Goal-Oriented Requirements Communication in New Product Development

Samuel Fricker
University of Zurich
fricker@ifi.uzh.ch
ABB Switzerland Ltd.
samuel.fricker@ch.abb.com

Tony Gorschek
Blekinge Institute of Technology
tony.gorschek@bth.se

Martin Glinz
University of Zurich
glinz@ifi.uzh.ch

Abstract

Product development organizations often distribute the responsibilities for requirements engineering over several roles. The collaboration of product management, concerned with market needs, and product development, concerned with the technological aspects of a product, is well established. Such shared responsibility provides advantages in the utilization of specific knowledge, skills, and resources. However, the collaboration leads to increased demands on coordination.

Novel concepts and models need to be investigated to support such collaborative requirements engineering. In this paper we focus on requirements communication from product management to a development team by proposing and evaluating the model of goal-oriented requirements communication. The model explains how efficiency and effectiveness of requirements communication can be increased and allows the utilization of established requirements engineering knowledge in a new way to address the task of requirements communication.

1. Introduction

Requirements engineering is a key activity for bringing new product development to success. Product managers seek opportunities for new product features by continuously eliciting needs from markets and company-internal stakeholders. Product managers are responsible for identifying creative ways of addressing these needs by carefully fostering product innovation. In addition, they steer product development towards directions beneficial for the company by prioritizing and selecting requirements and by shaping business objectives [31, 33]. Development teams define and formalize basic product functionality and assure product usability, without which a product would not be accepted on its markets [29].

The collaboration of product management and development is crucial for product success, but requires managing important communication and coordination challenges [19]. The communication of requirements from product management to the development team is of particular interest in this scenario. Such requirements communication occurs in the transition from scope definition for a product release to planning of the development project. The former is the responsibility of product management, and the latter the responsibility of development.

This interface between product management and development is not well understood. Questions concerning the quality of requirements specifications and the activities needed to achieve a shared understanding of requirements are of special interest. Too much detailing and quality improvement to the requirements can make it impossible for a product manager to pursue his key responsibility: taking decisions for steering the product evolution by monitoring markets and stakeholders. However, too badly specified requirements lead to ambiguity and misunderstandings that cause large corrective costs down the development road [9, 11].

The product manager also has a need for controlling the development project. The right product features implemented at the right time are key for product success. With too little control he risks projects lasting too long and producing inacceptable results. With too much control he nurtures an atmosphere of distrust, where personal conflicts can lead to project failure. Distance further complicates the task [6].

Such steering and control has been studied in the area of goal-oriented systems[1] [24]. Conceptualizing the interface between product management and development in terms of a goal-oriented system allows increasing efficiency and effectiveness of requirements communication. With feedforward, requirements can be specified with more impact and less effort. With feedback, the understanding of requirements can be assessed and managed.

This paper describes how requirements communication can be conceptualized as a goal-oriented system.

[1] Goal-oriented systems is a body of knowledge different from goal-oriented requirements engineering [34].

An industrial case shows the kind of thinking that facilitates using goal-oriented systems as a model of requirements communication. Finally, the goal-oriented systems model is used to structure established requirements communication concerns.

The paper is structured as follows. Section 2 describes the problems that motivated the development of the goal-oriented requirements communication model. Section 3 characterizes the principles of this model. Section 4 presents the case study, which describes the context targeted by the model. Section 5 relates the model to existing literature. Section 6 discusses the research results. Section 7 summarizes.

2. Motivation

The work presented in this paper was motivated by challenges identified in industry. It relates to a product development organization with about ten product managers and more than 100 R&D employees developing software-intensive systems. The organization was seeking to improve the requirements communication process between product management and development. The process was expected to minimize requirements specification effort while still assuring that the developed products were deemed acceptable by product management.

These expectations have two major implications on requirements communication. First, the requirements specifications that are handed over to development are not perfect. Some of the requirements remain tacit and are not documented. The requirements that are documented risk not satisfying established standards of good quality requirements specifications [22].

Second, understanding imperfect requirements requires experience in the domain and in the kind of products being developed. The alternative is to spend large resources for eliciting necessary requirements knowledge, or to risk implementing the wrong thing.

These problems have motivated researching requirements communication models, which do not make assumptions of perfect, or close to perfect, requirements specifications. At the same time, the model should be usable in an environment of experienced practitioners. The model should not only explain how the requirements receiver, i.e. the development team, should act, but also how the product manager and the team should collaborate.

3. Goal-Oriented Requirements Communication

Goal-oriented systems [24] are capable of satisfying these concerns. Goal-oriented systems have received considerable attention in systems sciences and are known under a number of names that designate special types of goal-orientation: regulation, control, and others. Figure 1 depicts requirements communication as a goal-oriented system.

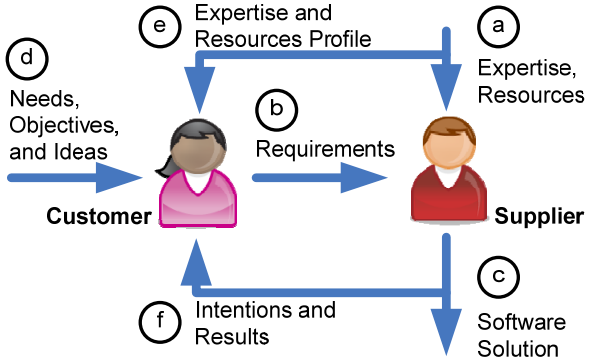

Figure 1. Goal-oriented systems applied to requirements communication. The development team uses (a) available expertise and resources and (b) requirements to realize (c) the software solution. The product manager uses (d) needs, objectives, and ideas, (e) information about the team's expertise and resource, and (f) information about the team's intentions and already achieved results to steer the team with (b) requirements.

The most fundamental concepts for conceptualizing a goal-oriented system are those of a *system goal* and of *performance*. In a requirements communication context, the system goal is to ensure that the output from the development team, the solution, is accepted by the product manager. The performance is the degree of acceptance that is achieved when the solution is handed over to the product manager.

Simple goal-oriented systems consist of two communicating elements. One is the element in terms of which the system goal is defined, the *goal-implementing element*. The other element, the *goal-seeking element*, generates states of a *goal-seeking variable*. In requirements communication, the development team represents the goal-implementing element. Inputs to the team include expertise and resources needed to realize the output, the solution. The team is steered by the requirements, the goal-seeking variable. The product manager represents the goal-seeking element. Primary inputs to the product manager include needs, company objectives, and innovative ideas. The product manager produces the goal-seeking variable, the requirements used to steer the development team.

A well-performing system contains *goal-seeking traits* that contribute positively toward achieving the system goal, hence allowing the system to perform better than other systems. In requirements communication, the goal-seeking traits are the traces from given

requirements to the design decisions made to implement the solution.

Four paradigms define to what degree a goal-oriented system can be optimized: informationless, feedforward, feedback, and full-information. These four paradigms are illustrated in Figure 2.

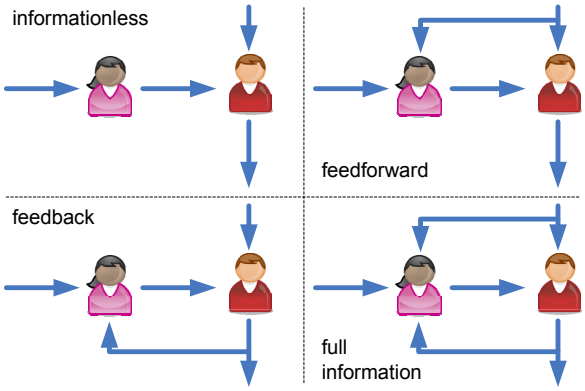

Figure 2. The four goal-oriented systems paradigms for optimizing requirements communication. Annotations in Figure 1.

The *informationless paradigm* is the most restrictive and the least powerful for achieving high performance. Here, the product manager formulates requirements and hands them over to the development team. The team then, given their expertise and resources, produces a solution that fits the requirements to best of their knowledge.

The performance of this system can be improved with the *feedforward paradigm*. Here, the product manager knows what resources will be used by the team. He also knows the degree of the team's expertise in relevant areas. This allows him to anticipate the team's outputs and adjust the requirements accordingly.

Alternatively, the performance of the system can be improved with the *feedback paradigm*. Here, the product manager uses the outputs from the development team to react to undesirable results and adjust the requirements accordingly. The earlier the feedback, the more powerful and less costly the adjustments become.

Feedback can be used by the product manager to support feedforward. Feedback allows him to learn about a team's expertise and resources to anticipate how to formulate requirements for that team. Such a combination of feedforward and feedback, called the *full-information paradigm*, corresponds to the most powerful paradigm for achieving high performance.

4. Case Study

Requirements communication performed according to the full-information paradigm should exhibit the following characteristics. The handed-over requirements specification does not necessarily comply with established quality criteria of requirements specifications such as those defined by standards [22]. Feedforward and feedback compensates for missing formal quality.

The product manager should know the development team. Rather than spreading his specification effort uniformly over the requirements specification, he tailors the requirements specification to the receiving team. By that he aims at maximizing the team's anticipated understanding.

The product manager seeks feedback from the development team to capture inadequate solution concepts that stem from requirements that turn out to be poorly specified. To correct misinterpretations of the requirements, he considers the traces between requirements and design decisions.

The better the product manager collaborates with the development team, the more the characteristics of the full-information paradigm for requirements communication become visible.

A case study [36] was performed to investigate the characteristics of requirements communication in a well-established collaboration between a product manager and a development team. Consistency of these characteristics with the model of goal-oriented requirements communication corroborates the validity of the model for such a situation.

For process improvement in the studied organization, a successful match between case study results and the characteristics of goal-oriented systems encourages institutionalizing techniques that support goal-oriented requirements communication.

Semi-structured interviews with the organization's most experienced product manager and most experienced architect were performed. The interviewed product manager had more than 20 years of experience in his role, the architect 19 years. Both were highly respected and trusted in their organization. They have a record of regular and successful collaboration with each other. The argumentation of the interviewees was expected to help understanding why feedforward and feedback are attractive requirements communication concepts.

In addition, a requirements specification for an in-house project was analyzed. The project was staffed with half of the organization's employees, which worked in four different countries. The specification was created by the product manager by considering company objectives, market needs, expectations of company-internal stakeholders, and technological capabilities of the organization. The requirements

specification, which the two parties considered a good one, contained approximately 500 requirements.

The remainder of this section presents the findings from the interviews and the analysis of the requirements specification. The quotes are from the interviews.

4.1 Assured Requirements Specification Qualities

Neither the product manager nor the architect expect requirements specifications to live up to standards. More important is the acceptability of the intended solution: *"It is quite often that a bad requirement creates a good solution design anyway."*

Still, the product manager assures a number of qualities of a requirements specification before it is handed over to the development team.

Validity: *"A requirement shall always be valid, which means that there shall be a customer demand behind it."*

Consistency: *"Requirements shall be consistent."*

Stability: *"Requirements should be stable for a market release time frame."*

Importance: *"Requirements are always ranked for importance because they need to be selected."* Only a small selection of possible requirements is communicated. Hence, the requirements specification contain only high-priority requirements.

Pre-traceability: *"The sources of the requirements shall be documented to provide feedback to their originator and to remember their rationale."*

Post-traceability (where pre-project development results exist): the specification refers to a number of product features, which have been implemented earlier by the development team.

4.2 Qualities Achieved while Communicating

During requirements communication, the two parties increase completeness, necessity, and correctness of both requirements and product design. These qualities are believed to be a result of successful requirements communication. Missing requirements and missing parts of the solution are detected. Scope is reduced depending on effort and timing estimates. Misinterpreted requirements are uncovered with consequent modification of the requirements and the tentative solution design.

Superfluous requirements do not make it to the final project scope: *"Only high-priority requirements will be implemented, because we will always be short on time and short on resources." "If the development team shows us that these cannot be implemented in time we will evaluate further cuts."*

Missing requirements are added when discovered: *"Requirements can never be more than 90% complete. We cannot write all details into the requirements – especially not those that are standard in the business." "The development team is very good at identifying forgotten requirements." "When the development team proposes a solution they find out that we would need other requirements also."*

Related to the discussion of missing requirements is the level of specification detail: *"When detailing requirements, I probably would not go below the competence of the technical specialist."* The better known a requirement is to the team, the shorter its specification is. Standardized functionality is simply referenced using titles and identifiers. On the other extreme, innovative requirements are enhanced with work results from collaborative design with domain experts.

Incorrect requirements are adjusted during requirements communication: *"Incorrect requirements that are changed exist, but are not frequent."*

To ensure that the requirements lead to an acceptable product, product management reviews design artifacts created by the development team as an answer to the requirements.

Superfluous design is removed from the final project scope: *"It is important that gold-plating[2] is not done, because the implementation takes longer and is more costly. There is always pressure to having something ready in time. Hence gold-plating always drops off at the end."*

Missing parts of the solution are added: *"If something is forgotten, we will notice." "Parts of the requirements are implemented by embedded software, by hardware, and by tool software. It is important to see how the division is made. The missing things are normally noticed in the design."*

Incorrect design gets adjusted: *"It is necessary that the solution is acceptable for the product manager, because he takes it to the customer." "It happens that we discover incorrect parts of the solution. These are uncovered in reviews of the design."*

4.3 Trust

The two parties establish trust through requirements communication: *"It is important that the development team believes that they have understood the requirements and that the product manager feels that the team will produce the right solution in time."*

Traceability between requirements and design is considered important for establishing trust: *"If I do not know how the various requirements will be implemented, the development team may create the wrong product." "It is necessary that the solution is justified by the requirements."*

[2] incorporating too many nice-to-have features

Shared expectations of effort and timing for implementing the requirements get established: *"The product manager needs to consider discrepancies before saying go ahead with implementation. If a proposed implementation is not within cost and time, it may be cancelled."*

4.4 Not Assured Specification Qualities

Requirements are further improved after requirements communication.

Testability: *"It is very difficult to state acceptance criteria beforehand, as they depend on the design of the solution."*

Other specification qualities are considered impossible to achieve or to be of low importance.

Unambiguity: *"Impossible, we are not that good in that."*

Formalized, concise formulations are considered to be solutions rather than requirements: *"Only if necessary. This is pointing to a solution*[3]*."* *"I would accept a description of an implementation that contains an algorithm."*

There is no strong opinion about the structure of the requirements specification: *"A reasonable structure suffices."* The structure of the requirements specification is adapted to its purpose such that it is easily comprehensible.

4.5 Evidence for Goal-Orientation

The case study confirms that, in the studied context, several traditional requirements qualities play a subordinate role in the requirements specification and are only achieved in the process of requirements communication. This is particularly the case for completeness, necessity, correctness, and unambiguity.

The product manager knows what resources are available to the development team. He refers to those in the requirements specification rather than specifying elaborate requirements. The product manager also knows the knowledge profile of the development team. He uses this knowledge to adjust the level of detail of the specification of different requirements.

The product manager seeks feedback from the development team. They exploit traces between requirements and design to discover misunderstandings and flaws in both the requirements specification and the product design. The project only proceeds when the intended solution is deemed acceptable by the product manager.

In situations such as the one described by the case study, employing a full-information goal-oriented approach for requirements communication turns out to be a well-working way of requirements communication and of steering a development team. Table 1 summarizes the value of feedforward and feedback and how feedforward and feedback are achieved.

Table 1. Value of feedforward and feedback.

Feedforward	
Guide development decisions	Product manager references existing development artifacts
Save specification effort	Product manager specifies just enough detail
Increase likelihood of understanding	Collaborative elaboration of innovative requirements
Feedback	
Take into account true R&D resource situation	Development team informs about requirement feasibility
Respond to interests of the development team	Development team shares its intentions
Identify misunderstandings	Product manager evaluates planned design
Transfer tacit knowledge	Product manager criticizes planned design
Build trust	Product manager confirms requirements understanding through accepting the justified solution

5. Related Work

Goal-oriented requirements communication can be used as a model to structure and interpret the role of established requirements engineering knowledge. Four threads of research can be distinguished, which the model complements with feedforward and feedback: learning, specification, negotiation, and transformation. Figure 3 shows how these topics can be positioned in the model. Such positioning can support method selection.

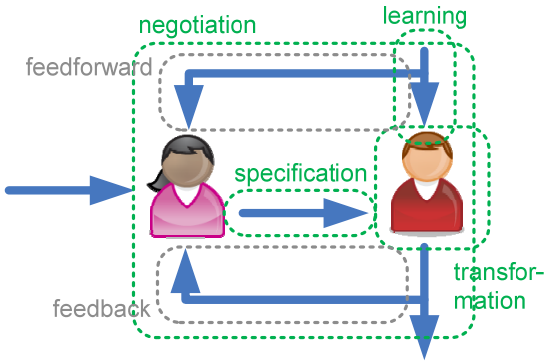

Figure 3. Positioning of established requirement engineering topics in goal-oriented requirements communication.

[3] It unnecessarily constrains the solution space.

5.1 Learning about Domain and Needs

Significant research has been dedicated to the learning aspect of requirements engineering under the term "elicitation". Assuming that a team is inexperienced in specific requirements-relevant topics, a number of techniques are known for eliciting stakeholder objectives and domain knowledge [16, 21]. Such elicitation can be systematized by following an inquiry cycle [30]. The inquiry can be further enhanced by domain and requirements analysis using appropriate modeling languages [7, 15, 27, 37], by prototyping the interfaces between the software solution and its environment [8, 17], or by employing collaborative design with domain experts [35].

Goal-oriented requirements communication looks at learning as an input to the development team that complements the requirements from product management. These requirements are intended for steering the development work and not for educating the team. When learning is required, the development team collaborates with other stakeholders and should inform the product manager about the changes to team knowledge.

5.2 Transformation (Formalization)

Much requirements engineering research has gone into employing formal methods for requirements specification instead of informal and natural language [1]. Increased precision and depth of a formal or almost formal specification is expected to support early validation of requirements and also ease the transition to design. A partial formalization can be achieved by using modeling languages [15, 27].

In goal-oriented requirements communication, requirements formalization work is delegated to the development team. This has two benefits. First, resources for the initial formulation of the requirements can be saved. Second, it can be checked whether the intentions of the product manager are adequately understood and addressed by the development team.

5.3 Specification

Some research has investigated how requirements can be specified in a high-quality manner. When requirements are expressed, mistakes in the perception and linguistic representation of facts can occur. Semantic language analysis helps to uncover and improve problematic formulations [32]. Specification styles and techniques are used to ensure that the stated requirements are understandable and testable [5, 14, 26]. Other work has investigated the necessary contents of requirements specifications. Taxonomies of requirements relevant-topics have been established to increase the likelihood that all relevant topics are addressed by a requirements specification [22, 23].

The hand-over of requirements is a central act in goal-oriented requirements communication. However, a perfect requirements specification is not an end in itself. Instead, the performance of requirements communication is addressed by tailoring the specification to their receiver (feedforward) and by checking the receiver's understanding (feedback).

5.4 Negotiation

Some requirements engineering research is dedicated to the topic of negotiation. Requirements negotiation is employed to address conflicts among stakeholders by following an appropriate process [20]. Requirements communication from a product manager to a development team is seen as one out of 16 negotiation constellations [13]. Successful negotiations between a project team and its stakeholders involve shifting of positions and expectations [28].

Goal-oriented requirements communication can be seen as a form of negotiation: a product manager and a development team negotiate with each other to come up with a good combination of requirements and design decisions for a development project. Such negotiation includes feedforward for preparing the negotiation and feedback for checking the match between requirements and implementation proposals, which is a basis for agreement [12].

5.5 Themes beyond Requirements Engineering

Feedback plays an important role in *iterative* and *incremental* software development [25], in particular in *agile* development. They all follow the basic idea of shortening the development cycle to promote early and continuous feedback to customers [2]. A collocated customer representative supports the project team in agile development. This allows the customer and the project team to learn from partial results, thus reducing project risk.

Iterative and incremental approaches help developing accepted software solutions with imperfect requirements. Goal-oriented requirements communication provides additional insights by suggesting that requirements understanding should also be addressed proactively by feedforward. Goal-oriented requirements communication can also be integrated into sequential, plan-driven methods by accepting feedback that does not necessitate software construction and testing.

Various forms of *reviews* are employed to prevent defects and to transfer knowledge between project members and stakeholders [3]. For example, ATAM [4] can be used for evaluating software architectures by tracing design decisions back to requirements.

However, reviews are performed after a specification has been created. Hence, they primarily support feedback when requirements communication has been concluded. Feedforward is not supported.

6. Discussion

6.1 Contributions

This paper has proposed a goal-oriented systems model for requirements communication from product management to a development team in a new product development context. The model allows drawing on the body of knowledge of systems sciences to increase the understanding of requirements communication.

Four paradigms have been presented that can be used to judge and optimize requirements communication methodology: informationless (the weakest paradigm), feedforward, feedback, and full-information (the strongest paradigm). The use of feedforward implies that a product manager prepares requirements communication by studying the requirements receiver. This allows him to tailor the requirements specification to increase the likelihood of requirements understanding. The use of feedback implies that a product manager studies work results from the development team. This allows him to react to inacceptable design decisions. The use of feedforward and feedback makes requirements communication more robust, hence allows imperfect requirements to be understood by the development team.

A case study was presented, which gives rich empirical results in order to form an understanding of the environment and thinking scheme in which goal-oriented requirements communication works and is accepted by practitioners.

It was shown how existing knowledge related to requirements communication can be structured and contrasted with the goal-oriented requirements communication model. The model shows that a development team should distinguish communication with the product manager from other related activities like learning about the domain and formalizing requirements. The model also helps a product manager to understand what activities the hand-over of a requirements specification should be embedded into. The model, finally, shows how negotiation, iterative and incremental software development, and reviews are related to requirements communication.

6.2 Validity

The presented work takes a constructivist research perspective [10] by showing why goal-oriented systems are a good model for requirements communication in the described case of mature, large-scale industrial product development.

Validity of the presented empirical data was addressed by triangulating from three data sources: two semi-structured interviews and one artifact analysis. The roles that were most concerned of requirements communication were interviewed. A representative requirements specification was analyzed. To ensure that the studied situation was understood, the first author of this paper collaborated with the organization for 1.5 years. Empirical results were presented in this study with as much detail as available space allowed.

Validity of the argumentation for the applicability of the goal-oriented system model was addressed by collaborating with the most experienced employees of the case organization. Their high level of experience gives confidence that the collected data reflected well-working practice. The interviewees confirmed that the research results made sense from their perspective. Practicability and impact of a systematic goal-oriented requirements communication process, however, should be further researched, for example by evaluating requirements communication methods that operationalize feedforward and feedback [12, 18].

6.3 Open Questions

Presented were research findings that fit a specific case. It remains open to what extent the presented findings can be generalized and, particularly, which are the exact conditions that make goal-oriented requirements communication attractive and effective. Hence, the model should be further researched, for example by replicating the study in different contexts.

It also remains open to compare the full-information goal-oriented approach for requirements communication with other requirements engineering approaches to study their similarities and differences, as well as their relative impact on requirements understanding, effort, and relationship between product management and development team.

7. Summary and Conclusions

Requirements engineering in a market-driven industrial environment involves balancing the capacity of handling large amounts of requirements against requirements specification quality. This poses particular challenges to requirements communication, because requirements cannot be handed over from product management to the development team in perfect quality.

This paper proposes a requirements communication model based on goal-oriented systems theory. Feedforward and feedback help increasing efficiency and effectiveness of requirements communication, while

becoming less dependent on the quality of a requirements specification. A case study is presented that shows the environment and thinking scheme in which such goal-oriented requirements communication works and is accepted by practitioners. The paper also shows how the model can be used to structure existing literature related to requirements communication.

Future work should refine the understanding of conditions for and effects of using goal-oriented requirements communication. Specific methods that operationalize the model should be investigated.

References

[1] D. Berry, "Formal Methods: The Very Idea. Some Thoughts About Why They Work When They Work", *Science of Computer Programming* 42 (2002), 11-27.

[2] B. Boehm, and R. Turner, Balancing Agility and Discipline: A Guide for the Perplexed, Addison-Wesley, 2004.

[3] M. Ciolkowski, O. Laitenberger, and S. Biffl, "Software Reviews: The State of the Practice", *IEEE Software* 20 (2003), 46-51.

[4] P. Clements, R. Kazman, and M. Klein, Evaluating Software Architectures: Methods and Case Studies, Addison-Wesley Professional, 2001.

[5] A. Cockburn, Writing Effective Use Cases, Addison-Wesley Professional, 2000.

[6] D. Damian, and D. Zowghi, "RE Challenges in Multi-Site Software Development Organisations", *Requirements Engineering* 8 (2003), pp.149-160.

[7] A. Dardenne, S. Fickas, and A. van Lamsweerde, "Goal-Directed Requirements Acquisition", *Science of Computer Programming* 20 (1991), pp.3-50.

[8] A. Davis, "Operational Prototyping: A New Development Approach", *IEEE Software* 9 (1992), pp.70-78.

[9] A. Davis, Software Requirements: Objects, Functions, and States, Prentice Hall PTR, 1993.

[10] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian, "Selecting Empirical Methods for Software Engineering". in: F. Shull, J. Singer, and D. Sjøberg, (Eds.), Guide to Advanced Empirical Software Engineering, Springer, 2008.

[11] S. Fricker, M. Glinz, and P. Kolb, "A Case Study on Overcoming the Requirements Tar Pit", *Journal of Universal Knowledge Management (J.UKM)* 1 (2006), pp.85-98.

[12] S. Fricker, T. Gorschek, and P. Myllyperkiö, "Handshaking between Software Projects and Stakeholders Using Implementation Proposals", Intl. Working Conference on Requirements Engineering: Foundation for Software Quality (RefsQ'07), 2007.

[13] S. Fricker, and P. Grünbacher, "Negotiation Constellations - Method Selection Framework for Requirements Negotiation", Intl. Working Conference on Requirements Engineering: Foundation for Software Quality (RefsQ'08), 2008.

[14] T. Gilb, "Advanced Requirements Specification: Quantifying the Quantitative", PSQT Conference, 1999.

[15] M. Glinz, S. Berner, and S. Joos, "Object-Oriented Modeling with ADORA", *Information Systems* 27 (2002), pp.425-444.

[16] J. Goguen, and C. Linde, "Techniques for Requirements Elicitation", IEEE Intl. Symposium on Requirements Engineering (RE'93), 1993.

[17] S. Gordon, and J. Bieman, "Rapid Prototyping: Lessons Learned", *IEEE Software* 12 (1995), pp.85-95.

[18] A. Griffin, and J. Hauser, "The Voice of the Customer", *Marketing Science* 12 (1993), 1-27.

[19] A. Griffin, and J. Hauser, "Integrating R&D and Marketing: A Review and Analysis of the Literature", *Journal of Product Innovation Management* 13 (1996), 191-215.

[20] P. Grünbacher, and N. Seyff, "Requirements Negotiation". in: A. Aurum, and C. Wohlin, (Eds.), Engineering and Managing Software Requirements, Springer, 2005, pp.143-162.

[21] A. Hickey, and A. Davis, "Elicitation Technique Selection: How Do Experts Do It?" 11th IEEE Intl. Requirements Engineering Conference (RE'03), 2003.

[22] IEEE, "IEEE Recommended Practice for Software Requirements Specifications", IEEE Standard 830-1998, 1998.

[23] ISO/IEC, "Software Engineering - Product Quality - Part 1: Quality Model", ISO/IEC Standard 9126-1, 2001.

[24] G. Klir, Facets of Systems Science, Kluwer Academic / Plenum Publishers, 2001.

[25] C. Larman, and V. Basili, "Iterative and Incremental Development: A Brief History", *Computer* 36 (2003), pp.47-56.

[26] S. Lauesen, Software Requirements: Styles & Techniques, Addison-Wesley Professional, 2002.

[27] OMG, "Unified Modeling Language (UML) Version 2.1.2", Object Management Group, 2007.

[28] P. Ovaska, M. Rossi, and K. Smolander, "Filtering, Negotiating and Shifting in the Understanding of Information System Requirements", *Scandinavian Journal of Information Systems* 17 (2005), 31-66.

[29] B. Paech, J. Dörr, and M. Koehler, "Improving Requirements Engineering Communication in Multiproject Environments", *IEEE Software* 22 (2005), 40-47.

[30] C. Potts, K. Takahashi, and A. Antón, "Inquiry-Based Requirements Analysis", *IEEE Software* (1994), pp.21-32.

[31] B. Regnell, and S. Brinkkemper, "Market-Driven Requirements Engineering for Software Products". in: A. Aurum, and C. Wohlin, (Eds.), Engineering and Managing Software Requirements, Springer, 2005, pp.287-308.

[32] C. Rupp, "Requirements and Psychology", *IEEE Software* (2000), pp.16-18.

[33] I. van de Weerd, S. Brinkkemper, R. Nieuwenhuis, J. Versendaal, and L. Bijlsma, "Towards a Reference Framework for Software Product Management", 14th IEEE Intl. Requirements Engineering Conference (RE'06), 2006.

[34] A. van Lamsweerde, "Goal-Oriented Requirements Engineering: A Roundtrip from Research to Practice", 5th IEEE Intl. Symp. on Requirements Engineering (RE'01), 2001.

[35] J. Wood, and D. Silver, Joint Application Development, Wiley, 1995.

[36] R. Yin, Case Study Research: Design and Methods, SAGE Publications, 2003.

[37] E. Yu, "Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering", 3rd IEEE Intl. Symposium on Requirements Engineering (RE'97), 1997.