Agile Software Development in Large Organizations

While agile practices can match the needs of large organizations—especially for small, collocated teams—integrating new practices with existing processes and quality systems will require further tailoring.

Mikael Lindvall
Fraunhofer Center for Experimental Software Engineering, Maryland

Dirk Muthig
Fraunhofer Institute for Experimental Software Engineering

Aldo Dagnino
Christina Wallin
ABB

Michael Stupperich
DaimlerChrysler

David Kiefer
John May
Motorola

Tuomo Kähkönen
Nokia

In recent years, the use of, interest in, and controversies surrounding agile methods have all increased dramatically—as has anecdotal evidence for agile methods' effectiveness in certain environments and for specific project types. Exactly in which environments and under what conditions agile methods work remains unclear, however. A development team at Motorola, for example, noted that, "a plethora of subjective evidence exists to support the use of agile development methods on non-life-critical software projects."[1] Yet the team found no information about using the approach for its particular development focus: mission-critical software products.

This shows the need for more evidence that a new technology works in a certain context before developers promote and deploy it on a larger scale. Although most organizations have similar needs, the need to see compelling evidence before adopting new methods looms greater in large organizations because of their complexity and the need to integrate new technologies and processes with existing ones.

To further evaluate agile methods and their underlying software development practices, several Software Experience Center (SEC) member companies initiated a series of activities to discover if agile practices match their organizations' needs. Although each organization evaluated agile methods according to its specific needs, here we attempt to generalize their findings by analyzing some of their common experiences in the particular context of large organizations with well-established structures and processes.

We base this analysis on experience collected and shared among four SEC members—ABB, DaimlerChrysler, Motorola, and Nokia—and focus on the following areas:

- identifying the business drivers that led to the evaluation of agile methods,
- ascertaining whether their pilot projects reached their goals,
- articulating lessons learned regarding incompatibilities with the project environment, and
- determining their conclusions regarding future use of agile methods.

Four SEC meetings and one electronic workshop (http://fc-md.umd.edu/projects/Agile/3rd-eWorkshop/summary3rdeWorksh.htm) in which the member companies shared experience on the application of agile methods provide the core data for our analysis. The "Sharing Agile Expertise through the SEC" sidebar describes these meetings in greater detail. We also collected experience reports internal to the companies that provided input to the meetings and gathered additional information from the member companies to clarify and refine the results. We have referenced reports that are publicly available. Unattributed quotes refer to unpublished reports and presentations proprietary to individual companies that are not publicly available.

We must admit that the data collection does not

Published by the IEEE Computer Society
0018-9162/04/$20.00 © 2004 IEEE

Sharing Agile Expertise through the SEC

Allan Willey, Motorola Labs

For the past five years, several large global businesses have been sharing experiences in software development practices. The member companies founded the consortium that makes this possible in 1999 so that they could share their experience while protecting its proprietary aspects. The Fraunhofer Center for Empirical Software Engineering, Maryland (FC-MD), and the Fraunhofer Institute for Empirical Software Engineering (IESE) in Germany agreed to facilitate the consortium and provide the legal umbrella to make this possible.

The member companies of the Software Experience Center consortium aim to improve their software competencies and development practices by actively sharing experiences with one another. The current SEC members are ABB, Boeing, DaimlerChrysler, Motorola, and Nokia. Fraunhofer directors Victor Basili and Dieter Rombach have served to guide the SEC, providing enormous benefits by assuring that key issues are adequately addressed.

Each of the member companies hosts three-day meetings, held semiannually on a rotating basis. The SEC Steering Committee, which draws its members from the participating companies and two Fraunhofer organizations, develops the program of topics for each meeting. An enduring theme has been the sharing of strategies for improving engineering productivity and product quality.

Topics addressed at these meetings have covered the gamut of current software engineering and system development issues, including

- agile methods,
- software process improvement,
- software and system quality metrics,
- software subcontractor management,
- engineering project management,
- requirements engineering,
- transition from the SEI CMM to the CMMi,
- Six Sigma and system development,
- R&D results deployment,
- system and software architectures, and
- product line practices.

Each company brings appropriate presentations to the three-day meetings. The steering committee then plays a key role in identifying and selecting topics that are of broad interest to the membership, so that each member can participate.

During the meetings, member companies make company-specific presentations on the selected topics. These presentations are covered by nondisclosure agreements. Each attendee receives a complete set of all presentations offered at each meeting. These company representatives then disseminate the results when they return to their day-to-day activities.

The information shared at SEC meetings has a variety of uses. For example, in one member company, the SEC Steering Committee representative prepared a trip report and posted it to an internal Web site, along with hot links to the presentations. At another company, meeting attendees made their trip report to an internal committee at a regularly scheduled internal meeting. All company participants have been satisfied with the extent of sharing they see and the information they gain through the exchange.

Allan Willey *is a fellow of technical staff at Motorola Labs. He received an MS in management sciences from George Washington University. Contact him at willey@motorola.com.*

follow the ideal scientific process: The data was defined after the pilot projects ended, it is mostly qualitative, and different people collected it. However, the collected data, drawn from about 15 different pilot projects influenced by eXtreme Programming[2] (XP) across four different organizations, provides a broad overview of the use of agile methods in large companies.

BUSINESS DRIVERS

Many small organizations have shown interest in agile methods because they seek alternatives to the traditional software development methodologies, which they find too cumbersome, bureaucratic, and inflexible. They also feel pressure to produce more at lower costs.

We found that these same needs drive large organizations as well. One Motorola team identified a need shared by all organizations in this study when they observed: "Software development teams face a continuous battle to increase productivity while maintaining or improving quality." This is indeed what drives most organizations to look for new ways to develop software.

Problems related to requirements supplied another common theme, and developers identified them as strong drivers among the four organizations. For example, because mandated ship dates require that software development begin after defining only a portion of the requirements, the organization must look for better ways to manage projects for which requirements are not yet fully specified.

Another problem typically arises when the requirements are passed along to the development team at a high level. This can make it difficult to decompose the requirements up front into detailed software specifications. This problem drives the organization to find ways to better understand the end users' real needs.

In addition, work on the specifications is time-consuming, and the resulting specifications are often obsolete by the time they are finalized.

In addition to the problems associated with ill-defined and high-level specifications, rapid changes

Before introducing new practices, the development team must understand their effects and implications.

in the requirements and other environmental factors drive the need to adapt swiftly to keep pace with evolving markets and technologies. This leads organizations to seek a flexible process capable of adapting to volatile requirements. The need to show early progress to the customer and present upper-level management a first version quickly also promote this behavior.

While *any* organization that develops software could encounter these problems, some are more likely to occur in large organizations. For example, some participants reported their disappointment with heavy process approaches and current quality systems that are too generic and complex to provide good support. Large organizations are more likely to implement defined development processes, including a system for assuring quality throughout the software development process. These processes and systems often put constraints on the development team, limiting what development practices they can and must use, which affects how quickly they can develop software.

When pressured to quickly deliver a product to capture a market opportunity, one software team reported that they felt they must fight two battles at the same time—one to develop a product in a short time and the other to fulfill the quality system's requirements. Finding alternative ways to develop software faster and more flexibly without compromising these organizations' high quality standards thus becomes essential.

APPLYING AND EVALUATING AGILE PRACTICES

Before introducing new practices, the development team must understand their effects and implications. Although the organizations participating in this study were aware of reports indicating that agile methods do indeed work, they remained unconvinced that these practices would work for them. For example, they needed to evaluate whether agile practices would increase productivity and reduce cycle time while maintaining the current level of quality and maintainability.

In addition, they questioned whether an established company could use agile practices to develop large, complex, safety-critical systems that would be maintained for decades. They also wanted to assess whether they could use agile practices to shrinkwrap product development.

To evaluate agile methods, these large organizations have conducted numerous pilot projects, studies, and other activities. The pilot projects we describe here offer a representative sample of how these organizations evaluated agile methods.

The pilots all used and tailored XP in some way, either using XP as is, choosing selected XP practices and incorporating them into their regular processes, or using XP terms to refer to the practices they already used. To cover the numerous variants, we simply say they were all influenced by XP. We emphasize that XP and agile methods are not interchangeable terms, but that XP is the most commonly used agile method today because it is the best documented and thus the easiest to implement.

Researchers at ABB applied and evaluated agile practices in several different ways. For example, they conducted a systematic study as a pilot XP project over 10 weeks. Along with piloting XP, the project members also evaluated software development lifecycle models and methodologies. In addition, on three occasions the developers used ADEPT, an evolutionary in-house lifecycle model that combines agile and traditional practices, in their evaluations of agile practices. ADEPT incorporates selected XP practices and is recommended for use when teams cannot implement a complete agile lifecycle model.

DaimlerChrysler researchers have made several partial attempts to apply agile methods, and they have extended their development approaches with selected XP practices. Typical applications include administrative, interactive software portal, Web, and embedded projects. DaimlerChrysler has conducted several XP practice studies. For example, the corporate researchers studied a software project in a business unit that embarked on a conventional project and later switched to XP.

At Motorola, researchers have applied agile practices in the form of XP several times. In one 18-month study, four separate teams developing embedded systems participated in a pilot project using XP. In another study, developers used XP for a project to develop a large safety-critical, real-time system with extremely high quality requirements.[1]

At Nokia, developers introduced XP practices on several projects. In addition, on at least three projects, developers defined and applied an in-house method that combined XP and traditional practices. In general, Nokia has adopted XP practices for projects when deemed appropriate.

APPLICATION RESULTS

An organization must consider several important aspects when introducing agile methods. From a business viewpoint, delivering high-quality software on time and within estimated cost and effort is essential. In the context of applying agile prac-

tices, understanding which aspects of the process became more agile is equally important.

Other aspects to consider include how difficult introducing and sustaining the practices will be, their desirable and undesirable side effects, and employees' satisfaction with using the methods. Based on internal acceptance testing and preliminary product test results, the ABB pilot project indicates that the resulting product exhibits higher quality than previous releases. In addition, the product meets its required scope, and the project only slightly exceeded the required delivery time.

The resulting code exhibited high quality, thanks both to pair programming, which prevented gold plating and complex design, and to automated tests, which prevented introducing errors or having them remain undiscovered. Continuous integration helped improve the quality of changes by quickly uncovering integration problems.

The increased agility was reflected in the speed with which the developers implemented change requests. Team members encountered fewer unpleasant surprises at the development cycle's end, and they shared a common view of the project. In addition, pair programming helped to spread information and knowledge throughout the team, and daily stand-up meetings improved work discipline.

Developers noted that they could easily learn XP without making major investments in tools or training.

DaimlerChrysler's experience demonstrated that using agile methods combined with constant testing and other classical QA techniques produced high-quality software throughout its projects. The practice of developing rough specifications instead of detailed ones also generated cost savings, which decreased the need for specification updates and resulted in further savings.

Agility increased in several ways. For example, flexibility grew through faster responses to changing requirements, and development velocity increased as implementations finished more quickly.

One DaimlerChrysler project team reported that using XP cut costs while achieving high levels of quality and customer satisfaction. The development team considered the project a success, due in large part to the impact of adopting XP practices. This project showed that adapting agile elements for use in a conventional project environment could lead to noticeable cost savings while maintaining high quality. Further, the communication between the project members and the customer improved substantially thanks to a variation on the onsite customer practice and the planning game practice.

Although the pilots conducted at Motorola all followed slightly different processes, they showed relatively consistent results based on data collected on both qualitative and quantitative aspects of the process. The data shows that the projects achieved quality levels comparable to or better than other processes. Defect density, for example, measured significantly better than the division average.

Developing rough specifications instead of detailed ones generated cost savings, which decreased the need for specification updates.

The development teams used custom formal technical reviews to ensure that the project produced a maintainable design. When surveyed, 82 percent of the pilot developers believed that the design and code generated using XP resulted in understandable, maintainable, and extensible software. In addition, 88 percent believed that the deliverables produced using XP would be adequate for future development and maintenance. The pilot teams also experienced a significant increase in engineer productivity compared to similar teams within Motorola that used different development processes.

A survey distributed to all developers involved in the pilot—29 respondents—showed that team morale increased, the learning curve for new engineers shortened, and test coverage improved. Among the respondents, 85 percent indicated that they enjoyed using XP, and 80 percent reported that they had more confidence in the design and code generated while they used pair programming than while they worked alone.

These pilot projects succeeded because the team produced and tested code earlier. In addition, they developed the system and executed it in smaller pieces. Altogether, this meant that the teams could detect and fix problems earlier.

All pilot projects this study covers had similar positive experiences. The subjective and objective measures indicate that these projects succeeded in terms of increased agility and made improvements to one or more of the following attributes: customer satisfaction, quality, productivity, and cost. In addition, most developers involved in the projects had positive experiences, team morale increased, and introducing the practices did not prove costly.

Overall, the teams applying the XP-influenced practices viewed their experience as very successful. These findings resemble other experiences with XP.[3-6]

LESSONS LEARNED

The greatest challenge to adopting agile practices involves integrating each pilot with the project environment's existing processes.

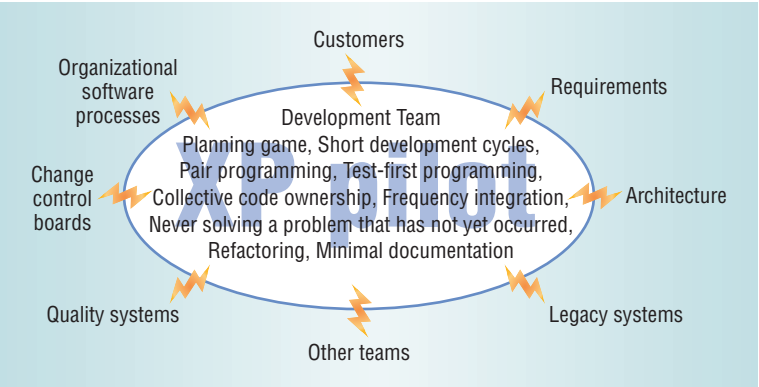

Figure 1. Tailoring XP to the organization. The lightning symbols show the incompatibilities between the XP pilot project and the environment. To maximize efficiency, organizations must resolve these incompatibilities.

Tailoring XP

All four organizations learned the absolute necessity of tailoring XP to their particular requirements. The experience at Nokia showed that introducing XP in a large organization without extensive tailoring is generally infeasible. Motorola had a similar experience: XP is not a one-size-fits-all software development process.

Figure 1 shows that the challenges lie not in the agile project itself and the new practices it puts in place, but in the interface between the new and existing practices. In a large organization, a project cannot be truly independent, but must interact with and follow the rules of the organization overall.

The amount of tailoring varied from project to project. In one example, a Motorola team tailored the process by modifying the customer role slightly, creating a baseline architecture, adding and modifying some documentation, and adding some formal reviews. Another Motorola team adopted some XP practices, dropped others, and supplemented others with traditional practices.[1] In this case, the mix of traditional and agile practices resulted in a situation in which "an outsider could easily interpret the process as a CMM-based process with some of the XP practices added to it."[1]

While some of the tailoring efforts aimed to make the practices work for a particular pilot project, most challenges related to making the pilot project work well within the organizational environment. DaimlerChrysler, for example, learned that it is essential to modify XP practices so that they align with the environment and the rest of the project. ABB reports a similar experience, noting that most difficulties were found in the project environment, not in XP. Motorola had a comparable experience and reports that the project postmortem revealed that—although some traditional practices, such as the change control board (CCB), clashed with XP practices—few defects could be traced directly to XP.[1]

We think this summarizes the differences between a large organization and a small one. Individual projects in a large organization often depend on their environment in several ways. For example, work is often distributed across several teams.[7] The team must be able to communicate and coordinate with other teams in the organization, and the developed software must integrate smoothly with a larger software system. In addition, the team must also fit into the standard processes and quality systems defined by the organization.

Cross-team communication support

Tailoring XP practices and adding support for cross-team communication presents an important lesson, one particularly prominent at Nokia.[7] Large organizations often distribute teams across several physical locations. However, XP practices aim to increase a software development project's agility by collocating the team. Consequently, XP practices do not address problems arising from communication and coordination between multiple teams.[7] This creates a need for more formal communication, such as meetings and documentation. As a result, communication between teams is less effective than within teams, creating additional developer overhead[7] and decreasing each project's agility.

Cultural differences between teams add to this problem. At Motorola, for example, project management noticed on several occasions that XP teams have difficulty interfacing with teams that use traditional development processes. For example, when two types of teams shared work on an interface, the XP team wanted just one or two pieces of the interface to work with at a time, while the traditional team wanted to develop and review the entire interface before providing it to the XP team.

Nokia believes that one solution to this problem lies in minimizing the need for cross-team communication. This is possible when, for example, each team is developing independent subsystems and occupies one physical space. Nokia's experience, however, shows that achieving and maintaining this alignment and architecture can be difficult in a large organization. Even though it might be possible to achieve such an alignment once, the architecture will evolve, causing loss of alignment. Determining how developers can reduce communication structure complexity to minimize the need for cross-team communication while maximizing synergies within teams remains an open question. Another approach that Nokia has been exploring to overcome this problem uses a continuum of facilitated cross-team workshops. Several projects at Nokia reported that these workshops, which amass people from different parts of organizations to perform a specific, well-defined task, can be used effectively to solve issues that span multiple teams.[7]

Refactoring and CCB clashes

XP encourages the somewhat controversial practice of *continuous refactoring*. This practice can easily clash with existing quality control systems, such as using a CCB to oversee management of changes to the source code. Refactoring encompasses changes to the system that leave its behavior unchanged and enhance its quality. These changes include simplicity, flexibility, understandability, and performance. XP's reliance on collective ownership means that any developer can change any line of code to refactor it. Refactoring, an integral part of XP, makes changing the codes easier, thereby allowing the implementation of changed customer requirements without breaking the design.

Most reports on refactoring consider it beneficial. One Nokia team even considered the practice a factor in a project's success "because the architects stayed with the project, refactored the architecture continuously and accomplished the survival and evolution of the architecture."[8] One Motorola project, however, encountered risks when large refactorings created some significant project defects.[1]

The controversy surrounding refactoring arises, however, because it runs contrary to the commonly applied practice of "if it isn't broken, don't fix it." In a culture that encourages a "get it right the first time" approach to development, many see the need to refactor as a process failure.[1]

Refactoring can also easily clash with using CCBs, which manage and limit code changes. A Motorola team reported that the refactoring clashed with the CCB's desire to minimize code base changes. To reconcile refactoring and the CCB, Motorola introduced several process modifications. For example, management encouraged each developer to think of refactoring ideas but to ask permission from the CCB before implementing them.[1]

Continuous integration and CCB clashes

Continuous integration is an XP practice that resembles refactoring in how it interacts with existing processes. Experience shows, for example, that this practice can easily clash with the CCB. Continuous integration means that developers should integrate and release code into the code repository whenever possible, at least every day. Continuous integration contributes to a project's agility by detecting and removing integration-related defects early and by dividing the integration work into smaller, easier to manage chunks.

Most projects reported positive experiences with continuous integration. ABB's experience, for example, showed that continuous integration combined with small releases guarantees the constant availability of an executable system. As a result, the team could always deliver working software when necessary.

Motorola experienced some difficulties when the CCB began exercising its power to control which changes were integrated. The board met weekly to plan the next build, so integrations took place weekly. This clash with the CCB decreased the project's agility. On one occasion, the CCB postponed integration of a minor defect fix. Meanwhile, the code underwent significant changes before the CCB approved the fix. By this time, the file version with the defect fix differed significantly from the mainline version and, consequently, the merge became nontrivial and had to be done manually.[1]

Large organizations often follow defined software processes, which can result in double work when projects apply new practices.

Organizational software processes

Large organizations often follow defined software processes, which can result in double work when projects apply new practices that have not been well integrated with these traditional processes. This applies to both input and output from the project and raises the following issues.

Scope and delivery planning. A team at ABB experienced double work when traditional processes overlapped or conflicted with agile practices. Traditionally within ABB, the program defined the scope and delivery time for the project in advance, as it did for the XP pilot. The program, for example, developed plans up front, with little or no development team participation. The program also required that the documentation be frozen before design and implementation started. Developing plans without involving the team and freezing documentation clashes with XP's core ideas.

Traditional requirements management. By design, XP expects coarse input requirements, but ABB's traditional approach to running programs that applied XP required delivering detailed requirements to the development project in advance. These requirements did not take the form of user stories, had not been defined with involvement from developers, and were seldom accurate by the time development actually started. For the XP pilot, this resulted in double work because the project team analyzed and decomposed the market requirements twice, first into traditional project requirements at the program level, then into user stories and engineering tasks during the pilot itself.

Traditional acceptance test management. XP expects acceptance tests to be run continuously during development, but ABB traditionally calls for the

Having mature processes already in place ensures that efforts to become more agile do not turn a mature organization into a chaotic one.

project to design and implement the code first, then let another team perform the product acceptance test. This procedure caused the developers to view the acceptance tests done in the XP pilot as internal only, thus they required another round of acceptance tests after the pilot had been finished. The double acceptance test occurred because customers and quality systems often require running such tests independently, with little or no developer involvement. In addition, testers often perform these tests on a system for which many different teams may have developed the components. Thus, the acceptance test the project conducts serves as a test of the component.

Quality management. XP asserts that when properly used, pair programming eliminates the need for formal reviews. But eliminating formal reviews clashes with traditional quality systems. In most cases, the pilot projects found that they needed an additional layer of quality assurance.

One Motorola team, for example, introduced pair programming and replaced the mandated formal technical reviews with informal reviews.[1] When the team later reviewed test cases formally, they found missing tests and identified ways to improve the test suite.

In Motorola's complex environment, a pair of people will not be able to consider all effects on the entire system, which makes it unlikely that pair programming will eliminate all mistakes during coding. Further, project management deemed valuable the discussions that take place in reviews involving developers with many different viewpoints. They thus concluded that formal code and test case reviews can complement pair programming. Motorola also prevented another of its teams from skipping the rigorous review process because that group developed products that could affect public safety.

DaimlerChrysler expected that agile processes would not mesh with the project's more conventional environment. This is exactly what happened: Both the control and quality management departments demanded the same documentation as proof of project progress for this special case, just as they would for any standard project. The project partners therefore decided to treat the project as a conventional one on the outside, satisfying all requirements defined by quality management. For example, the team set up quality gates and quality plans in accordance with standard templates. As usual, the contractor kept track of its own project performance, such as engineering and manpower. Although this would not have been necessary in XP, the procedure did help avoid conflicts with the control department and other organizational units.

FUTURE PROSPECTS

Based on this experience, ABB concluded that any organization that develops manageable system elements with small teams could try XP. Further, it determined that nothing prevents large organizations from trying XP on a small scale. As most XP practices can, by themselves, be useful in traditional development projects, they can become part of the toolbox offered to projects. Some practices have already spread outside the development team. A broader implementation of XP, however, would require changes to ABB's culture and current quality system.

DaimlerChrysler acknowledges that mature processes require activities such as quality management, documentation, and measurement. It also realized that reducing time to market has become increasingly difficult. The company concluded that agile practices can help *mature* organizations become more flexible. Having mature processes already in place will ensure that efforts to become more agile do not turn a mature organization into a chaotic one. DaimlerChrysler thus identifies agile methods as another tool in the software process improvement toolbox that includes, for example, the SEI's Capability Maturity Model.

The pilot projects' success convinced the Motorola team that it is possible to use XP to develop large, complex, safety-critical systems with long life cycles. Motorola's developers note, however, that integrating an agile process into a company with a culture that favors more traditional development processes can be difficult. The standard XP process needs tailoring to better mesh with an organization's specific circumstances. Carefully introducing agile processes yields positive results, however.

Nokia notes that small software development teams are more productive than large ones. Thus, they strive to apply the most appropriate software method for the task at hand and view agile methods as another tool in the software process improvement toolbox.[9] Nokia decided that XP works best for small, independent, collocated projects and that using selected agile techniques will become increasingly common. Agile methods will primarily influence other processes. Hybrid processes of different agile influence levels will be the primary means for applying agile development ideas. To achieve orga-

nizational agility, Nokia has defined a set of agile software engineering patterns that organizational units can use to select agile practices that fit them instead of trying to apply a one-size-fits-all solution.

Based on the experiences of the organizations we have studied, we believe agile practices match the needs of large organizations, especially for small, collocated teams. Even so, integrating new practices with existing processes and quality systems that govern the conduct of software development requires further tailoring. The challenge here lies not in applying agile practices to a project, but in efficiently integrating the agile project into its environment. To fully benefit from agile practices, organizations must better define the interfaces between the agile team and its environment, thus avoiding the double work caused by the conflict between agile practices and traditional ones.

These obstacles will not stop large organizations from using agile methods, especially as the promising results in pilot projects increase interest in them. Now that the hype about agile methods has been substantiated with real-world results, many project teams will view agile methods as a useful resource. As these projects identify a few practices that suit them well, they will then implement them as part of their regular processes. By using this hybrid approach, these organizations can maintain existing quality systems while becoming more agile so that they can serve their customers better. ■

Acknowledgments

We thank the following for contributing to this article: at Motorola, Azeem Ayoob, Matthew Baarman, Jason Bowers, Erik Melander, Andrij Neczwid, David Noftz, Rekha Raghu, and Jerry Drobka; at DaimlerChrysler, Jan-Peter van Hunnius and Kurt Schneider; at Nokia, Jouni Jartti, Kari Känsälä, and Jari Vanhanen. We also thank the SEC Steering Committee that made it possible to write this report: Victor Basili, Fraunhofer Center for Experimental Software Engineering Maryland, and Dieter Rombach, Fraunhofer Institute for Experimental Software Engineering; Martin Bollinger and Manfred Schoelzke, ABB; Thilo Schwinn, DaimlerChrysler; and Allan Willey, Motorola. We also thank the following people for helping improve the article: Patricia Costa, Forrest Shull, and Roseanne Tesoriero Tvedt for ideas and feedback; Ioana Rus and Michelle Shaw for reviewing the text; and Jen Dix for proofing.

References

1. J. Bowers et al., "Tailoring XP for Large System Mission Critical Software Development," *Proc. 2nd XP Universe and 1st Agile Universe Conf. on Extreme Programming and Agile Methods*, Springer, 2002, pp. 100-111.
2. K. Beck, *Extreme Programming Explained: Embracing Change*, Addison-Wesley, 1999.
3. D. Karlström, "Introducing Extreme Programming—An Experience Report," *Proc. 3rd Int'l Conf. Extreme Programming and Agile Processes in Software Eng.*, Springer, 2002, pp. 24-29.
4. J. Grenning, "Launching Extreme Programming at a Process-Intensive Company," *IEEE Software*, Nov./Dec. 2001, pp. 27-33.
5. C. Poole and J. Huisman, "Using Extreme Programming in a Maintenance Environment," *IEEE Software*, Nov./Dec. 2001, pp. 42-50.
6. B. Rumpe and A. Schröder, "Quantitative Survey on Extreme Programming Projects," *Proc. 3rd Int'l Conf. Extreme Programming and Flexible Processes in Software Eng.*, Springer, 2002, pp. 95-100.
7. T. Kähkönen, "Agile Methods for Large Organizations—Building Communities of Practice," *Proc. Agile Development Conf.* (ADC 04), IEEE CS Press, 2004, pp. 2-11.
8. J. Vanhanen, J. Jartti, and T. Kähkönen, "Practical Experiences of Agility in the Telecom Industry," *Proc. 4th Int'l Conf. Extreme Programming and Agile Processes in Software Eng.*, Springer, 2003, pp. 279-287.
9. K. Känsälä, "Good-Enough Software Process in Nokia," LNCS 3009, Springer, 2004, pp. 424-430.

Mikael Lindvall *is a scientist at the Fraunhofer Center for Experimental Software Engineering, Maryland. His research interests include agile methods, software process improvement, software architectures, and experience and knowledge management. Lindvall received a PhD in computer science from Linköping University, Sweden. Contact him at mikli@fc-md.umd.edu.*

Dirk Muthig *is a department head at the Fraunhofer Institute for Experimental Software Engineering. His research interests include product line engineering, system architectures, and variability management. Muthig received a PhD in computer science from the Technical University of Kaiserslautern, Germany. Contact him at dirk.muthig@iese.fraunhofer.de.*

Aldo Dagnino *is a senior principal scientist in software engineering at the US Corporate Research*

Center of ABB. His research interests include software architectures, software processes, and knowledge-based technologies. Dagnino received a PhD in systems design from the University of Waterloo, Canada. Contact him at aldo.dagnino@us.abb.com.

***Christina Wallin** is a software engineering consultant. She was previously at ABB and is now at Process Key. Her research interests include software development processes. Wallin received a Licenciate in computer science from the University of Mälardalen, Sweden. Contact her at christina.wallin@processkey.se.*

***Michael Stupperich** is a senior researcher at DaimlerChrysler. His research interests include engineering processes for embedded systems, distributed software engineering, and agile process models. Stupperich received a Diploma in computer science from the University of Karlsruhe. Contact him at Michael.Stupperich@DaimlerChrysler.com.*

***David Kiefer** is an engineering manager at Motorola. His research interests include public safety communication systems with an emphasis on mission-critical voice over IP. Kiefer received a BS in computer engineering from the University of Illinois at Chicago. Contact him at David.Kiefer@motorola.com.*

***John May** is a principal staff engineer at Motorola. His research interests include agile methods, software productivity metrics, and call processing design patterns. May received an MS in engineering from Cornell. Contact him at John.May@motorola.com.*

***Tuomo Kähkönen** is a process development manager at Nokia Technology Platforms. His research interests include process modeling, process assessments, and agile software development methods. Kähkönen received an MS in industrial engineering and management from Lappeenranta University of Technology, Finland. Contact him at tuomo.kahkonen@nokia.com.*