Integration by communication: Knowledge exchange in global outsourcing of product software development

Expert Systems

Article

DOI: 10.1111/j.1468-0394.2012.00640.x

Integration by communication: knowledge exchange in global outsourcing of product software development

Baldur Kristjánsson, Remko Helms and Sjaak Brinkkemper

Department of Information and Computing Sciences, Utrecht University, P.O. Box 80.089, 3584 CC Utrecht, The Netherlands

Email: baldurkr@gmail.com; r.w.helms@uu.nl; s.brinkkemper@uu.nl

Abstract: *Global outsourcing is a growing trend among independent software vendors. In these projects like other distributed work, distances have negative effects on communication and coordination, directly impacting performance. We present a normative model designed to address this issue by improving communication and knowledge exchange. The model consists of six distinct practices and a tool blueprint, each coming with practical guidelines. It is based in part on two case studies of Dutch software vendors who have successfully outsourced part of their activities to an Eastern European outsourcing vendor, and validated by a panel of six experts from industry and the scientific community. It is concluded that knowledge exchange in global software outsourcing is a by-product of efforts to enhance communication and coordination, rather than specific technical solutions. By committing to sharing knowledge, emphasizing transparency and integrating the outsourcing team into their organizations, customers from the product software business can realize the benefits of global outsourcing.*

Keywords: knowledge management, global software outsourcing, offshore outsourcing, product software development

1. Global outsourcing for software vendors

For some years, global outsourcing of software development has been an emerging trend, some refer to it as an 'irreversible megatrend' (Rao, 2004), and a recent survey has shown that a substantial portion of IT jobs at larger companies may be shifting offshore (Thibodeau, 2008). Global software-outsourcing providers can provide scarce resources in a cost-competitive manner, and this has caught the attention of independent software vendors (ISVs) who are in the core business of developing software, but see an opportunity in the outsourcing of part of their software development activities (Herbsleb & Moitra, 2001).

A recent study among 100 executives of software vendors indicated that knowledge transfer is a key challenge when they outsource (part of) their development activities to a software-outsourcing provider (Aditi, 2008). This is in line with findings of similar studies in the general outsourcing literature (Gupta *et al*., 2007; Oshri *et al*., 2007; Lacity *et al*., 2009). Software development is a complex process and often involves complex products. Hence, outsourcing is only possible if the receiving party understands the details and context concerning the processes and the products. Gaps in knowledge on the vendor side can compromise performance in projects, and knowledge gained during development projects may leave the customer company when the outsourcing relationship ends. But knowledge transfer in software development can be difficult as developers are known to be reluctant when it comes to documenting knowledge created in projects (Briand, 2003). Furthermore, not all knowledge can be codified and tacit knowledge is known to be 'sticky', that is embodied in a particular person, and can be difficult to transfer (Szulanski, 1996). Also geographical, time and cultural distance (Carmel & Agarwal, 2001; Fabriek *et al*., 2008), which may result in less informal communication and thus diminishing knowledge exchange, only add to pre-existing concerns about transferring knowledge back and forth to and from an external provider.

A number of studies, in both the general and the software-outsourcing literature, have addressed the topic of knowledge transfer in outsourcing (Tiwana, 2003; Cusick & Prasad, 2006; Layman *et al*., 2006; Kotlarsky *et al*., 2008). It involves mainly qualitative (case) studies that research what influences knowledge transfer and they identify successful practices or recommendations to foster knowledge transfer in outsourcing projects. Although there are some common threads in the studies, there is not an overall model or framework that combines these insights and provides practical recommendations for ISVs who engage in outsourcing software development activities.

Therefore, in this research we combine the insights from the literature on knowledge transfer in software development outsourcing with the insights of two empirical case studies of Dutch vendors who outsource part of their development activities to Eastern Europe. The result is a normative model for knowledge transfer that addresses three main phases in software-outsourcing projects and contains six best practices and a blueprint for a knowledge infrastructure to support knowledge exchange in the development phase of the project. This tool blueprint is further elaborated by means of an explorative prototype. To validate the normative model, it has been reviewed along with the prototype by six external experts, three from the software industry and three from the scientific community. Their feedback confirms the usefulness of the current model and also that the model is not limited to the specific situation at the case study company but can be applied by ISVs and outsourcing providers in general.

The remainder of the paper is organized as follows: In Section 2, the research method is briefly introduced. In Section 3, related work and previous research on the topic is explored. Section 4 summarizes the results of the two case studies. In Section 5, the aforementioned model is presented and an overview given. In Section 6, the recommendations in the model are further elaborated. Section 7 presents the prototype. Section 8 summarizes an evaluation by a panel of experts. Finally, Section 9 presents the conclusions and implications of the research.

2. Research method

For this research, we adopted the fundamental approach of Design Science Research (DSR), as proposed by Hevner *et al.* (2004) and Vaishnavi and Keuchler (2004). The philosophy behind DSR is that scientific knowledge can be generated by means of constructing an artefact (Hevner *et al.*, 2004; Vaishnavi & Keuchler, 2004). In its core, DSR is a problem-solving process (Hevner *et al.*, 2004; Peffers *et al.*, 2008). Today DSR has achieved considerable popularity among researchers because of the ability to support the development and observation of artefacts that are often the subject of investigation in several research domains. Artefacts usually represent something new and not yet existing in nature (Simon, 1996). Such artefacts can be in the form of constructs, model, methods, instantiations or better theories (Vaishnavi & Keuchler, 2004).

The general methodology of DSR, as proposed by Vaishnavi and Keuchler (2004) is depicted in Figure 1. The general methodology consists of five steps, providing the opportunity to iterate some of the steps if the process outcomes present areas for improvement. We elaborate this process in greater detail below.

The first step, *Problem Awareness*, is the realization that there is a particular problem in business, society or science. Here, we explored the problem of knowledge exchange in the global outsourcing of product software development in a case study and studied related work and previous research in the literature, presented in Section 3. Once the problem had been defined and the solution domain explored, it was possible to make a *suggestion* (an early draft) of a possible solution in the form of two artefacts – a normative model for knowledge exchange and a prototype of a supporting Product Knowledge Center. The artefacts were then further

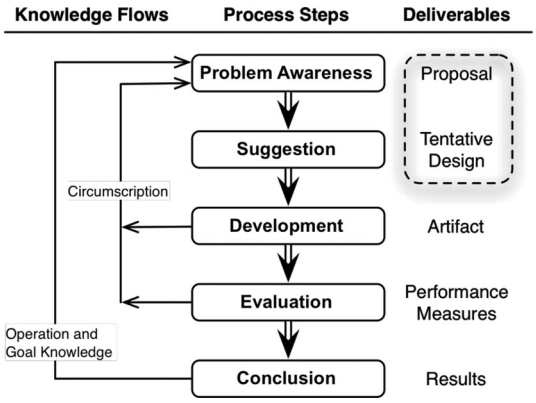

Figure 1: *Design science research cycle.*

developed and revised using input from two case studies (i.e. two outsourcing projects) presented in Section 4. Based on interviews and document study, we collected data on the knowledge exchange practices and tried to link them with described practices in the literature. After comparing the results of the case studies, we finalized the model and corresponding practices as well as the supporting tool. The revised versions of the artefacts are elaborated in Sections 5–7. *Evaluation* consisted of expert validation of the artefacts by an expert panel, presented in Section 8. Finally, *conclusions* are drawn from the design activities and empirical studies. They are explained in the conclusions and further research section.

As indicated, exploratory case study research was applied during the *Suggestion* and *Development* phase of the DSR cycle. Case studies are typically conducted to provide deeper understanding of the phenomena under study. They investigate complex real-life issues, involving humans and their interactions with ever-changing and complex technology (Yin, 1994). In this research, case studies have been applied to better understand knowledge exchange in outsourcing relations. It helped to put the common threads that we found in literature into perspective and to develop an integrated vision on how to (pragmatically) deal with knowledge exchange in the context of global software outsourcing. By conducting two case studies, it was also possible to do a cross-case analysis to identify similarities and differences.

3. Related work

3.1. Role of knowledge sharing in outsourcing success

One of the first studies to research the relationship between explicit and tacit knowledge sharing and outsourcing success was by Lee (2001). He found a positive relationship and the success of outsourcing was further influenced by the organizations' capability to create, integrate and leverage knowledge. Cramton (2001) has researched factors that influence knowledge exchange processes in dispersed teams and concludes that the dispersion itself, along with mediated communication, results in reduced visibility of collaboration and poorer performance. This is called the mutual knowledge problem, where mutual knowledge is defined as 'knowledge that communicating parties share in common and know they share'. Distributed work also both breaks down memory systems that are enacted in collocated teams, and reduces the amount of informal exchange (coffee room discussions).

3.2. Bridging knowledge gaps in outsourcing relations

An empirical study of 209 outsourced software projects by Tiwana (2003) aimed at identifying situations where the 'black box' approach in software outsourcing (transferring domain knowledge in the form of a formal requirements document) is insufficient. The conclusion is that conceptually novel projects, where the problem domain is totally new to the vendor, and projects with so-called process novelty, where the tools, technologies and/or methods are new to the customer, need appropriate adjustments in knowledge level to be successful. Totally novel projects (where both situations apply) require extensive customer–vendor collaboration across all phases of the development process. Tiwana

(2003) suggests five integration mechanisms for bridging the knowledge gaps. Firstly, development coordination (process support) tools are recommended since there is evidence that they enhance knowledge sharing across space and time. Secondly, ongoing customer–vendor interaction helps surface requirements not captured in specifications. Thirdly, it is suggested that customer and vendor join forces in architectural design efforts. Fourthly, a more mature development process leads to a higher level of knowledge integration (Balaji & Ahuja, 2005). Finally, the whole team should have as much a collaborative development experience as possible.

3.3. Knowledge management best practices in outsourcing

Cusick and Prasad (2006) propose several recommendations for success in management of global software development projects. The most notable ones with regard to knowledge exchange are: Require both structured and unstructured communication, track all issues that come up during the project, maintain channels of informal and formal communication, retain domain expertise both onshore and offshore and transition key staff to production support.

Oshri *et al*. (2008) have elaborated on this and identified factors that contribute to difficulty in transferring knowledge between remote sites in different countries. They found that the diversity of contexts exacerbates the 'stickiness' of information. Furthermore, different local routines for working, training and learning impedes shared understanding of practices and knowledge. Related to this they found that differences in skills, expertise, infrastructure, tools and methodologies hinder knowledge transfer. Finally, also the lack of prior experience of working together and changes in team membership hinder knowledge transfer.

The same authors, Kotlarsky *et al*. (2008), also developed a model that depicts four different types of coordination mechanisms that can positively impact knowledge processes in globally distributed projects. Organizational design mechanisms (1), comprising hierarchies, teams and networks, can be used to facilitate knowledge flows. Work-based mechanisms (2), such as plans, specifications and documentation, are important for making knowledge explicit. Technology-based mechanisms (3), which involve tools for capturing, processing, storing and retrieving information, can amplify knowledge management processes. Finally, social (interpersonal) mechanisms (4), involving communication and working relationships, help to establish social capital of employees.

Layman *et al*. (2006) studied the use of Extreme Programming in a global software development setting. They were interested in finding out how a globally distributed team could use a development methodology that relies very much on informal communication. They conclude after an extensive study of a customer in the USA outsourcing its development to the Czech Republic that it can be successful, and make conjectures and recommendation based on the case. Firstly, it is essential to have a well-defined customer authority that is able to make conclusive decisions about requirements and scope. Secondly, having a key member of one team (who speaks both languages) co-located with the other team can improve communication. This member then effectively works with both teams. Thirdly, prompt responses to asynchronous queries have a positive impact. Mailing list software is therefore recommended. Finally, the whole team should be provided with continuous access to process and product information. Use globally available tools to record and monitor project status on a daily basis.

3.4. Information technology support for knowledge exchange

Herbsleb (2007) suggests exploring tool support for storing, enriching and retrieving information that can be classified as 'project memory' in a wide sense, which can be useful in what he calls the communication-starved context of global software outsourcing. Exploring collaborative solutions that have grown out of open-source projects is one way to address this research area. Sengupta *et al*. (2006) focus on application knowledge migration and management in global outsourcing, and stress that the sources of this knowledge are more than the formal artefacts, and that these sources can be acquired with collaborative tools although there are limits to it.

The aforementioned studies have some common threads. First of all, they suggest some kind of integration of both teams, either by partial physical co-location or intense collaboration on some activities. Secondly, informal communication is emphasized as an important vehicle for information and knowledge sharing. And thirdly, they encourage the use of globally accessible collaborative process support tools for tracking issues, keeping both sides up to date on progress and communicating formally. These common threads have been the starting point for the knowledge exchange model that we have developed and is discussed in Section 5.

4. Case studies: a European nearshore outsourcing provider

In order to understand communication practices in the outsourcing collaboration of product software vendors and outsourcing providers, we conducted two case studies of Dutch software vendors. Each of the vendors currently subcontracts part of their development activity to Levi9 Global Sourcing, a Dutch-owned outsourcing vendor with development centres in four Eastern European countries. The unit of analysis in both case studies is a software development outsourcing project, both of which were perceived as successful from the viewpoint of customer and vendor. The customers were satisfied and were both planning on continuing their relationships with Levi9.

Data collection took place in February and March 2010 in the Netherlands and Serbia and consisted of semi-structured interviews with both customer and vendor team members, tool and document reviews and participant observation.

After a short summary of each of the cases, they are compared, their similarities and contrasts are highlighted and identified practices are pointed out. The analysis of the cases provided valuable input for the design of the model described in Sections 5 and 6.

The names of two software companies, their products and the technologies they are built on are changed for reasons of objective comparability and confidentiality.

4.1. TreasurySoft

TreasurySoft is a software vendor with a line of three products to support financial management within organizations.

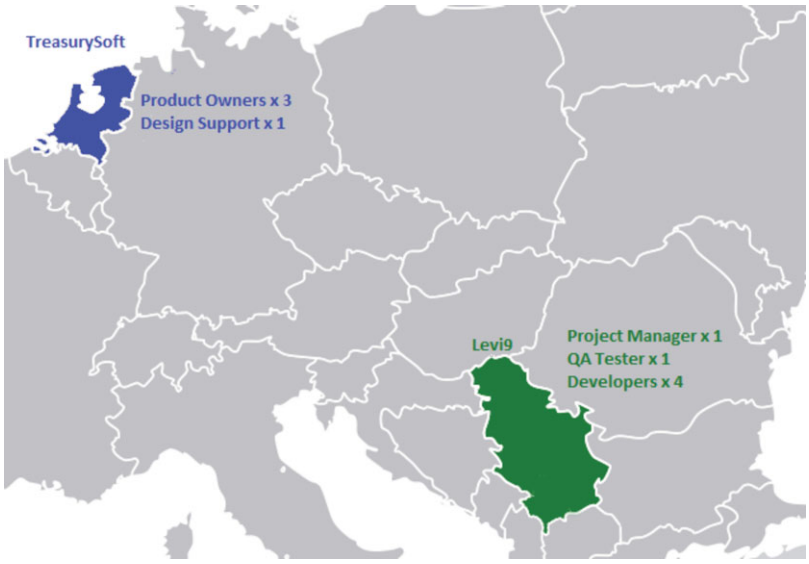

Figure 2: *Geographical distribution in TreasurySoft project.*

One of them, WebTreasury, is web-based. After two unsuccessful attempts to build the WebTreasury application, they turned to Levi9 in order to acquire affordable and available expertise in novel Web 2.0 technologies and try a new way of working.

Currently, a six-person team in Serbia is working on building the new version of WebTreasury. TreasurySoft has three employees working on product management, requirements and functional specifications, and one senior architect who oversees technical aspects, supports the team with specific issues and visits them on average every 3 months. Figure 2 illustrates the geographical distribution of the teams.

The project was kicked off by means of the visit of six TreasurySoft employees to Serbia. They were trained in Scrum and some of them had collaborative sessions with the Levi9 team on solution architecture. Additionally, the TreasurySoft members trained the Levi9 team in the existing application version and shared knowledge of the solution domain.

The team utilizes the Scrum framework to manage the project and delivers working software increments in 4-week cycles. Meetings at the start and end of each cycle are held using sophisticated videoconferencing equipment. Other communication takes place with voice over IP, instant messaging, email or using Levi9's own web-based issue tracking and task management system. Special sessions called 'grooming meetings' are held via teleconference to discuss new functional specifications in detail, and following those meetings they are updated. Tools for managing issues, documents and source code are all provided by Levi9.

Documentation requirements by the customer are not clear-cut. However, the vendor team regularly produces technical documentation according to their own standards, and tests are thoroughly documented. Both customer and vendor agree that more focus on documentation is needed before the system can be put into production.

4.2. ERPSoft

ERPSoft is a large software vendor providing organizations of all sizes with business solutions. It offers a sophisticated product line, ERPWeb, which is sold as a service and targeted at small- and medium-size organizations. Due to market pressures, ERPSoft needed immediate resources for developing a new product in the ERPWeb line aimed at local governments in the Netherlands. These resources were not available internally at ERPSoft and therefore they turned to Levi9.

Currently, a team of 10, distributed across Serbia (four team members) and Ukraine (six team members), is actively working on their second ERPWeb project for ERPSoft, who decided to start working with both development centres in order to be able to scale up the outsourcing activity more easily. The project manager is situated in Serbia, with a development team lead in each development centre. The two QA testers are located in Ukraine. ERPSoft has five employees committed partly or fully to the project – four in requirements management and functional design, and one who oversees architecture and technical design efforts. Figure 3 illustrates the geographical distribution in the ERPSoft project.

Since ERPSoft is a large vendor, they have an organized training program for new employees. They kicked off the outsourcing effort by inviting the whole outsourcing team for a 2-week training session in their headquarters where the team learned all about their product line architecture and design patterns. Also, some informal mixing of the teams took place. This was helpful both for getting the teams involved in working for ERPSoft and getting to know both their fellow team members in the other development centre and at the customer.

Despite being spread over two locations, with the product owners in the third location, the team uses Scrum to manage the development and delivers internal releases in 4-week cycles. Daily status meetings (i.e. Daily Scrum) are held via teleconference. The teams also utilize email, instant messaging and a discussion forum provided by ERPSoft, for day-to-day communication regarding functional specifications, technical design and other issues. Other components of a common tooling infrastructure, such as source code control and issue tracking, is also provided by ERPSoft, further integrating the outsourcing team into their way of working.

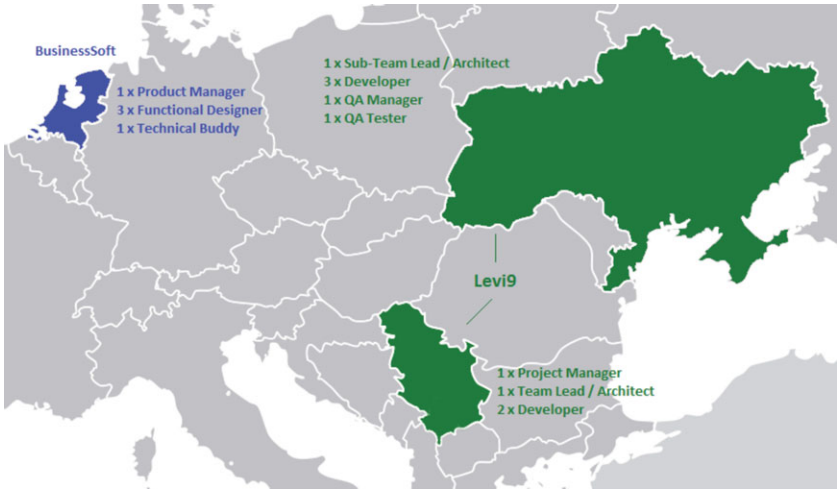

Figure 3: *Geographical distribution in ERPSoft project.*

All technical designing is done by the team members from ERPSoft. The vendor team compares this to cooking using a recipe. As a result, there is no requirement on behalf of ERPSoft that the vendor team document their work extensively, apart from code comments. However, since the QA testing is managed by the vendor, functional tests are well documented and reusable by the customer at a later point.

4.3. Comparison and identified practices

Table 1 summarizes the comparison of the key aspects of knowledge exchange and communication in the two cases.

The projects mainly differ on the level of control exercised by the customer. ERPSoft emphasizes integrating the vendor team into their organization and harmonizing technical design across products, resulting in Levi9 almost acting like a remote department of ERPSoft. The team working for TreasurySoft works like a contractor, leading architecture and design efforts, but the customer remains in charge of the content of the product on a daily.

It was also observed that only ERPSoft had a strategy for documentation and transferability of development. The strategy is simply to integrate the team as much as possible into their own organization so that knowledge is transferred through frequent interaction between the team and the organization. In the TreasurySoft project, the requirements towards documentation are somewhat implicit, and handover to production had not been planned at the time of the study.

From the cases and their comparison, by linking the observations to previous research and taking into account the subjective views of the respondents, the following practices and conventions that directly promote knowledge exchange used in the projects can be identified:

- Organizing kick-off activities in the same location for as long as needed involving all team members, both from customer and vendor. According to interviews with Levi9 team members, this is not done in all projects, but has very positive impact on collaboration and communication where it is applied.
- Having a common technical project infrastructure accessible to all team members. This is always highly recommended by Levi9 when initiating new partnerships.

Table 1: *Case comparison*

Aspect	*TreasurySoft*	*ERPSoft*
Team size and locations	Six members, co-located	10 members in two countries
Kick-off activities	Organized by vendor	Organized by customer
Tooling infrastructure	Scrum → Customer DCMS → Vendor Common and accessible to all team members, exclusively hosted by Levi9	ERPWeb, ERPSoft working procedures → Vendor Common and accessible to all team members, mostly hosted by ERPSoft
Communication infrastructure	Videoconferencing Skype VoIP and IM Email	Skype VoIP and IM Email
Functional issue resolution	Work item comments Skype IM and voice calls Grooming meetings	Email Discussion forum Skype IM and voice calls Regular visits
Technical assistance and control	Design support, occasional visits, low but adequate availability	'Technical buddy', regular visits, high availability (part of job description)
Technical documentation	No formal requirements, expected to be delivered by vendor	Not required from vendor, customer does it
Transferability of development	Difficult, due to low level of knowledge of technical platforms at customer	Very good, architecture and technical design pattern owned by customer

- Open communication between parties on a daily basis about functional and technical issues, as well as project status, using multiple channels. Levi9 has experience with many platforms for communication in distributed development and applies it in its projects.
- Committing one or more members of the technical staff of the customer to knowledge sharing. This was seen as a constructive rule by the Levi9 team members.

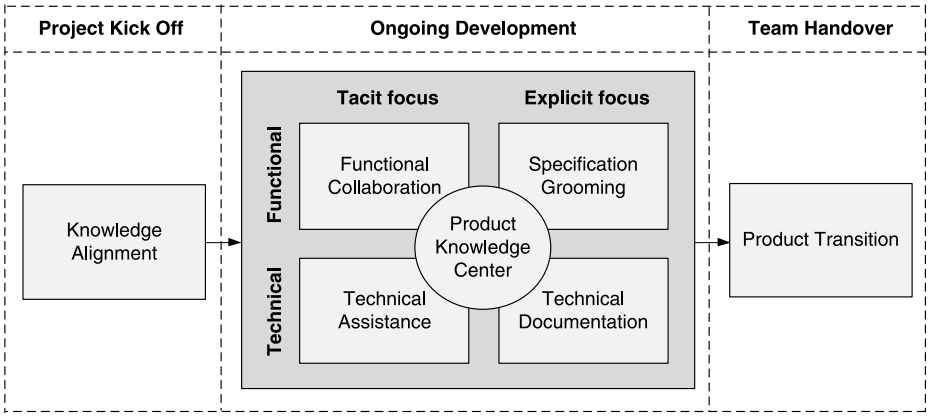

Figure 4: *Model for knowledge exchange in global software outsourcing.*

It needs to be noted that in both cases, a dominant contextual factor is that the outsourcing customers are software companies with a high level of knowledge and expertise in both software development and the business domain they are developing for. This is an enabling factor for the practices mentioned above. The teams were able to 'plug in' to the software vendors in terms of business domain and application knowledge, just like a group of new employees at the vendor, with little problems in spite of organizational boundaries and distances.

5. A normative model for knowledge exchange

Based on our research of previous research and the two case studies presented in the previous section, we have developed a normative model for knowledge exchange in global software outsourcing, which comprises six knowledge management practices and a tool blueprint. The problem addressed by those practices is weakened communication lines and knowledge exchange in global outsourcing projects when compared to those that run on a single site. The main focus of the model is knowledge about the application product. Application product knowledge is created and shared between the customer and the vendor as the development project progresses. In those cases, there is an existing application or application framework that needs to be built upon, the initial application product knowledge of the customer needs to be shared at the start of the project.

The model is based on findings from two case studies of successful global software development projects, where two Dutch ISVs outsourced new product development to Levi9 in Eastern Europe. In addition, existing research on knowledge exchange in global software outsourcing was reviewed and linked to the findings. The model is shown in Figure 4.

Each of the six practices, represented by the rectangles in Figure 4, addresses different aspects of communication and knowledge exchange in three stages of the outsourcing of product software development. Both customer and vendor possess knowledge about how to develop software, which creates opportunities for richer knowledge exchange than in the case of a software-outsourcing vendor collaborating with a non-technical customer. We propose that both customer and vendor pay attention to this fact and suggest a variety of practices with guidelines for successful application.

The core required knowledge that needs to be exchanged before and during the outsourcing of a software project is application knowledge, that is knowledge about the software product being built. We distinguish two main types of application knowledge, functional and technical. *Functional knowledge* is defined as knowledge associated with the functionality (desired or implemented) of the software product, for example interface flows and business logic and *technical knowledge* is knowledge associated with the implementation of that functionality, for example technical design patterns and production code.

We also distinguish between explicit and tacit knowledge. *Explicit knowledge* is knowledge that can be captured and codified, for example as text, images or audio. *Tacit knowledge* is the implicit dimension of knowledge rooted in the experience, skills and attitudes of the one who beholds it (Polanyi, 1966; Nonaka, 1994).

The first phase of the model involves Project Kick-off and marks the start of the customer–vendor collaboration. During this phase, we propose one practice: *Knowledge Alignment*. This practice emphasizes the need for both parties to mutually adjust, establish informal personal connections, exchange necessary knowledge to start the project and collaborate on upstream activities such as architecture.

Based on the aforementioned idea of different types and dimensions of knowledge, we propose four different practices for the Ongoing Development phase, which is also the main part of the model. *Functional Collaboration* is ongoing live communication and collaboration between customer and vendor on a day-to-day basis about desired functionality and products that are derived from functional specifications. *Specification Grooming* focuses on functional knowledge but from the explicit viewpoint, with emphasis on transferring and enhancing documented knowledge. Functional specifications, already accepted internally by the customer, are systematically reviewed and clarified in sessions with the vendor's development team, and subsequently revised. *Technical Assistance* is where a technical expert (architect or senior developer) from the customer has the formal role of assisting the vendor development team in emergent issues, for example technical design, integration with existing products or harmonization of product line architecture. *Technical Documentation* emphasizes writing down what is known about the technical implementation. What needs to be documented is defined by the audience (user) of the documentation.

Additionally a tool blueprint, the *Product Knowledge Center*, links the four practices and supports them. A prototype of the Product Knowledge Center is described in Section 7.

The third and final phase of the model involves Team Handover. At the point in time when responsibility for the application product shifts to another team than the original development team, the practice of *Product Transition* is applied. Here, the need to transfer the necessary knowledge and train the receiving team (whether a team at the outsourcing supplier or the customer) is emphasized.

6. Knowledge exchange practices

In this section, each of the six knowledge exchange practices is defined in further detail. Practical guidelines, previous research and theoretical links are also mentioned for each respective practice.

6.1. Knowledge alignment

The practice of knowledge alignment is defined as *mutual adjustment and integration of customer and vendor knowledge when kicking off the collaboration*. The purpose is to prepare the outsourcing team to work for the customer organization. At the start of a software-outsourcing project, it is very probable that the outsourcing team does not have all necessary knowledge to execute the project. Tiwana (2003) concludes from his research that non-routine software projects, where novelty on a conceptual level (business domain knowledge) and/or development process level is present, call for integration of customer and vendor knowledge.

To bridge these gaps, both vendor and customer team members should be convened in one location for kick-off activities. Team-building activities and social introductions as well as collaboration on architecture and technical project infrastructure, or person-to-person knowledge transfer about the existing software of the customer relevant to the project, lays the basis for informal communication and knowledge sharing during the project. Formal training sessions and assimilation of the new outsourcing team to the customer organization can also be part of this program.

Rottman and Lacity (2006) make similar recommendations. The customer should visit the offshore supplier to build personal connections, the offshore employees should be integrated into the development team and the training of offshore employees can be related to internal training efforts. Convening all team members in one location establishes conditions for knowledge sharing in distributed work (Rottman, 2008).

In the TreasurySoft case, the team members travelled to Levi9's premises in Novi Sad for a three day kick-off session dedicated to Scrum training, architecture and application knowledge sharing. All team members were thus successfully convened and team building took place. At ERPSoft, the outsourcing teams from Serbia and Ukraine travelled to the Netherlands for kick-off training. The team members were trained as if they were new employees of ERPSoft, which demonstrates a deliberate will to integrate the team into the company.

6.2. Functional collaboration

The practice of Functional Collaboration is *ongoing live communication and collaboration between customer and vendor on a day-to-day basis about desired functionality.* This relates closely to one of the recommendations from Tiwana (2003), that customer–vendor interaction should be ongoing. Although this activity takes place in most software projects, it can be neglected, for instance by adapting the 'black box' mentality and writing specifications that are 'thrown over the wall'. By adopting Functional Collaboration, the customer commits to working with the vendor towards full understanding of specifications, surfacing requirements that were not known up-front, and resolving issues that come up as quickly as possible. The vendor also commits to requesting knowledge from the customer. Functional Collaboration is facilitated by informal and personal ties between the different sides as well as different modes of communication.

The customer should be as accessible as possible. The unavailability of the owner of a functional design to clarify specifications, share knowledge, discuss compromises or respond to suggestions from developers can be a source of inefficiency in software development projects. Additionally, attention should be paid to weak spots in business domain knowledge. The outsourcing team may come from a different business domain and has possibly been chosen on base of technical knowledge only. It might therefore be required to transfer the relevant business domain knowledge to the vendor.

Functional Collaboration can be seen as both an organizational design mechanism (committing customer representatives to sharing functional knowledge) and social (interpersonal) mechanism that, if in place, both relies on and stimulates working relationships between the two sides (Kotlarsky *et al*., 2008).

At TreasurySoft, collaboration about functionality took place via JIRA (the common web-based portal for software developers) and Skype. Customer representatives were available most of the time, and team members have been sharing domain knowledge openly. In the ERPSoft case, the teams used a multitude of platforms to communicate about functionality, including email, teleconference and discussion forums. The outsourcing team members were very satisfied with the level of access they got to the product owners at ERPSoft.

6.3. Specification grooming

The practice of Specification Grooming is a *systematic review, clarification and revision of approved functional specifications.* Members of the vendor team review a new (possibly partial) specification and note any issues and questions that have to be resolved before the functionality can be implemented. Then, in a meeting, the customer and vendor discuss these issues and resolve them. The customer subsequently updates the functional specification so that it will still be representative for the actual functionality in the final product.

The name of the practice is borrowed from an optional practice in the Scrum methodology called Backlog Grooming (or Backlog Maintenance Meeting), which refers to a meeting with all team roles where the Product Backlog is

further explained, estimated and revised. It creates a dialogue between the team and the stakeholders (Hazrati, 2010).

Since the customer is an IT organization, some internal work has been done on building the specification, but when it is handed over to development, issues might arise. Some of these issues can be attributed to incompleteness of the specifications, others to the lack of business domain or application knowledge of the developers. Both problems need to be addressed.

To make the most of this practice, grooming sessions should be formal in the sense that all outsourcing team members who rely on the specification should be prepared and present, and all relevant personnel from the customer should be there to answer questions and address the issues. Decisions of the meetings should be documented appropriately and specifications should be updated, which should reduce probability of errors in implementation and test design, and facilitate the creation of user manuals.

Specification Grooming emphasizes both formal/structured and informal/unstructured communication, as recommended by Cusick and Prasad (2006). Functional application knowledge is made explicit and its codification is revised and kept up-to-date after a semi-structured session of clarification and, in some cases, exchange of ideas and opinions. Additionally, the grooming sessions are a platform for live exchange of domain expertise as recommended by the same authors.

In the case studies, the TreasurySoft team had regular formal grooming sessions with the customer team. Relevant team members were invited and prepared for the sessions, and the specifications were updated after meetings. In the ERPSoft case, the customer and vendor teams relied more on ad-hoc communication.

6.4. Technical assistance

Technical Assistance is defined in the current context as committing an expert from the customer side to the project that supports knowledge development at the vendor side and transfers new knowledge back to his organization. This person can for instance be an architect who gets the vendor team started with technical design on the customer's terms, or a senior developer who has experience from similar projects. The role of such a boundary-spanning knowledge broker is not new in global software outsourcing and has been described by Johri (2008).

The person responsible for Technical Assistance, here called the technical liaison, has a similar role for building the knowledge level at the vendor, as a cultural liaison has for reducing cultural distance (Carmel & Agarwal, 2001). The team member in the role effectively plays both sides, since he is a part of the customer organization but also works closely with the outsourcing team, especially if he is co-located with them some or all of the time.

Three practical guidelines are associated with this practice. First, the customer organization needs to explicitly recognize that supporting the vendor team technically is not a side project secondary to other tasks, but one of the primary functions of the employee who takes on the role. Second, partial or full co-location further increases the value of this practice. Layman *et al*. (2006) recommend co-location in order to establish a communication conduit, or linking pin, that works closely with both teams. Third, the technical liaison should make sure that new application knowledge created in the vendor team does not leave the customer company.

At TreasurySoft, a senior software architect was dedicated to the project. Working on the outsourced project was recognized one of his responsibilities. He was perceived as very busy, but had in the opinion of both parties 'just enough' time to fulfil his role. The technical liaison visited Levi9 a number of times but did not have a fixed amount of co-location or time dedicated to the project. ERPSoft committed two senior technical staff partially to supporting the developers in Novi Sad and Kiev. Supporting the team is not an unrecognized side task at ERPSoft, but an integral part of their job. Also, at least one of the 'technical buddies' usually visited Novi Sad at the end of a sprint.

6.5. Technical documentation

Technical Documentation can be defined as *making knowledge about the technical implementation of a software product explicit and accessible.* Documentation is accepted as a necessary by-product of software engineering. Documents associated with a software project act as a communication medium between team members are a system information repository used by those maintaining the software and tell users how to use and administer the application (Sommerville, 2007). Seen from the perspective of knowledge management, documentation is a systematic effort to capture explicit application knowledge.

Documentation has many potential roles in an ongoing software project and after it, but also limitations. In outsourced projects, it is important to determine the actual role of documentation and the needs that drive it. Directly related to that is the budget for documentation, since the customer has to be willing to pay for an adequate level of documentation, both user documentation (end-user and technical reference) and developer documentation. Therefore, the customer should determine the objectives, audience and extent of the project documentation. There are limitations to making knowledge about an application product explicit by describing it in structured documents, and it needs to be acknowledged that there can be more effective ways of transferring knowledge about a software product than by means of documentation.

Furthermore, explicit application knowledge does not only reside in formal, printable documents but also in source code as comment lines and in process support information systems as tickets, work items, defect reports and test reports. There are also even more informal sources of application knowledge, such as data from configuration management applications (e.g. log messages) and email messages (Sengupta *et al*., 2006)

In the model, the technical documentation practice only refers to the documentation of technical implementation. Documentation of desired and implemented functionality is covered by the practice of Specification Grooming.

Documentation requirements in the TreasurySoft project were somewhat vague. Some technical documentation was done, but team members believed more would be required when the application would go into production. The respondent at TreasurySoft was conscious about this lack of direction and provided suggestions for improvement in the context

of the project. ERPSoft has complete responsibility of technical documentation in their case, and documentation needs are internally determined by ERPSoft. This reflects the fact that ERPSoft was exclusively responsible for the technical design of the product being developed and tested at Levi9.

6.6. *Product transition*

The goal of the Product Transition practice is to *make sure that the level of application product knowledge within a new responsible team is of an acceptable level.* It is applicable when the customer organization or another outsourcing team takes over work on the product after the original outsourcing team has finished its contractual obligations. Thus, the transfer can take place before, during or after transition from development to production maintenance and support.

The activities that should be considered here are reviewing and finalizing documentation, de-briefing the outsourcing team and training new responsible personnel – any unfinished business associated with the four practices prescribed for ongoing development. A strategy for transfer should be determined – for instance whether to dismantle the project infrastructure and transfer the application knowledge offline rapidly or make all information available online and committing the original team partly to sharing their knowledge over an extended period of time. Cusick and Prasad (2006) suggest transitioning key development staff to production support. It depends on the context and human resource strategy whether key development staff could and should be moved to support or if they are assigned the responsibility to transfer their knowledge to support staff that is new to the application product.

During the data collection phase of the case studies, both projects were ongoing, which resulted in limited data on knowledge-transfer plans. However, respondents from the TreasurySoft project agreed that plans needed to be made for transition were the product to be handed back to TreasurySoft or to a support team at Levi9. At ERPSoft, there was little concern about knowledge transfer since two architects already had good constant overview of the development work.

7. The Product Knowledge Center: a tool blueprint

The Product Knowledge Center can be defined as *a hosted platform for electronic knowledge exchange.* While face-to-face or voice communications are often the most effective ways to resolve complex issues, electronic platforms for such exchange have the benefit of being reusable by everyone with an interest, including those who didn't participate in it. In distributed development, asynchronous electronic forms of communication are more common than in co-located development and can as such be leveraged as 'technical mousetraps' to capture knowledge about the specifics of complex software applications.

The Product Knowledge Center (presented here as a blueprint in the form of an exploratory prototype) is a general web-based platform for knowledge and information sharing in the context of a software-outsourcing project. It internally integrates discussion threads, instant messaging, a participatory knowledge base and a document repository, and offers bidirectional external integration with email. Each instance of the PKC is a project or outsourcing relationship (a program/perpetuating project).

The basic idea for each part is as follows:

- A *discussion forum* with the basic features as such, but discussions have a status like tickets (issues) and can only be marked as resolved with an accepted answer. The forum sends email notifications to those who have already contributed to the thread when new posts appear. Accepted answers can be published to the knowledge base and linked to documents in the document repository.
- Web-based *instant messaging* (IM), where availability of team members is shown. IM conversations can be published to the knowledge base with a title and edited to suit the needs. To facilitate implementation, the IM should be based on open standards, for example Jabber (XMPP).
- A Q&A style product *knowledge base* with the characteristics of a wiki. All registered users can contribute to the knowledge base by adding new content or modifying existing articles. Articles can be linked to documents in the document repository.
- A *document repository* with versioning and document summaries. In the overview for each document, a list of all related discussions and knowledge base articles is shown.

The Product Knowledge Center has a link to each of the practices in the model.

- During Knowledge Alignment, the Product Knowledge Center should be set up for both sides of the team and the team members should be trained in using it.
- The Product Knowledge Center is a tool for Functional Collaboration and Technical Assistance, offering a discussion forum integrated with email and instant messaging. Some of the tacit knowledge possessed by team member is thus captured and made reusable. Both communication formats can be promoted to a more formal element, a knowledge base article, or linked to a document that was updated as a result of the collaboration.
- The Knowledge Base can support all the practices in the model. The knowledge captured is typically specific to the project or application at hand, and formulated as a detailed answer to a question, which does not belong to a certain document or project artefact.
- The Product Knowledge Center has a standard document repository for Specification Grooming and Technical Documentation.
- During Product Transition, the Product Knowledge Center is the main point of entry for documented knowledge about the product.

To illustrate the tool blueprint, mock-up screens were created with examples of user interface and content of the Product Knowledge Center.

Figure 5 shows the first example screenshot. The main menu is on the left, and under the menu is the contact list for instant messaging, which consists of the project team members. In the top section, full-text search is available. The main section shows the discussion forum, called Current Issues in the main menu.67

The user can choose whether he sees all issues, only active issues or only resolved issues. In the list, new issues (issues

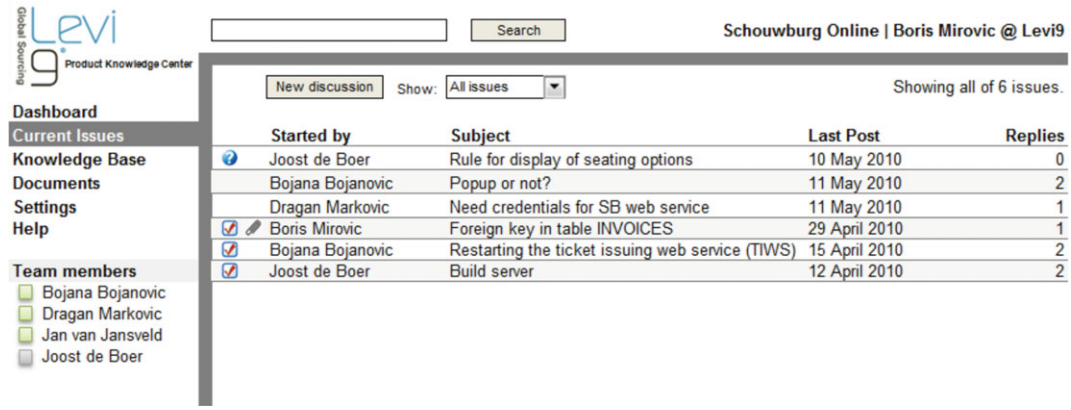

Figure 5: *Product Knowledge Center – current issues.*

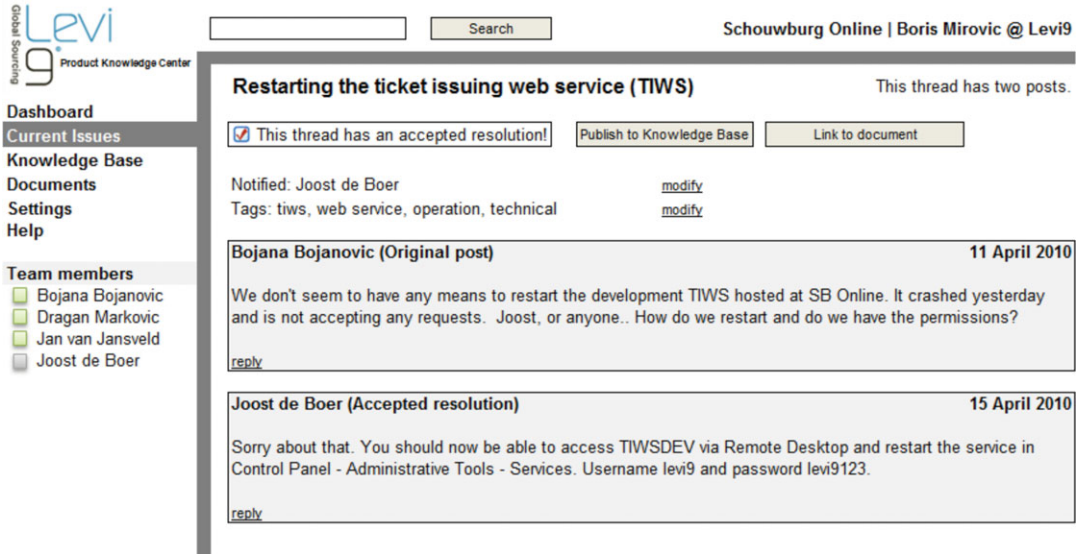

Figure 6: *Product Knowledge Center – accepted resolution.*

without reply) are highlighted, as well as closed issues with an accepted resolution. If documents are attached to a post, it is shown in the list as well. Accepted resolutions can be published as draft articles to the Knowledge Base. Figure 6 shows an example of an accepted resolution.

Forum discussions with accepted resolutions can also be linked to documents in the document repository. This creates a cross-reference to the document that was updated based on the discussion. All content (discussions, knowledge base articles and documents) can be tagged with keywords. An example of a list of knowledge base articles is shown in Figure 7.

By default, all articles are displayed (with paging if they exceed a full screen) but the user can also click a tag to narrow down the list. A thumbs-up sign beside an article title means that a sufficient number of team members have recommended the article. Any user can edit the content of any article, similar to a wiki.

In Figure 8, an example of a summary page from the document repository is shown. The page shows the version

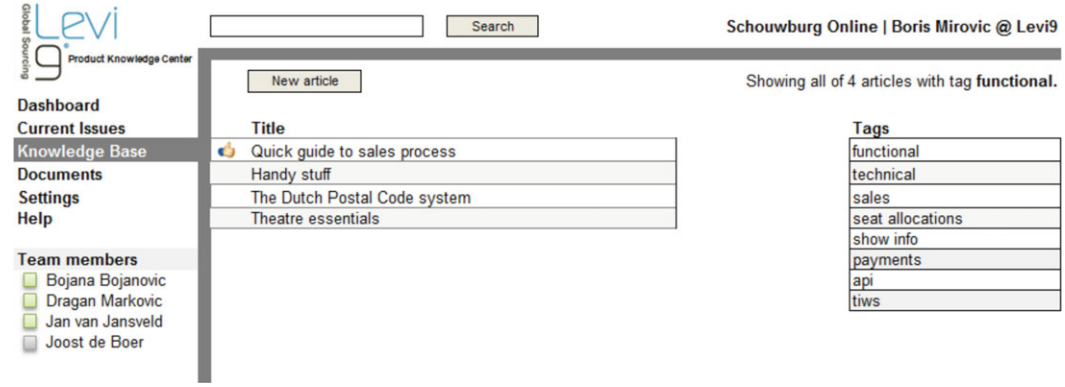

Figure 7: *Product Knowledge Center – knowledge base article.*

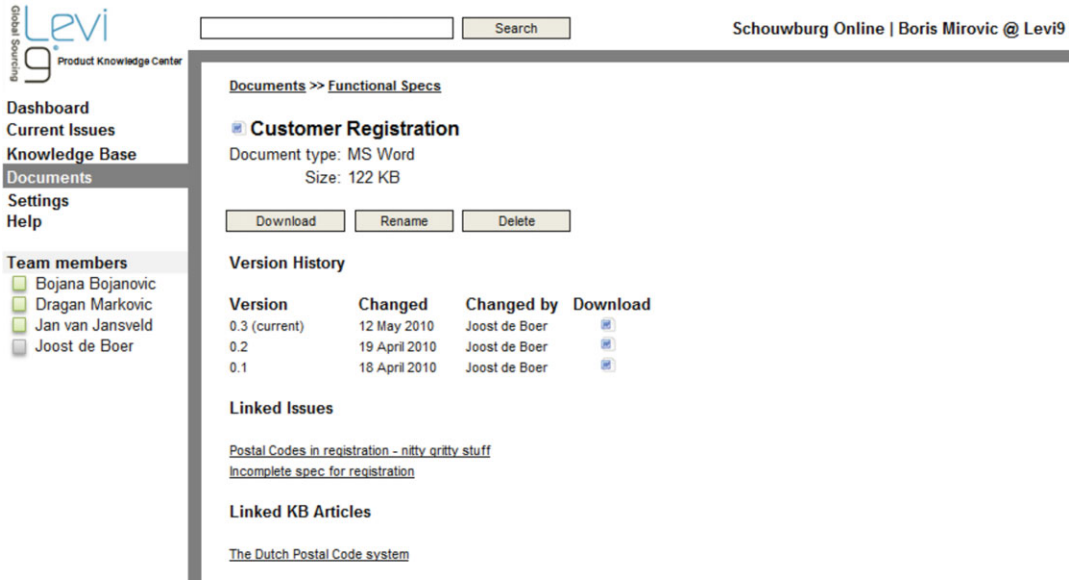

Figure 8: *Product Knowledge Center – document summary.*

history of the document, as well as linked issues (discussion threads) and knowledge base articles. In addition to the four main features mentioned above, there are some important supporting features. Users are logged on automatically on an intranet, and a dashboard is available that highlights the latest content.

8. Evaluation

To evaluate the utility of the model and revise it, content validation was applied by means of expert evaluation (Burns, 2000). Six experts were interviewed and asked to state openly their opinions about the content of the artefacts. The expert panel consisted of three experts from practice and three experts from the scientific community. Table 2 shows the credentials of the experts.

The interviews were structured but consisted of open questions, thus enabling the experts to liberally state their opinions and ideas for improvement. The following questions were asked about the draft model:

- What is your first impression?
- For each practice, judge: (a) the name, (b) the definition and intent of the practice and (c) the guidelines given.
- Do you agree with the proposed relationships and links?
- Do you believe anything is missing from the model?
- Is anything redundant or overdone in the model in your opinion?
- Do you have any further comments?

Table 2: *Expert panel*

Name	*Position*
Researcher 1	Associate Professor at Hogeschool van Amsterdam Principal Consultant and Partner at VKA
Researcher 2	Full Professor at Maastricht University Executive Manager at Ernst & Young Advisory
Researcher 3	Assistant Professor at Utrecht University and Member of the Board of Advisors at BusinessBase and Yunoo.nl
Industry Expert 1	Development Manager at TreasurySoft
Industry Expert 2	Application Design Manager at ERPSoft
Industry Expert 3	Senior Software Developer at Levi9 Global Sourcing Balkan

Positive remarks about the model were that it was good to distinguish practices by knowledge dimension and functional versus technical focus, that it looked 'rather clear', 'relatively clear' and 'clean'. One reviewer stated that the model was '[...] balanced, not too simple and not too complex'.

The model included boxes with two principles (transparency/visibility and customer commitment) that, despite a consensus among the reviewers that they were general success factors, were removed from it after two reviewers argued that the model would look better without them – one said that they were 'baggage'.

Another important result of the review is that the proposed prototype was linked to the model as a container for all the types of knowledge exchange. It replaced another practice in the draft, namely 'informal exchange', which was perceived as vague by two of the reviewers. All agreed that informal exchange was important in itself, which led to the definition of Functional Collaboration practice.

Other points of criticism were that the term 'grooming' sounded unfamiliar and that the term 'phase' might as well refer to the software development cycle.

The following guidelines were added or emphasized in the practices as a result of the review:

- Knowledge Alignment: Team-building and creating a basic level of trust.

- Technical Assistance: A key staff member to maintain personal contact with the developers.
- Product Transition: Knowledge should be of 'an acceptable level'. Also, choice of strategy was pointed out as a necessary guideline – online transfer (ongoing access to collaborative workspace and previous development team) or offline transfer (detailed artefacts submitted to receiving team, and they have to speak for themselves).

As for the question whether anything was missing from the model, two reviewers did not think so. Other reviewers brought up different suggestions with little agreement – team member roles, the process itself, formal progress reporting, governance and the project charter.

For the prototype, the following questions were asked:

- What is your first impression?
- What do you think about the proposed non-standard features of the forum?
- Do you think any more aspects should be elaborated in the prototype?
- Can you think of any more features?
- Do you think any features are unnecessary?
- Do you have any further comments?

Positive first impressions included 'very simple, which is very important, since KM is about simplicity, makes implementation a lot easier'. Two reviewers were somewhat confused about the use of tags/keywords and the use of summaries. This was improved from the draft shown to the reviewers to the version shown here.

Regarding the non-standard forum features, three thought they were supportive and had no comments. One of them stated that they were actually quite standard, except for the resolution feature. One reviewer stressed the importance of threads having an owner, and that topics could have a place despite the product/project focus of each instance of the Product Knowledge Center. Finally, one thought it should be a requirement that the forum was open and accessible to anyone, not just the project team.

Most reviewers suggested one or more new features, some of which were incorporated into the final version. Features added as a result of the review were full-text search, automatic pass-through logon, instant messaging, defining who can 'resolve' a thread and publishing the end conclusion.

One reviewer did not see new contributions in the tool and expressed scepticism about using forums in collaborative work, except with the help of a 'killer application' and a 'killer reason' to use it. He also added that email was the current 'killer application'.

9. Conclusions and further research

9.1. Contribution

Outsourcing software development activities seems attractive because of the promise of lower cost when outsourcing activities to low-wage countries. However, there are several barriers that need to be overcome to actually achieve these lower costs. One of these barriers is proper knowledge exchange between the customer and the vendor. Based on insights from the case studies and literature, we presented a normative model for Knowledge Exchange in Global Software Outsourcing that comprises six practices as well as a blueprint for a tool to support these practices. The model recognizes that knowledge needs to be exchanged at different stages of a software development project and not only once, for instance at the start of the project. We defined six practices that are based on case study insights as well as knowledge management literature, and aim to exchange both functional and technical knowledge that is either tacit or explicit.

An overview of the practices is shown in Table 3 showing the name of the practice, the phase to which it belongs, the goal of the practice and the impact if the practice is not or only partly applied. The model has been reviewed by external experts, including senior sourcing consultants, who found the proposed practices relevant and valuable for application in real-world situations.

Concerning the tool blueprint, the results of the case studies and expert interviews reflect some skepticism towards deploying specific technical solutions with before unseen features. The main recommendation is to focus on ways to connect the teams by employing good communication and co-ordination measures, and existing collaborative platforms can be deployed as a Product Knowledge Center. The tool blueprint can be used as a guide to required features of such a platform.

This study can be considered as contribution to the further understanding of factors that drive knowledge exchange in global software outsourcing. The case study results confirm what has previously been indicated by research in this domain, that knowledge exchange is closely intertwined with communication, collaboration and co-ordination. These aspects have all been addressed in previous research, and the findings of our study indicate that by addressing knowledge exchange specifically, as well as emphasizing communication, can have positive impact on project success in software outsourcing.

Summarizing, the contribution of our research is that it proposes practices for knowledge exchange in global software outsourcing and bundles them in an overall model and suggests a tool for supporting the practices. Both the model and practices are easy to understand and can find its way into the industry. The suggested tool is in an early prototype stage of development and cannot be distributed yet but can serve as a blueprint for further development.

9.2. Limitations

The limitations are best discussed using the validity threats as specified by Yin (1994): construct validity, internal validity and external validity.

Construct validity is about establishing correct operational measures for the concepts that are being studied. We applied the following measures to mitigate the threat of construct validity. First of all, multiple sources of evidence were used (interviews, data from collaboration platforms and documents) to establish concepts that are recognized by several sources. Secondly, case study reports were reviewed by the project managers at Levi9 and the model, practices and the tool were reviewed by the expert panel. The reviews confirm that the model and practices are a reliable reproduction of the interviews and observations.

Table 3: *Overview of suggested knowledge exchange practices and their impact*

Phase	*Practice*	*Goal*	*Impact*
Kick-off	Knowledge alignment	Mutual adjustment and integration of customer and vendor knowledge when kicking off the collaboration	Lack of knowledge about application, process or domain at the developer team may lead to iterations later in the development process caused by 'errors'
Ongoing development	Functional collaboration	Ongoing live communication and collaboration between customer and vendor on a day-to-day basis about desired functionality	Unavailability of people to clarify specifications, share knowledge, discuss compromises or respond to suggestions from developers results inefficiency and delays in software development projects
Ongoing development	Specification grooming	Systematic review, clarification and revision of approved functional specifications	Issues and questions that have not been resolved before actual implementation of the functionality may lead to incorrect implementation and hence iterations in developments or even defective applications when unnoticed
Ongoing development	Technical assistance	Committing an expert from the customer side to the project that supports knowledge development at the vendor side and transfers new knowledge back to his organization	Lack of assistance lengthens the learning curve of the vendor team. Re-inventing the wheel because knowledge and lessons learned are not transferred to the customer organization.
Ongoing development	Technical documentation	Making knowledge about the technical implementation of a software product explicit and accessible	Poor documentation leads to ineffective communication and errors. Furthermore, it makes maintenance of the application more difficult and inefficient.
Handover to production support	Production transition	Make sure that the level of application product knowledge within a new responsible team is of an acceptable level	Lack of handover makes maintenance and support of the application more difficult and inefficient

Internal validity is not applicable in our situation as the case studies concern exploratory research rather than explanatory research, because the focus is more on developing a new knowledge-sharing model than validating hypotheses concerning the model. However, the underlying assumption of our research is that application of the model and corresponding practices will result in more successful software outsourcing, a claim that cannot be justified by this research alone.

Finally, there is external validity that refers to the possible generalization of the results to other cases. The selected cases are both in the nearshoring of product software development to Eastern Europe. Although the provider in both cases is the same, its clients for both studied projects are different. Despite the differences between the companies, they seem to find common ground in the model and practices established in this research. Hence, we feel confident that the results are also valid for other software companies and nearshore locations, especially as the cases are not considered very rare cases. However, one might put a different emphasis on the knowledge exchange practices considering the unique contextual factors in other situations.

9.3. Further research

Further research could look into the external validity of the model, or examine more similar cases at offshore vendors located further away from the customer, for example in India or China. Since the scope of this research is limited to outsourcing of the activities of software vendors, the fit of the findings to other models for distributed software development could also be investigated. Another direction of further research concerns studying the relation between application of the knowledge exchange practices and increased software development performance. Both qualitative and quantitative research methods could be applied to study this relation.

Acknowledgements

We would like to express our thanks to Levi9 Global Sourcing in Amsterdam and Novi Sad, and to the Dutch software vendors, for providing insight into their activities. The panel of experts also provided valuable input for the study.

References

ADITI. (2008) Going offshore: a primer for ISVs. *Aditi*. Available at http://www.aditi.com/Pdf/ISV Offs horing primer.pdf (accessed 31 May 2010).

BALAJI, S. and M.K. AHUJA (2005) Critical team-level success factors of offshore outsourced projects: a knowledge integration perspective, in *38th Annual Hawaii International Conference on System Sciences (HICSS '05)*. 3–6 January 2005, Big Island, HI, USA. Los Calamitos, CA: IEEE Computer Society.

BRIAND, L.C. (2003) Software documentation: how much is enough, in *Proceedings of the 7th European Conference on Software Maintenance and Reengineering (CSMR '03)*. 26–28 March 2003, Benvenito, Italy. Los Calamitos, CA: IEEE Computer Society.

BURNS, R.B. (2000) *Introduction to Research Methods*, London: Sage Publications.
CARMEL, E. and R. AGARWAL (2001) Tactical approaches for alleviating distance in global software development, *IEEE software*, **18**, 22–29.
CRAMTON, C.D. (2001) The mutual knowledge problem and its consequences for dispersed collaboration, *Organization Science*, **12**, 346–371.
CUSICK, J. and A. PRASAD (2006) A practical management and engineering approach for offshore collaboration, *IEEE Software*, **23**, 20–29.
FABRIEK, M., M. VAN DEN BRAND, S. BRINKKEMPER, F. HARMSEN and R. HELMS (2008) Reasons for success and failure in offshore software development projects, in *Proceedings of the 16th European Conference on Information Systems (ECIS 2008)*, Golden, W., T. Acton, K. Conboy, H. van der Heijden and V.K. Tuunainen (eds), 9–11 June 2008, Galway, Ireland, 9–11.
GUPTA, A., S. SESHASAI, S. MUKHERJI and A. GANGULY (2007) Offshoring: the transition from economic drivers toward strategic global partnership and 24-hour knowledge factory, *Journal of Electronic Commerce in Organizations*, **5**, 123.
HAZRATI, V. (2010) Backlog grooming: who, when and how, *InfoQ*. Available at http://www.infoq.com/news/2010/05/backlog-grooming (accessed 13 May 2010).
HERBSLEB, J.D. (2007) Global software engineering: the future of socio-technical coordination, in *29th International Conference on Software Engineering (ICSE 2007)*, Briand, L. and A. Wolf (eds), 20–26 May 2007, Minneapolis, MN, USA. Los Calamitos, CA: IEEE Computer Society, 188–198.
HERBSLEB, J.D. and D. MOITRA (2001) Global software development, *IEEE software*, **18**, 16–20.
HEVNER, A.R., S.T. MARCH, J. PARK and S. RAM (2004) Design science in information systems research, *Management information systems quarterly*, **28**, 75–106.
JOHRI, A. (2008) Boundary spanning knowledge broker: an emerging role in global engineering firms, in *38th Annual Frontiers in Education Conference (FIE 2008)*. 22–25 October 2008, Saratoga Springs, NY, USA.
KOTLARSKY, J., I. OSHRI and P.V. FENEMA (2008) *Knowledge Processes in Globally Distributed Contexts*, Basingstoke: Palgrave Macmillan.
LACITY, M.C., S.A. KHAN and L.P. WILLCOCKS (2009) A review of the IT outsourcing literature: insights for practice, *The Journal of Strategic Information Systems*, **18**, 130–146.
LAYMAN, L., L. WILLIAMS, D. DAMIAN and H. BURES (2006) Essential communication practices for Extreme Programming in a global software development team, *Information and software technology*, **48**, 781–794.
LEE, J.N. (2001) The impact of knowledge sharing, organizational capability and partnership quality on IS outsourcing success, *Information & Management*, **38**, 323–335.
NONAKA, I. (1994) A dynamic theory of organizational knowledge creation, *Organization science*, **5**, 14–37.
OSHRI, I., J. KOTLARSKY and L. WILLCOCKS (2008) Missing links: building critical social ties for global collaborative teamwork, *Communications of the ACM*, **51**, 76–81.
OSHRI, I., J. KOTLARSKY and L.P. WILLCOCKS (2007) Managing dispersed expertise in IT offshore outsourcing, lessons from Tata consultancy services, *MIS Quarterly Executive* **6**, 53–65.
PEFFERS, K., T. TUUNANEN, M. ROTHENBERGER and S. CHATTERJEE (2008) A design science research methodology for information systems research, *Journal of Management Information Systems*, **24**, 45–77.
POLANYI, M. (1966) *The Tacit Dimension*, London: Routledge & Kegan Paul.
RAO, M.T. (2004) Key issues for global IT sourcing: country and individual factors, *Information Systems Management*, **21**, 16–21.
ROTTMAN, J.W. (2008) Successful knowledge transfer within offshore supplier networks: a case study exploring social capital in strategic alliances, *Journal of Information Technology*, **23**, 31–43.
ROTTMAN, J.W. and M.C. LACITY (2006) Proven practices for effectively offshoring IT work, *Sloan management review*, **47**, 56–63.
SENGUPTA, B., S. CHANDRA and V. SINHA (2006) A research agenda for distributed software development, in *28th International Conference on Software Engineering*. 20–28 May 2006, Shanghai, China. Atlanta, GA: Association for Information Systems.
SIMON, H.A. (1996) *The Sciences of the Artificial*, Cambridge, MA: The MIT Press.
SOMMERVILLE, I. (2007) *Software Engineering*, Harlow, England: Pearson Education.
SZULANSKI, G. (1996) Exploring internal stickiness: impediments to the transfer of best practice within the firm, *Strategic Management Journal*, **17**(Winter Special Issue), 27–43.
THIBODEAU, P. (2008) U.S. IT jobs take a hit: one in four going offshore, *CIO.com*. Available at http://www.cio.com/article/470089/U.S._IT_Jobs_Take_a_Hit_One_In_Four_Going_Offshore (accessed 31 May 2010).
TIWANA, A. (2003) Knowledge partitioning in outsourced software development: a field study, in *24th International Conference on Information Systems (ICIS 2003)*. 14–17 December 2003, Seattle, WA, USA. Atlanta, GA: Association for Information Systems, 259–270.
VAISHNAVI, V. and W. KEUCHLER (2004) Design research in information systems, *Association for Information Systems*. Available at http://home.aisnet.org/displa ycommon.cfm?an = 1&subar ticlenbr = 279 (accessed 11 December 2009).
YIN, R.K. (1994) *Case Study Research: Design and Methods*, London: Sage Publications.

The authors

Baldur Kristjánsson

Baldur Kristjánsson currently works as an IT consultant at Skyrr in Reykjavík, Iceland, and is a recent MSc graduate from the Business Informatics programme of the Department of Information and Computing Science at Utrecht University, The Netherlands. He received his BSc in Computer Science in 2003 with a Business Minor from Reykjavík University, and has since worked in diverse IT functions such as software engineering, business analysis and project management at Eimskip, the Icelandic Medicines Agency and Kaupthing Bank. His main research and professional interests are software product management and software process improvement.

Remko Helms

Remko Helms is an assistant professor at the Department of Information and Computing Science at Utrecht University, The Netherlands. He received his MSc and PhD from Eindhoven University of Technology, The Netherlands. Currently, his research focuses on knowledge management and social media applications in the manufacturing and software industry, with a particular interest in knowledge sharing and transfer practices and structural analysis of (online) communities using social network analysis. He serves on several program committees, for example ECKM and PAKM, and acts as a reviewer for several major IS conferences, for example ECIS and ICIS, and IS journals, for example International Journal on Information Management and Information Systems Journal. End of 2009, he was a visiting researcher to Melbourne University, Victoria, Australia.

Sjaak Brinkkemper

Sjaak Brinkkemper is full professor of organization and information at the Institute of Department and Computing Sciences of the Utrecht University, the Netherlands. He leads a group of about 30 researchers specialized in product

software entrepreneurship. The main research themes of the group are methodology of product software development, implementation and adoption and business-economic aspects of the product software industry. He holds an MSc and a PhD in Mathematics and Computer Science from the University of Nijmegen. He has published eight books and about 150 papers on his research interests: software product development, information systems methodology, meta-modelling and method engineering. He serves on the program committees of numerous conferences and workshops (CAiSE, ICSOC, ECIS, WITS, OOIS, REFSQ, WICSA, ICSOB), and serves on the steering committee of CAiSE, ECIS and ICSOB. He is a member of the Editorial Board of the MIS Quarterly, European Journal of Information Systems and the Journal of Database Management.