Handshaking with Implementation Proposals: Negotiating Requirements Understanding

Samuel Fricker, *University of Zurich and Fuchs-Informatik AG*

Tony Gorschek, *Blekinge Institute of Technology*

Carl Byman, *ABB*

Armin Schmidle, *ABB Switzerland*

A bidirectional process for agreeing on product requirements proves effective in overcoming misunderstandings that arise in the traditional handoff of requirements specifications to development teams.

Requirements engineering focuses on good specification practices but has yet to find working solutions for effective requirements communication. Inadequate communication and tacit assent to a demanding customer's requests make it hard to fully understand a project's requirements.

We're using a negotiation process, called *handshaking with implementation proposals*,[1] to communicate requirements effectively—even in situations where almost no written requirements exist and where distance separates the customer from developers. Handshaking is an efficient technique that uses architectural options[2] as a way to understand requirements, to make implementation decisions that create value, and to establish the foundation for a stable project. The handshaking process supports the communication between a company's product management and its development organization, with the former acting as a customer of the latter.[3]

We describe the communication challenges, solutions, and lessons learned in developing the handshaking process and the results of applying it at 10 European and Asian project development sites of Asea Brown Boveri (ABB) and Danaher Motion Särö (DHR). The "Industry Partners" sidebar characterizes these organizations.

Background

On many occasions we found product managers who felt unable to convey the desired meaning of their requirements and architects confronted with requirements that were too fragmentary for sound solution design. This resulted in requirements cycle times that senior management considered unacceptable. In this difficult situation, some product managers understandably began architecting the solution themselves, turning to increased control in an attempt to ensure rapid development of an acceptable product.

Development teams responded, sometimes fiercely, to this micromanagement and loss of autonomy. Architects stopped accepting requirements from product managers at great personal risk of being laid off. In their eyes, it wasn't product management's job to design the product. Furthermore, such design decisions risked breaking the product or weren't realizable within a meaningful timeframe.

0740-7459/10/$26.00 © 2010 IEEE

Handshaking with implementation proposals was developed as an answer to requirements communication problems. To be useful for industry product development, the communication approach had to be practical and effective for both small-scale collocated and large-scale distributed development.

Good-Enough Requirements

According to a widely held belief, a development project's success is highly dependent on its requirement specification's quality. Standards encourage requirement specifications to be correct, unambiguous, complete, consistent, ranked, verifiable, modifiable, and traceable. We support the view that documentation should be "as good as possible," but only to the extent that quality isn't penalizing the engineering processes.

Requirements specifications should be good enough and adapted to the situation at hand. Inadequately specified requirements lead to ambiguity and misunderstandings that cause large corrective costs down the development road. However, too much detail and quality improvement retards the delivery of development results while also increasing specification costs and unnecessarily constraining the solution space.

But what are good-enough requirements? Answers from experienced developers and product managers helped us identify some criteria.[4] The developers expected the requirements they received to be valid and traceable back to real customer needs and company strategy, to be selected for implementation on a priority basis, and to be consistent and stable for the targeted market release.

Experienced product managers adapted specifications to their receivers—namely, the development teams. They chose to collaborate directly with the team and stakeholders to elaborate new-to-the-world product features before they specified the identified requirements. They specified in detail features known to stakeholders but new to the team and provided access to domain experts who could support the team in properly interpreting these requirements. However, they also assumed that requirements considered standard for a given product were obvious, and they specified them only coarsely, if at all. They expected the development team to refine the resulting incomplete specification.

Even though these criteria were useful in establishing some aspects of good-enough requirements, we came to consider requirements quality as a moving target.

Industry Partners

Asea Brown Boveri (ABB) is a leader in power and automation technologies that enable utility and industry customers to improve their performance while lowering their environmental impact. The ABB group of companies operates in about 100 countries and employs more than 110,000 people.

Danaher Motion Särö (DHR) develops software and hardware equipment for navigation, control, fleet management, and service of automated guided vehicle (AGV) systems. More than 50 AGV system suppliers worldwide use DHR technologies and expertise in their own products. The headquarters and R&D center are located in Särö, Sweden, with 85 employees.

Handshaking with Implementation Proposals

We addressed the moving target of good-enough requirements by replacing requirements handoff with a bidirectional communication process. In this process, product management takes the customer role and uses requirements to control the development results. The development team takes the supplier role, proposing designs and their impacts to communicate their intentions. Requirements are "good enough" if the customer accepts the planned solution.

To increase communication efficiency, the customer tailors the requirements specification to a specific supplier by investing the most effort in defining novel requirements and by specifying deltas toward already-known requirements. If the supplier lacks important expertise, the customer assists in finding and building competence in this area. Focusing on the supplier's needs requires knowledge of the supplier's expertise, but it also concentrates the specification effort where it's most needed and mitigates misunderstandings.

To ensure solution acceptance, the supplier generates rapid feedback by proposing the design for those product *themes* that are critical for stakeholder satisfaction or hard to correct when committed to. A theme might relate to a product feature, an important part or subsystem of the solution, or a development increment. Its source is often a product roadmap or an architectural concern, and it is defined just broadly enough to explore design alternatives and their impacts on requirements and project planning.

The customer reviews the proposals and selectively adjusts requirements or suggests product design changes when catching wrong assumptions and misunderstandings. In communicating these changes to the supplier, the customer includes domain knowledge and business rationales that motivate them. This rapid feedback builds on lessons

Title:	Event Logging
Requirements:	RID-110 Log user activities, RID-130 Event subscription, RID-190 ■, RID-200 ■, RID-230 ■,
Assumptions:	Compliance to regulation ■ needed.
Design:	System structure (per drawing to the right). Event logger interface: Function_EL1 ■, Function_EL2 ■. Host application support functions: Function_HA1 ■, Function_HA2 ■. Event logger behavior: ■
Impact:	Minimum ■ Kbytes memory in field device. Minimum ■ Mbytes memory in supervision device. Well-defined interfaces ease source-code integration into existing devices.
Dismissed alternatives:	None investigated.
Open issues:	Evaluate impact on third-party devices. Workshops for defining integration strategy and effort estimation.
Scope:	All devices from current portfolio.

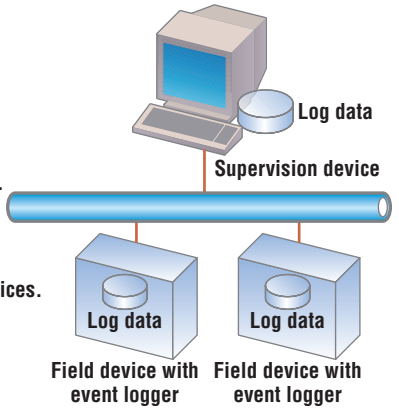

Figure 1. Industry example of an implementation proposal. The brown boxes represent proprietary data from a real implementation proposal.

from prototyping and iterative development.[5,6] It lets suppliers express their intentions as early as possible, with minimal but sufficient formality and with the customer in mind.

Handshaking assists such requirements communication with a template, the implementation proposal, for documenting the supplier's design proposals. It also provides a straightforward negotiation process for agreeing on the proposals. The negotiations engage the customer and supplier in a learning process for understanding what constitutes good-enough requirements. Observing how requirements are interpreted and matched with product design lets the customer gauge the supplier's expertise and resources. This knowledge enables further tailoring of the requirements specification. The more the communication partners handshake, the better they become at anticipating problems of understanding and acting in an increasingly proactive manner.

Implementation Proposals

Implementation proposals support communication by providing a form for recording architecture decisions[7] and design rationales.[8] They document requirements understanding by providing traceability between customer requirements and supplier design decisions for a given theme. Figure 1 shows an example implementation proposal extracted from a seven-page document.

A basic implementation proposal contains three attributes that relate the design to the requirements for a given theme. Several attributes can extend this basic template to support effective requirements negotiation and project planning. Table 1 summarizes these attributes.

The *basic attributes* are mandatory for effective handshaking. They establish traceability between customer interests and supplier intentions for a given theme. *Negotiation attributes* enrich the argumentation for understanding requirements correctly by documenting investigated alternatives, by revealing assumptions and expected impact of the planned product, and by capturing any remaining open issues that need agreement. *Planning attributes* support project planning by defining how the solution will be realized and its costs.

Companies can tailor the template to support company-specific processes, adding attributes such as identifiers, executive summaries, and revision histories.

In contrast to goal and system modeling approaches, implementation proposals don't enforce any specification formalism. Instead, the proposal author specifies the design on a case-by-case basis in the way believed to be most efficient. Meaning arises from connecting the design to the requirements it intends to fulfill.

Practitioners use the same template for sketching a proposal on a whiteboard, presenting slides, and defining the sections of an implementation-proposal document. They start by specifying the basic attributes to be flexible and fast in early negotiation phases, extend the specification when necessary, and comprehensively document the agreement reached.

Handshaking negotiations are effective when the supplier has expertise in the implementation proposal's theme. This expertise is also needed for subsequent project work and can come from the supplier's own employees, consultation with domain experts, and prototyping.

Handshaking Process

Handshaking leads the customer and supplier through requirements communication in three phases: taking a position, negotiating, and confirming agreement.[9] Figure 2 illustrates this process.

Table 1

Implementation proposal template description

Attribute group	Attribute	Description
Basic attributes	Title	The theme addressed by the implementation proposal—a product feature, an important solution part or subsystem, or a development increment.
	Requirements	The requirements addressed by the implementation proposal. What does the proposed design imply for the customer?
	Design	The design proposed to address the requirements. What parts and functionality of the solution must be created or modified? What structure, style, and rules will be followed? What technologies will be used? What are the interfaces?
Negotiation support	Assumptions	Interpretation of the requirements in terms of preconditions that make the design meaningful.
	Impact	Impact of the design for the customer, supplier, and other stakeholders in terms of advantages, limitations, and risks.
	Dismissed alternatives	Alternative designs to address the requirements. How was each design investigated, and why was each alternative dismissed?
	Open issues	Actions required by the customer to progress with the negotiations—for example, confirm the proposal or provide more information.
Planning support	Scope	Parts of the solution that the design affects.
	Necessary activities	Activities needed to realize the design—the inputs for effort estimation and project planning.
	Effort estimate	Estimated effort to realize the design with arguments for why the supplier believes the estimate is correct.

During positioning, both the customer and the supplier share their expectations and intentions in the form of requirements and implementation proposals. The customer and supplier negotiate as many implementation proposals for a given requirements specification as they need to achieve a shared understanding of the requirements. Typically, the product manager specifies requirements before the development team starts creating implementation proposals. However, we also observed development teams that proactively approached product management with implementation proposals. In either case, the information exchange lets the two parties recognize where they understand each other and where conflicts exist.

During negotiation, both parties seek to resolve conflicts. The customer corrects unacceptable proposals by adjusting misinterpreted and unfeasible requirements and by adding missing requirements. The supplier proposes alternative solutions to correct wrong assumptions, unacceptable impacts, and excessive costs. All the negotiations we observed ended with an agreement. Conceivably, however, a decision not to collaborate with each other could result.

To confirm agreement, the two parties update the requirements and implementation proposals according to their agreement. Senior management should review the agreed implementation proposals to verify the successful conclusion of handshaking and to launch the ensuing collaboration phases.

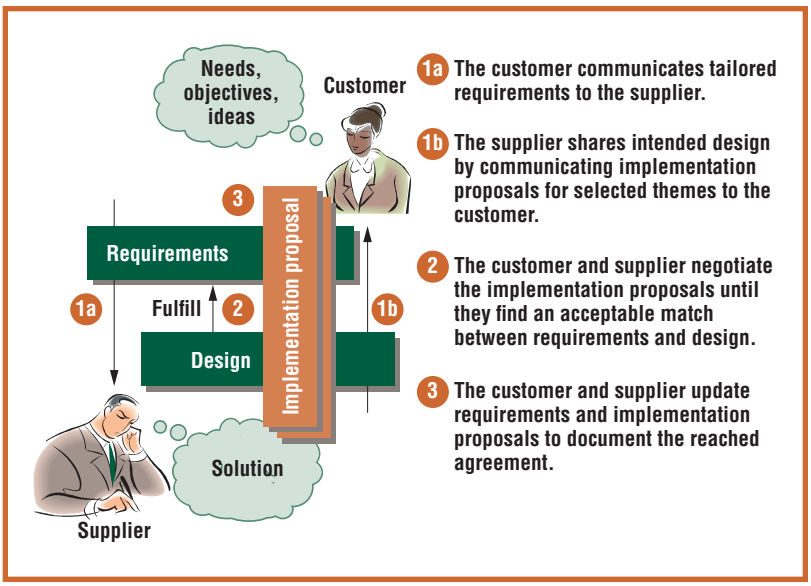

Figure 2. Handshaking process. An implementation proposal describes how a solution design fulfills requirements for a given theme. The more of the known requirements that go through the handshaking process, the more likely the customer will accept the solution.

Practitioners launch handshaking to elaborate and agree on requirements after the customer has defined a development project's vision. Concluded handshaking results support ensuing scope negotiations and project planning, where

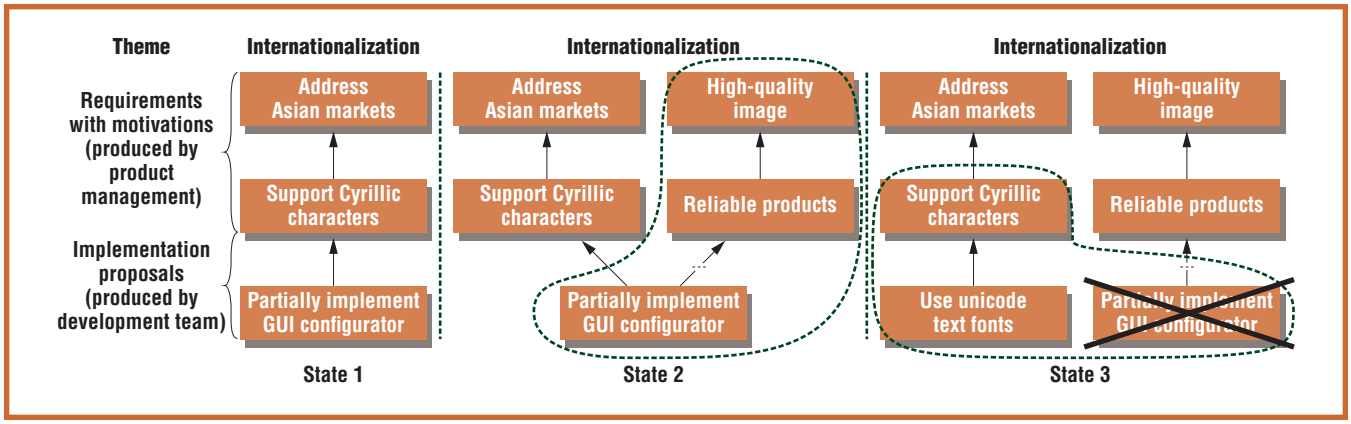

Figure 3. Implementation proposal evolution. Requirements and design changes resemble a goal-contribution tree that evolves as a result of handshaking negotiations. The dotted lines highlight the changes between two states.

the supplier combines the implementation proposals into a specification that defines the overall development effort's scope.

Implementation proposals help reach an agreed understanding of requirements but don't necessarily reduce feasibility and completeness risks, which are better addressed by techniques such as prototyping and system analysis.

Surprises during the execution of a development project might necessitate renegotiating implementation proposals, which is a normal part of the project's change-management process.

Handshaking Negotiations: Industry Example

A collaborating customer and supplier share the common goal of developing a high-quality product. However, they differ in their specific product interests, responsibilities, and competencies. This raises the risk of fundamentally misunderstanding each other.[10]

We observed many such misunderstandings between product managers and development teams. In one typical case, a development team received a requirement to support Cyrillic characters in a project. The product manager introduced the requirement in response to the company's interest in the Asian market.

The product manager and development team discussed the intended design and its purpose to help bridge the mental worlds between them. The team expressed implementation proposals in an "if-then" language pattern: "If we partially implement the planned GUI configurator (proposed design), we can support the Cyrillic characters (requirement)." In Figure 3, State 1 illustrates the team's proposal.

The development team gave their proposal to the product manager. He felt that the proposed design had a negative effect on a so-far-unstated requirement—specifically, that "reliable products" had to support the company's "high-quality" image (Figure 3, State 2). This comment encouraged the team to defer the GUI configurator implementation and to propose the alternative of using unicode text fonts. The product manager accepted this proposal (Figure 3, State 3).

Development teams expressed implementation proposals without sticking to a specific graphical formalism. Instead, they followed the proposal structure we've outlined by employing text and ad hoc drawings. Product managers added, removed, and modified requirements while reacting to the implementation proposals. In one case, a product manager adjusted two thirds of the initial requirements during a first handshaking negotiation round.

We analyzed the evolving negotiations for coaching purposes with goal trees, as shown in Figure 3. We borrowed several argumentation patterns from the goal-oriented requirements-engineering field,[11] which enhanced the handshaking negotiations. In addition to using the basic goal-contribution pattern (Figure 3, State 1), the practitioners combined design decisions, elaborated side effects of a design (Figure 3, State 2), and considered alternative designs (Figure 3, State 3).

Lessons Learned

Using handshaking with implementation proposals to replace requirements handoffs led to recognized improvements at ABB and DHR. The lessons learned revealed the strengths and limits of handshaking that companies should consider when adopting and further developing the practice.

The first and second authors introduced handshaking to the organizations. They collected the lessons learned from 10 development projects by inter-

viewing product managers and architects after they had applied the handshaking process. The third and fourth authors worked independently as leading architects in three of these projects.

All the development projects targeted new features for existing software products or product lines of software-intensive systems. The projects lasted from three months to two years with staffs ranging from four to more than 50 engineers. Seven projects were collocated, and three were distributed over three development sites in Europe and Asia.

The projects identified from two to 22 themes (average 10, median 9) for implementation proposals to address. The practitioners used their expertise in the product domain, rather than formal criteria, to define which themes and requirements merited the negotiation process.

An implementation proposal addressed up to 51 requirements (average 8, median 4). One proposal addressed no requirements but instead justified the proposed design with assumptions about stakeholder needs. Full implementation proposal specifications ranged from seven to 28 pages long (average 16, median 15).

Handshaking for one implementation proposal required efforts ranging from three to 50 person-days and lasting from one week to five calendar months. An important effort driver was the theme size in terms of affected system components.

Strengths

Handshaking improved requirements. Development teams identified missing requirements, hidden expectations, and tacit domain knowledge by proposing their development intentions. The proposed design, explicit assumptions, expected design impact, and suggested alternatives facilitated rich discussions for reaching good agreements. Practitioners judged that the improved requirements decreased defect-related costs by about 40 percent.

Handshaking allowed detailed requirements engineering to be delegated to the development teams. The teams elicited requirements from product users and domain experts. The teams needed this knowledge to create implementation proposals, and it helped them increase product usability and identify new features for future product improvements. Product managers could reduce their workload and focus on steering development projects, while giving the development teams more freedom.

Handshaking improved identification, analysis, and selection of variants. Implementation proposals supported grouping requirements into development themes of adequate granularity for identifying implementation alternatives. Development teams determined the product functionality needed to support the requirements, the changes required in the current product, and the effort needed for realizing the proposals. When the development teams submitted completed implementation proposals to their product managers, they could state clearly what they intended to do and ask, "Is it what you want? This is what we intend to do." This clarity created pressure to properly evaluate and select implementation alternatives.

Handshaking promoted win-win negotiations. The teams usually had strong opinions about the best product functionality to choose. Still, they expected product management to critically review their proposals. The understanding of possibilities and limitations enabled product managers to identify issues that stakeholders needed to agree on. It also let them correct development decisions that were inconsistent with market and stakeholder needs. This critical examination of implementation proposals, rather than simply proceeding with implementation, helped enhance the realized software's value and increase the stakeholders' satisfaction.

Handshaking helped development organizations achieve deep requirements understanding and prepare for projects. The more the implementation proposals were elaborated, the more they became a part of the organization's vocabulary and a focal point for coordinating planned work results. The implementation proposals documented how key parts of the solution had to be implemented, integrated, and verified, hence reducing the remaining design effort and risks in subsequent project phases.

Handshaking increased the amount and quality of decision-making information. The product manager received more and better information that helped steer the development projects and express requirements more concisely. Improved information also helped the teams increase planning precision and, hence, better adhere to promises. Over time, this reduced the need for change management during implementation. It also increased trust and accelerated requirements communication.

Finally, handshaking encouraged development organizations to reduce their projects' duration. The longer a development project lasted, the more difficult and risky it was to agree and commit to a proposed implementation. With shorter development cycles, the practitioners could postpone issues to the moment when insights had been gained from development and from product use.

Practitioners judged that the improved requirements decreased defect-related costs by about 40 percent.

Development teams repeatedly accepted badly specified requirements for which they successfully proposed accepted solutions.

Limitations

Product managers didn't always receive what they wanted. Development teams refused, in some cases, to create implementation proposals, arguing that they didn't believe the requirements were valid. Hence, a product manager who uses the handshaking process must depend on the development team's good will. We consider this limitation a kind of quality control. Product managers should be able to justify their requirements—for example, by demonstrating their importance for achieving company objectives or satisfying customer needs.

In another case, a development team was unable to create an acceptable implementation proposal. The proposed solution, which the team had already partially implemented, contained circular dependencies and exhibited performance and scalability problems. The company reacted by supporting the team with an experienced software architect. A development team must have experience in the kind of problems and solutions it's proposing.

Requirements that were good enough for the supplier weren't necessarily understandable to practitioners who didn't participate in the handshaking negotiations. Development teams repeatedly accepted badly specified requirements for which they successfully proposed acceptable solutions. Outsiders, even domain experts, had difficulty grasping the meaning of these requirements and reusing them in other projects. Hence, handshaking leads to good agreements and requirements understanding but can't be used to improve specification quality and to achieve general-purpose requirements understandability.

One development team was confronted with requirements from two product managers, rather than just one. The team had difficulty receiving final acceptance of their implementation proposals because the product managers belonged to different organizations and had conflicting views and no overall authority. In such a constellation, an additional mechanism is needed to resolve customer conflicts.

The teams used lists of themes and requirements, and not implementation proposals, to document the scope of their development projects. The one-theme focus of implementation proposals didn't directly support scope negotiations. However, the requirements understanding achieved with handshaking made it easier to commit to a scope because product management and the development organization had studied and agreed on development strategy and effort.

Impact on Distributed Development

The three distributed development projects required more elaborate requirements communication than smaller-scale collocated projects. The architect extended the basic handshaking process in steps that integrated subteams into the negotiations:

1. The architect identified and selected key alternatives. The architect's holistic understanding of available assets and expertise increased the development organization's efficiency in creating a meaningful proposal.
2. The architect negotiated with selected subteam leaders. These engineers proposed the necessary changes in the subsystems for which they were responsible and estimated the effort for realizing these changes. The architect facilitated decision making and ensured the design's consistency across subteams.
3. The architect handshaked with product management, which reviewed the implementation proposal for its impact. Issues that affected stakeholders negatively were returned to the architect, who elaborated further alternatives.
4. Senior management authorized agreements, checking that the right people were involved in the negotiations and that the agreement suited the organization. Their approval empowered the development organization to start using the agreed implementation proposal.

The architect performed steps 1 to 3 iteratively. Changes on the development side affected product management and vice versa. He executed the first iteration informally to see how the involved parties reacted on the first proposal. The second iteration required sufficient effort in writing and reviewing the implementation proposal to reduce correctness, feasibility, and consistency risks to acceptable levels. Step 4 preceded project planning.

The adjusted handshaking process enabled the organization to coordinate development across sites. The implementation proposals provided holistic views of given design topics for assigning responsibilities and defining interfaces. Engineers expressed their ideas and potential design effects for other subteams and for stakeholders. This helped reveal design errors that would otherwise have shown up only during integration or verification.

Impact on Project Planning

Handshaking with implementation proposals im-

proved project planning accuracy. Five projects with comparable staffing collected standardized measurements in a history database. This let us correlate handshaking practices with adherence to delivery dates.

Four projects covered 67 percent or more of their scope with implementation proposals and missed their deadlines by no more than 12 percent. One project covered only 13 percent of its scope and missed its deadline by 57 percent. The analysis of variance (ANOVA) indicated a correlation coefficient of 0.95 between handshaking coverage and deadline adherence. The linear regression's standard error was 0.083. These figures suggest that development project predictability correlates with the percentage of known requirements covered by accepted implementation proposals.

Architects confirmed this correlation. They estimated that handshaking decreased planning errors by 50 to 70 percent compared with post-planning reviews of design specifications. Handshaking helped stabilize release projects. Development teams stopped a never-ending stream of change requests from project management and other stakeholders by proactively formulating their intentions in an implementation proposal and sharing it with stakeholders. This explicit position became an effective tool for initiating discussions, forcing decisions about project direction, and aligning the team's work with other teams' development.

Handshaking also improved estimation accuracy: The developers that wrote an implementation proposal estimated its realization effort. The development team's first-hand knowledge of necessary activities, local engineering practices, and individual development performance improved project planning and effort estimation.

Finally, handshaking enhanced negotiations of project scope. The development teams used implementation proposals as building blocks for planning development projects. The analysis behind the proposals gave credibility to a team's argument for what they could deliver within a given time. Hence, the teams could strengthen their position in the negotiations and set realistic expectations. This helped balance project scope with staffing and deadlines and thus avoid project delays and disappointments.

Evolution of Handshaking Practice

As with most new technologies, the handshaking process faced an initiation and learning threshold. Initiation involved training and defining tools such as implementation-proposal templates. Learning meant that at first development teams created more implementation proposals than were actually necessary and in too much detail for handshaking.

Handshaking in the second and third consecutive projects was more like calibration. The teams found appropriate detail levels for the implementation proposals, streamlined the overall process, and created a common vocabulary for the practices and tools. Many benefits reported in this article were already apparent in the first project, but they became obvious and confirmed in iterations two and three.

The coverage and quality of implementation proposals decreased in the fourth iteration. This decrease became obvious in iteration five. The teams created too few of the necessary and planned proposals and specified them only fragmentarily. Product management reviewed the proposals less diligently. As a result, many problems returned that handshaking had alleviated—for example, missed deadlines and increased defects and misunderstandings. At iteration six, the development organization launched measures to correct this negative trend.

The project teams attributed the devolvement to an increase in confidence. Over time, engineers and managers had become habituated to accurate estimates and to few misunderstandings and defects. They began cramming more features into a project. The resulting increases in resource demands left insufficient time spent on handshaking. Requirements communication began reverting to the same form as before: one-way delivery.

We encourage practitioners to test handshaking with implementation proposals for requirements communication.[12] Start writing and negotiating implementation proposals for customer requirements that are vague or volatile or that are critical for mutual understanding. Implementation proposals can be quick and dirty initially and refined after reaching a first agreement. They helped ABB and DHR use domain knowledge and experience to agree on requirements understanding efficiently and became an important input for robust project planning.

To seize the full benefits and ensure consistent use of handshaking in your organization, the technique must be properly institutionalized and managed. This requires initial effort, but the returns become rapidly evident.

About the Authors

Samuel Fricker is a postdoctoral researcher at the University of Zurich and a senior consultant at Fuchs-Informatik AG, a Switzerland-based leader in requirements engineering. His work includes standardization activities in software product management and research on stakeholder collaboration. Fricker has a doctorate in informatics from the University of Zurich. He's a member of the IEEE and a supporting member of the International Requirements Engineering Board. Contact him at fricker@ifi.uzh.ch.

Tony Gorschek is an associate professor of software engineering at Blekinge Institute of Technology (BTH). He also manages his own industry consultancy company. His research interests include requirements engineering, technology and product management, process assessment and improvement, quality assurance, and innovation. Gorschek has a PhD in software engineering from BTH. He's a member of the IEEE and the ACM. Contact him at tony.gorschek@bth.se or visit www.gorschek.com.

Carl Byman is a system architect at ABB. His interests include product development processes, process improvement, and technology in different areas. Byman received an MSc in engineering physics from Uppsala Tekniska Högskola. Contact him at carl.byman@se.abb.com.

Armin Schmidle is a senior project manager in substation automation system engineering tools at ABB Switzerland. His interests include project management, software engineering, and requirements management. Schmidle received an electrical engineering degree from the University of Applied Sciences, Furtwangen, Germany. Contact him at armin.schmidle@ch.abb.com.

References

1. S. Fricker, T. Gorschek, and P. Myllyperkiö, "Handshaking between Software Projects and Stakeholders Using Implementation Proposals," *Proc. 13th Int'l Working Conf. Requirements Eng.: Foundation for Software Quality (REFSQ 07)*, Springer, 2007, pp 144–159.
2. R. Kazman et al., "The Architecture Trade-off Analysis Method," *Proc. 4th IEEE Int'l Conf. Eng. of Complex Computer Systems* (CCC 98), IEEE CS Press, 1998, pp. 68–78.
3. A. Griffin and J. Hauser, "Integrating R&D and Marketing: A Review and Analysis of the Literature," *J. Product Innovation Management*, vol. 13, no. 3, 1996, pp. 191–215.
4. S. Fricker, T. Gorschek, and M. Glinz, "Goal-Oriented Requirements Communication in New Product Development," *Proc. Int'l Workshop on Software Product Management*, IEEE CS Press, 2008, pp. 27–34.
5. S. Gordon and J. Bieman, "Rapid Prototyping: Lessons Learned," *IEEE Software*, vol. 12, no. 1, 1995, pp. 85–95.
6. C. Larman and V. Basili, "Iterative and Incremental Development: A Brief History," *Computer*, vol. 36, no. 6, 2003, pp. 47–56.
7. J. Tyree and A. Akerman, "Architecture Decisions: Demystifying Architecture," *IEEE Software*, vol. 22, no. 2, 2005, pp. 19–26.
8. A. Dutoit et al., *Rationale Management in Software Engineering*, Springer, 2006.
9. S. Fricker and P. Grünbacher, "Negotiation Constellations—Method Selection Framework for Requirements Negotiation," *Proc. 13th Int'l Working Conf. Requirements Engineering: Foundation for Software Quality*, Springer, 2008, pp. 37–51.
10. L. Bucciarelli, "Between Thought and Object in Engineering Design," *Design Studies*, vol. 23, no. 3, 2002, pp. 219–231.
11. A. van Lamsweerde, "Goal-Oriented Requirements Engineering: A Guided Tour," *Proc. 5th IEEE Int'l Symp. Requirements Eng.* (RE 01), IEEE CS Press, 2001, pp. 249–261.
12. S. Fricker, *Pragmatic Requirements Communication: The Handshaking Approach*, Shaker Verlag, 2009; www.ifi.uzh.ch/rerg/research/handshaking.