8-2010

Knowledge Transfer in Distributed Software Support with a Traceability Framework

Kannan Mohan
Department of Statistics and Computer Information Systems Zicklin School of Business Baruch College, kannan_mohan@baruch.cuny.edu

Balasubramaniam Ramesh
Department of Computer Information Systems J. Mack Robinson College of Business Georgia State University, bramesh@gsu.edu

Lan Cao
Department of Information Technology College of Business and Public Administration Old Dominion University, lcao@odu.edu

Knowledge Transfer in Distributed Software Support with a Traceability Framework

Kannan Mohan
Department of Statistics and Computer Information Systems
Zicklin School of Business
Baruch College
Kannan_mohan@baruch.cuny.edu

Balasubramaniam Ramesh
Department of Computer Information Systems
J. Mack Robinson College of Business
Georgia State University
bramesh@gsu.edu

Lan Cao
Department of Information Technology
College of Business and Public Administration
Old Dominion University
lcao@odu.edu

ABSTRACT

Distributed software support has become increasingly common especially due to the rampant globalization of the software development process. Transfer of contextual knowledge across stakeholders involved in such distributed environments is critical to successful product development and support. Timely transfer of appropriate knowledge about issues faced in production environments to support teams and within support teams can result in significant economic benefits. Failure to share contextual knowledge and lack of common understanding significantly impact the quality of support. We draw from the literature on knowledge transfer and adaptive structuration theory to develop a theoretical basis and an approach to knowledge transfer in distributed software support contexts. Based on a multi-site case study, we develop a traceability framework to enhance shared understanding among team members. A qualitative evaluation of the usefulness of our traceability-based approach to knowledge transfer is presented.

Keywords:

Distributed software support, knowledge transfer, traceability, adaptive structuration theory.

INTRODUCTION

Several software development activities, including software support, routing programming, and complex design, are being sourced to different parts of the world (Oshri, Kotlarsky and Willcocks, 2007). Outsourcing of software support activities to distributed teams has become an established practice. Consider a large U.S. investment bank (FinCo) that uses an outsourcing vendor MidCo that is based in India. FinCo deploys complex middleware systems to manage its transactions. MidCo manages the daily operations of the middleware system and ensures their reliability. MidCo is also responsible for keeping the system updated and creating new configurations that meet the evolving needs of FinCo. While MidCo has several teams located in India, it also uses some onsite teams at the customer sites. The onsite teams are primarily focused on developing a greater understanding of FinCo's operations so that MidCo can effectively support the operations of FinCo's middleware systems. The offshore support personnel also visit FinCo's various locations, primarily to interact with the onsite personnel. In this scenario, how can MidCo's onsite personnel transfer their knowledge about FinCo's needs and problems to teams operating offshore? Such knowledge transfer across distributed teams has been recognized as a challenging task (Lin, Geng and Whinston, 2005; Oshri et al., 2007). Motivated by this challenge, our research seeks to answer the question, *"How can effective knowledge transfer among distributed software support teams be facilitated?"*

We draw from Adaptive Structuration Theory (AST) (DeSanctis and Poole, 1994) to develop guidelines for the development of an approach to answer this question. Based on a multi-site case study, we develop a traceability-based solution to enable knowledge transfer among distributed teams. Traceability, the ability to describe and follow the life of a conceptual or physical artifact (Ramesh and Jarke, 2001), has been widely used in software development to understand the development and use of various software artifacts like requirements, design elements, source code, test cases, etc. In this research, we present traceability as an effective knowledge transfer mechanism.

The paper is organized as follows: First, we discuss the theoretical bases for this research. We then present our case study and the development of our traceability framework. A preliminary qualitative evaluation of our approach is then described. Finally we discuss implications to research and practice.

THEORETICAL BACKGROUND

Knowledge transfer is defined as the process of knowledge diffusion from knowledge sources to knowledge recipients (Joshi, Sarker and Sarker, 2005). It is the process by which an organization recreates and maintains a complex set of routines in another setting. The objective of such knowledge transfer is to improve the receiver's understanding of the organizational processes and systems (Argote, 1999). Several areas of research including knowledge management, organizational science, software development, and strategic management have emphasized the importance of knowledge transfer across various organizational entities (Kwan and Cheung, 2006). Specifically, knowledge transfer in distributed contexts has gained considerable attention in past research (Desouza, Nissen and Sørensen, 2008). While much of the extant research is focused on understanding and evaluating the factors that affect knowledge transfer (for example, different types of organizational boundaries and the transfer of knowledge across them (Carlile, 2002; Carlile, 2004), nature of knowledge (Nonaka, 1994; Zander and Kogut, 1995), and absorptive capacity (Chen, 2004; Cohen and Levinthal, 1990)), our research is unique in its focus on facilitating knowledge transfer using a traceability framework grounded both theoretically in AST and empirically in the context studied.

In the following sections, we draw from AST to guide the development of an effective knowledge transfer approach for distributed software support activities.

Use of AST to Guide Knowledge Transfer

AST which highlights the use of Structuration theory in IS (Jones and Karsten, 2008), provides a suitable theoretical basis for the development of an approach to facilitate knowledge transfer. AST provides insights into how people incorporate technologies in their work practices (DeSanctis and Poole, 1994). AST focuses on "social structures, rules and resources provided by technologies and institutions as the basis for human activity" (DeSanctis and Poole, 1994). Social structures guide the performance of various activities. AST comprises of two central concepts – structuration and appropriation. Structuration is defined as "the act of bringing rules and resources from an advanced information technology or other structural sources into action". AST posits that the structure within technology and structure in action are two important components of structuration that are continually intertwined, shaping one another. Application of specific rules or resources within a specific context is referred to as appropriation. Different structural features of technology may be appropriated by different individuals in different ways (DeSanctis and Poole, 1994). AST has primarily been applied to examine the adaptation of advanced information technologies that enable organizational change (Dennis and Garfield, 2003) and the dynamics of global virtual teams (Maznevski and Chudoba, 2000). In this study, we use AST as our theoretical foundation to develop a knowledge transfer approach for distributed software support.

Implications for Knowledge Transfer

Our research is based on the premise that knowledge transfer is a process of social interaction which can benefit from structuration and appropriation. Drawing from AST, we argue that structure in the knowledge transfer process and the way in which this structure is appropriated by different stakeholders in the knowledge transfer process impacts outcomes of knowledge transfer. Also, since prior literature on software development has emphasized the importance of structured acquisition of process and product knowledge (Ramesh, 2002; Ramesh and Jarke, 2001), we argue that the impact of structure in the knowledge transfer process and the way in which it is appropriated will likely have a positive impact on decision outcomes in the context of software support. The implication of AST for knowledge transfer is that we need to consider the '*structure' used to represent knowledge elements* and the *'structure' used to guide the process of knowledge transfer*. Whereas the former refers to the definition of the type of knowledge elements and the relationships among them, the latter refers to the activities involved in knowledge transfer.

In the following section we explain how these implications of AST help us develop a traceability-based solution.

Traceability Based on AST

Traceability has been used in software engineering as a technique to describe and follow the life of any artifact developed during the software development lifecycle (Ramesh and Jarke, 2001). Traceability has been considered critical in managing changes in software development, helping development and support teams understand the impact of changes and establish consistency among software artifacts. The use of traceability for knowledge transfer involves creating a traceability model and a process. This maps to the implications of AST discussed above as follows:

1. Traceability framework as 'structure' for knowledge - Software development organizations commonly use traceability matrices or tables to create links among customer requirements with design elements, source code modules, test cases, etc. Past research suggests that more sophisticated traceability practices are necessary to facilitate the transfer of process knowledge in software development (Ramesh and Jarke, 2001). In such a practice, a traceability framework (i.e., a semantic network of knowledge about artifacts and processes followed to develop these artifacts) is developed. Traceability frameworks provide a shared vocabulary or meta-data about knowledge that needs to be shared among team members. They provide structure in terms of what components of knowledge are valuable for documentation and sharing among the participants in software development and support activities.
2. Traceability framework as 'structure' for the process of knowledge transfer - A traceability framework needs to be complemented by a process that developers can follow to document and transfer knowledge. A comprehensive traceability framework should suggest a structured process for creating, sharing, and updating knowledge chunks during the development and support processes.

In summary, implementation of the guidelines discussed above entails the development of a traceability framework to enable the structuration and appropriation of knowledge transfer in distributed software support. Based on a case study, we develop a traceability framework that includes a traceability model that explicates the structure of knowledge and a process to enable knowledge transfer in distributed software support. We then present a preliminary qualitative evaluation of our approach to examine its usefulness in knowledge transfer.

TRACEABILITY-BASED KNOWLEDGE TRANSFER FOR DISTRIBUTED SOFTWARE SUPPORT

Phase 1: Development of Traceability Framework

Case Study

One of the organizations in our case study (MidCo) supports the use of middleware systems in large companies in the financial services industry. The second organization (SCCo) develops, deploys and provides support for supply chain management systems in different industries.

MidCo and SCCo were selected as appropriate for our study for the following reasons: (1) Both were involved in providing distributed support and were facing challenges in knowledge transfer among teams located onsite and offshore; and (2) There were significant differences in the way their support operations were distributed. While MidCo had very few employees onsite, SCCo had a significantly larger number of onsite employees.

MidCo is a large IT services company that provides both software development and support services to a variety of clients. We investigated MidCo's support provided to a large financial institution (FinCo). FinCo uses a middleware system that facilitates the storage and transmission of transactional messages across a variety of their systems. MidCo was involved in the deployment and daily support of this middleware system. FinCo operates in multiple locations around the world. A few of MidCo's employees reside onsite at these locations while several MidCo teams are located in India. MidCo employees who are onsite have the opportunity to acquire significant amounts of domain, technical, and process knowledge about FinCo's operations. MidCo requires its support employees in India to visit the various FinCo locations to facilitate the transfer of knowledge from their onsite employees.

SCCo, which is involved in developing and supporting supply chain solutions for various vertical industries like apparel and pharmaceutical, has several teams located in the US and a few teams located in India. SCCo's clients are primarily located in the US. Many of SCCo's developers are involved both in the development and support of their products. Our study investigated support activities associated with a warehouse management system (WMS), a core product of the company. Occasionally, developers from SCCo's US location visit their office in India to transfer domain and technical knowledge. These visits often last several months.

Data Collection and Analysis

Data collection was done through semi-structured interviews with project managers and developers at both organizations. Three developers and two project managers from MidCo, and five project managers from SCCo participated in our study. Interviews focused on understanding the support processes and the knowledge needs of support personnel. Interviews included discussions about several support scenarios commonly faced by developers and the processes followed to address these issues. The discussions typically spanned several projects that the informants had worked on. Whenever possible, interviews were recorded and transcribed. Detailed notes were taken during interviews. Each interview lasted at least one and a half hours and each informant was interviewed multiple times (1 to 3 times). Several documents such as change management process descriptions, design specifications, change request descriptions, and systems such as change management systems pertaining to the development and support of various products were also examined to understand the nature of the product being supported. Data analysis followed open, axial, and selective coding techniques, commonly used as part of the grounded theory method (Strauss and Corbin, 1990). Open coding is the process of breaking down, examining, comparing, conceptualizing and categorizing data. It is aimed at revealing essential ideas found in the data. Labeling and discovering categories are the two tasks involved in open coding. In labeling, discrete events and ideas receive a label. Categories are discovered by finding related phenomena or common concepts or themes in the data. These themes are grouped into common headings thereby leading to categories and sub-categories. Axial coding aims at further developing the categories and their properties, and establishing relationships among them. Selective coding is used to integrate the categories to form a theoretical framework. These coding techniques resulted in the concepts in the traceability framework. Segments of data were labeled initially with codes. These codes were then categorized and relationships between categories were created, resulting in the traceability framework.

Traceability Framework

We first describe the problems with the knowledge transfer process faced by both organizations. Then we present the traceability framework (developed based on our case study), which includes a traceability model (shown in Figure 1) and a process. The need for and challenges to knowledge transfer that were observed in the case study sites informed the identification of the knowledge elements and steps in the process in the traceability framework.

Problems and challenges in knowledge transfer in distributed software support

1. *Lack of standards for documenting relevant knowledge to facilitate knowledge transfer:* The participants in the process do not know what types of knowledge are candidates for knowledge transfer. In MidCo, onsite maintenance managers have significantly more technical and domain knowledge as well as the expertise that is required to solve the problem than offshore support personnel. However, onsite personnel are rarely involved in solving day to day production support issues. Instead, they largely play more managerial roles while production support is handled by teams in India. Periodically, one or two team members from MidCo's Indian location visit FinCo for knowledge transfer. They work with the onsite managers to learn about the domain, technology being used and the nature of various production issues. When they return to their teams, they are responsible for transferring the knowledge acquired from such visits to the rest of their team. Though there is an informal process in place for knowledge transfer, since there is no structure that guides the process, knowledge transfer does not happen as expected. It often becomes difficult for the team members who visit FinCo for knowledge transfer purposes to actually transfer this knowledge to their colleagues at MidCo. Similarly, there have been concerns in SCCo about the adequacy of offshore teams' domain and technical knowledge. The lack of structure in documenting knowledge about production issues and the system itself have been identified by SCCo as the primary reason for this problem.

2. *Inability to comprehensively track issues:* At MidCo, production support starts with offshore maintenance personnel receiving information from a client or the middleware system itself about a problem. Many issues are automatically generated by the middleware system which is capable of recognizing alerts. The issues are examined and assigned to one or more team members, often along with high-level guidelines on how to address the issues. Issues are sent directly to teams in India and the team members contact onsite personnel for help only in complex and rare situations. However, the issues are sometimes not tracked in a systematic way and often the resolution of issues is significantly delayed. SCCo has also been facing considerable challenges in ensuring that the offshore developers receives the issues accurately and react in a timely manner.

3. *Inability to map solutions to issues:* At SCCo, support scenarios involve a developer getting a phone call from a client about an issue. The developer immediately tries to identify the root cause of the problem and attempts to address it. The maintenance request, details about the cause, the reproduction of the problem, and the possible or implemented solution are logged into a tracking system. For simple issues, such documentation is done as the issues are being resolved. However, for complex issues the documentation is often done later. In the recent past, the support process has gradually shifted to SCCo's teams in India. Developers and managers in the US location receive calls about production issues and delegate some of them to the teams in India. However, these knowledge elements are not captured as traceable artifacts. For example, if the root cause identified in one issue has caused other problems cannot be easily retrieved to help investigate and design the solutions for the current issues.

4. *Lack of knowledge reuse:* Both at SCCo and MidCo, the development and support teams did not share a common structure or vocabulary that facilitates knowledge capture and reuse. They were unable to easily retrieve and assemble past solutions to address current problems. The knowledge captured in the current tracking system was not easily transferrable to other situations and other stakeholders.

To address the above identified challenge, we developed a traceability framework to provide structure for both knowledge and process.

Traceability framework: (A) 'Structure' for knowledge

The support process usually begins with a *production issue* reported by the end-users or by the system itself. A production issue is a malfunction that occurs while the system is in operation and hinders the end-users from performing their function. Support personnel examine the issue, attempt to reproduce it in a test environment, to identify the *root cause* of the issue. The root cause could be errors introduced in the past in parts of the source code, design errors, erroneous data, erroneous processes followed by end-users, etc. After the support team examines the root cause, the *changes* that need to be done to address the issue are identified. These changes impact several *design and documentation elements*. Design specifications, models, and textual documentation of requirements, design, code, and system operation procedures are examples of design and documentation elements. A change may range from a simple modification to a configuration to a significant modification of the design. Requirements that lead to major design changes are reported to development teams. Often these are assigned to be implemented in the next version of the product. However, stopgap measures are taken to prevent the recurrence of the production issue until such a more permanent solution is implemented. Implemented changes are documented in a change management system. This documentation includes the identity of the team member who was responsible for the issue and a textual description of the issue and the solution. Ideally, any documentation that is associated with the design elements that in which changes are incorporated is also updated. *Decisions* that result in *resolutions* regarding the changes incorporated and the *rationale* behind them are also documented. Any conflicts among the decisions or design elements are also recorded. Finally, the change is examined for any general characteristics that could be used in implementing future changes or in redesigning the system. If the support or development team identifies any *patterns* that emerge from the changes that they have worked on, they document these patterns as well so as to enable reuse of solutions used in this scenario in future scenarios.

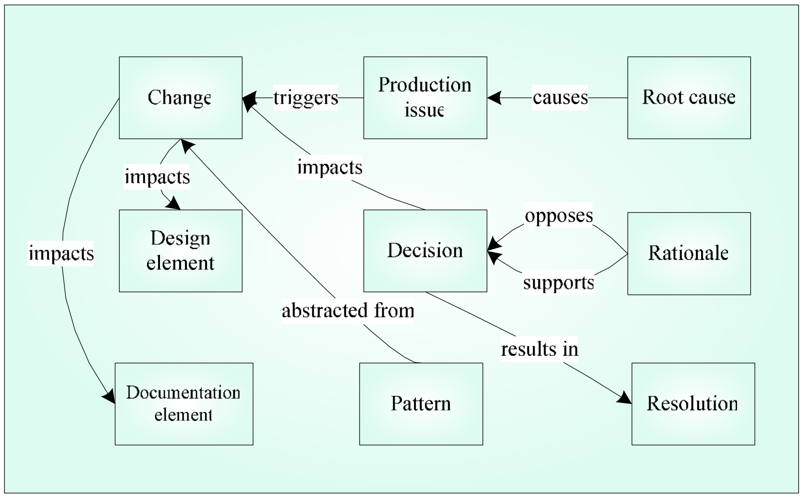

Figure 1: Traceability Model

The traceability model provides a way to structure the knowledge that is generated and used during production support. How can the developers actually use this model in their daily work processes? Below, we describe a process that developers can follow in using our traceability model to effectively transfer knowledge.

Traceability framework: (B) 'Structure' for process of knowledge transfer

Development and support teams should select development environments and technology infrastructure that is adequately equipped to facilitate traceability. Such an infrastructure should enable the development and support teams to follow a structured process to document traceability knowledge, thereby enabling effective knowledge transfer. We present a structured traceability-based knowledge transfer process using a scenario for handling a production issue at MidCo. We have developed this process based on the insights provided by the challenges faced in establishing effective knowledge integration practices in the focal organizations.

1. *Customizable traceability frameworks:* Software development and support organizations should acquire and deploy an infrastructure that can facilitate the use of project specific traceability frameworks. This involves the ability to custom-define knowledge elements and links among them. Development and support teams can decide at the onset of the project on the types of knowledge elements that play the most critical role in helping the teams effectively support their clients and define these elements in the traceability infrastructure. For instance, they can define the traceability structure by adapting the traceability model presented in Figure 1.
2. *Comprehensive tracking of issues through shared understanding:* A team member in the offshore location receives a call from the client regarding the inordinate accumulation of messages in a server that lead to the failure of the server. The team member immediately logs the issue in a collaboration tool as a high priority issue and assigns responsibility. The onsite manager receives a note through this system about the issue. Though the issue has been assigned to a team member in India, the onsite manager decides to monitor progress due to the criticality of the issue. Throughout the process of working on such issues, all the stakeholders involved develop a shared understanding.
3. *Mapping solutions to issues utilizing the traceability model:* Offshore team members document ‘production issues’ as a knowledge element in the traceability model and link it to the tasks defined in the collaboration system. As the team member makes progress in addressing the issue, the decisions made, the rationale, etc are documented as traceability knowledge. The onsite manager monitors this knowledge and the production system to ensure that the issue is being resolved appropriately. S/he links this knowledge element to a document that describes how a similar issue was resolved in the past. The team members in India accesses this knowledge through the collaboration system and if needed through knowledge elements in the traceability infrastructure. The exchange continues till the issue is resolved. All exchanges and documents that are shared during the process are linked to the knowledge elements documented in the traceability infrastructure.
4. *Knowledge reuse through a shared vocabulary and retrieval process:* A shared process provides a common ground in terms of both a shared vocabulary and a set of steps that will help effectively retrieve knowledge about past projects. Decisions taken by one team can always be reviewed and used as an input for other scenarios faced by other teams.

The most important aspect of our approach is that it provides structure to the social interactions in the way in which knowledge is explicated. The stakeholders involved in the knowledge transfer process need to codify knowledge before it is understood and used by others. This explication process is structured using the traceability model and the process described above. This way, the developers do not create unstructured documents or irrelevant knowledge chunks. The use of a structure enables the sender and receiver to clearly communicate and understand the knowledge being transferred.

Though our approach provides a structure to the knowledge transfer process, it should be noted that different developers can ‘appropriate’ the approach differently. As highlighted by DeSanctis and Poole (1994), developers may use different ‘appropriation moves’. For example, they may relate past structures that they had used in transferring knowledge and blend it with the structures provided by our approach. The study of how different stakeholders may use this approach in different settings is a topic of ongoing research.

Figure 2 (adapted from (DeSanctis and Poole, 1994)) summarizes how AST guides the development of our traceability framework. The elements of AST have been adapted for knowledge transfer, and four parts of the knowledge transfer process in distributed software support as part of social interaction among stakeholders involved in the software support.

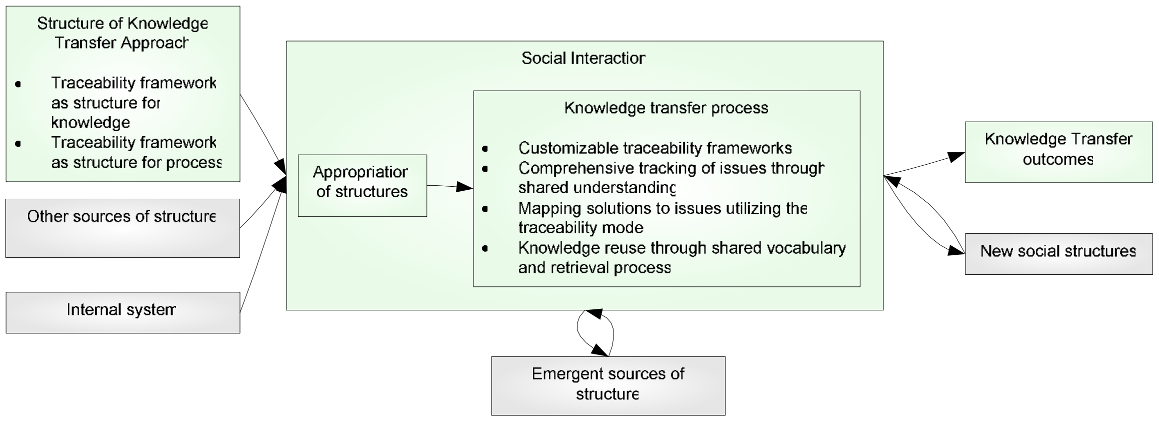
Structure of Knowledge Transfer Approach
Traceability framework as structure for knowledge
Traceability framework as structure for process
Other sources of structure
Internal system
Social Interaction
Appropriation of structures
Knowledge transfer process
Customizable traceability frameworks
Comprehensive tracking of issues through shared understanding
Mapping solutions to issues utilizing the traceability mode
Knowledge reuse through shared vocabulary and retrieval process
Knowledge Transfer outcomes
New social structures
Emergent sources of structure

Figure 2: Adaptation of AST for Knowledge Transfer (adapted from (DeSanctis and Poole, 1994)) [1]

Phase 2: Preliminary Evaluation of our Approach

As a first step in evaluating the usefulness of our approach, we conducted a qualitative evaluation. We presented the traceability model and the process to five experienced development managers from SCCo who are involved in distributed software support. The participants were then asked to provide feedback on the usefulness of this approach in enabling knowledge transfer and in turn on improving the quality of distributed support. Following is a summary of this qualitative evaluation:

1. *Value of the framework in guiding knowledge transfer:* The informants recognized that the lack of a structured way of collaborating and transferring knowledge was the primary impediment for effective knowledge transfer. Therefore, the use of a framework to structure knowledge transfer was viewed as very valuable. It provides clarity and ensures uniform acquisition and sharing of knowledge among distributed team members. Further, it will provide a much needed structure to the way the team members interacted.
2. *A unified process:* The informants commented that their past attempts at managing and transferring knowledge were inhibited by the lack of interoperability with the most commonly used development and support tools. They suggested that a primary reason why developers are reluctant to consult documentation is the lack of structure in the documentation which makes access to relevant information difficult. Our approach addresses this issue by providing not only a structure to manage relevant knowledge, but also by presenting a process for establishing traceability for knowledge transfer.
3. *Balance in knowledge acquisition and transfer:* One of the primary problems discussed by the informants relates to scoping the type of knowledge that needs to be transferred. They report that the framework offers such a balance without tipping the scale towards making knowledge acquisition and transfer as an expensive overhead. The framework provides structure and guidance for knowledge transfer thereby avoiding expensive documentation and transfer of extensive amounts of knowledge.
4. *Scale of implementation*: Though the approach was considered to be potentially useful, organization-wide implementation of such an approach was recognized to be challenging. Since different development and support teams are used to interacting with one another in specific ways using different approaches and tools, bringing them together to use a unified approach was seen as the primary challenge in implementing this approach.

[1] It should be noted that while we recognize the importance of all elements of AST, we specifically draw from parts that are shown in green in Figure 2. We specifically focus just on the structure of knowledge transfer and not on other sources of structure.

IMPLICATIONS TO RESEARCH AND PRACTICE

This research takes the first step in suggesting an adaptation of AST in the development of a knowledge transfer approach for software support teams. This research is also unique in providing a theoretical basis for the use of traceability in software support activities, because much of the prior work on traceability is motivated by practice and lacks theoretical grounding. Drawing from AST, we suggest the use of structure in social interactions to improve knowledge transfer. This research has critical implications for software support teams that are globally distributed. The traceability framework developed here can be used as a reference framework and may be tailored to suit project specific needs.

A more detailed evaluation of the usefulness of our approach through quantitative studies is a subject of ongoing research. We are currently in the process of developing a system to implement our approach.

REFERENCES

1. Argote, L. (1999) Organizational Learning: Creating, Retaining, and Transferring Knowledge Kluwer Academic, Boston.
2. Carlile, P.R. (2002) A Pragmatic View of Knowledge and Boundaries: Boundary Objects in New Product Development, *Organization Science*, 13, 4, 442-455.
3. Carlile, P.R. (2004) Transferring, Translating, and Transforming: An Integrative Framework for Managing Knowledge across Boundaries, *Organization Science*, 15, 5, 555-568.
4. Chen, C.-J. (2004) The Determinants of Knowledge Transfer through Strategic Alliances, *Academy of Management Proceedings*, H1-H6.
5. Cohen, W., and Levinthal, D. (1990) Absorptive Capacity: A New Perspective on Learning and Innovation, *Administrative Science Quarterly*, 35, 1, 128-152.
6. Dennis, A.R., and Garfield, M.J. (2003) The Adoption and Use of Gss in Project Teams: Toward More Participative Processes and Outcomes, *MIS Quarterly*, 27, 2, 289-323.
7. DeSanctis, G., and Poole, M.S. (1994) Capturing the Complexity in Advanced Technology Use: Adaptive Structuration Theory, *Organization Science*, 5, 2, 121-147.
8. Desouza, K.C., Nissen, M., and Sørensen, C. (2008) Managing Knowledge Transfer in Distributed Contexts, *Information Systems Journal*, 18, 6, 559-566.
9. Jones, M.R., and Karsten, H. (2008) Giddens's Structuration Theory and Information Systems Research, *MIS Quarterly*, 32, 1, 127-157.
10. Joshi, K.D., Sarker, S., and Sarker, S. (2005) The Impact of Knowledge, Source, Situational and Relational Context on Knowledge Transfer During Isd Process, *38th Annual Hawaii International Conference on System Sciences (HICSS'05)*, IEEE Computer Society.
11. Kwan, M.M., and Cheung, P.-K. (2006) The Knowledge Transfer Process: From Field Studies to Technology Development, *Journal of Database Management*, 17, 1, 16-32.
12. Lin, L., Geng, X., and Whinston, A.B. (2005) A Sender--Receiver Framework for Knowledge Transfer, *MIS Quarterly*, 29, 2, 197-219.
13. Maznevski, M.L., and Chudoba, K.M. (2000) Bridging Space over Time: Global Virtual Team Dynamics and Effectiveness, *Organization Science*, 11, 5, 473-492.
14. Nonaka, I. (1994) A Dynamic Theory of Organizational Knowledge Creation, *Organization Science*, 5, 1, 14-37.
15. Oshri, I., Kotlarsky, J., and Willcocks, L.P. (2007) Global Software Development: Exploring Socialization in Distributed Strategic Projects, *Journal of Strategic Information Systems*, 16, 1, 25-49.
16. Ramesh, B. (2002) Process Knowledge Management with Traceability, *IEEE Software*, 19, 3, 50-52.
17. Ramesh, B., and Jarke, M. (2001) Towards Reference Models for Requirements Traceability, *IEEE Transactions on Software Engineering*, 27, 1, 58-93.
18. Strauss, A., and Corbin, J. (1990) Basics of Qualitative Research: Grounded Theory Procedures and Techniques Sage Publications, Newbury Park, CA.
19. Zander, U., and Kogut, B. (1995) Knowledge and the Speed of the Transfer and Imitation of Organizational Capabilities: An Empirical Test, *Organization Science*, 6, 1, 76-92.