

Handshaking: Negotiate to Provoke the Right Understanding of Requirements

Version: 1.2

Samuel Fricker

fricker@ifi.unizh.ch

University of Zurich and Fuchs-Informatik AG

CH-8050 Zurich, Switzerland

Phone: +41 44 635 7128

Fax: +41 44 635 6809

Tony Gorschek

tony.gorschek@bth.se

Blekinge Institute of Technology

PO Box 520

SE-372 25 Ronneby, Sweden

Phone: +46 457 385817

Fax: +46 457 27125

Carl Byman

carl.byman@se.abb.com

ABB AB

SE-721 83 Västerås, Sweden

Phone: +46 21 325 000

Fax: +46 21 186 521

Armin Schmidle

armin.schmidle@ch.abb.com

ABB Switzerland Ltd.

CH-5500Baden, Switzerland

Phone: +41 58 585 8119

Fax: +41 58 585 3582

Submission date: August 25, 2008. Revision 1.1: April 24, 2009. Revision 1.2: July 2, 2009.

Keywords:

D.2.1.e Methodologies < D.2.1 Requirements/Specifications < D.2 Software Engineering < D Software/Software Engineering

D.2.10.f Methodologies < D.2.10 Design < D.2 Software Engineering < D Software/Software Engineering

Word count (excluding authors and tables): 4600 + 3 pictures (600) + 1 table (200)

Development teams that accept fragmentary and vague customer requirements have to cope with uncertainty and disorientation. Inadequate communication and submission to a dominating customer makes it hard to know if requirements are understood or not. Requirements engineering, with its focus on good specification practices, had yet to find working solutions for effective requirements communication.

We are using a negotiation process called Handshaking with Implementation Proposals¹ to make such requirements communication work – even in situations where written requirements are almost nonexistent and distance separates customers from developers. Handshaking is an efficient and effective technique that uses architectural options² to achieve requirements understanding, to create value with implementation decisions, and to establish the foundation for a stable project.

This article focuses on the interface between product management and the development organization in companies, where product management acts as a customer towards development³. We describe the requirements communication challenges we encountered, explain how we addressed these challenges with the Handshaking process, and share the lessons we learned at European and Asian development sites from ABB and Danaher Motion. The side bar *Industry Partners* characterizes these organizations.

(side bar)

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 environmental impact. The ABB group of companies operates in around 100 countries and employs more than 110,000 people.

Danaher Motion Särö AB (DHR) develops software and hardware equipment for navigation, control, fleet management, and service of automated-guided-vehicle systems. More than 50 automated-guided-vehicle 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.

Background

On a number of occasions we found product managers that 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 turned – understandably – to increased control and started to architect the solution themselves in an attempt to assure rapid development of an acceptable product.

This micro management and loss of autonomy prompted fierce reactions from the concerned development teams. Architects stopped accepting requirements from product managers at great personal risk of being laid-off. In their eyes, it was not the job of product management to design the product. Product management's design decisions risked breaking the product or were not realizable within meaningful time.

Handshaking with Implementation Proposals was developed as an answer to this requirements communication problem. To be useful for practical product development, it was important that the requirements communication approach was practical and effective for both small-scale collocated and large-scale distributed development.

Good-Enough Requirements

According to a widely held belief, the success of development projects is highly dependent on the quality of requirements specifications. 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 is not penalizing the engineering processes.

Requirements should be good-enough and specification quality adapted to the situation at hand. Too badly specified requirements lead to ambiguity and misunderstandings that cause large corrective costs down the development road. However, too much detailing and quality improvement retards the delivery of development results, increases requirements specification cost, and unnecessarily constrains the solution space. But what is good-enough? Answers from experienced practitioners allowed us to identify some criteria but at the same time led us to consider requirements quality as a moving target⁴.

Experienced practitioners expected the requirements handed-over to development to be valid and traceable back to real customer needs and company strategy. The requirements had to be consistent and stable for the targeted market release. Only the highest-priority ones were selected for implementation.

Experienced product managers adapted each requirements specification to the intended users of the specification, namely the development team. They elaborated new-to-the-world product features together with the team and stakeholders. They refined the specification of features known to stakeholders but new to the team and provided access to domain experts that supported the team to properly interpret these requirements. However, they assumed requirements considered standard for a given product to be obvious, hence specified them only coarsely if at all. The development team was responsible to refine the resulting incomplete requirements specification.

Handshaking with Implementation Proposals

We addressed the moving target of good-enough requirements by replacing requirements hand-over with a bidirectional communication process. In this process, product management takes the customer role and uses requirements to control the development results. Development takes the supplier role and proposes design and design impact to inform about its development intentions. Requirements are good-enough if the planned solution is acceptable to the customer.

To increase communication efficiency, the customer tailors the requirements specification to the specific supplier by investing most effort for defining novel requirements and by specifying deltas towards already-known requirements. If important knowledge is missing, the customer assists the supplier in finding and building up lacking competence. Such focus on the requirements receiver

allows investing specification effort where it is most needed and mitigates problems of requirements understanding, but requires knowledge of the supplier's expertise.

To assure 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. The customer reviews the proposals to catch wrong assumptions and misunderstandings. The customer then transfers domain knowledge and business rationale to the supplier by selectively adjusting requirements and suggesting justified changes to product design. This rapid feedback builds on lessons from prototyping⁵ and from iterative development⁶ by allowing the supplier to express his intentions as early as possible, with minimal but sufficient formality, and with the customer in mind.

Handshaking assists such requirements communication with a template called Implementation Proposal for documenting the supplier's proposals and with a straight-forward negotiation process for agreeing on these proposals. The negotiations engage the customer and the supplier in a learning process for understanding what constitutes good-enough requirements. Observing how requirements are interpreted and matched with design allows the customer to 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 can act in an increasingly proactive manner.

Implementation Proposals

Implementation Proposals document requirements understanding by recording traceability between the customer's requirements and the supplier's design decisions for a given theme. A theme may be related to a product feature, to an important part or subsystem of the solution, or to a development increment. Many themes are derived from product roadmaps. The breadth of a theme is chosen to allow exploring alternatives and evaluating their impact on requirements and project planning.

A number of attributes extend the basic template to support effective requirements negotiation and project planning. The negotiation attributes document investigated alternative solutions and provide a justified recommendation of the favored alternative. The planning attributes define how the solution will be realized and what the cost of the solution is. The side bar *Implementation Proposal Template* describes the minimal, negotiation, and planning attributes.

(side bar)

Implementation Proposal Template

Implementation Proposals are a form of architecture decisions⁸ and design rationale⁹, which support requirements communication. The following table describes the template used for specifying Implementation Proposals.

Minimal	Title	The theme addressed by the Implementation Proposal, e.g. a development increment or a product feature.
	Requirements	The requirements addressed by the Implementation Proposal. What does the proposed design imply for the customer?
	Design	Description of the design proposed to address above requirements. What parts and functionality of the solution shall be created or modified? What structure, style, and rules shall be followed? What technologies shall be used? What shall be the interfaces?
Negotiation	Assumptions	Interpretation of above requirements in terms of preconditions to 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 above requirements, how each design has been investigated, and why each has been dismissed.
	Open Issues	Actions required by the customer to progress with the negotiations: e.g. confirm proposal or provide more information.
Planning	Scope	Parts of the solution that are affected by the design.
	Necessary Activities	Activities needed to realize the design. These activities are an input for effort estimation and project planning.
	Effort Estimate	Estimate of the effort needed to realize the design with arguments for why the estimate is believed to be correct.

The *minimal attributes* are mandatory for effective Handshaking. They establish traceability between customer expectations and supplier intentions for a given theme. The *negotiation attributes* enrich the argumentation for negotiating the right understanding of requirements. They allow surfacing concerns related to the acceptance of the planned product and capture the investigated options towards a reached agreement. The *planning attributes* support project planning. They document the cost side of the Implementation Proposal. Additional attributes like *identifier*, *executive summary*, *revision history*, etc. that support company-specific processes are added as part of template tailoring.

Figure 1 shows an example of an Implementation Proposal extracted from a seven-page document. The open issues attribute indicates that Handshaking has not been concluded yet. The dark bars indicate anonymization.

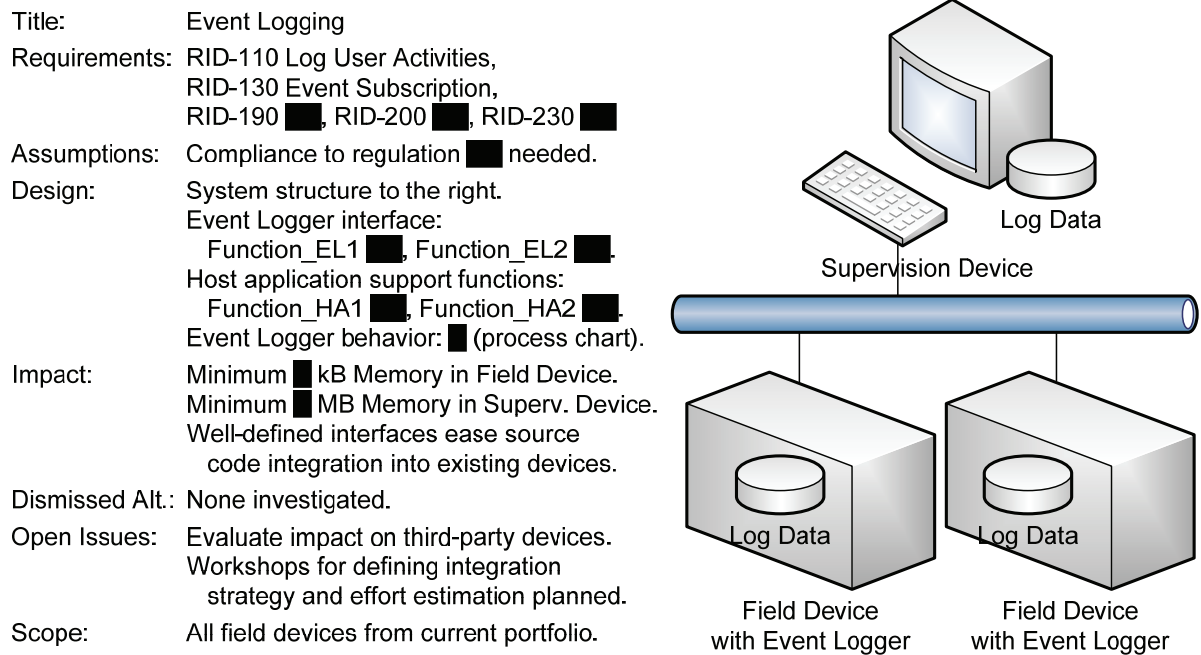


Figure 1: Industry example of an Implementation Proposal.

In contrast to goal and system modeling approaches, Implementation Proposals do not enforce any specification formalism. Instead, their author pragmatically specifies the design in the way believed to be most efficient on a case-by-case basis. Meaning arises from connecting the design to the requirements that are intended to be fulfilled.

Practitioners use the same template for sketching Implementation Proposals at the whiteboard, for defining the bullets on presentation slides, and for defining the section headings of Implementation Proposal documents. They start specifying the minimal attributes to be flexible and fast in early negotiation phases, extend the specification when necessary, and are comprehensive when an agreement is documented.

Handshaking assumes that the supplier has expertise in the theme of the Implementation Proposal. Building on experienced employees, consulting domain experts, and prototyping help acquiring such knowledge. This knowledge supports not only the requirements negotiations but also subsequent project work.

Handshaking Process

The Handshaking process leads the customer and the supplier through requirements communication by passing the three phases *take a position*, *negotiate*, and *confirm agreement*¹⁰. Figure 2 illustrates.

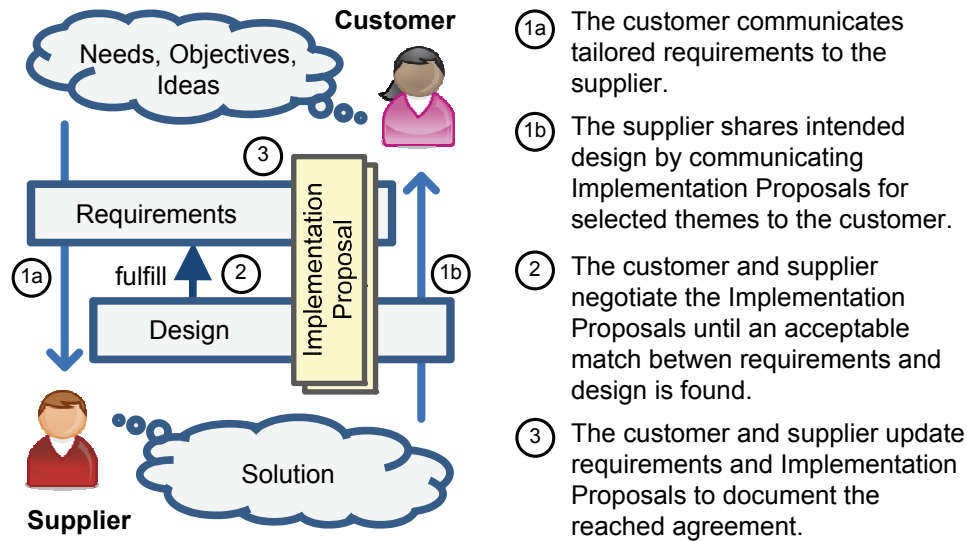


Figure 2: Handshaking process. An Implementation Proposal describes how solution design fulfills requirements for a given theme. The more requirements are handshaked the more likely the solution is acceptable to the customer.

During positioning, both the customer and the supplier share their expectations and intentions in the form of requirements and Implementation Proposals. Typically, the product manager specifies requirements before the development team starts creating Implementation Proposals. However, we also experienced development teams that proactively approached the product management with Implementation Proposals. This information exchange allows the two parties to recognize where they understand each other and where conflicts exist.

During negotiation, both parties seek to resolve conflicts. The customer corrects unacceptable proposal by adjusting misinterpreted and unfeasible requirements and by complementing missing requirements. The supplier proposes alternative solutions to correct wrong assumptions, unacceptable impact, and excessive cost. All negotiations we observed ended with an agreement. Conceivable, however, may also be a decision not to collaborate with each other. The side bar *Handshaking Negotiations – An Industry Example* shows an example of a negotiation.

To confirm agreement, the two parties update the requirements and the Implementation Proposals according to the reached agreement. Senior management should review the Implementation Proposals to verify the successful conclusion of Handshaking and to launch the ensuing phases of the collaboration.

Handshaking Negotiations – An Industry Example

A customer and a supplier that collaborate share a common goal: to develop a product of quality. However, their specific interests, responsibilities, and competencies related to the product are different. This leaves the two parties with the risk of fundamentally misunderstanding each other⁷.

We observed many product managers and development teams with such misunderstandings. In one typical case, a team was confronted with internationalization when it received the requirement “support Cyrillic characters.” The product manager motivated this requirement by the interest of the company to “address Asian markets”.

Discussions of the intended design and the purpose of that design helped to bridge the mental worlds between the team and its product manager. The team expressed Implementation Proposals with the language pattern “if ..., then” “If we partially implement the planned GUI configurator (proposed design), we can support Cyrillic characters (requirement).” Figure 3 State 1 illustrates the team’s proposal.

The development team approached the product manager with the Implementation Proposal. He felt that the proposed design had a negative effect on a so far unstated requirement “reliable products” needed to support the “high-quality image” of the company (Figure 3 State 2). This comment encouraged the team to defer the implementation of the GUI configurator and to propose the alternative “use Unicode text fonts”. This was accepted by the product manager (Figure 3 State 3).

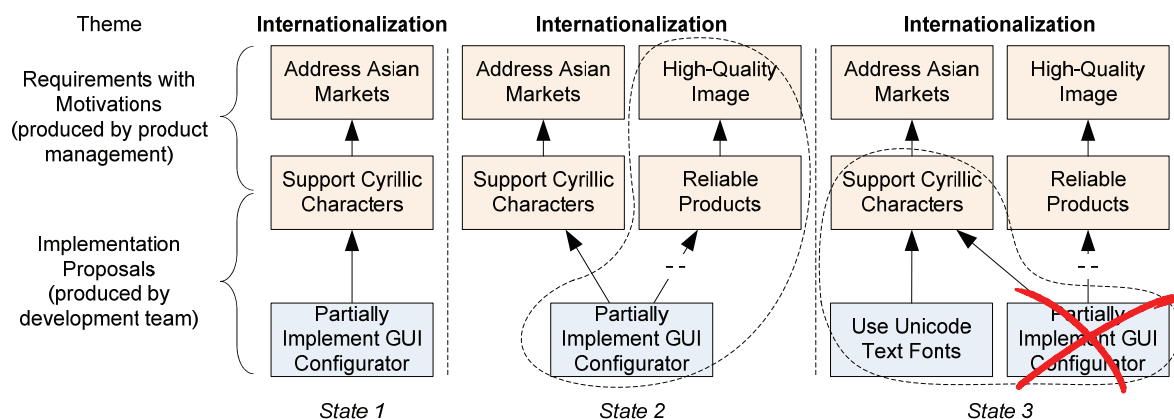


Figure 3: Goal-contribution tree that evolves as a result from Handshaking negotiations. The dotted lines highlight the change between two states.

Development teams expressed Implementation Proposals without sticking to a specific graphical formalism. Instead they followed the structure described in the side bar *Implementation Proposal Template* by employing text and ad-hoc drawings on different media. Product managers added, removed, and modified requirements while reacting to the Implementation Proposals. In one

observed case a product manager adjusted two thirds of the initial requirements during a first Handshaking negotiation round.

We analyzed the evolving negotiations using goal trees as suggested by Figure 3 when called for coaching purposes. We borrowed a number of argumentation patterns from the field of goal-oriented requirements engineering¹¹ that usefully enhanced the Handshaking negotiations: in addition to the basic goal-contribution pattern (Figure 3 State 1), design decisions were combined, alternative designs considered (Figure 3 State 3), and side-effects of a design elaborated (Figure 3 State 2).

Practitioners launch Handshaking to elaborate and agree on requirements after the vision for a development project has been defined, e.g. through a high-level release plan. Concluded Handshaking results support ensuing scope negotiations and project planning, where the supplier combines the Implementation Proposals into a specification that defines the overall scope of the development effort.

Implementation Proposals help reaching an agreed understanding of requirements, but do not necessarily reduce feasibility and completeness risks. These are better addressed by techniques like prototyping and system analysis. Surprises during the execution of a development project may necessitate re-negotiating Implementation Proposals, which is a normal part of the project's change management process.

Lessons-Learned

Replacing requirements hand-over by Handshaking with Implementation Proposals led to recognized improvements at ABB and DHR. We also appreciated the limits of Handshaking and used the lessons-learned for decision-making regarding use, tailoring, and further development of the practice.

The first and second authors introduced the Handshaking to the organizations. They collected the lessons-learned from ten development projects by interviewing 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 development projects targeted at developing new features for existing software products or product lines of software-intensive systems. The projects lasted between three months and two years and were staffed with four to more than 50 engineers. Seven projects were collocated and three distributed over three development sites in Europe and Asia.

The projects identified between two and 22 themes (average 10, median 9) to be addressed by Implementation Proposals. The practitioners used their expertise in the product domain, rather than formal criteria, to define which themes and requirements were worth the negotiation process.

An Implementation Proposal addressed between zero and 51 requirements (average 8, median 4). The one addressing zero requirements justified the proposed design with assumptions about stakeholder needs. Full Implementation Proposal specifications had between 7 and 28 pages (average 16, median 15).

Handshaking of one Implementation Proposal required between three and 50 person-days effort and lasted between one week and five calendar-months. An important effort driver was the size of the theme 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, the explicit assumptions, the expected impact of the design, and the suggested alternatives facilitated rich discussions for finding good agreements. Practitioners judged that the resulting requirement improvements decreased defect-related cost by about 40%.

Handshaking allowed delegating detailed requirements engineering to the development teams. The teams elicited requirements from product users and domains experts. This knowledge was necessary for creating Implementation Proposals. It also allowed the teams to increase product usability and to identify new product features for future product improvements. Product managers could reduce their workload and focus on steering development projects, while giving more freedom to the development teams.

Handshaking improved identification, analysis, and selection of variants. Implementation Proposals allowed grouping requirements into development themes of adequate granularity for identifying implementation alternatives and asking “what product functionality is needed to support the requirements?”, “how should the current product be changed?”, and “how much effort is required?” The development teams submitted completed Implementation Proposals to their product managers by stating: “Is this what you want? This is what we intend to do.” This statement created pressure for proper evaluation and selection of implementation alternatives.

Handshaking promoted Win-Win negotiations. The teams usually had a strong opinion of what the best choice of product functionality would be. Still, they expected product management to critically review their proposals. The understanding of possibilities and limitations enabled product managers to identify issues that needed to be agreed with stakeholders and to correct development decisions inconsistent with market and stakeholder needs. This critical examination of Implementation Proposals, rather than simply proceeding with implementation, helped to enhance the value of the realized software and to increase the satisfaction of stakeholders.

Handshaking allowed development organizations to achieve deep understanding of requirements and to prepare for development projects. The further the elaboration of Implementation Proposals progressed, the more they became a part of the organization’s vocabulary and the 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 reduced remaining design effort and risk of the subsequent project phases.

Handshaking increased the amount and quality of decision-making information. The product manager received more and better information that helped steering the development projects and expressing requirements more concisely. Improved information also helped the teams to increase planning precision, hence to better adhere to promises. Over time, this led to reduced need for

change management during implementation, increased trust, and accelerated requirements communication.

Handshaking, finally, encouraged development organizations to reduce the duration of development projects. 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, issues could be postponed to the moment where insights had been gained from development and from product use.

Limitations

Product managers did not always receive what they wanted. Development teams refused in some cases to create Implementation Proposals arguing they did not believe that the requirements were valid. Hence, a product manager that uses the Handshaking process is dependent on the goodwill of the development team, which we consider a kind of quality control. He should be able to justify his requirements, for example by demonstrating their importance for achieving company objectives or for satisfying customer needs.

In another case a development team was unable to create an acceptable Implementation Proposal. The proposed solution, which the team already had partially implemented, contained circular dependencies and exhibited problems related to performance and scalability. The company reacted by supporting the team with an experienced software architect. A development team needs to be experienced in the kind of problems and solutions it is proposing an implementation for.

Requirements good-enough for a specific receiver were not necessarily understandable for others. Development teams repeatedly accepted badly specified requirements for which they successfully proposed accepted Implementation Proposals. Outsiders, even domain experts, had difficulties in grasping the meaning of these requirements and in reusing them for other projects. Hence, additional improvement of requirements quality is needed when the supplier is not known in advance, such as in tendering.

One development team was confronted with requirements from two, rather than from one product manager. The team had difficulties in receiving final acceptance of their Implementation Proposals because the product managers, which belonged to different organizations, had conflicting views and no overall authority. In such a constellation, an additional mechanism is needed to resolve the conflicts between the customers.

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 did not directly support scope negotiations. The requirements understanding achieved with Handshaking, however, made it easier to commit to a scope because development strategy and effort had been studied and agreed.

Impact on Distributed Development

The three distributed development projects required more elaborate requirements communication than smaller-scale collocated development. The architect extended the basic Handshaking process

by integrating sub-teams into the negotiations. This assured that the sub-teams understood relevant requirements and allowed coordinating product design among these teams:

1. The architect identified and selected key alternatives. His holistic understanding of available assets and expertise increased the efficiency of creating a proposal that was meaningful for the development organization.
2. The architect negotiated with selected sub-team leaders. These engineers proposed necessary changes in the solution's subsystems they were responsible for and estimated effort for realizing these changes. The architect facilitated decision making and assured that the design was consistent across sub-teams.
3. The architect handshaked with product management. Product management reviewed the Implementation Proposal by reflecting on the impact of the proposed solution. Issues that affected stakeholders negatively were returned to the architect who elaborated further alternatives.
4. Senior management authorized agreements. Senior management checked that the right people were involved in the negotiations and that the agreement suited the organization. Their approval empowered lower-level management to start using the Implementation Proposal.

The architect performed steps 1-3 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. With the second iteration, sufficient effort was invested in writing and reviewing the Implementation Proposal to reduce correctness, feasibility, and consistency risks to acceptable levels. Step 4 preceded planning of the development project.

The adjusted Handshaking process enabled the organization to coordinate development across sites. The Implementation Proposals provided a holistic view of a given design topic that allowed assigning responsibilities and defining interfaces. Engineers expressed their ideas and potential effects of their design for other sub-teams and for stakeholders. This helped surfacing design errors that otherwise would only have been identified during integration or verification.

Impact on Project Planning

Handshaking with Implementation Proposals improved accuracy of project planning. Five projects with comparable staffing collected standardized measurements in a history database. This allowed investigating the correlation of Handshaking practices with adherence to delivery dates.

Four projects covered 67% or more of their scope with Implementation Proposals and missed their deadlines by no more than 12%. One project covered only 13% of its scope and missed its deadline by 57%. The ANOVA analysis indicated a correlation coefficient of 0.95 between Handshaking coverage and deadline adherence. The standard error of the linear regression was 0.083. These figures suggest that predictability of development projects correlate with the amount of created Implementation Proposals.

Architects confirmed this correlation. They estimated that Handshaking decreased planning errors by 50-70% compared with post-planning reviews of design specifications. Handshaking helped

stabilizing release projects, improved estimation accuracy, and enhanced negotiations of project scope.

Stabilized development projects: product management and other stakeholders confronted the teams with a never-ending stream of changing requirements. Implementation Proposals allowed project teams to proactively formulate their intentions for a development project and to share these intentions with stakeholders. This explicit position became an effective tool for initiating discussions, for forcing decisions about project direction, and for aligning the team's work with the development of other teams.

Improved estimation accuracy: Implementation Proposals authors estimated effort. Their first-hand knowledge of necessary development activities, local engineering practices, and individual development performance improved project planning and effort estimation.

Enhanced negotiations of project scope: the development teams used Implementation Proposals as building blocks for planning development projects. The analysis behind the Implementation Proposals made the argumentation of what can be delivered within given time credible. Hence, the teams could strengthen their position in the negotiations and set realistic expectations. This allowed balancing project scope with staffing and deadlines and helped to avoid project delays and disappointments.

Evolution of Handshaking Practice

As with most new technologies, the adoption of Handshaking with Implementation Proposals was confronted with an initiation and learning threshold. Initiation involved training and defining tools such as templates. Learning meant that the development teams initially created more Implementation Proposals than actually necessary and that these Implementation Proposals were too detailed or contained items not necessary for Handshaking.

Handshaking in the second and third consecutive project can be characterized as calibration. The teams found appropriate levels of detail for the Implementation Proposals, streamlined the overall Handshaking process, and created a common vocabulary for the practices and tools. Many of the benefits reported in this article were obtained already in the first project, but were obvious and confirmed in iterations two and three.

Implementation Proposal coverage and quality decreased in the fourth iteration. This decrease became obvious in iteration five. The teams created too few of the necessary and planned Implementation Proposals and specified them only fragmentarily. Product management reviewed Implementation Proposals less diligently. The implication was a return to many of the problems that were alleviated with Handshaking: missed deadlines and increases in defects and misunderstandings. At iteration six, the problem was recognized and measures launched to compensate this negative trend.

The project teams attributed the cause of this devolvement to an increase in confidence. Over time, engineers and managers had become habituated to accurate estimates and to the only small number of misunderstandings and defects. Consequently, they attempted to increase the amount of features

crammed into a project. The resulting increases in resource demands led to insufficient time spent for Handshaking. Requirements communication was beginning to take the same form as before: one-way deliveries of requirements.

(Conclusion)

We would like to encourage practitioners to test Handshaking with Implementation Proposals for requirements communication¹¹. A good place to start writing and negotiating Implementation Proposals is where customer requirements are most vague or volatile and where agreement on requirements understanding is critical. Implementation Proposals can be quick and dirty initially and subsequently refined when a first agreement has been reached. Implementation Proposals permitted ABB and DHR to efficiently utilizing experience and domain knowledge, and have turned into an important input for robust project planning.

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

Citations

1. S. Fricker, T. Gorschek, and P. Myllyperkiö, "Handshaking between Software Projects and Stakeholders Using Implementation Proposals," International Working Conference on Requirements Engineering, Trondheim, Norway, 2007.
2. R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson, J. Carriere, "The Architecture Tradeoff Analysis Method," Fourth IEEE International Conference on Engineering of Complex Computer Systems, Monterey, CA, USA, 1998.
3. A. Griffin, J. Hauser, "Integrating R&D and Marketing: A Review and Analysis of the Literature," Journal of Product Innovation Management 13 (1996), 191-215.
4. S. Fricker, T. Gorschek, and M. Glinz, "Goal-Oriented Requirements Communication in New Product Development," International Workshop on Software Product Management, Barcelona, Spain, 2008.
5. S. Gordon, and J. Bieman, "Rapid Prototyping: Lessons Learned," IEEE Software 12 (1995), pp.85-95.
6. C. Larman, and V. Basili, "Iterative and Incremental Development: A Brief History," Computer 36 (2003), pp.47-56.
7. L. Bucciarelli, "Between Thought and Object in Engineering Design," Design Studies 23 (2002), pp.219-231.
8. J. Tyree, A. Akerman, "Architecture Decisions: Demystifying Architecture," IEEE Software 22 (2005), 19-26.

9. A. Dutoit, R. McCall, I. Mistrik, B. Paech, Rationale Management in Software Engineering, Springer-Verlag Berlin Heidelberg, 2006.
10. S. Fricker, P. Grünbacher, „Negotiation Constellations – Method Selection Framework for Requirements Negotiation,” 13th International Conference on Requirements Engineering: Foundation for Software Quality, Montpellier, France, 2008.
11. A. van Lamsweerde, “Goal-Oriented Requirements Engineering: A Guided Tour,” 5th IEEE International Symposium on Requirements Engineering, Toronto, Canada, 2001.
12. S. Fricker, “Handshaking: Pragmatic Requirements Communication,” July 2009; <http://www.ifi.uzh.ch/req/research/handshaking/>.