# 23 MIND THE GAP! Understanding Knowledge in Global Software Teams

Aini Aman
*School of Accounting, Faculty of Economics and Business*
*Universiti Kebangsaan Malaysia*
*Bangi Selangor, Malaysia*

Brian Nicholson
*Manchester Business School*
*Manchester, United Kingdom*

**Abstract** *This paper presents a conceptual framework and preliminary empirical analysis of knowledge gaps between global software team members in the UK and India. Drawing on episodes from rich case study evidence of a UK software firm based in the UK and software development sites in India, the conceptual framework is used to explore the data and to understand the knowledge gaps encountered. These are in relation to the level of knowledge, educational background, and experience of members. This study unpacks the notion of knowledge of software development into domain, technical, and application knowledge and considers the implications of prior knowledge, experience, and education background. It is anticipated from the preliminary findings that a practical and theoretical contribution will improve our understanding of the complexities of knowledge in such global arrangements.*

**Keywords** Knowledge management, global software teams

*Please use the following format when citing this chapter:*

Aman, A., and Nicholson, B., 2008, in IFIP International Federation for Information Processing, Volume 267, Information Technology in the Service Economy: Challenges and Possibilities for the 21st Century, eds. Barrett, M., Davidson, E., Middleton, C., and DeGross, J. (Boston: Springer), pp. 321-330.

# 1 INTRODUCTION

This paper aims to present a conceptual framework and preliminary analysis of the causes and implications of knowledge gaps in offshore software development work, when teams are separated by time, distance, and culture (Carmel and Tija 2005; Sahay et al. 2003). There has been limited prior research focusing on knowledge in offshore software development. Exceptions include Sarker and Sahay (2004) and Nicholson and Sahay (2004) who have improved our understanding of knowledge transfer issues and embeddedness of knowledge. However, the issue of what constitutes gaps in knowledge when developers are distributed globally and the implications of the gaps remains under-researched.

The theoretical framework draws on three linked concepts of tacit–explicit knowledge, knowledge types, and level of knowledge. Tacit knowledge is unarticulated and can be acquired informally or through practical experience, while explicit knowledge is the codifiable knowledge that can be acquired through formal study (Lam 1997; Polanyi 1966). Understanding explicit knowledge requires the understanding of tacit knowledge within the context of social interaction (Thompson and Walsham 2004). Second, a literature search revealed several authors who have identified similar category types of software development knowledge including domain, application and technical knowledge (Curtis et al. 1988; Waterson et al. 1997). Third, each member in a software development team possesses different levels of prior knowledge derived from their unique education background and set of experiences (Lam 1997; Waterson et al. 1997). The differences in the types and levels of knowledge owned by software development team members are referred to as knowledge gaps. Thus, the research question driving this inquiry is concerned with the causes and implications of knowledge gaps in offshore software development.

The empirical basis for our analysis is a longitudinal case study of a UK-based software firm with a development center in India where projects are split between teams in India and the UK. India is widely regarded as the world's leading venue for offshore software.[1]

The rest of this paper is organized as follows. In the next section, we present our theoretical framework, followed the by research methodology and a case description. Subsequently we present the analysis, and finally, we conclude this paper with anticipated theoretical and practical contributions.

# 2 THEORETICAL FRAMEWORK

This section presents a theoretical framework which was not developed *a priori* but according to the emerging themes found in the data analysis and intensive literature reviews and was refined during data analysis as events emerged. This inductive approach to theorization has been adopted by Barrett and Walsham (1999), Walsham and Sahay (1999), Nicholson and Sahay (2001), and Barrett et al. ( 2005).

[1]A. T. Kearney produces a location-attractiveness index, available at www.atkearney.com.

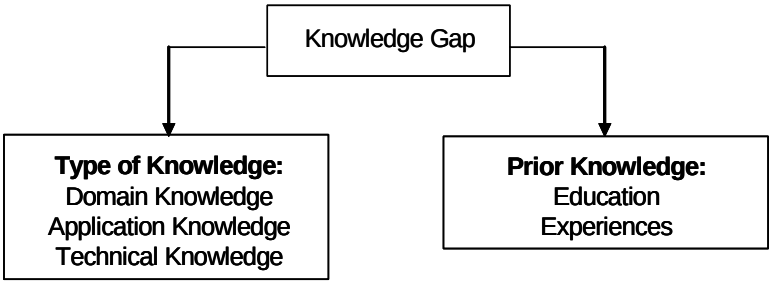


**Figure 1. Conceptual Framework**

In order to understand the process of managing knowledge gaps in offshore software development, we extend Curtis et al. (1988) on the types of knowledge using the ideas of prior knowledge (Lam 1997) and focusing on the experiences and educational background. We also take into consideration that the process of integrating knowledge situated in a range of sites is difficult because of the differences in the norms of participation and the physical context of work that varies across sites (Sole and Edmondson 2002). Our basic theoretical framework is shown in Figure 1.

## 2.1 Type of Knowledge

Domain knowledge is "business knowledge" that represents the formal and informal rules and practices of the particular company, which is needed in requirement analysis and design (Curtis et al. 1988, Sahay et al. 2003). Domain knowledge presents challenges when team members are globally distributed. For example, Nicholson (1999) provides a case study of global software development for UK housing benefits. India has no equivalent social security system and thus Indian developers had no immediate intuitive conception of the inputs, outputs, processes, and functions of such a system. By contrast, domain knowledge about the UK social security system was already held by UK-based developers, all of whom had been born and brought up in the UK. Some had even directly engaged with the UK housing benefits system as a user when making housing benefit claims.

Technical knowledge is concerned mainly with coding (Curtis et al. 1988) but may also include use of tools such as compilers, interpreters, and debuggers for generating code or high-level programming languages such as C, C++, Pascal, or Java. A developer should be able to produce and test the source code for software based on his prior knowledge of the relevant requirements, architecture, and design documentation (Curtis et al. 1988; Quintas 1993). For example, a developer needs to have this knowledge in order to understand the design document and translate it into an implementation language or machine-readable form.

Application knowledge is the ability to perform "soft" analysis such as requirements elicitation, relationship management, resource allocation, and tracking in the software development process (Curtis et al. 1988). Application knowledge provides a "mapping"

between domain knowledge and technical knowledge that is relevant for the particular software being developed (Guarino 1997, p. 11). When performing requirements analysis, the developer has to use domain and technical knowledge in order to understand the requirement and translate it into a technical design document that will have meaning to other developers (Walz et al. 1993).

### 2.2 Prior Knowledge

Educational background inevitably differs between software team members depending on how they gain their technical knowledge. They acquire their technical knowledge either through formal education or training (Lam 1997) from various institutions. An illustration of the implications of this is provided by Nicholson and Sahay (2004) in the case of a UK-based software house with a subsidiary in India. The Indian subsidiary recruited graduates from some of the second- and third-tier colleges in India which had a stronger emphasis on technical knowledge than did the first-tier colleges (Nicholson and Sahay 2004). In addition, some of the skills of these lower-tier graduates were found to be outdated, with a majority of them having had their programming course on COBOL, with little or no experience with newer languages like Java and Visual Basic. This technical emphasis clashed with the UK firm's desire for application knowledge involving creative, out-of-the-box thinking, which tend to be taught in higher-tier institutes of technology.

Domain and application knowledge can be acquired through experience (Sahay et al. 2003) that are organized and embedded in organizational routines and procedures (Lam 1997). Each team member in offshore software development work brings disparate experiences into the project that contribute to the differences in the levels of experiential knowledge among team members. The case of UK social security benefits described earlier illustrates how the onshore team in the UK might face difficulties in communicating their knowledge to an offshore team in India, which has no direct, first-hand experience of making a social security claim. Making such tacit knowledge explicit in any requirements specification to be passed to Indian developers represents a major challenge.

## 3 RESEARCH METHODOLOGY

We adopt an intensive case study approach (Walsham 1995; Yin 1994) carried out in a UK-based company named Alp (a pseudonym) engaged in offshore software development in India. Alp was established in 1997 with 40 staff in the UK and India. During 1999, Alp entered into a partnership with a local company in Mumbai for offshore software development in order to take advantage of low labor costs. The partnership with the small local company called ABS in India lasted for almost two years. Alp (UK) set up a wholly owned offshore software development center in India in July 2002. The type of software development work sent to India depends on the type and nature of the projects. Typically, the onshore team in the UK will perform the requirements elicitation and some high-level technical architecture. The offshore team in India will undertake the design and the rest of the development work such as coding and testing. Communica-

tions and discussions between onshore and offshore teams are informal through telephone, e-chat, and e-mail. The focus of the analysis in Alp is on the ATI Co (a pseudonym) project phases one and two. ATI Co currently has 180 staff and decided to set up a data warehouse on construction information and web-based delivery to subscribers. Phase one involved the pay-per-view process as a trial with one portal, while phase two was expanding that for any portal with a subscription engine. Phase one had a four-person team full-time for five months, whereas phase two was extended over a further five months.

The research began with a historical reconstruction starting from Alp's inception of operations offshore and following the subsequent operations during the period of the study between 2001 and 2005. Interview information collected at periodic intervals in India and the UK was cross-checked where possible against other sources of evidence such as documentation and company reports. Data collection used a variety of methods including unstructured and semi-structured interviews, documentation reviews, and observations taken in the UK and during field trips to India. Interviewing, repeat interviewing, and triangulation of sources over time using various techniques of data collection provided multiple perspectives and information on emerging concepts and allowed for cross-checking (Eisenhardt 1989; Pettigrew 1990). Documents such as functional specifications, project plans, project reports, test cases, e-chat between onshore and offshore teams, and other documents related to software projects and work procedures were reviewed. In addition, observations of team interactions and meetings during projects were undertaken during the fieldwork in the UK and India.

There have been a total of 93 hours of interviews with 30 individuals including directors, operation directors, project managers, project leaders, senior developers, developers, quality analyst, administrator, marketing and technical support (Table 1). Interviews were handled in different ways, including face-to-face copresent, conference call, telephone and e-chat. Interviews lasted for at least 1 and a maximum of 2.5 hours. The interviews were transcribed and subsequently summarized. The data, once collected, was analyzed by identifying themes related to knowledge gaps. Transcripts were coded into the categories and a theoretical summary produced. The theoretical framework used was not developed *a priori* but developed according to the emerging themes found in the data analysis and intensive literature reviews. Later, an intermediate report was presented to the operations manager in one of the case companies for further explanation and verification of our interpretations.

**Table 1. Details of Interviews**

| Job Title | UK | India |
|---|---|---|
| Director, Operations Director | 30 hours | 4 hours |
| Project Manager, Consultant | 20 hours | 2 hours |
| Project Leaders | – | 5 hours |
| Senior Developer, Developer, QA Analyst | – | 13 hours |
| Administration, Marketing, Technical Support | 2 hours | 6 hours |
| Customer | 4 hours | – |
| Total hours | 56 hours | 37hrs |

# 4 ANALYSIS AND DISCUSSION

This section presents specific episodes during ATI projects to illustrate the causes and implications of knowledge gap according to the types of knowledge and prior knowledge.

## 4.1 Type of Knowledge

***Episode 1: Domain Knowledge***. In ATI phase one, Robert, a project manager in Alp, gathered client requirements and sent a document consisting of a series of flowcharts and text-based system requirements to Samir, a senior developer in India. Later, Robert explained the requirements to Samir through e-chat but Samir had difficulty understanding them. As Samir explains,

> *I did not understand what Robert was trying to say and Robert did not understand what I was trying to say. It was very difficult to get the same understanding in virtual space.*

The misunderstanding was related to the gaps in domain knowledge between Robert and Samir. Robert had a better understanding of the requirement than Samir about the client and their business environment as he lives in the UK and had worked closely with the client previously, while Samir had never met the client, never been to the UK, and never worked with the client. Samir could not understand the users of the application, why they needed the information, what types of information needed to be stored, or how the user would access the information. Again, Samir explains,

> *In one incident, the client wants something to be displayed on the screen. When I looked into it, I had difficulty to understand….the problem is I did not understand the client's business completely and how it runs.*

Without appropriate domain knowledge, Samir had no intuition about the requirement (Winograd 1995) and had to make his own assumptions based on his knowledge about the business environment in India, forming his framework of meaning to make sense of the requirement. However, Samir had no experience of a similar company to ATI in India and had difficulty in imagining the functions, inputs, outputs, and interface of the completed application in use. Domain knowledge is required to develop "a picture in their mind of the context" (Cramton 2001, p. 359). The implication is that he was not able to fill in the gaps with his experiences of similar systems in India. According to Samir,

> *We did not have enough knowledge about the client's business. We had to assume a lot of things…in most of the cases, the client talks in terms of business, which is quite different from the places where we work in India; the working culture, the processes, and the standards. We couldn't imagine.*

***Episode 2: Application and Technical Knowledge***. A further episode was encountered in ATI phase two. Jeevan, an onshore project manager, has to communicate to Pratap, a project leader in India who often claimed that he did not have sufficient information to perform the tasks. He clarifies,

> *The client tells the onshore team about the requirements, and the onshore team forwards the requirements to us. We have to assume. We didn't have enough information about the system. Questions were not adequately answered, and the information wasn't going down to the developer level.*

The cause of such a claim is the gap in application and technical knowledge between Pratap and Jeevan. According to Pratap, Jeevan did not have appropriate technical knowledge and because of this, despite his application knowledge of the relevant techniques, he was unable to envisage the program code execution and thus explain the requirement in the way that the Indian team could understand:

> *He should understand what the client wants and be able to explain with data, to foresee the possible design or coding and to communicate it to the offshore teams.*

Application knowledge requires imagination of the requirement in the technical knowledge such as the functions, processes, and interface of the system from the requirements. Winograd (1995, p. 69) explains that the programmer needs to be able to "visualize what the program will be like and what can be done with it....[because the] abstract representation, such as written descriptions, flow charts and object class hierarchies cannot provide a grounded understanding." Jeevan's inability to explain with data and foresee the possible design or coding when communicating with Pratap, may result in inaccurate assumptions (Cramton 2001). The implication is that Pratap had to imagine the functions, processes, and interface of the system from the requirements and his imagination is conditioned or filtered by his knowledge, which is derived from the Indian context.

## 4.2 Prior Knowledge

***Episode 1: Education Background***. The differences in educational background between Robert and Samir the contribute to knowledge gap. Robert has a degree in Software Engineering from the UK, while Samir only has a technical certificate from a local technical college in India. Samir had to write using nontechnical terms when communicating with Robert. As a director in India explains,

> *He is not going to understand code or technical parts. They have to write in the language that he understands. It's all about giving information in a form that can make sense to the other person.*

***Episode 2: Experiences***. Experiences between onshore and offshore teams also differ. In ATI phase two, the offshore teams had to produce a design based on a given

requirement by the onshore team. Tom, a designer in the UK, tried to explain to Pratap using a design document sent through e-mail. However, he could not include all of the details in the document as some of the knowledge could not be made explicit. Tom explains,

> *We try to explain as many assumptions as possible formally. Every document that we write has assumptions in it but the assumptions do not tell you so much about the business. There will always be implicit assumptions in the design.*

Because of the implicit assumptions, Pratap was always trying to figure out what was happening in the UK without having much experience and knowledge to do that effectively. For example, Pratap suggested a colorful interface design but Tom knew that the client would not accept this as he knew that the client corporate color was black and red. Such information was not formally articulated in the requirements or subsequent documents but constituted part of the tapestry of background domain knowledge or experiences about the client held tacitly by Tom and other UK-based developers. Tom explains,

> *They often suggest different alternatives. Every time, we will go back to the initial design that we gave them. That is because we have experience in UK cultures and business environments or because we probably did not give all the information that we probably could have done. It's those things that are almost intangible; things we know just by living in this country.*

## 5 CONCLUSION

The research question of this paper is concerned with taking some first steps in improving our understanding of the causes and implications of knowledge gaps in offshore software development. The main anticipated theoretical contribution is in the form of the conceptual framework. We unpack the notion of software development knowledge including domain, application, and technical knowledge (Curtis et al. 1988) and the implications of prior knowledge (Lam 1997). Preliminary findings indicate the importance of the concept of imagination (Winograd 1995), which was demonstrated when Indian software developers used their imagination to understand the domain knowledge and make assumptions on the required design that were sometimes inaccurate. Imagination is required to integrate the system as a whole or to connect diverse things and is linked to prior knowledge.

The practical contribution of the research is in the conceptual framework, which enables analysis of potential for knowledge gaps between team members, and the cognitive "filters" on imagination and interpretation, which domain knowledge presents in particular. Software development teams may use the concepts to diagnose failures and better understand the potential breakdowns. They may benefit from understanding the kinds of knowledge that are necessary to communicate in the projects on which they are working. Formal training in the knowledge types and potential breakdowns would be of benefit and this paper presents a starting point in the form of case studies for discussion.

## References


Barrett, M., Cooper, D. J., and Jamal, K. 2005. "Globalization and the Coordinating of Work in Multinational Audits," *Accounting, Organizations and Society* (30:1), pp. 1-24.

Barrett, M., and Walsham, G. 1999. "Electronic Trading and Work Transformation in the London Insurance Market," *Information Systems Research* (10:1), pp. 1-22.

Carmel, E., and Tija, P. 2005. *Offshoring Information Technology: Sourcing and Outsourcing to a Global Workforce*, Cambridge, UK: Cambridge University Press.

Cramton, C. D. 2001. "The Mutual Knowledge Problem and its Consequences for Dispersed Collaboration," *Organization Science* (12:3), pp. 346-371.

Curtis, B., Krasner, H., and Iscoe, N. 1988. "A Field Study of the Software Design Process for Large Systems," *Communications of the ACM* (31:11), pp. 1268-1287.

Eisenhardt, K. M. 1989. "Building Theories from Case Study Research," *Academy of Management Review* (14:4), pp. 532-550.

Guarino, N. 1997. "Understanding, Building, and Using Ontologies," LADSEB-CNR, National Research Council., Padova, Italy (http://ksi.cpsc.ucalgary.ca/KAW/KAW96/guarino/guarino.html).

Lam, A. 1997. "Embedded Firms, Embedded Knowledge: Problems of Collaboration and Knowledge Transfer in Global Cooperative Ventures," *Organization Studies* (18:6), pp. 973-996.

Nicholson, B. 1999. *The Process of Software Development Across Time and Space: The Case of Outsourcing to India*, unpublished Ph.D. thesis, Salford University.

Nicholson, B., and Sahay, S. 2001. "Some Political and Cultural Issues in the Globalisation of Software Development: Case Experience from Britain and India," *Information and Organization* (11:1), pp. 25-43.

Nicholson, B., and Sahay, S. 2004. "Embedded Knowledge and Offshore Software Development," *Information and Organization* (14:4), pp. 329-365.

Pettigrew, A. 1990. "Longitudinal Field Research on Change: Theory and Practice," *Organization Science* (1:3), pp. 267-292.

Polanyi, M. 1966. *The Tacit Dimension*, London: Routledge.

Quintas, P. 1993. "Introduction—Living the Lifecycle: Social Processes in Software and Systems Development," Chapter 1 in *Social Dimensions of Systems Engineering: People, Processes, Policies and Software Development*, P. Quintas (ed.), London: Ellis Horwood Limited, pp. 1-7.

Sahay, S., Nicholson, B., and Krishna, S. 2003. *Global IT Outsourcing*, Cambridge, UK: Cambridge University Press.

Sarker, S., and Sahay, S. 2004. "Implications for Space and Time for Distributed Work: An Interpretive Study of US-Norwegian Systems Development Teams," *European Journal of Information Systems* (13), pp. 3-20.

Sole, D., and Edmondson, A. 2002. "Situated Knowledge and Learning in Dispersed Teams," *British Journal of Management* (13), pp. 17-34.

Thompson, M. P. A., and Walsham, G. 2004. "Placing Knowledge Management in Context," *Journal of Management Studies* (41:5), pp. 725-747.

Walsham, G.,and Sahay, S. 1999. "GIS for District-Level Administration in India: Problems and Opportunities," *MIS Quarterly* (23:1), pp. 39-65.

Walsham, G. 1995. "Interpretative Case Studies in IS Research: Nature and Method," *European Journal of Information Systems* (4), pp. 74-81.

Walz, D., Elam, J., and Curtis, B. 1993. "Inside a Software Design Team: Knowledge Acquisition, Sharing and Integration," *Communications of the ACM* (36:10), pp. 62-77.

Waterson, P. E., Clegg, C. W., and Axtell, C. M. 1997. "The Dynamics of Work Organization, Knowledge and Technology During Software Development," *International Journal Human-Computer Studies* (46), pp. 79-101.

Winograd, T. 1995. "Heidegger and the Design of Computer Systems,." in *Technology and the Politics of Knowledge*, A. Feenberg and A. Hannay (eds), Bloomington, IN: Indiana University Press, pp. 108-127.

Yin, R. K. 1994. *Case Study Research: Design and Methods* (2nd ed.), Thousand Oaks, CA: Sage Publications.

## About the Authors

**Aini Aman** is a senior lecturer at Universiti Kebangsaan Malaysia (UKM), Malaysia. Since 1993, she has been involved in teaching, research, and consultancy projects in the broad area of auditing, accounting, and business. Her Ph.D. involved an in-depth study of the offshore software development process in the UK, India, Bangladesh, and Malaysia. Her post doctoral study involved work with Brian Nicholson on the project "Risk and Control of Offshore Outsourcing of Accounting Services" commissioned by the Institute of Chartered Accountants in England and Wales. Aini's current research is on accounting outsourcing in Malaysia and her current consultancy work is an evaluation of e-government orojects in Malaysia. Aini can be contacted at aini@ukm.my.

**Brian Nicholson** is a senior lecturer at Manchester Business School. Since 1995, he has been involved in teaching, research, and consultancy projects in the broad area of global outsourcing of software and other business processes. This has involved work in India, China, Costa Rica, Iran, Egypt, Malaysia, and Bangladesh. Brian's research at the firm level has resulted in several influential publications in international journals and a book, *Global IT Outsourcing* (Cambridge University Press, 2003). Policy-level consultancy studies have been undertaken for the governments of Costa Rica and Iran to stimulate software exports. In the case of Costa Rica, this resulted in production of a national level strategy. Recent work has been commissioned by the Institute of Chartered Accountants in England and Wales, "Risk and Control of Offshore Outsourcing of Accounting Services." Brian can be contacted at brian.nicholson@mbs.ac.uk.