Quality in the Brazilian Software Sector–Survey 2001], Ministry of Science and Technology (MCT), 2001; http://ctjovem.mct.gov.br/index.php/content/view/34854.html (in Portuguese).
2. "Software Industry Statistics for 1991–2005," *Enterprise Ireland*, 2006; www.nsd.ie/htm/ssii/stat.htm.
3. D.J. Storey, *Entrepreneurship and the New Firm*, Croom Helm, 1982.

About the Authors

Ita Richardson is a senior lecturer at the University of Limerick and head of operations at Lero—the Irish Software Engineering Research Centre. She's also project leader on the Science Foundation Ireland-funded Global Software Development for Small to Medium Sized Enterprises project. Her research interests are software process improvement and global software development, particularly in small companies. She received her PhD in software engineering from the University of Limerick. She's a member of the IEEE Computer Society, the Irish Computer Society, and the British Computer Society and is a chartered engineer. She's on the editorial board of *Software Process Improvement and Practice*. Contact her at Lero—the Irish Software Eng. Research Ctr., Dept. of Computer Science & Information Systems, Univ. of Limerick, Limerick, Ireland; ita.richardson@ul.ie.

Christiane Gresse von Wangenheim is a professor at the Universidade do Vale do Itajaí (Univali) and a consultant at Incremental Tecnologia. Her research interests are in software process improvement, focusing on small companies. She's also a project management professional and assessor of the Brazilian Process Improvement Model MPS.BR. She received her PhD in production engineering from the Federal University of Santa Catarina and her PhD in computer science from the University of Kaiserslautern. She's a member of the IEEE Computer Society, the Project Management Institute, and the Working Group ISO/IEC JTC1/SC7/WG24—SE Life-Cycle Profiles for Very Small Enterprises. Contact her at Univali, Rod. SC 407, Km 04, 88122-000 São José/SC, Brazil; gresse@univali.br.

4. R.L. Daft, *Organisation Theory and Design*, 4th ed., West Publishing, 1992.
5. D.L. Johnson and J.G. Brodman, "Tailoring the CMM for Small Businesses, Small Organizations, and Small Projects," *Elements of Software Process Assessment and Improvement*, K. El Emam and N.H. Madhavji, eds., IEEE CS Press, 1999, pp. 239–259.
6. M.C. Paulk, "Using the Software CMM in Small Organizations," *Proc. Joint 16th Pacific Northwest Software Quality Conf. and 8th Int'l Conf. Software Quality*, 1998, pp. 350–360; www.pnsqc.org/proceedings.
7. T.P. Rout et al., "The Rapid Assessment of Software Process Capability," *Proc. 1st Int'l Spice Conf.*, Centre for Software Eng., Dublin City Univ., 2000, pp. 47–56.
8. T. Mäkinen, T. Varkoi, and M. Lepasaar, "A Detailed Process Assessment Method for Software SMEs," *Proc. 7th European Software Process Improvement Conf.*, 2000, s.1-14–1-16.
9. C. Gresse von Wangenheim, A. Anacleto, and C.F. Salviano, "Helping Small Companies Assess Software Processes," *IEEE Software*, vol. 23, no. 1, 2006, pp. 91–98.
10. M. Sanders, ed., *The Spire Handbook—Better, Faster, Cheaper: Software Development in Small Organisations*, Centre for Software Eng., Dublin City Univ., 1998.
11. W. Hayes et al., *Handbook for Conducting Standard CMMI Appraisal Method for Process Improvement (SCAMPI) B and C Appraisals (Version 1.1)*, tech. report CMU/SEI-2005-HB-005, Software Eng. Inst., 2005.
12. E. Ronney, P. Olfe, and G. Mazur, "Gemba Research in the Japanese Cellular Phone Market," *Proc. 12th Symp. Quality Function Deployment*, QFD Inst., 2000, pp. 358–374.
13. I. Richardson, K. Ryan, and E. Murphy, "Development of a Generic Quality Function Deployment Matrix," *Quality Management J.*, vol. 9, no. 2, 2002, pp. 25–43.
14. J. Karlsson and K. Ryan, "A Cost-Value Approach for Prioritizing Requirements," *IEEE Software*, Sept./Oct. 1997, pp. 67–74.
15. K. Wiegers, *Software Requirements*, 2nd ed., Microsoft Press, 2003.
16. J. Karlsson, C. Wohlin, and B. Regnell, "An Evaluation of Methods for Prioritizing Software Requirements," *Information and Software Technology*, vol. 39, 1998, pp. 939–947.
17. M.B. Chrissis, M. Konrad, and S. Shrum, *CMMI Guidelines for Process Integration and Product Improvement*, Addison-Wesley, 2003.
18. W. Harrison, H. Ossher, and P. Tarr, "Software Engineering Tools and Environments: A Roadmap," *Proc. Conf. Future of Software Eng.*, ACM Press, 2000, pp. 261–277.
19. R. Balzer and V. Gruhn, "Process-Centered Software Engineering Environments: Academic and Industrial Perspectives," *Proc. 23rd Int'l Conf. Software Eng.* (ICSE 01), IEEE CS Press, 2001, pp. 671–673.
20. G. Cugola and C. Ghezzi, "Software Processes: A Retrospective and a Path to the Future," *Software Process Improvement and Practice*, vol. 4, no. 3, 1998, pp. 101–123.