Reverse Presentations

A Client-Driven Method for Requirements Engineering in Offshore Software Development

Requirements engineering is frequently cited as one of the most critical stages in software development. In an offshore setting, this criticality is even increased by distance challenges. The paper presents a client-oriented method for requirements validation in offshore software development. The method aims at creating a common understanding of the future system by means of "reverse presentations". This core element of the method facilitates the transfer of knowledge across social worlds for the purpose of validation. Case studies confirm the method's fit with the offshore context as well as its positive impact on the inter-organizational interaction and control.

DOI 10.1007/s12599-010-0100-1

The Authors

Dr. Martin Wiener (✉)
Department of Information Systems
University of Erlangen-Nuremberg
Lange Gasse 20
90403 Nuremberg
Germany
martin.wiener@wiso.uni-erlangen.de
url: http://www.wi3.uni-erlangen.de

Dipl.-Inform. Rolf Stephan
Institute of Applied Informatics
and Formal Description Methods
(Institut AIFB)
KIT Karlsruhe Institute of Technology
Kaiserstraße 89
76133 Karlsruhe
Germany
rolf.stephan@kit.edu
url: http://www.aifb.kit.edu

Received: 2009-07-31
Accepted: 2010-03-05
Accepted after two revisions by the editors of the special focus.
Published online: 2010-04-23

This article is also available in German in print and via http://www.wirtschaftsinformatik.de: Wiener M, Stephan R (2010) Reverse Presentations. Eine kundenorientierte Methode zur Anforderungsvalidierung in der Offshore-Softwareentwicklung. WIRTSCHAFTSINFORMATIK. doi: 10.1007/s11576-010-0221-7.

1 Introduction

Companies are increasingly relocating some or all of their software development activities to vendors in low-wage countries like India, China, or Russia, the so called offshore software development (OSD) (e.g., Carmel and Tija 2005; Dibbern et al. 2008; Heeks et al. 2001; Sahay et al. 2003; Willcocks and Lacity 2006). One of the key drivers for client organizations to engage in OSD projects is to reduce labor costs (e.g., Apte and Mason 1995; Currie et al. 2003; Rottman and Lacity 2004; Schaaf 2004). Other potential OSD benefits include the access to a large pool of highly skilled workers, the reduction of development time, as well as the proximity to markets and customers (O Conchuir et al. 2009a, 2009b). Against this background, there is a growing debate on which development tasks can and which cannot be offshored. One stream of research suggests that activities like coding, testing, and bug fixing are a better fit for offshore locations, while more complex activities like requirements engineering (RE) are better to be carried out onshore (Carmel and Tija 2005). However, cost savings are limited as long as only low-value adding tasks are included in an OSD project (Edwards and Sridhar 2005). Heeks et al. (2001) argue that clients need to move their OSD relationships up the value chain to reap greater benefits. One possible way to do so is to offshore RE tasks.

Recent surveys confirmed the inherent difficulty (Cheng and Atlee 2007) and growing importance of RE in research and practice (Van Lamsweerde 2000a). RE is often cited as a crucial stage in the software development process (e.g., Browne and Rogich 2001; Evaristo et al. 2005; Hanisch and Corbitt 2007; Hofmann and Lehner 2001; Maciaszek 2001; Kotonya and Sommerville 1998). This is because mistakes during this early project stage cascade into the later stages (Browne and Rogich 2001; Edwards and Sridhar 2005; Stephan 2005). For instance, Boehm and Basili (2001; first published in Boehm 1987) found that mistakes during the requirements phase can cost up to one hundred times more than coding errors. The special nature of OSD projects even increases the criticality of the RE phase by posing unique challenges to the client (Sangwan et al. 2007). Those challenges may primarily arise from cultural, geographic, linguistic, and time zone differences between the client and the vendor country (Dibbern et al. 2008),

and significantly impact the collaboration with the offshore employees (Winkler et al. 2008).

In spite of growing literature on OSD (Wiener 2010), prior research focused primarily on the later stages of the software development lifecycle (Grinter et al. 1999). There are only a few studies addressing the critical RE phase of OSD (Yadav et al. 2007). This can be attributed to the relative newness of the area (Boehm et al. 2001) and the common belief that only more mechanical phases like coding are suitable for OSD (Yadav et al. 2007). The majority of recent studies view RE in OSD projects in a dyadic manner: either they suggest a face-to-face or a distributed RE approach. By contrast, we were able to identify a single study, by Carmel and Tija (2005), which initially discusses a combination of these RE approaches based on the nature of an OSD project. With regard to OSD-specific methods or tools, we only found a generic management framework for RE best practices (Bhat et al. 2006). Although prior literature suggests a multitude of RE methods, we do not know whether these methods can address the specific RE challenges in a global project setting. This paper aims to fill this gap by presenting as well as conceptually and theoretically refining a method for requirements validation in OSD. Given the novelty of the phenomenon and the paucity of research in this area (Damian and Zowghi 2003) as well as the high complexity (Briggs and Grünbacher 2002), it is important to provide researchers and practitioners with a structured approach for doing RE in OSD projects.

In our study, we use a design science research approach. According to Hevner et al. (2004, p. 77), "design science [...] creates and evaluates IT artifacts intended to solve identified organizational problems". Such artifacts may comprise constructs, models, instantiations, and methods. Our overall goal is to fill the existing knowledge gap in the OSD domain with the contribution of a RE method that is tailored to the validation of requirements in this specific domain (Zowghi 2002). Here, we aim "to bridge practice to theory rather than theory to practice" (Holmström and Ketokivi 2009, p. 65). Thus, the entry point for our research was a practical solution that worked. This is referred to as a client-/context-initiated approach (Peffers et al. 2007) because it starts with a design science solution and "researchers work backward to apply rigor to the process retroactively" (p. 56). The use of this approach is consistent with Agerfalk and Fitzgerald's (2006) observation that practice is ahead of research in terms of RE in OSD projects, and that there is a need to better conceptualize and theorize fundamental underpinnings. To ensure the method's scientific value added our research is guided by Hevner et al.'s (2004) principles for design science research. These include the application of a rigorous process to identify an important business problem, to design the artifact, to evaluate the design, and to communicate the results to appropriate audiences.

The paper is structured as follows: the next chapter positions our work in the context of prior research. We then design, present and initially evaluate our RE method, and conclude by discussing implications for practice and research.

2 Theoretical Background

RE is one of the most challenging aspects of software development (Yadav et al. 2009) as it determines "what the [software] system should do" (Crowston and Kammerer 1998, p. 227). Van Lamsweerde (2000a, p. 5) defines RE as "the identification of the goals to be achieved by the envisioned system, the operationalization of such goals into services and constraints, and the assignment of responsibilities for the resulting requirements to agents such as humans, devices, and software." Sommerville (2007) classifies requirements by their level of abstraction (user vs. system requirements) and their origin (functional vs. non-functional requirements). In this paper, we concentrate on functional user requirements due to our focus on business, not technical aspects.

According to Byrd et al. (1992), RE typically involves a group of analysts working with (end) users to establish a common understanding of organizational needs related to the software system to be developed. Several authors (e.g., Browne and Rogich 2001; Cheng and Atlee 2007; Hanisch and Corbitt 2007; Jarke and Pohl 1994) broadly discuss RE as a three-step process:

1. *Capturing*: Gathering and eliciting the requirements (from users);
2. *Specification*: Analyzing, modeling, and documenting the captured requirements in some explicit fashion (e.g., activity, data flow, entity-relationship, state, and use case diagrams as well as screen prototypes);
3. *Validation*: Checking and showing the correctness (in terms of "fit for purpose") of the specified requirements.

In this paper, we basically adopt this process but add "targeting" as fourth and initial RE phase. This phase aims to ensure a common understanding with regard to the basic functionality of the software system, and hence a target-oriented capturing, specification, and validation of relevant requirements. Especially in OSD, a targeting phase of valid requirements can be regarded as particularly important as conflicts, misunderstandings, and misinterpretations easily arise in such a project (Carmel 1999; Winkler et al. 2008).

2.1 Existing RE Methods

Prior literature suggests a multitude of RE methods. In the following well-established methods are briefly described and discussed (Geisser et al. 2007): VOLERE is one of the most comprehensive RE methods (Robertson and Robertson 2006), supporting all phases along the RE process. However, it requires direct communication and physical presence of stakeholders and, therefore, does not account for the specific requirements of a distributed RE (Geisser et al. 2007). EWW (EasyWinWin) is a RE method that supports the capturing of requirements by leveraging collaborative technologies (Grünbacher and Boehm 2001). It is based on a groupware system that enables the involvement and interaction of key stakeholders. By developing a shared vision as well as by creating win-win situations, the method aims at reaching a fundamental consensus on relevant requirements among stakeholders. According to Geisser et al. (2007, p. 200), EWW is currently "the only method for a collaborative requirements capturing which was also frequently used in practice". However, due to its focus on a single RE phase, it does not support later phases (Geisser and Hildenbrand 2006). Further, physical meetings still play a crucial role when applying EWW. Thus, greater modifications are required to allow the use of this method in a distributed setting. ARENA (Anytime, Anyplace Requirements Negotiation Aids) is a web-based tool which, at least partially, transfers the EWW method to a distributed context (Geisser et al. 2007). A major limitation is that this tool only

supports asynchronous work as it completely replaces the original EWW groupware (Seyff et al. 2005). Moreover, it is not possible to integrate external applications into ARENA due to missing interfaces and specific requirements of EWW meetings (Grünbacher and Boehm 2001).

Within all RE methods mentioned above, decisions (e.g., requirements selection) are based on subjective qualitative assessments by stakeholders. Prior literature suggests two methods which include quantitative aspects to support the objective selection of requirements (Geisser et al. 2007). These are: CVA (Cost-Value Approach) (Karlsson and Ryan 1997) and QWW (Quantitative WinWin) (Ruhe et al. 2002). Using Saaty's (1980) Analytic Hierarchy Process (AHP), both methods possess a strong mathematical foundation, which was proven to be suitable for prioritizing software requirements (Karlsson et al. 1998). However, both methods concentrate on one specific RE phase and provide only limited guidance for other phases.

In order to unify and extend the advantages of several other methods (EWW, CVA and QWW), Geisser et al. (2007) propose the so called DisIRE (Distributed Internet-Based Requirements Engineering) method. Even though this method is designed for a distributed context and provides support along the entire RE process, it is neither aimed at addressing the client perspective nor the social aspects of RE. **Table 1** summarizes and classifies the presented RE methods (in alphabetical order) (see also Van Lamsweerde 2000a).

In summary, only two methods provide cross-phase support along the RE process (DisIRE in a distributed RE context and VOLERE in a face-to-face context). Geisser et al. (2007) emphasize the importance of a continuous method support along the RE process, particularly in a spatially scattered project. On the one hand such support avoids media breaks and loss of information; on the other hand it facilitates document consistency as well as traceability of requirements changes and mutual dependencies across requirements (Sommerville 2007). A major limitation of both methods is their focus on the vendor point of view. They aim at enabling a vendor company to systematically capture, specify, and validate requirements (Geisser et al. 2007), thereby neglecting coordination and control aspects from a client perspective. Furthermore, DisIRE and VOLERE concentrate on structuring the RE process and defining associated steps for each process phase. It is suggested that a more structured RE method will lead to more clearly defined and understood requirements (Carmel 1999; Kotonya and Sommerville 1998). However, structured methods may also inhibit socialization (Hanisch 2001), which can cause miscommunications (Hanisch and Corbitt 2007). Therefore, Thanasankit (2002) argues that (especially in global projects) organizations need to incorporate social aspects of RE.

2.2 RE Challenges in OSD

RE has always been a problematic area of software development (Browne and Rogich 2001). Problems come from a variety of sources, including human limitations (Davis 1982) and the "high dynamic complexity" (Briggs and Grünbacher 2002). OSD, as opposed to traditional onshore, collocated software development, even exacerbates these problems due to four major factors: cultural (Krishna et al. 2004; Rao 2004), geographic (Carmel and Agarwal 2002; Rao 2004), linguistic (Rao 2004; Zatolyuk and Allgood 2004), and time zone differences (Rao 2004; Rottman and Lacity 2006) between the client and vendor country. All of these factors imply a certain type of distance and, thus, "may be referred to as different categories of offshore-specific client-vendor distance measurers" (Dibbern et al. 2008, p. 341). Those distance measurers lead to specific (social) challenges which can affect both single RE phases and the entire RE process.

Time zone differences between the client and the vendor country enforce communication related issues as they shorten time slots for voice or video conversations (Carmel 1999). Due to the limited overlap of working hours, OSD projects show a tendency to heavily rely on document-based communication in the RE process. In their study on the interplay between conflict and culture in globally distributed RE, Damian and Zowghi (2003) found that the simple exchange of documents is a very poor strategy for clearly communicating requirements.

Language differences are a significant problem in achieving a common understanding of the required software functionality between client and vendor (Layman et al. 2006), and increase the potential for misunderstandings (Hanisch and Corbitt 2007). Sarker and Sahay (2004) observed such issues including mismatch in preferred language for conversation and misinterpretation of conversation style. Additionally, language barriers may affect the transfer of relevant knowledge to the offshore vendor. According to Damian and Zowghi (2003), this challenge is particularly meaningful in the case of non-English speaking clients.

Geographic differences make face-to-face contact between the client and vendor teams difficult and/or expensive. Thus, geographic distance represents a significant barrier to the interactions between system users, analysts, and developers. First, it limits formal communication, i.e., the active participation and collaboration of stakeholders in the RE process (Damian and Zowghi 2003). Second,

Table 1 Overview of selected RE methods

RE method	Supported RE phase(s)	Decision basis	Support of distributed RE	Tool support
ARENA	Capturing	Qualitative	Yes	Yes
CVA	Selection	Quantitative (AHP)	Yes	No
DisIRE	All	Quantitative (AHP)	Yes	Yes
EWW	Capturing	Qualitative	Limited (physical meetings still crucial)	Limited (groupware)
QWW	Selection	Quantitative (AHP)	Yes	No
VOLERE	All	Qualitative	No (physical presence required)	No

it reduces the opportunities for informal communication between the stakeholders, resulting in lower levels of trust, lower awareness of local working context, and lower transparency of working progress at remote sites (Layman et al. 2006).

Cultural differences between the contracting parties cause a number of social challenges to the RE process, such as developing trust between team members, accounting for communication preferences, and creating a cultural sensitivity (Hanisch et al. 2001). In addition, different cultural backgrounds can lead to unrealistic performance expectations, for instance, when national holidays or religious festivals are disregarded (Sarker and Sahay 2004). If not effectively addressed, these challenges may lead to conflicts and decrease in trust (Winkler et al. 2008). Here, the development of a shared context and shared meanings between client and vendor are essential to reduce the complexity of an OSD project (Hanisch and Corbitt 2007). This is especially important when communication-intensive tasks like RE are offshored.

As already stated by Beath (1987) and Kirsch (1997), software development is not only a technical but also a social process, which demands close collaboration among a diverse set of stakeholders. This is particular true for the RE stage as it requires a higher degree of communication than other stages in software development (Hanisch and Corbitt 2007). Distance challenges in OSD affect the social process of RE, i.e., the coordination of and the control over the vendor (Winkler et al. 2008), making it difficult to achieve and validate a consistent understanding of requirements across social worlds (Damian and Zowghi 2003). Thus, RE in OSD needs to carefully consider existing distance challenges (Zowghi 2002). Failure to fully understand the required system features may result in budget and schedule overruns and, ultimately, in damaged client-vendor relationships (Damian et al. 2006; Layman et al. 2006).

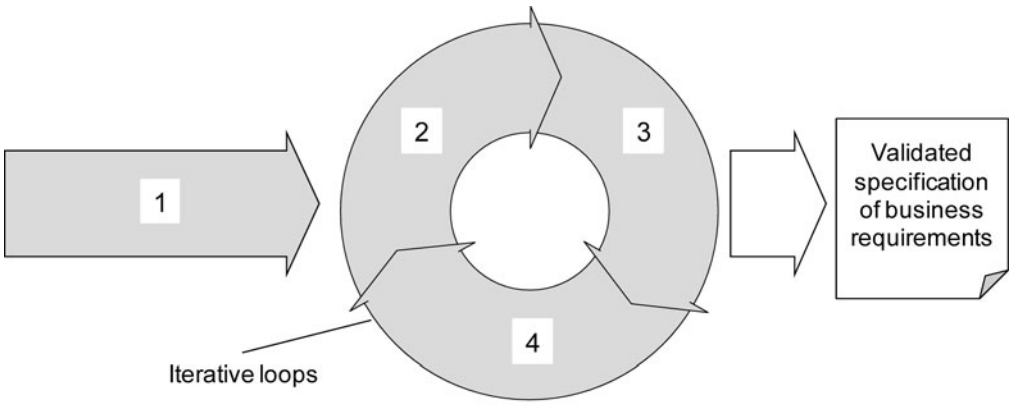
Furthermore, RE in OSD projects requires significant interaction between the contracting parties (Yadav et al. 2009). Thus, it is typically conducted at the client location. Here, collocated teams of users as well as business and system analysts work closely together, finally communicating the requirements to the development staff at the offshore location (Damian and Zowghi 2003; Ramesh and Dennis 2002). In an attempt to improve the cost arbitrage, OSD client organizations increasingly consider distributed RE as a possible alternative to the classical face-to-face approach (Yadav et al. 2009). This is also picked up by an emerging stream of research (Bhat et al. 2006; Damian and Zowghi 2003; Edwards and Sridhar 2005; Evaristo et al. 2005; Nath et al. 2006; Ocker et al. 1995). In a distributed scenario, analysts and developers located offshore interact in a virtual mode with the clients located onshore to capture, specify, and validate requirements. On the one hand, a shift to virtual teams could lead to cost savings, exploit existing time differences to reduce cycle time, and take advantage of the distributed team members' expertise in developing robust requirements artifacts (Damian et al. 2000). On the other hand, RE is a communication-intensive task and, therefore, heavily affected by the above mentioned distance challenges (Edwards and Sridhar 2005). The great number and diversity of these challenges indicate that the total offshoring of RE (which means that all RE tasks are performed in a distributed mode) can be considered a risky endeavor and is still uncommon in practice (Nath et al. 2008; Yadav et al. 2009).

While distributed RE "may be desirable in achieving economy of resources", face-to-face RE may be helpful in considering social aspects "so that lasting relationships [...] may be formed, and RE activities achieved" (Hanisch and Corbitt 2007, p. 793). The multitude of pros and cons of both basic approaches (Nicholson and Sahay 2004) suggests that OSD may require a more flexible RE approach, balancing these two extremes. Taking into account the specific OSD context and RE phase, the selected location strategy can be closer to the distributed or the face-to-face end of the continuum (Carmel and Tija 2005).

3 Design of a Method for Requirements Validation in OSD

No matter how much effort the contractual partners have put into the specification of the requirements, it will contain some gaps, conflicts, or potential for misunderstandings and misinterpretations. In an onshore context, an "inadequate" specification is often uncritical. For example, German firms frequently have a small number of strong relationships with local software vendors (Buchta et al. 2004) who possess extensive domain know-how. They expect from their vendors to independently close existing gaps or resolve conflicts in the specifications document. Although such an RE approach may work in a national setting, it will probably not work in an international one. This can be traced back to distance challenges in OSD, possibly resulting in a weak relationship and an insufficient knowledge transfer between client and vendor. Often, it is unclear what the offshore vendor has really understood. For instance, a Swiss bank may take it for granted that a software vendor with experience in the banking industry knows about the country-specific rounding logic when calculating the account balance. With regard to OSD, it is at least questionable whether a comparable Indian vendor is aware of such "minor" differences. For this reason, particularly in OSD, it is critical to validate the vendor's understanding of the requirements. Our RE method for OSD exactly addresses this issue. Instead of solely communicating the requirements to the vendor, the client asks the vendor to capture, specify, and present the requirements ("Reverse Presentation") for validation purposes in an iterative manner. Based on these presentations the client can evaluate the vendor's understanding of the software system to be developed.

At this point, it has to be noted that OSD basically comprises both captive offshoring and offshore outsourcing of software development activities (Wiener 2010). Consequently, the term client may refer to either an organization (e.g., a domestic customer company or an IT service provider) which operates an offshore subsidiary (*internal client*) or an organization which cooperates with an offshore third-party vendor (*external client*); (Batra et al. 2006; Davis et al. 2006). In this paper, we concentrate on (non-captive) software offshore outsourcing, i.e., "external clients", as we assume that it is particularly difficult to establish a common understanding in such a setting. The paper's main focus lies on business requirements. This can be reasoned by the high criticality of these requirements in a software project in general and an OSD project in particular. Nevertheless, we believe that Reverse Presentations might also help in communicating and validating technical requirements across contexts.



	1 Targeting	2 Capturing	3 Specification	4 Validation
Key technique	Representative use cases	Use cases	Data flow, state diagrams, etc.	Reverse Presentations
Goal	Conveyance of *basic* vendor understanding	Development of *detailed* vendor understanding	Documentation of vendor understanding	Ensurance of *common* understanding
Predominant location strategy	Face-to-face	Face-to-face and/or distributed	Distributed	Face-to-face

Fig. 1 Overview of RPM process

3.1 Business Needs and Design Goals

According to Hevner et al. (2004, p. 79), "framing research activities to address business needs assures research relevance". Based on our practical experience in the OSD field and the results of our literature review, we were able to identify three major business needs with regard to an OSD-specific RE method:

- *Concentration on the client perspective*: Established RE methods predominantly focus on the vendor perspective (Geisser et al. 2007). Due to their lack of client orientation, these methods widely disregard the particularly high importance of client control in an OSD project (Rustagi et al. 2008).
- *Combination of basic approaches*: Existing RE methods as well as prior literature on RE approaches in OSD typically depend on or favor either face-to-face or distributed RE. In contrast, we do not know of any RE method seeking to combine the benefits that both approaches have to offer (Carmel and Tija 2005; Hanisch and Corbitt 2007; Nicholson and Sahay 2004).
- *Consideration of social aspects*: Primarily resulting from cultural, geographic, linguistic, and time zone differences between client and vendor, OSD poses unique distance challenges to RE (Bhat et al. 2006; Damian and Zowghi 2003). Traditional methods do not cater for these more social aspects of RE (Zowghi 2002) or even inhibit these aspects (Hanisch 2001).

The identified business needs translate directly into the design goals for our method (Peffers et al. 2007). By concentrating on the client perspective the method aims at improving the control over the offshore vendor (*control improvement*). By combining the advantages of face-to-face (e.g., relationship building) and globally distributed RE (e.g., cost reduction) it aims at reducing the overall efforts associated with an OSD project (*effort reduction*). Most importantly, by considering the social aspects of OSD it aims at bridging the existing distance challenges (*context fit*) in order to create and validate a consistent understanding of the requirements across social worlds. Such an understanding may contribute to a more accurate, complete, correct, and robust definition of requirements, and therefore to the success of an OSD project. The stated design goals also serve as basis for the initial evaluation of our method in Sect. 4 (Hevner et al. 2004).

Moreover, the identified business needs support the development of a new RE method. According to Damian and Zowghi (2003), researchers should pay due attention to developing RE methods that address the specific characteristics and issues of an OSD project. Here, Zowghi (2002) claims that especially "social issues are at the heart of many of the problems in RE and [...] cannot solely be addressed by the currently available technical methods" (p. 54). She concludes that novel RE approaches and methods need to be sought.

3.2 Method Process

The Reverse Presentations Method (RPM) extends existing RE process models by the initial targeting phase, resulting in a four-phase process. The input for the RPM process is a basic vision of the software system and the certainty that such a system is realistic (Geisser et al. 2007). The process output is a validated specification of business requirements. This specification may serve as basis for the negotiation of requirements with the offshore vendor and the translation into technical requirements by the vendor. As indicated in **Fig. 1**, the RPM builds on an iterative approach which combines elements of face-to-face and distributed RE. The reverse character of this approach emphasizes the social aspects of OSD.

(1) *Targeting*: The client selects and describes representative use cases. Due to our focus on functional user requirements, use cases are an appropriate modeling technique (Wiegers 2005). Representative refers to a high-level description of cross-sectional system functions from

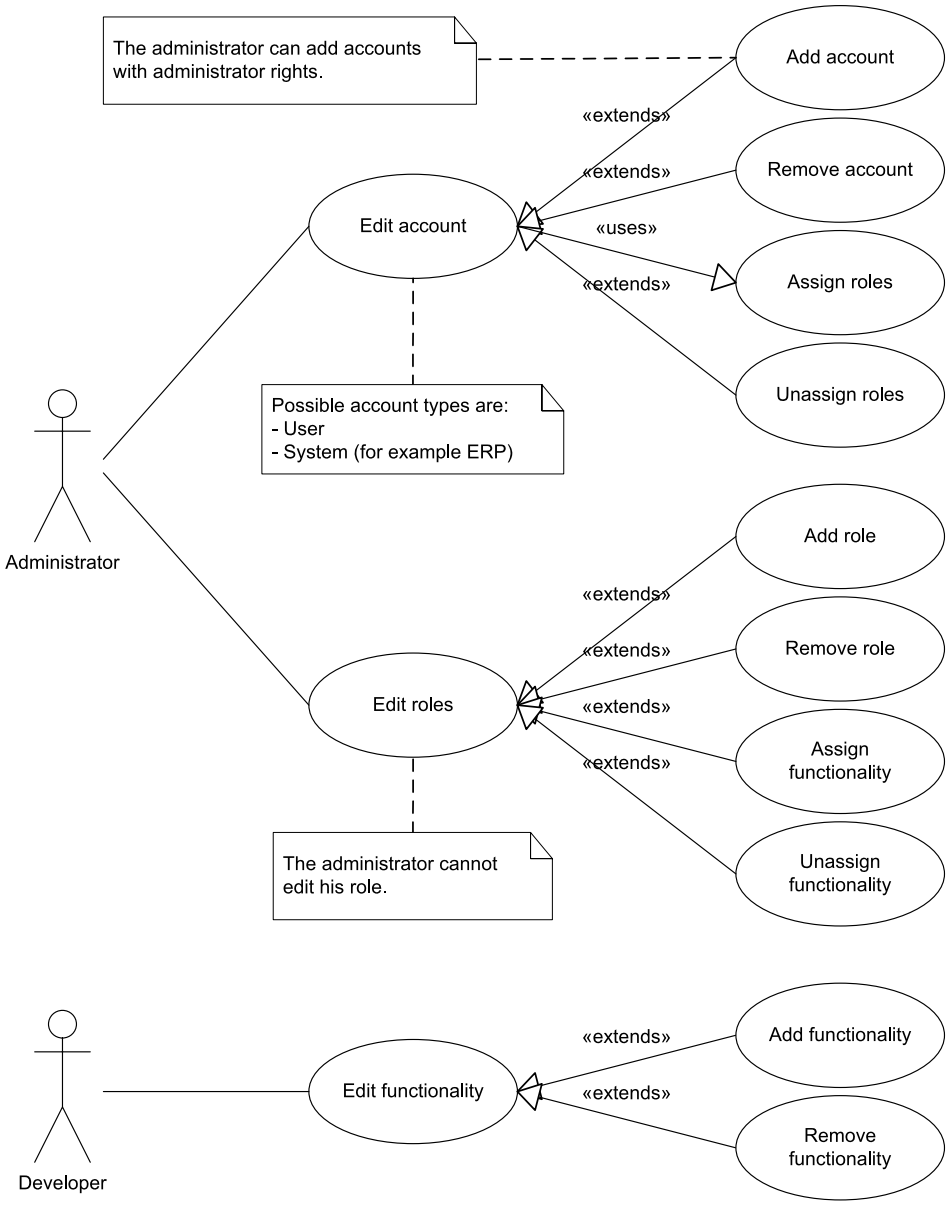


Fig. 2 Exemplary use case: "user account management"

a user perspective (see **Fig. 2** for an example). This description excludes any technical details and attempts to convey an overall impression of the targeted software system. For use case illustration, the client can use both professional modeling tools (e.g., Rational Rose, Enterprise Architect) and standard office tools. Dependent on the complexity of the respective software module, typically up to ten use cases are defined for each module. The representative use cases are not formally verified and make no claim to be complete.

In an initial meeting, the client presents and hands over the defined use cases to the vendor. This meeting usually takes place face-to-face at the client location as it also serves for getting to know the stakeholders and building trust among them (Damian et al. 2003). The goal is to give the vendor a basic understanding of the development task. Based on this rudimental information, the vendor can identify internal business and system analysts with relevant experiences and propose a strategy for requirements capturing.

If the OSD project deals with the reengineering of a legacy system, the client may provide the vendor with access to this system. This would complement the know-how transfer because it strengthens the vendor's understanding of the business context. However, it can never replace an intensive capturing phase as new requirements are particularly critical. Further, a too extensive review of the legacy system may also negatively influence later RE phases (e.g., a certain prejudice hampering new innovative solution statements).

(2) *Capturing*: The requirements capturing typically takes place onshore at the client location by means of face-to-face interviews (Yadav et al. 2009). However, recent research rather supports the idea of distributed capturing (e.g., Bhat et al. 2006). In line with Carmel and Tija (2005), the RPM does not predefine one specific location strategy but suggests a more flexible approach: based on project, system, and vendor characteristics (Choudhury and Sabherwal 2003), the OSD partners need to select a suitable location mix for capturing requirements. **Table 2** gives a high-level guideline for selecting the best-fit strategy by comparing

Table 2 Comparison of face-to-face and distributed requirements capturing

Criterion	Face-to-face capturing	Distributed capturing
Project size	Large	Small
System complexity	High	Low
System criticality	High	Low
System innovativeness	High	Low
Vendor domain know-how	Limited ("beginner")	Very high ("expert")
Vendor relationship	Weak (initial)	Strong (long lasting)
Major advantages	• Building of relationship • Improvement of control • Lower potential for misunderstandings	• Exploitation of time differences • Reduction of costs (e.g., for travelling)

the two basic location strategies in terms of suitability (criteria in alphabetical order) and major advantages. It has to be added that the feasibility of a pure distributed capturing strategy in OSD is still subject of controversial discussions (Nath et al. 2008; Yadav et al. 2009).

Based on the selected capturing strategy, the vendor refines and completes the uses cases provided in the targeting phase, and adds other relevant use cases. Checklists, templates, or mind maps can support this work. The goal is that the vendor gets a more comprehensive and detailed picture of the software system to be developed.

(3) *Specification*: The requirements specification builds on the developed use cases. First, the vendor extracts business requirements from the use cases. Then, the vendor structures, details, and models these requirements in some explicit fashion (e.g., data flow or state diagrams). Finally, the vendor summarizes the captured requirements in the form of a written document. This document can be enriched by first prototypes ranging from static screen prototypes to dynamic functional prototypes (Lichter et al. 1993). The goal of this phase is that the vendor documents his system understanding.

Please note that it is almost impossible to specify a software system without any gaps or conflicts. This can primarily be attributed to human limitations (Davis 1982) as well as moving targets during the course of the project (Briggs and Grünbacher 2002). Only formal specification methods claim to converge towards full completeness and integrity. Even though these formal methods are essential in some business applications (e.g., critically important financial software systems), their effort cannot be justified in most business applications and their use in practice is limited (Fraser et al. 1994; Van Lamsweerde 2000b).

(4) *Validation*: The validation phase represents the core of our method. In this phase, the vendor takes on the role of the client and proactively explains the functionality of the software system to the client (*client pull*). Here, the client only takes on a passive role. This proceeding can be regarded as a reversion of the classical approach in which the client explains the future system to the vendor (*client push*).

Based on a Reverse Presentation, the client tests and evaluates the vendor's understanding, and decides on next steps: either he accepts the vendor presentation and triggers the translation of the presented business into technical requirements; or he asks the vendor to iteratively refine the requirements by an additional loop of capturing, specification, and validation. The overall goal of this (final) phase is to ensure a sufficient understanding on the vendor side before the technical specification of the system begins.

The Reverse Presentation typically takes place in a face-to-face workshop at the client location. In this workshop, the vendor uses traditional presentation media (e.g., slides, flipcharts, and first prototypes). Due to the personal interaction, the client may better estimate the vendor's understanding by also observing behavioral aspects, in particular the presenter's body language. As the name indicates, body language is concerned with the activities of elements of the human body: hand movements, facial expressions, eye contact, posture, proxemics, body rhythms, and speech (Jenkins and Johnson 1977). To attain the most complete and accurate picture of the vendor understanding, the client needs to consider these elements of the "human communications subsystem". In their research on the relevance of body language to the information analyst in the development of management information systems, Jenkins and Johnson (1977, p. 46) conclude that body language "is too powerful a communication construct to be ignored". According to these authors, body language possesses the advantage that some expressions such as fear, anger, and interest are fairly independent from culture. However, just as verbal language, body language tends to vary between cultures in some aspects. Therefore, the client needs to be alert to these variances when interpreting the nonverbal behavior from different cultures.

3.3 Method Roles and Responsibilities

RE requires interaction between numerous stakeholders from both the client and vendor organization. With regard to the RPM, stakeholders can be assigned to six roles. All of these roles can be filled by a single person or a group of persons alternatively. On the part of the client, relevant roles (in alphabetical order) are:

- *Domain expert*: Brings in significant experience in business domain, represents user perspective, serves as central source for requirements elicitation;
- *Project manager*: Monitors and controls OSD project performance from a client point of view;
- *System designer*: Acts as interface between business and IT department, describes and designs system functionality.

Consciously, the RPM does not include the project sponsor as it focuses on content-related RE aspects (i.e., the establishment and validation of a common understanding concerning the requirements between client and vendor), not economical ones (e.g., the selection of requirements based on an analysis of their

Table 3 Overview of roles and their responsibilities along the RPM process

Role	Targeting	Capturing	Specification	Validation
Client				
Domain expert	Check suitability of use cases	Coordinate input process; Deliver input	(Answer emerging questions)	Check common understanding
Project manager	Check suitability of use cases	Decide on capturing strategy		Accept or reject presentation
System designer	Select and define use cases; Present use cases	Deliver input	(Answer emerging questions)	Check common understanding
Vendor				
Business analyst	(Analyze legacy system)	Collect input; Refine and complete use cases	Structure, detail, model, and document (client) input	Present specifications document
Project manager	Check use cases; Select business and system analysts	Support decision making for capturing strategy		Test internal understanding
System analyst	(Analyze legacy system)	Collect input; Refine and complete use cases	Structure, detail, model, and document (client) input; Develop first prototypes	

costs and benefits). Relevant roles (again in alphabetical order) for the vendor are:

- *Business analyst (or subject matter expert)*: Brings in basic knowledge of business domain, possesses experience with selected requirements capturing strategy, represents business perspective;
- *Project manager*: Monitors and controls project performance from a vendor point of view;
- *System analyst*: Possesses experience with selected capturing strategy and relevant software systems, brings in technical perspective.

These six roles take over different responsibilities within the RPM process. The role-specific responsibilities are outlined in **Table 3** (roles sorted by perspective and alphabetical order).

In addition to the activities stated in **Table 3**, the project managers on either side are responsible for continuously monitoring and controlling the project performance (in terms of cost, quality, and time) across all phases.

4 Initial Method Evaluation

According to Peffers et al. (2007, p. 56), an "evaluation could include any appropriate empirical evidence or logical proof". In an initial step towards evaluating the RPM, we apply a twofold approach in line with Hevner et al.'s (2004) guideline on design evaluation. First, we describe experiences with the method use from a practical perspective (What worked and what did not, in what circumstances?) and compare these experiences with the stated design goals (*observational evaluation*). Second, we take on a more theoretical perspective by comparing the method with existing RE methods as well as relating it with relevant research in the broader context of organizational learning and knowledge integration (*descriptive evaluation*).

4.1 Observational Evaluation

4.1.1 Empirical Exploration

In order to gain empirical insights into the RPM usage, we conducted multiple case studies with German-speaking client companies in the last five years. According to Hevner et al. (2004), case studies are especially applicable for the purpose of observational evaluation. A brief overview of the cases is given in **Table 4**. In five cases (Insur1, Logis1, Manu1, Insur2, and Trav1), one member of the research team gained access to the respective client companies and consulted them in their OSD initiatives. Here, participant observations and review sessions with key informants enabled us to develop the RPM, to gather client feedback on the method (Peffers et al. 2007), and to iteratively refine it (Markus et al. 2002). This course of action provided us with "essential feedback to the construction phase as to the quality of the design process and the design product under development" (Hevner et al. 2004, p. 85).

For the purpose of initial method evaluation, we conducted one-hour interviews with one IS senior (project) manager from each case partner. The interview language was German. All interviews were held in a semi-structured manner and took place via conference call (four interviews) or face-to-face meeting (two interviews) between July and October 2009. In the interviews, the participants were asked to characterize their OSD projects (e.g., on three-point Likert scales) and to describe their experiences with the RPM. While three of the six case partners already made formal usage of the RPM (adherent to the process and roles described in Sect. 3), the other three partners used the method informally or only applied single elements of the RPM depending on specific situations (e.g., Reverse Presentations at critical project stages).

In an effort to ensure the open nature and authenticity of the informants' statements, we decided to keep written records of all interviews rather than to record them (Urquhart 2001). We transcribed the interviews immediately after each interview session. To extract case-specific findings, we encoded and

Table 4 Case overview

Case	Formal RPM usage			Informal RPM usage		
	Insur1	Logis1	Manu1	Insur2	Serv1	Trav1
Sector	Insurance	Transportation	Manufacturing	Insurance	IT services	Travel
Interviewee	Senior project manager	Senior project manager	Chief software architect	Steering board member	Senior delivery manager	Head of application development
Number of projects	1	>10	1	2	>20	>20
(Average) size [person-year]	70 PY	15 PY	350 PY	60 PY	0,5 PY	10 PY
System	Core insurance system	Machine control system	CAD/CAM application	Data warehouse system	Web applications	Travel reservation system
Complexity	High	Medium	High	Medium	Low	High
Criticality	Medium	High	High	High	Low	High
Innovativeness	Medium	Medium	High	Low	Medium	Medium
Vendor country	India	India	India	India	Ukraine/India	India
Domain know-how	Limited	Limited	Advanced	Limited	Very high	Limited

structured each transcript. The coding involved the identification of positive and negative experiences with the RPM in consideration of the stated design goals. Building on the individual findings, we performed a cross-case analysis (Yin 2003) in order to identify similarities and differences throughout the cases.

4.1.2 Findings

Table 5 summarizes the key statements from the case interviews[1] and relates the coded user experiences from the six cases to the corresponding design goals.

In summary, the conducted interviews indicate that the RPM largely fulfills two of its three design goals. First, the method addresses the existing distance challenges between German-speaking clients and especially Asian software vendors (*context fit*), thereby increasing the degree of bilateral communication and common understanding between the OSD partners. Second, it enhances the level of client control in terms of controlling the communication flows, the project progress, and the understanding of the system requirements on either side (*control improvement*). By contrast, the interviews do not clearly point to the promise that the RPM also fulfills its third design goal (*effort reduction*). Comparative in-depth case studies and experiments including quantifiable measures are necessary for examining the method's influence on the total efforts of an OSD project.

4.2 Descriptive Evaluation

To descriptively evaluate the RPM, we start with a comparison of our method with the DisIRE method which we perceived as the most elaborated RE method (see Sect. 2.1). Both methods possess similar conceptual strengths: cross-phase support, flexibility, intuitive usability, realistic expectations, suitability for distributed context, and trust building (Geisser et al. 2007). Due to the smaller scope of the RPM along the RE process, not all DisIRE strengths can be included in the comparison as some of them refer to the requirements selection phase (e.g., hierarchical application of AHP). A major weakness of the RPM is the limited tool support. Currently, the method is not associated with a specific software tool but can be partially supported by standard modeling tools like Rational Rose or Enterprise Architect. However, by suggesting Wiki technologies (Geisser et al. 2007), DisIRE as well provides only basic tool support with regard to the capturing, specification, and validation phase.

Overall, DisIRE aims at assisting software vendors to systematically capture, select, specify, and validate requirements in a distributed context. Here, it focuses on structuring the RE process and providing tool support along this process. While the RPM also defines a basic structure of the RE process, it focuses on the client perspective and goes a step further by incorporating social aspects of RE (Thanasankit 2002). Here, the method's reverse character aims at addressing the OSD-specific RE distance challenges. This may facilitate the establishment and validation of a common understanding of the requirements between project partners.

In this respect, the RPM is based on the same assumptions as *exploratory prototyping* where the builder seeks to demonstrate through a design, artifact, or behavior how he understands the future system should operate. This type of prototyping is suited when the problem is not fully understood (Lichter et al. 1993). It can be used to identify gaps in assumptions and domain knowledge on the part of the vendor, but as well to find out what the client really wants. Even though, (exploratory) prototyping provides a discussion basis for all stakeholders along the software development process, it typically relies on a shared vision of the future system and the interaction with system users (Lichter et al. 1993). After having produced an early working version of the system, users work with this prototype to test its usability and/or functionality, and to give feedback to the developers. As opposed to prototyping, the RPM only covers the initial stages of the development process (including the generation of a common vision and understanding) and focuses on the interactions within the core project team (analysts, designers, subject matter experts, etc.), thereby

[1] Interviewees' statements have been translated to English.

Table 5 Coding of user experiences and design goals

Case	Key statement	User experience (code)	Design goal
Insur1	"Reverse Presentations had a positive effect on the social interactions within the project team."	Bridging of cultural and geographical distance	Context fit
Logis1	"The method's reverse character increased the degree of both formal and informal communication between the project partners."		
Insur1	"The generally high cost pressure in OSD projects often prevents the execution of an additional iteration of Reverse Presentations."	Difficulty in stringent application	
Manu1	"The rigid application of the method requires at least basic domain know-how on the part of the offshore vendor. This is particularly true for the refinement of the representative use cases."		
Serv1	"Reverse Presentations are particularly suitable for OSD projects with Asian software vendors."	Aptitude for Asian culture	
Logis1	"The RPM is an effective way for dealing with the Asian culture."		
Manu1	"Especially the Asian team members' tendency of simply saying 'yes' and the resulting lack of feedback can be mitigated by means of Reverse Presentations."		
Serv1	"It is questionable whether the use of Reverse Presentations is also beneficial in a nearshore context due to the cultural proximity of the contractual partners in such a setting."		
Serv1	"The method facilitates the controlling of the communication flows between the client and vendor teams."	Process and progress control	Control improvement
Manu1	"The method's structured process supports the controlling of the project progress on the vendor side."		
Logis1	"Reverse Presentations can be used to assess the vendor's degree of (business) understanding."	Control of vendor understanding	
Trav1	"Often, client firms realize too late that certain knowledge has not reached the vendor. Here, Reverse Presentations may improve the transparency of the knowledge transfer."		
Logis1	"The RPM helps to identify aspects and features of the software system which have been disregarded thus far."	Control of own understanding	
Insur2	"The method supports the identification of specification gaps."		
Manu1	"The quality of the presentation, and thus the level of control heavily depend on the person presenting."	Subjective character of presentation	
Insur1	"The RPM reduced the overall project efforts by lowering the number of early mistakes and required iterations."	Decrease in number of iterations	Effort reduction
Insur1	"The required face-to-face meetings for the Reverse Presentations lead to a significant increase in travel costs."	Increase in travel efforts	
Logis1	"It is difficult to measure the method's impact on project efforts."	Unclear effect on project efforts	
Manu1	"The efforts associated with the RPM are quite high. However, one hopes that the avoidance of early mistakes lowers the total efforts."		
Insur2	"People doubt that the method might contribute to a decrease in project efforts. Otherwise, one would observe a more stringent method application."		
Serv1	"It is doubtful that the method reduces the project efforts but it may reduce the number of mistakes."		

supporting the bidirectional knowledge transfer.

Furthermore, the issue of requirements capturing, specification, and validation is the generation of valid knowledge of a system that can be built in future and which fits its stakeholders' needs. In this regard, the RPM's idea of reversing the roles and making the vendor to present the client's requirements may be one way of improving how the interactions between "*perspective making and taking*" (Boland and Tenkasi 1995) take place. Perspective making is the process whereby a community of knowing develops and strengthens its unique knowledge. Perspective taking is the process whereby such communities interact and utilize their distinctive knowledge with the goal of understanding the view of other communities. With regard to RE in OSD, the knowledge transfer between the OSD partners is not a problem of exchanging or making data commonly available; rather it is a problem of perspective making and taking. A Reverse Presentation – the core element of the

RPM method – addresses this problem. The preparation of the presentation may support the development of a strong perspective (*perspective making*) by the client and the vendor "necessary to do important knowledge work" (Boland and Tenkasi 1995, p. 357). The actual presentation may improve the ability to appreciate and include the knowledge of the other community as well as to establish a common understanding (*perspective taking*).

In the sense of Boland and Tenkasi (1995), a Reverse Presentation can be regarded as a (*design*) *boundary object* because it represents an "artifact that is shared between two or more actors at the border of two social worlds" (Bergman et al. 2007, p. 550). These authors define a design boundary object "to be any representational artifact that enables knowledge about a designed system [...] to be transferred between social worlds and that simultaneously facilitates the alignment of stakeholder interests populating these social worlds by reducing design knowledge gaps" (p. 551). Operationally, the four essential features of such an object are the capability to promote shared representation, to transform design knowledge, to mobilize for action, and to legitimize design knowledge. All of these features may be applied to a Reverse Presentation: it conveys representations of the requirements that can be shared between the OSD partners; it transforms the knowledge of both client and vendor in order to further refine design knowledge within the OSD project team; it mobilizes stakeholders for action (e.g., by uncovering gaps in knowledge and agreements); finally, it needs to be consistent with the client understanding of the future system in order to be accepted.

5 Conclusions and Outlook

Agerfalk and Fitzgerald (2006) highlight that practice is ahead of research in terms of RE in OSD, and that there is an evident need to better conceptualize and theorize fundamental underpinnings. The RPM represents a RE method used for requirements validation in OSD projects. In this paper, we presented and conceptually refined this method based on empirical and theoretical findings as well as carried out an initial method evaluation.

Similar to existing RE methods the RPM provides cross-phase support by structuring the necessary steps along the RE process. Such a support is particularly important in a globally distributed project (Geisser et al. 2007). However, in contrast to other methods, the RPM focuses on the perspective of an OSD client and also takes into account social aspects of RE in OSD. These aspects can be regarded as a root cause for many RE problems and cannot solely be addressed by existing, more technical methods (Zowghi 2002). The core idea of the RPM is to create and validate a common understanding of the future system between client and vendor by means of Reverse Presentations. This core element of the RPM may be an effective way of improving the interactions between the development of a strong understanding on both sides ("perspective making") and the establishment of a common understanding ("perspective taking") (Boland and Tenkasi 1995). In this sense, a Reverse Presentation may act as a design boundary object which facilitates the transfer and validation of knowledge about a software system across social worlds (Bergman et al. 2007). An insufficient knowledge transfer and validation within the RE stage is a major source for unsatisfactory results of an OSD project (Wiener 2006). Mistakes in this early project stage may escalate to later stages (Browne and Rogich 2001; Edwards and Sridhar 2005; Stephan 2005), thereby jeopardizing the project success and, ultimately, the client-vendor relationship (Damian et al. 2006; Layman et al. 2006).

The paper has several important implications for practice. Most importantly, it clearly shows the need for OSD clients to pay adequate attention to the existing distance challenges in RE. Numerous informants acknowledged that the reverse character of the RPM can make a significant contribution to bridge the cultural and geographical distance. Especially in projects with Asian vendors, the method's reverse approach positively influences the degree of inter-organizational interaction and understanding. To increase their level of control, OSD clients should rely on a structured and iterative RE process. Such a process may be used to control the communication flows, the project progress, and the vendor understanding. As remarked by two case partners, this process may also enable the client to test and refine its own understanding of the system to be developed. In addition, the inclusion of a dedicated targeting phase and the early integration of the offshore vendor in the RE process may reduce the likeliness of moving targets and early mistakes, thereby decreasing the overall efforts of an OSD project.

The above implications for practice must be viewed against some limitations. First, the RPM is explicitly designed for addressing the social distance challenges of RE in OSD projects. One might wonder whether it might also be used in onshore, collocated software development projects. There would appear to be no reason it could not be used; however, there are method elements that are intended to support essential OSD project characteristics that might not apply to other projects. Thus, for traditional software development, the RPM may be perceived as "over-engineered" for some contexts. Second, we so far excluded the cost-benefit analysis and selection of business requirements as well as the specification of technical requirements from the method scope. We believe that the early RE phases, i.e., the generation and validation of a consistent understanding of the business requirements across social worlds, are a key to OSD project success. However, we may consider building an integrated, cross-stage version of the RPM at a later stage. Third, basic domain know-how on the part of the selected vendor is crucial for the RPM to work. No vendor can afford to build up the required knowledge just to try to compete for a contract. Finally, a major shortcoming of the method is the limited tool support. To some extent, this can be explained by its focus on social aspects of RE. Nevertheless, we are currently in the process of identifying and evaluating appropriate RE tools. In a next step, we plan to integrate these tools in the RPM.

In conclusion, two major opportunities for future research emerge from this paper. One case partner questioned whether the RPM is also suitable for a nearshore context. Case studies of near- and offshore projects involving the same client would help to further understand the method contributions in both contexts. Finally, the paper's principal aim was to present a method for requirements validation in OSD and to determine how well this artifact works, not to examine

Abstract

Martin Wiener, Rolf Stephan

Reverse Presentations

A Client-Driven Method for Requirements Engineering in Offshore Software Development

Reverse Presentations is a method for requirements validation in offshore software development. In this paper, the authors present and conceptually refine this method and carry out an initial evaluation. The method provides cross-phase support and is characterized by a structured and iterative validation process. In contrast to existing methods, it focuses on the client perspective and takes into account social distance challenges. The method aims at creating a common understanding of the future system by means of "reverse presentations". This core element of the method facilitates the transfer of knowledge across social worlds for validation purposes. Case studies with clients confirm that the method fits well with the offshore software development context. The cases point to the method's positive impact on the interorganizational interaction and control.

Keywords: Offshore outsourcing, Software development, Requirements validation, Reverse presentations method, Knowledge transfer

or prove why it works. "This is where design-science and behavioral-science researchers must complement one another" (Hevner et al. 2004, p. 88).

Acknowledgements

Inputs from Ulrich Remus, Michael Reinhardt, Bianca Vogel and Florian Fischl are gratefully acknowledged. The authors thank the Associate Editors and two anonymous reviewers for their valuable comments and suggestions. They are also deeply indebted to all the case partners who devoted considerable time and provided them with the information upon which this paper is based.

References

Agerfalk PJ, Fitzgerald B (2006) Flexible and distributed software processes: old petunias in new bowls? Communications of the ACM 49(10):27–35

Apte UM, Mason RO (1995) Global disaggregation of information-intensive services. Management Science 41(7):1250–1262

Batra D, Sin T, Tseng SY (2006) Modified agile practices for outsourced software projects. In: Proc 12th Americas conference on information systems (AMCIS), Acapulco

Beath CM (1987) Managing the user relationship in information systems development projects: a transaction governance approach. In: Proc 8th international conference on information systems (ICIS), Pittsburgh

Bergman M, Lyytinen K, Mark G (2007) Boundary objects in Design: an ecological view of design artifacts. Journal of the Association for Information Systems 8(11):546–568

Bhat JM, Gupta M, Murthy SN (2006) Overcoming requirements engineering challenges: lessons from offshore outsourcing. IEEE Software 23(5):38–44

Boehm B (1987) Industrial software metrics top 10 list. IEEE Software 4(5):84–85

Boehm B, Basili VR (2001) Software defect reduction top 10 list. Computer 34(1):135–137

Boehm B, Grünbacher P, Briggs RO (2001) Developing groupware for requirements negotiation: lessons learned. IEEE Software 18(3):46–55

Boland RJ, Tenkasi RV (1995) Perspective making and perspective taking in communities of knowing. Organization Science 6(4):350–372

Briggs RO, Grünbacher P (2002) EasyWinWin: Managing complexity in requirements negotiation with GSS. In: Proc 35th Hawaii international conference on system sciences (HICSS), Hawaii

Browne GJ, Rogich MB (2001) An empirical investigation of user requirements elicitation: comparing the effectiveness of prompting techniques. Journal of Management Information Systems 17(4):223–249

Buchta D, Linß H, Röder H, Ziegler R (2004) IT-Offshoring und Implikationen für den Standort Deutschland. Unpublished Article, AT Kearney

Byrd TA, Cossick KL, Zmud RW (1992) A synthesis of research on requirements analysis and knowledge acquisition techniques. MIS Quarterly 16(1):117–138

Carmel E (1999) Global software teams. Prentice Hall, Englewood Cliffs

Carmel E, Agarwal R (2002) The maturation of offshore outsourcing of information technology work. MIS Quarterly Executive 1(2):65–78

Carmel E, Tija P (2005) Offshore information technology: sourcing and outsourcing to a global workforce. Cambridge University, Cambridge

Cheng BHC, Atlee JM (2007) Research directions in requirements engineering. In: Proc 29th international conference on software engineering (ICSE), Minneapolis

Choudhury V, Sabherwal R (2003) Portfolios of control in outsourced software development projects. Information Systems Research 14(3):291–314

Crowston K, Kammerer EE (1998) Coordination and collective mind in software requirements development. IBM Systems Journal 37(2):227–241

Currie WL, Desai B, Khan N, Wang X, Weerakkody V (2003) Vendor strategies for business process and applications outsourcing: recent findings from field research. In: Proc 36th Hawaii international conference on system sciences (HICSS), Hawaii

Damian DEH, Zowghi D (2003) An insight into the interplay between culture, conflict and distance in globally distributed requirements negotiations. In: Proc 36th Hawaii international conference on system sciences (HICSS), Hawaii

Damian DEH, Eberlein A, Shaw MLG, Gaines BR (2000) Using different communication media in requirements negotiation. IEEE Software 17(3):28–36

Damian DEH, Eberlein A, Shaw MLG, Gaines BR (2003) An exploratory study of facilitation in distributed requirements engineering. Requirements Engineering Journal 8(1):23–41

Damian DEH, Lanubile F, Mallardo T (2006) The role of asynchronous discussions in increasing the effectiveness of remote synchronous requirements negotiations. In: Proc 28th international conference on software engineering (ICSE), Shanghai

Davis G (1982) Strategies for information requirements determination. IBM Systems Journal 21(1):4–30

Davis GB, Ein-Dor P, King WR, Torkzadeh R (2006) IT offshoring: history prospects and challenges. Journal of the Association for Information Systems 7(11):770–795

Dibbern J, Winkler J, Heinzl A (2008) Explaining variations in client extra costs between software projects offshored to India. MIS Quarterly 32(2):333–366

Edwards HK, Sridhar V (2005) Analysis of software requirements engineering exercises in a global virtual team setup. Journal of Global Information Management 13(2):21–41

Evaristo R, Watson-Manheim MB, Audy J (2005) E-collaboration in distributed requirements determination. International Journal of E-Collaboration 1(2):40–55

Fraser MD, Kumar K, Vaishnavi VK (1994) Strategies for incorporating formal specifications in software development. Communications of the ACM 37(10):74–86

Geisser M, Hildenbrand T (2006) A method for collaborative requirements elicitation and decision-supported requirements analysis. In: Ochoa SF, Roman GC (eds) Advanced

software engineering: expanding the frontiers of software technology. Springer, Boston, pp 108–122
Geisser M, Heinzl A, Hildenbrand T, Rothlauf F (2007) Verteiltes internetbasiertes Requirements-Engineering. WIRTSCHAFTS-INFORMATIK 49(3):199–207
Grinter R, Herbsleb J, Perry D (1999) The geography of coordination: dealing with distance in R&D work. In: Proc international ACM SIGGROUP conference on supporting group work, Phoenix
Grünbacher P, Boehm B (2001) EasyWinWin: a groupware-supported methodology for requirements negotiation. ACM SIGSOFT Software Engineering Notes 26(5):320–321
Hanisch J (2001) Requirements engineering during virtual software development: achieving balance. In: Proc information resources management association (IRMA) international conference, Toronto
Hanisch J, Corbitt B (2007) Impediments to requirements engineering during global software development. European Journal of Information Systems 16(6):793–805
Hanisch J, Thanasankit T, Corbitt B (2001) Exploring the cultural and social impacts on the requirements engineering processes – highlighting some problems challenging virtual team relationships with clients. Journal of Systems and Information Technology 5(2):1–19
Heeks R, Krishna S, Nicholson B, Sahay S (2001) Synching or sinking: global software outsourcing relationships. IEEE Software 18(2):54–60
Hevner A, March S, Park J, Ram S (2004) Design science in information systems research. MIS Quarterly 28(1):75–105
Hofmann HF, Lehner F (2001) Requirements engineering as a success factor in software projects. IEEE Software 18(4):58–66
Holmström J, Ketokivi M (2009) Bridging practice and theory: a design science approach. Decision Sciences 40(1):65–87
Jarke M, Pohl K (1994) Requirements engineering in 2001: (virtually) managing a changing reality. Software Engineering Journal 9(6):257–266
Jenkins AM, Johnson RD (1977) What the information analyst should know about body language. MIS Quarterly 1(3):33–47
Karlsson J, Ryan K (1997) A cost-value approach for prioritizing requirements. IEEE Software 14(5):67–74
Karlsson J, Wohlin C, Regnell B (1998) An evaluation of methods for prioritizing software requirements. Information and Software Technology 39(14–15):939–947
Kirsch LJ (1997) Portfolios of control modes and IS project management. Information Systems Research 8(3):215–239
Kotonya G, Sommerville I (1998) Requirements engineering: processes and techniques. Wiley, Toronto
Krishna S, Sahay S, Walsham G (2004) Managing cross-cultural issues in global software outsourcing. Communications of the ACM 47(4):62–66
Layman L, Williams L, Damian DEH, Bures H (2006) Essential communication practices for extreme programming in a global software development team. Information and Software Technology 48(9):781–794
Lichter H, Schneider-Hufschmidt M, Züllighoven H (1993) Prototyping in industrial software projects – bridging the gap between theory and practice. In: Proc 15th international conference on software engineering (ICSE), Baltimore
Maciaszek L (2001) Requirements analysis and systems design: developing information systems with UML. Addison-Wesley, Toronto
Markus ML, Majchrzak A, Gasser L (2002) A design theory for systems that support emergent knowledge processes. MIS Quarterly 26(3):179–212
Nath D, Sridhar V, Adya M, Malik A (2006) The effect of user project monitoring on the performance of virtual teams in the requirements analysis phase of off-shored software projects. In: Proc INFORMS conference on information systems and technology (CIST), Pittsburgh
Nath D, Sridhar V, Adya M, Malik A (2008) Project quality of off-shore virtual teams engaged in software requirements analysis: an exploratory comparative study. Journal of Global Information Management 16(4):24–45
Nicholson B, Sahay S (2004) Embedded knowledge and offshore software development. Information and Organization 14(4):329–365
O Conchuir E, Agerfalk PJ, Olsson HH, Fitzgerald B (2009a) Global software development: where are the benefits? Communications of the ACM 52(8):127–131
O Conchuir E, Olsson HH, Agerfalk PJ, Fitzgerald B (2009b) Benefits of global software development: exploring the unexplored. Software Process Improvement and Practice 14(4):201–212
Ocker R, Hiltz SR, Turoff M, Fjermestad J (1995) The effects of distributed group support and process structuring on software requirements development teams: results on creativity and quality. Journal of Management Information Systems 12(3):127–153
Peffers K, Tuunanen T, Rothenberger MA, Chatterjee S (2007) A design science research methodology for information systems research. Journal of Management Information Systems 24(3):45–77
Ramesh V, Dennis AR (2002) The object-oriented team: lessons for virtual teams from global software development. In: Proc 35th Hawaii international conference on system sciences (HICSS), Hawaii
Rao MT (2004) Key issues for global IT sourcing: country and individual factors. Information Systems Management 21(3):16–21
Robertson S, Robertson J (2006) Mastering the requirements process, 2nd edn. Wesley, Upper Saddle River
Rottman JW, Lacity MC (2004) Twenty practices for offshore outsourcing. MIS Quarterly Executive 3(3):117–130
Rottman JW, Lacity MC (2006) Proven practices for effectively offshoring IT work. MIT Sloan Management Review 47(3):56–63
Ruhe G, Eberlein A, Pfahl D (2002) Quantitative winwin – a new method for decision support in requirements negotiation. In: Proc 14th international conference on software engineering and knowledge engineering, Ischia
Rustagi S, King WR, Kirsch LJ (2008) Predictors of formal control usage in IT outsourcing partnerships. Information Systems Research 19(2):126–143
Saaty TL (1980) The analytic hierarchy process. McGraw-Hill, New York
Sahay S, Nicholson B, Krishna S (2003) Global IT outsourcing: software development across borders. Cambridge University, Cambridge
Sangwan R, Bass M, Mullick N, Paulish DJ, Kazmeier J (2007) Global software development handbook. Auerbach, Boca Raton
Sarker S, Sahay S (2004) Implications of space and time for distributed work: an interpretive study of US-Norwegian systems development teams. European Journal of Information Systems 13(1):3–20
Schaaf J (2004) Offshoring: globalisation wave reaches services sector. Deutsche Bank Research Economics 45:2–15
Seyff N, Hoyer C, Kroiher E, Grünbacher P (2005) Enhancing GSS-based requirements negotiation with distributed and mobile tools. In: Proc 14th IEEE international workshop on enabling technologies: infrastructure for collaborative enterprise. Linkoping, pp 87–92
Sommerville I (2007) Software engineering. Wesley, Harlow
Stephan R (2005) Kommunikation und Wissenstransfer – Schlüsselfaktoren für erfolgreiche Offshore-Projekte. In: Hermes HJ, Schwarz G (eds) Outsourcing – Chancen und Risiken, Erfolgsfaktoren, rechtssichere Umsetzung. Haufe, Freiburg, pp 221–229
Thanasankit T (2002) Requirements engineering – exploring the influence of power and Thai values. European Journal of Information Systems 11(2):128–141
Urquhart C (2001) An encounter with grounded theory: tackling the practical and philosophical issues. In: Traut E (ed) Qualitative research in information systems: issues and trends. Idea, London
Van Lamsweerde A (2000a) Requirements engineering in the year 00: a research perspective. In: Proc 22nd international conference on software engineering (ICSE), Limerick
Van Lamsweerde A (2000b) Formal specification: a roadmap. In: Proc 22nd international conference on software engineering (ICSE), Limerick
Wiegers KE (2005) More about software requirements: thorny issues and practical advice. Microsoft, Redmond
Wiener M (2006) Critical success factors of offshore software development projects – the perspective of German-speaking clients. Gabler, Wiesbaden
Wiener M (2010) Offshore software development – a multi-perspective research framework and agenda. In: Tagungsband der Multikonferenz Wirtschaftsinformatik (MKWI), Göttingen
Willcocks LP, Lacity MC (2006) Global sourcing of business & IT services. Palgrave, New York
Winkler J, Dibbern J, Heinzl A (2008) The impact of cultural differences in offshore outsourcing – case study results from German-Indian application development projects. Information Systems Frontiers 10(2):243–258
Yadav V, Nath D, Sridhar V, Adya M (2007) Investigating an 'agile-rigid' approach in globally distributed equirements analysis. In: Proc 11th Pacific-Asia conference on information systems (PACIS), Auckland
Yadav V, Adya M, Sridhar V, Nath D (2009) Flexible global software development (GSD): antecedents of success in requirements analysis. Journal of Global Information Management 17(1):1–31
Yin RK (2003) Case study research: design and methods, 3rd edn. Sage, Thousand Oaks
Zatolyuk S, Allgood B (2004) Evaluating a country for offshore outsourcing: software development providers in the Ukraine. Information Systems Management 21(3):28–33
Zowghi D (2002) Does global software development need a different requirements engineering process? In: Proc 24th international conference on software engineering (ICSE), Orlando