Knowledge Problems in Corrective Software Maintenance - A Case Study

Maria Alaranta
Department of Industrial Engineering and Management, School of Science and Technology, Aalto University, Finland
maria.alaranta@aalto.fi

Stefanie Betz
Karlsruhe Institute of Technology, Germany
stefanie.betz@kit.edu

Abstract
Corrective software maintenance derives its importance not only from the huge costs it induces but also its importance to customer relations and ultimately, the software company's revenues. In this paper, we address the knowledge problems in the corrective software maintenance process by applying the lens of Transactive Memory System (TMS). TMS is a theory of knowledge coordination in groups.

We carry out a case study of the troubled corrective software maintenance process of a multinational ICT company. We discuss our empirical case from three key TMS manifestations: specialization, credibility and coordination. Our results show that TMS could offer a fruitful avenue for understanding and managing knowledge problems that lead to poor performance in such a process. Designers of a global software maintenance process should not only pay attention to the process structure and supporting IS but also to facilitating a well-functioning TMS.

1. Introduction

The estimates for the relative cost of maintaining software and managing its evolution have varied from over 50% [15] to over 90% [5] of its total cost. It has been estimated to consume over 75% of the Fortune 1000’s IS budget [1]. Software maintenance refers to "the modification of a software product after delivery, to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment."[8 p.47] Software maintenance tasks can be further divided into correction, adaptation and perfection of operational software [15].

This paper focuses on corrective software maintenance. It derives its importance not only from the huge costs it induces but also its importance to customer relations and ultimately, the software company’s revenues. Together with other front-end activities, corrective software maintenance solves clients’ problems and questions. This can lead to a long-term relationship that is satisfying for both the client and the software company.

Prior literature on software maintenance has addressed issues such as scale economies in batching software maintenance [1], benefits of departmentalization in software development and maintenance [21], key problems in software maintenance [15] and the key problems in front-end support in corrective software maintenance [10]. This literature has shown that key problems in software maintenance continue to be of managerial nature [15] [10]. These prior contributions provide a comprehensive understanding of the range of problems.

It has been found that software maintenance is knowledge intensive work [25] [Cf. 2] that is plagued by inadequate information and uncertainties [27]. In corrective software maintenance, some key sources of ambiguity and uncertainty include, for example: poor documentation, meeting scheduled commitments, interdependencies with other components in the product in question [15], communication with the users and others, interaction and interfaces, and user involvement [10]. Our research question is: What knowledge problems trouble corrective software maintenance?

Our empirical data comes from the corrective software maintenance process of a multinational company X in the field of ICT. At the first quick glance, our impression of the empirical case was that a recent merger-related organizational change had resulted in a well-defined and well-run process that follows well-accepted principles of incident management. Yet, customer dissatisfaction was high and key performance indicators low. Our initial analysis indicated that a key problem was that of knowledge flows and coordinating knowledge in the process.

Transactive Memory Systems (TMS) was chosen as the theoretical perspective to this study as it is a theory that explains how people in collectives encode, store, use, and coordinate their knowledge to achieve [16]. As Wenger [23] (p. 204) puts it, “structuring of an organization is clearly an exercise in structuring..."
In this paper, we present an empirical case where problems of structuring a transactive memory to support software maintenance are evident. The empirical case illustrates problems in all three manifestations of TMS: Specialisation of tasks, Credibility of other group members and Coordination of actions. By doing so, this paper aims at contributing to the literature on software maintenance by providing a detailed account of knowledge problems related to corrective software maintenance.

The rest of this paper is structured as follows: in the next Section we discuss briefly the theoretical perspective to study the Transactive Memory System. Then, we describe the methodological choices. The empirical findings of the case are presented in Section 4 and discussed in section 5. The next Section, Section 6 concludes this paper and offers suggestions for further research.

1. Transactive Memory Systems

Transactive memory is a shared conceptualization about who knows what in a group. Combining individuals’ or groups’ transactive memories and communications (transaction) between them forms a transactive memory system. TMS transactions include encoding, storing and retrieving information [23-25].

Participants of TMS assume responsibility for different areas of knowledge [7] in terms of the know-what, know-how and know-why in the knowledge area in question (Quinn et al. 1996 in [16]). Through this structure, TMS allows its members to use other as memory resources and thereby access information well beyond their own cognitive capacities [16]. When TMS exists, it causes specialization and division of labour [20] as well as mutual trust on others’ knowledge and smooth, coordinated task processing [13]; all crucial for the efficient execution of corrective software maintenance in a global setting.

TMS research has traditionally addressed information flows in stable dyads and (small) groups [3, 23, 25] [13] but recent research has extended its use to various types of groups [16], teams [6, 13, 17] and organizations [19-20, 23]. Besides providing insights on corrective software maintenance, this paper aims at taking a step towards contributing to the literature on TMS in large groups by discussing an empirical case of corrective software maintenance where weak TMS results in poor efficiency. This paper presents initial insights on the specific problems that the context of a multinational organization under organizational change presents to TMS in action.

Prior research has suggested that a key prerequisite for TMS to develop is cognitive interdependence. I.e. each member’s actions have an impact on others’ outcomes and members are dependent on each other to the extent they can’t guarantee outcomes by performing the task alone. This interdependence can stem from a group reward structure, a task structure where one member’s output becomes the next members’ input, or task complexity [3].

Prior research has identified three stages that are involved in the creation and maintenance of TMSs: the use and currency of directories, information allocation, and retrieval e.g. [9, 16]. It has also identified three manifestations of a well-functioning TMS: specialization of tasks, credibility of other group members’ knowledge, and coordination of actions [13] [14]. Our empirical work focuses on analyzing problems in the manifestations of a well-functioning TMS.

Earlier research has applied these concepts to both small [13], medium-sized [6, 13, 17] and large [9, 16] groups, including both on-site [13] and distributed [13, 16-17] teams. Efficiency via specialization is facilitated by a differentiated knowledge structure; i.e. that there are areas of (lower-order) knowledge that one knows and others do not know. Mutual meta-information is needed to enable others to retrieve knowledge relevant to them [25]. Integrated structure refers to a situation where common (yet differentiated in details of content) lower-order knowledge storages allow integration; i.e., developing a shared higher-order conceptions [25]. The differentiated yet integrated structure is also a precondition for building new knowledge and innovation. Through these structures, TMS allows members to access knowledge that they would otherwise not have and thus allows groups to perform better.

Group members are willing to develop differentiated knowledge if they can rely on others to provide other task-critical knowledge [7]. In particular, when the task is beyond individual TMS members’ cognitive capacities, each member’s outcome is dependent on the knowledge of others in the same group. In traditional TMS theory, beliefs about a group member’s task-relevant expertise serve as a basis for task assignment and specialization [7]. Credibility of others’ knowledge allows a group member not only to receive the knowledge but also to trust it and be confident in relying on it [13]. Traditional TMS theory suggests small, stable groups develop and validate a shared understanding of who knows what through the experience of working together [3].

The final manifestation of TMS is a smooth, efficient coordination of action towards reaching the goal. This requires the members to have a good understanding of where the knowledge resides and how knowledge from different sources fits together [13].
Traditional TMS theory suggests that this is based on a shared mental model of who knows what.

However, not all dyads and groups succeed in building a well-functioning TMS. Misleading stereotypes may guide the allocation of responsibility [23]. Duplicate knowledge or one group having access to all knowledge decreases dependency dramatically [25]. When a process is built to channel knowledge away from expertise, things may be forgotten. Or, if expertise is in dispute, knowledge may fall in cracks. [23]

Developing a TMS may face major challenges in a distributed process [17], and in particular in globally distributed teams such as software maintenance in a multinational company. High turnover that hinders the long-term development of TMS is a problem of distributed software work in general [20] and it is a prominent problem in several studies on software maintenance [10][15]. Distributed teams may also lack prior experience of working together or suffer from decreased communications and misunderstandings. Staff transfers and recruitments aggravate this problem. [20].

In addition, corrective front-end support is plagued with problems of communicating with and understanding users, issues related to interaction and interfaces, and software having been written elsewhere [10]. All these make it difficult to encode knowledge to repositories.

Prior literature offers some guidance for overcoming some of these problems. TMS’s range from differentiated to integrated, and the style of TMS should be matched to the task; the first supporting innovations and creation of new knowledge and the second facilitating greater efficiency [23].

In activities like corrective software maintenance, where a client reaches the organization, the first contact person should not be a mere telephone operator. Instead, the first contact point should be a TMS expert that knows the expertise structure in the organization (i.e. who knows what in the company) and is thus able to assign the problem directly to the person that is able to solve it [23].

It has also been noted that TMS in organizations can be supported by computer-based information systems [23][19][20]. However, building such a system is bound to be challenging as much of the knowledge is contextualized, tacit and/or ever-changing. Therefore, it has been suggested that the IS should consist of a meta-memory directory as meta-memory is less volatile and perhaps more explicit than the knowledge itself [19]. Also, more recent research has suggested that virtual workspaces may help the effectiveness of knowledge-sharing [17].

2. Methodological choices

We needed detailed information from multiple organizational levels and stages in the maintenance process to capture relevant meanings and nuances. Qualitative studies are particularly suitable for matching these unique needs. Thus, a case study was chosen as the method of this study as it allows a detailed, in-depth scrutiny of the phenomenon in its real-life settings [26][4]. A study of a single case is an appropriate strategy for revelatory studies [26]. A multi-tier global software maintenance process covering various product ranges offers a particularly good setting for observing a wide range of managerial problems. We opt to carry out a single case study in order to theorize about the nature of managerial problems in such situations.

Corrective software maintenance at multinational Case X. Our study focuses on the corrective software maintenance process of a multinational company in the field of ICT that has its headquarters in Europe. The organizational context of the case is turbulent: the company has gained its current form via a merger a few years earlier, and has gone through a series of organizational restructuring efforts before and after the merger. Currently, it has over 50 000 employees. It has hundreds of products in its portfolio and the main product consists of integrated hardware and software components. The company’s main product consists of both hardware and software components.

At the time of our data collection in 2009, Case X had succeeded in integrating its corrective software maintenance processes after a merger. The new process has one to four stages depending on the complexity of the maintenance problem (see Figure 1). The idea is to optimize the process so that the number of unsolved cases is decreased at each stage.

![Figure 1. Corrective Software Maintenance at Case X](image)

The process starts when a customer reports a problem and the front-end keys the problem into the IS as a case. At each stage, the engineers try to solve the problem within the specific paradigm of the stage in question and within a timeframe defined by the service level agreements. If they can’t solve the error within the timeframe, the problem is forwarded to the next stage. The foci for troubleshooting cases at Stages 1-4 are:

- Stage 1 (S1): a local customer front-end where the error reports from customers are received, recorded
in the supporting IS and solved if possible based on re-used data; otherwise forwarded to Stage 2.
- Stage 2 (S2): centralized maintenance centers where the case is solved if it is doable by analyzing technical queries; otherwise forwarded to S3.
- Stage 3 (S3): product centers where the case is solved if possible by analyzing trouble reports; otherwise forwarded to S4.
- Stage 4 (S4): R&D develops a preventive resolution (often in batches) based on error reports.

The process ends when a solution is delivered to and accepted by the customer. If the customer doesn’t accept the solution, the case won’t be closed but will instead be returned to the engineer working on the case.

The process follows well-accepted principles of incident management in the sense that incidents are detected and recorded; they are classified and initial support is provided, often in the form of a workaround; they are investigated and diagnosed; resolution and recovery actions are carried out when possible; and each case is eventually closed. In addition, the case company has established procedures for incident ownership, monitoring, tracking and communication both with the client and with the different actors in the software maintenance process.

In sum, our first impression was that Company X had implemented a well-defined and well-run corrective software maintenance process. Yet, at the time of collecting the data for this study, the management was not pleased with the process performance. In particular, they were worried by the amount of customer complaints and also bad internal performance indicators. One key concern of the customers is that the corrective process takes a long time which in turn causes long down-times to the services they provide to their clients.

In order to understand the nuances of the case better, we conducted an initial problem analysis in the form of a rich picture (see Figure 2). This analysis resulted in a holistic understanding of the process and its problems, and lead us to understand that knowledge flows and coordinating knowledge in the process were key issues in this case.

**Data collection.** To get a detailed understanding of the case, we collected data via semi-structured interviews that were complemented with direct observations of case handling and documents provided by the case organization.

We conducted semi-structured interviews with 47 interviewees representing all relevant actors and organizational levels within the case organization; including engineers, managers, process developers and IT; as well as all stages of the maintenance process.

The initial list of interviewees was composed in cooperation with the case organization. Each interview lasted about 1-2 hrs. Most interviews were carried out in English, but English was a second language for most interviewees. The interviews were tape-recorded and transcribed. The interview themes were:
- Respondent background: career history, current title, job description and role in the process.
- The process: Reflections on the overall process and each stage: Who does what? What goes well? What problems are encountered? How are they solved? Why?
- Recent changes: who did what? What went well? What problems were encountered? How were they solved? Why?
- Lessons learned: What should have been done differently? What should be done in the same way? Development ideas?

Questions varied according to the respondent’s role and participation in the maintenance process.

The interview data was supplemented by field notes and transcriptions from 8 meetings and one process development workshops with 18 participants representing various stakeholders. We also obtained documents (e.g. organizational charts, change plans, process charts, instructions, examples of tickets) from the case organization.

**Data analysis.** In this research, data collection and analysis were intertwined [18]. There were at least two field researchers in each interview. Typically, one acted as the main interviewer and the other took notes and asked for further clarifications. The researchers discussed surfacing process problems and other emerging themes after each interview. Using theoretical sampling, new informants were chosen to either confirm or challenge the emerging conceptualizations. Also, new interview questions were formed based on the emerging themes to see if the next interviewees could further explain or deny the theme. Finally, the data collection stopped when a state of theoretical saturation with respect to a particular theme was reached [18].

Thereafter, we coded the data for problems, constructed a work-flow chart of the process and drew a rich picture of the knowledge flows in the process.

To identify the problems, we identified instances where the interviewees explicitly described something as a problem, challenge or equivalent. On top of this, we operated on the rather straight-forward assumption that anything an interviewee mentioned to hamper the work was a problem (at least subjectively perceived by that interviewee).

The work-flow chart mapped the relevant activities in the process to the actors and time-lines. This view was complemented by an analysis performed by applying the rich picture technique [11] to understand knowledge flows. Besides the knowledge flows, key components of the rich picture were knowledge...
repositories (individuals, organization culture, transformations, structures, ecology, and external archives [22]) and computer-based information systems. We also mapped the problems identified in this chart to improve our understanding of problems related to encoding, storing and retrieving knowledge in the case. Figure 2 presents an extract of the rich picture.

Thereafter, we constructed a textual narrative of the maintenance process including the TMS-related problems. Interview quotes are included in the narrative to improve the credibility of this report. Then, we analyzed the data to understand what influences the TMS problems in the maintenance process.

The results will be discussed with key interviewees and academic colleagues before submitting a final version of this manuscript. These practices will serve to improve the construct and internal validity of the findings [26][12].

3. Findings: knowledge-flow problems in the corrective software maintenance process

The process is plagued by problems such as lack of cooperation and collaboration, the lack of shared goals, and insufficient communication with the client. A frequently quoted problem is that the overall logic of the process makes service slow. Even when a problem can only be solved by R&D, it goes through all the stages before being assigned to an R&D engineer. One root of this problem is that the key performance indicators for the four stages are not well-aligned. A head of technical support stated “The objectives aren’t same for the different [stages]” and a process and IS owner added that “[the stages] are sub-optimizing their operations. They aren’t seeing the big picture”. By this he refers to the fact each stage is rewarded by the number of cases they solve. Consequently, engineers often work on cases with only minor chances of succeeding, and only forward the case when the formal rules of the IS force them to do so. By doing so, they waste both the client’s time and their own resources. In a similar vein, service level agreements with the client and the internal agreements contradict each other at some points.

The head of level 2 remarked “We are feeling like ... [we would be working in] different organizations” and that the “[hand-over] between the [stages] does not work”. Moreover, also when the knowledge between the different stages is transferred, it isn’t necessarily perceived reliable by the next stage. As one Stage 3 engineer confirms: “We didn’t really trust the information that came from other levels”. This leads to a redoing of things, since the engineers at the different stages are supposed to work with the knowledge given by the previous stages. Instead of using this knowledge, they ask the customer the same questions again and run the same tests. With this they again waste their resources and the clients’ time.

This is partly due to the continuous chain of organizational changes. The process was reshaped after an M&A that took place a few years before this study was carried out. In this restructuring, the two organizations were integrated and Stage 2 was added to the process. To build S2, some engineers were transferred from other units and countries, others recruited externally. The ramp-up was somewhat painful, and building up the desired skills took time during which quality problems were evident.

Still, at the time of our data collection, it seems that organizational rules are not clear to every stakeholder. A head of level 2 declared “The decision making between the organizations has to be unified".
general, the stages lack awareness about the other stages in the process. One manager declared “there is lack of awareness what the other [stages] are doing”.

Another problem is that customers frequently complain about the process being “a black box”. “Neither the tool or process support that the customer would get active and frequent ... feedback about what is going on with the case” explained the head of care services. Thus, the customer is largely uninformed about the actual status of the case-handling. This is partly aggravated by the fact that there’s no clear ownership of a case. At each stage, an engineer gets assigned to a case according to a queuing system. But, the same engineers are also assigned other tasks such as installing new systems. Sometimes, they find these tasks more important or interesting which can result in delays in case-handling.

In addition to the problems that plague the end-to-end process, each stage has some specific problems:

**Local customer front-end (S1):** At this stage, the engineers receive the error reports and key them into the process-support IS as cases. They record data on e.g. customer identification, the contract and the case in question. After recording the data, the engineers at Stage 1 solve the problem if it is doable by reusing existing knowledge. Other cases should be escalated immediately to subsequent stages. However, sometimes they resist the process rules and try to solve the case by themselves. One interviewee explained that: “S1’s skill level is too high, thus they don’t want hand over the case [to the next stage]”.

On the other hand, when a case is forwarded to S2, the initial case description created by S1 may be insufficient; especially if the case will be forwarded to subsequent stages. One technical support engineer told us: “...we have an office here in [the interviewee’s location] and they are very competent people and the tickets are done well. On other places they aren’t done so well and it takes time.” An engineer from Stage 2 explained “When you receive a case, you hope that it’s properly documented, but this is something that in 99% the cases are lacking information”. As a result, there is a strong mutual distrust between stages S1 and S2.

**Centralized maintenance centers (S2):** At this stage, the engineers solve non-design-defect related problems through intensive analysis of symptoms and test bed simulations. They also record the newly diagnosed symptoms in the IS as well as create and document new solutions for knowledge re-use.

S2 has two key problems regarding collaboration in the maintenance process. First, there is a communications gap between S1 and S2 and thus they ask the customer the same questions again. One interviewee explained: “[Stage 2] doesn’t communicate with [Stage 1] and then they asking after 1.5 months the same questions...”. In addition, S2’s geographically dispersed sites don’t share their knowledge with each other. One manager remarked “We have these engineer groups between [Stage 2 units], that should communicate related to their technology. But what is missing is the community thing even though they are the same team.” Centralized maintenance centers also vary in terms of quality: We have certain [Stage 2’s] that have been around for a while. [...] Some other Stage 2’s are lacking competencies. This aggravates the problem of highly competent S1 centers being unwilling to forward cases to the next stages in the process.

**Product centers (S3):** This stage tackles with problems in product design. If a final solution can’t be found within a (given) short time frame, a workaround needs to be developed. However, S3 engineers don’t share the customer-oriented mindset, and thus “spend 5% of their time in support and 95% in making things work and trying to understand”, as one interviewee declared.

In addition, “A huge communication gap between [Stages 2 and 3] exists” and “[S1] and [S3] need to understand each other and learn how to interact which each other”. Again, delivering a solution to the customer is slowed down as S3 is not able to get the necessary knowledge from the stages that are earlier in the process.

**R&D (S4):** Engineers at Stage 4 develop preventive resolutions for the cases escalated to them. R&D has two main problems concerning the corrective software maintenance process: the engineers at that stage are not focusing on maintenance and they lack a customer-oriented mindset. One engineer explained us that “The most important task of the [S4] is to launch new products – maintenance is number two.” Another engineer stated that “R&D is not customer-oriented. R&D is driven by the deadlines of product launches”. In addition, one manager declared “the engineers have difficulties in writing documents that the customer can understand”.

**IS Support:** A new IS tool was implemented simultaneously with the M&A-related large-scale organizational change that took place a few years before carrying out this study. The purpose of the IS is to support the corrective software maintenance process. The managers are pleased with the new IS: “[...] we got this good tool, this is perfect. This brings the company together. One tool!” However, the engineers complain that “[the new IS] supports the processes but not the problem resolution”. By this they mean that the IS doesn’t contain the background knowledge needed. In particular, various interviewees shared the view that the information stored in the IS at one stage is not understandable for engineers at the subsequent stages. As a result, at each
stage, “...an engineer ... asks again the same questions from the customers.” (Head of Stage 2) Another manager clarified: “Sometimes I think we are losing in communication a lot. Between [stages] you will always lose something. There are always different mindsets between the [stages]”. In addition, some front-end engineers explained that before the organizational change, they could contact for example an R&D engineer they knew from previous cases. But, this know-where was broken when the new process and IS were implemented as the IS was to route the cases to the subsequent level according to criteria such as expertise and capacity. The engineers also claimed that for some problems, the long-standing relationships had been the key for fast resolution.

In addition, the case company has implemented an IS tool for knowledge reuse as some problem types appear frequently. However, very few solutions are keyed into this system. Some interviewees explained that engineers may not be willing to key in their innovative solutions into the IS because they see it as unproductive work from their personal perspective. In addition, one interviewee shared the following view with us: “... when you think of it from the viewpoint of an individual engineer, they have knowledge power. That is, you know that your own position in the organization is secured as long as you have something unique knowledge that others don’t have. Why would you then share that information with anybody else?”

Perhaps even more importantly, several interviewees held the view that the engineer’s identity lies at the heart of this problem: engineers don’t solve problems – or share knowledge on the solutions – to earn money for themselves or the company, but the value for them lies in possessing and developing excellent engineering skills. Some interviewees suggested that if the IS would offer a forum where an engineer could get peer recognition for a particularly good solution, engineers would be more willing to use it.

4. Discussion

The 4-staged process combines an integrated TMS within each stage with differentiated TMS across the stages. This design could allow efficiency within stages and innovative problem-solving across stages [Cf. 23]. But, Case X was not able to reap these benefits. As one interviewee puts it: “In paper [it] looks good, practically [it] doesn’t work.”

In this section we discuss our findings to show that many knowledge problems in our case organization and prior literature are linked to an ill-functioning TMS. The empirical findings are summarized in Table 1.

<table>
<thead>
<tr>
<th>TMS Manifestations</th>
<th>Problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specialisation of tasks</td>
<td>• Lack of collaboration (Case X)</td>
</tr>
<tr>
<td></td>
<td>• Insufficient communication with customer (Case X), [10]</td>
</tr>
<tr>
<td></td>
<td>• heterogeneous objectives and decision making (Case X)</td>
</tr>
<tr>
<td></td>
<td>• No clear ownership of the cases</td>
</tr>
<tr>
<td></td>
<td>• Initial case description is insufficient (Case X)</td>
</tr>
<tr>
<td></td>
<td>• Skill level too low or too high (Case X)</td>
</tr>
<tr>
<td></td>
<td>• No shared knowledge (Case X)</td>
</tr>
<tr>
<td></td>
<td>• Establishing a process [10]</td>
</tr>
<tr>
<td>Credibility of other group members</td>
<td>• Lack of trust (Case X)</td>
</tr>
<tr>
<td></td>
<td>• Knowledge is not perceived reliable (Case X)</td>
</tr>
<tr>
<td></td>
<td>• Communication gap (Case X), [10]</td>
</tr>
<tr>
<td></td>
<td>• Lack of customer-oriented mindset (Case X)</td>
</tr>
<tr>
<td>Coordination of actions</td>
<td>• Lack of shared goals (Case X)</td>
</tr>
<tr>
<td></td>
<td>• Lack of awareness (Case X)</td>
</tr>
<tr>
<td></td>
<td>• Performance indicators are not well-aligned (Case X)</td>
</tr>
<tr>
<td></td>
<td>• Big picture is not seen (Case X)</td>
</tr>
<tr>
<td></td>
<td>• Process is a black box (Case X)</td>
</tr>
<tr>
<td></td>
<td>• IS supports the process but not the problem resolution (Case X)</td>
</tr>
<tr>
<td></td>
<td>• Defining a process [10]</td>
</tr>
</tbody>
</table>

Table 1. Summary of Findings

As we focus on the knowledge problems in the process of corrective software maintenance, our findings and results don’t speak of many other relevant problems such as product quality, employee skills or firm-level organizational problems.

**Ill-functioning TMS in the software maintenance process.** The context of a multinational company under an organizational change seems to be particularly challenging for TMS. The implementation of maintenance process Stage S2 as a part of the post-M&A transformations caused a disruption in the existing TMS. To build it, staff transfers and recruitments were needed; a potential problem area also suggested by prior literature on TMS [20] and software maintenance [10][15]. As the stages lacked experience of working together, communication was at times minimal and misunderstandings frequent; a finding also in line with existing research [20].

We observed problems related to each of the key manifestations of TMS: specialization of tasks, credibility of other group members’ knowledge and
coordination of actions [13]. A key problem related to specialization of tasks can be observed between Stages 1 and 2. Many S1 engineers are knowledgeable also on S2 tasks; i.e., they can access the distributed team’s knowledge independently and hence are not dependent on teamwork to solve the problem. Prior literature on transactive memory has suggested that this could happen to couples [25] and our data extends this finding also to distributed teams. This directs S1’s focus away from their primary task: either solving problems by reuse of data or producing a problem description and forwarding them to S2. S2 engineers are formally dependent on S1 in getting the case descriptions. However, the way S1 encodes the knowledge to the IS is not sufficient and understandable to the subsequent stages of the process. Thus, S2 engineers contact the client directly instead of relying on S1 which is time-consuming and annoys the clients.

Expertise is in dispute at this point of the process, and knowledge seems to fall in the cracks and thus needs to be recreated at the subsequent level. This insight has been suggested by existing theoretical TMS literature on groups [23] and our research presents an empirical example of this concerning a case of global, distributed software work.

There is also a problem of lateral knowledge flow between S2 sites. The engineers at different sites possess knowledge on different solutions, but instead of integrating their experiential knowledge and possibly creating even better solutions, they remain in their geographical silos. This is a remarkable problem as integrative processes are among the most important transactive events in creating new knowledge for the group [23]. In our case, this translates into poor knowledge re-use; e.g. S1 not having enough template solutions to assist the client fast and efficiently. This lack of integration may be caused by the fact that to a certain extent, the different sites possess almost duplicate knowledge [Cf. 25] and therefore are not dependent on each other to solve any particular case, and thus fail to build this horizontal axis into the organization’s TMS.

Credibility problems are also evident in the case. The unbalance in specialization between S1 and S2 was aggravated by the fact that after quality problems during the implementation of the S2 departments, many S1 engineers remained unsure whether S2 engineers could actually solve the problems.

As already mentioned, the engineers further down the process stream did not find the knowledge produced by S1 credible and useful and thus turned to the client for additional knowledge. In some cases, S2 engineers expressed clearly their distrust in S1: “S1 people don’t really know what they are doing. They aren’t enough technical persons”, said one S2 engineer. When beliefs of this type become stereotypes – no matter whether they are justified or not – they start shaping the allocation of responsibility [23]; in this case in a harmful manner.

In a dynamic global software maintenance process, there are no small, stable groups that could develop and validate a shared understanding of who knows what through the experience of working together [3]. Prior literature has suggested that in dramatic situations such as catastrophe relief, credibility could be created through action [16]. In our empirical case, trust-creation was harmed by the initial performance problems of S2. Anecdotal evidence from informal discussions with key managers a few months after our data collection shows that when S2’s performance was brought to the desired level, credibility through action was built, too, and specialization between the two front-end stages improved.

Overall, the case process shows clear problems in coordination of actions. A key problem is that more complicated cases flow through each stage before being assigned to an engineer in S3 or S4, which results in a slow process and down-time to clients. A key problem here is the first contact person’s role (S1) to only record the problem, try to solve it by knowledge reuse and if this fails, forward it to S2. Instead, the first contact person should be a TMS expert that knows who knows what in the company. This way, the first contact point could assign the case directly to the stage that is able to solve it [23]. Traditional TMS theory suggests that this is based on a shared mental model of who knows what. In a large, distributed organization such as our case company, it is not possible for the front-end person to be familiar with every engineer individually. However, the front-end engineers can have a thorough understanding of the roles and groups within the company, and they could be able to forward the case directly to the group of engineers (e.g. process stage) that is capable of solving the problem. Besides outstanding TMS knowledge, such a front-end person – or perhaps a team - needs enough skills to determine the nature of the client’s problem. After this initial step, an internal front-end person in that group could forward the case to a specific engineer; this could possibly be (semi-) automated as it is in the case company.

**Partial IS support for TMS.** Besides examining TMS problems with regard to specialization, credibility and coordination, our case study illustrates that a computer-based information system can support TMS in software maintenance. This finding is in concordance with earlier contributions by [23][19][20] which suggest IS can play a key role in providing a basic platform for the TMS in such distributed collaborative work.
However, our findings show that an information system designed to fit and support a maintenance process design may lead only to a partially functioning TMS. The IS works well in providing a basic understanding of what cases exist and allocating them to engineers with suitable skills when forwarded to a new stage. However, it is not alone sufficient to perform critical TMS tasks in such a maintenance process: directing the case to the right group of experts (e.g., stage) as already described, and conveying adequately rich knowledge for actually solving the problem. Stages S2-S4 experience remarkable trouble with interpreting and using the knowledge produced by S1. Each of the process stages keys in information that is relevant to their specific focus on solving the problem and thus understandable in their context. But, this information is often either illegible or irrelevant for other stages. This is in concordance with earlier research [19][14], which suggests that building TMS supporting IS is likely to be difficult because of the contextual and tacit nature of the knowledge. This finding may also indicate that the TMS members do not have an understanding of how the pieces of knowledge from different sources fit together. In addition, the IS does not support knowing or finding out who might have the missing knowledge [Cf. 23].

Besides the problems related to what can be achieved by technology, IS support for TMS in software maintenance may also be hampered by members’ unwillingness to participate in TMS. One problem which occurs throughout the process is that members lack clear knowledge of where the knowledge resides. In particular, existing solutions for many frequently appearing customer problems are not available for re-use. This is because, the engineers are not willing to key the solutions into the IS as indicated by the process design, and the process design does not support other forms of know-where such as searching past cases to find the persons that solved them. In particular, the interviewees felt that the engineers’ identity, that values skilled problem-solving over smooth processes, was one of the key reasons for resisting the IS designed for knowledge re-use. This reflection on personal characteristics may be an interesting finding. This is because, contrary to the earlier studies that apply rational lenses, this finding provides an initial insight to the non-rational and somewhat postmodern side of TMS and software maintenance problems.

Based on this analysis, we suggest that in order for an organization to build a well-functioning TMS, it needs a combination of supporting IS and a work-flow structure with more human-centric strategies that allow conveying rich, contextual knowledge and also cross-functional collaboration where needed. One attempt towards this goal applied by the case company was job rotation aimed at giving the engineers a possibility to learn about other stages’ context. Before the organizational changes, many engineers had personal relationships with others from different stages, and thus they could dynamically form cross-functional teams for problem-solving. But, the organizational changes broke the existing TMS, and the new process and IS supported only partial construction of the new TMS. This brings us to an important insight: planning for a radical modification in a software maintenance process is clearly an exercise in planning a new form of TMS [Cf. 23]. One dimension a planner can think of is the balance between differentiation and integration. It should allow both specialization and bringing together the specialized knowledge to create better solutions for clients’ software problems.

5. Conclusion

This research takes a step towards understanding knowledge problems of corrective software maintenance. We study them from the perspective of transactive memory systems – a theory that speaks of how groups coordinate knowledge. Our case study concerns the corrective software maintenance of a multinational ICT company. The company has implemented a seemingly well-designed process yet it has severe problems in terms of customer satisfaction and internal performance indicators.

We discuss our empirical case from three key TMS manifestations: specialization, credibility and coordination. Our analysis shows that the key problems in the case process are linked to these. Thus, we suggest that TMS could offer a fruitful avenue for understanding and managing problems that lead to poor performance in such a process.

Our analysis makes a useful contribution to the literature in showing that whilst designing the process structure and support IS are essential components for well-running corrective software maintenance, the planners of such processes should also pay attention to a well-functioning TMS. It is also of importance to match the style of the TMS (integrated or differentiated) with the task. Thus, the planners should also include other, more human centric strategies to ensure that more sticky, tacit knowledge can flow to where it is needed.

Our findings contribute also to the TMS literature by showing that several insights developed for dyads and small groups could be important also in understanding the dynamics of large, distributed teams.

As our research focuses on the knowledge problems in corrective software maintenance, we suggest that future research addresses the influence of
other important problem areas such as how the characteristics of the firm, customers, personnel and products may trouble the corrective software maintenance process. A more detailed study of the role of computer-based information systems in supporting TMS as well as an in-depth scrutiny of the human-centric strategies are interesting topics for future research. Another limitation of our study is that we were only allowed access to the front-end but no direct access to the clients. We suggest that future research studies also the inter-organizational TMS including the client.

12. References


