User Knowledge Transformation through Design: A Historical Materialism Perspective

Yutaka Yamauchi

Kyoto University Graduate School of Management

Yoshida Hon-machi, Sakyo-ku, Kyoto 606-8501 JAPAN

yamauchi@gsm.kyoto-u.ac.jp

phone: +81 75-753-3536

Abstract

Through observation of an accounting system development project, this study examines how user knowledge of work, organization, and information system is transformed. The study employs the framework of historical materialism to explicate the interplay of knowledge and material conditions. The findings suggest that contradictions within the material condition, both in the resulting designs and in relations between users and developers, engender crises and trigger alteration of existing knowledge, and that new knowledge is created and validated through negotiating and specifying material designs. Knowledge transformation is shown to be historical in that knowledge is based on material condition at a certain moment and therefore is subject to change due to contradictions in the material condition. Therefore, often knowledge is transformed only partially as users and developers settle on a design to replicate existing practices with new features designed for different practices, producing contradictions for further transformation. The historical interplay of knowledge and material conditions paints a holistic picture of knowledge transformation through information system design.

Keywords: Knowledge transformation, materiality, historical materialism, contradictions

User Knowledge Transformation through Design:
A Historical Materialism Perspective

INTRODUCTION

In designing a new information system, users often have to transform their knowledge of existing work, organization, and technology; a new information system tends to contradict the previously established ways of working. Scholars have discussed how knowledge transformation is accomplished for both users and IT developers. This includes constructive conflict resolution to help overcome user resistance (Robey & Farrow, 1982; Robey, Farrow, & Franz, 1989). Barki and Hartwick (2001, p. 202) discussed the problem solving approach, wherein “a synthesis is sought, integrating all parties’ perspectives,” is critical to resolving conflicts. Open communication whereby participants can voice opinions freely is considered a key process in conflict resolution (Barki & Hartwick, 1994; Robey et al., 1989; Robey & Farrow, 1982; Salaway, 1987). An important aspect of this interaction is users’ reflections on their assumptions (Boland, 1978; Bostrom, 1989; Majchrzak, Lim, & Chin, 2005; Urquhart, 2001). In the participatory design tradition, transformation of user knowledge has been central. Most users are unfamiliar with the possibilities that new technologies offer (Ehn, 1993; Greenbaum & Kyng, 1991). The challenge is that learning these new possibilities involves “transcendence” (Greenbaum & Kyng, 1991, p. 277) of deeply ingrained knowledge.

This study applies the theory of knowledge transformation to information system design. Carlile (2002; 2004) offered a comprehensive theory of knowledge transformation with an emphasis on boundary objects. Although the present study does not address boundary spanning specifically but is about processes of information system design, the core thesis of knowledge transformation can be applied to information system design. Carlile wrote that transforming knowledge “refers to a process of altering current knowledge, creating new knowledge, and validating it within each function and collectively across functions” (Carlile, 2002, p. 445). He interpreted knowledge transformation as political because knowledge invested in practice is at stake; hard-won knowledge needs to be surrendered. When novelty arises in a situation, “the path-dependent nature of knowledge has negative effects” (Carlile, 2004, p. 557) and knowledge transformation is triggered. Therefore, knowledge transformation encompasses alteration of existing hard-won knowledge through political conflicts in contrast to simple learning.

He further explained that materiality played a critical role in transformation. Boundary objects, i.e., objects shared by involved parties across boundaries, help these persons understand and negotiate trade-offs as well as engage in collaborative design development of a system that functions for both parties involved. If one party makes a change in the object, the consequence of this change for the other party can be made visible. Parties then can negotiate using this object. Bechky (2003) additionally showed that different occupational members overcome mutual misunderstandings by transforming their understanding using boundary objects. In the information system design context, Levina and Vaast (2005, p. 335) discussed the manner in individuals become boundary spanners who produce locally useful boundary objects. These objects do not automatically lead to transformation. Levina (2005) demonstrated that diverse players in collaborations sought control by responding to an object that others had produced. This insight that material objects play a key role in creation and validation of new knowledge is important in the information system context.

While theories of knowledge transformation provide clear explanation of how knowledge is transformed once the transformation is initiated by novelty, it remains unclear how knowledge transformation unfolds throughout the design process. Particularly, the relationship between design and knowledge transformation remains to be explored. While we know that novelty triggers knowledge transformation, we know less when and how the novelty arises. We need to examine what conditions within information system design make novelty arise. Furthermore, boundary objects are used to negotiate and resolve conflicts, but often conflicts probably cannot reach a finite resolution. In such a case, how is design determined without conflicts being resolved? The relationship of design and knowledge transformation is complicated because even if knowledge is transformed, a workable design may not be derived, and even if a design is derived, knowledge may not be completely transformed. We do not know how these two factors unfold in interaction with each other, thus a holistic framework is needed to understand knowledge transformation in information system design.

Historical materialism, a Marxian theory of applying dialectics to concrete, material society, offers a framework to explicate these aspects in a holistic manner. This perspective helps elucidate how knowledge transformation proceeds in interaction with transformation of material conditions (Marx, 1992; Marx & Engels, 1976). Specifically, the historical and dialectical perspective suggests that material conditions and knowledge transformation are reflexively related. From this, it becomes clear that knowledge transformation cannot take place all at once; knowledge is tied to the material conditions in which contradictions remain and trigger further transformation of knowledge. Power relations and conflicts are also intertwined to the material conditions and thereby subject to change as material conditions change. The historical perspective offers a holistic picture of knowledge transformation in information system design.

This study analyzes how users transform knowledge and design information systems. For this purpose, the author investigated an accounting information system development project; real-time interactions between users and developers were observed and recorded throughout the design phase. Although the analysis includes developers' as well as users' knowledge transformation, the study puts more weight on user knowledge. The challenge for the IT developers was to understand the work of the particular accountants and design a new information system that would help them. IT developers have "a wider view of business processes than users because they could look beyond a particular division to see its operations in relation to other divisions" (Pawlowski & Robey, 2004, p. 659). In contrast, the challenge for users is to unlearn a particular way they see accounting work and the use of an information system and to explore new possibilities for leveraging IT for their work.

Subsequently, the concept of historical materialism is addressed; in particular its application to information system design is illustrated. After describing research design and methodology, the case is summarized and detailed accounts of specific designs are reported. The subsequent discussion explores the theoretical significance and implications of the revealed historical material processes in light of this analysis. Finally, the concluding remarks are provided.

HISTORICAL MATERIALISM FOR KNOWLEDGE TRANSFORMATION IN INFORMATION SYSTEM DESIGN

Historical Materialism

In this study the Marxian theory of historical materialism is selectively appropriated to explain a specific case of information system design. Karl Marx developed the concept of historical materialism as a way to bring Hegelian dialectical theory into the materialist domain. While Hegel discussed ideational change processes through interactions of dialectical forces, Marx tried to understand the dialectical social change whose basis is material economic processes. Because this domain of information system design is far removed from what Marx sought to understand, we must appropriate the fundamental theoretical perspective in historical materialism rather than be constrained by the specific theoretical details. Today a number of problems in Marx's framework and later theoretical developments have been clarified; thus we do not need to rely on such concepts as the labor theory of value or the union of theory and practice. Nonetheless, this study makes an effort to be faithful to the core perspective of the theory.

Many prior studies have used a dialectical framework to explain organizational change (Benson, 1977; Bresser & Bishop, 1983; Carr, 2000; Seo & Creed, 2002; Van de Ven & Poole, 1995). Based on "a logic of opposition" (Robey & Boudreau, 1999, p. 168), the dialectical perspective explains organizational change as a dynamic interplay of opposing forces. It has been extensively applied in information system development, which is largely political and full of conflict (Bjerknes, 1991; Cho, Mathiassen, & Robey, 2006; Mathiassen, 1998; Robey, Ross, & Boudreau, 2002). In particular, Robey et al. (2002) used the dialectical theory to explain how ingrained knowledge opposes new knowledge associated with a novel system. Carlo, Lyytinen, and Boland (2012, p. 1104) showed that an organization can be mindful, as opposed to mindless, by "orchestrating and balancing deliberate interactions between opposing poles—like control versus autonomy, mindful versus mindless, and global integration versus local variation—across organization and over time." This study proposes a theoretical framework rooted in historical materialism, where materiality is emphasized as well as dialectics (Swedberg, 2008).

A few words are needed to explain how this notion of material condition is related to recent debates on materiality. In using materiality, Marx emphasized everyday economic activities for producing means of subsistence. Marxian material conditions encompass not only forces of production, e.g., technologies, but also relations of production and labor processes; his interest was not in the enduring material objects (Leonardi, 2012). Furthermore, in this study, one important materiality is IT design. The design is a concrete technological object, however in the middle of the designing process, the design does not yet exist as concrete material object, i.e., an information system. The design exists only on paper (materiality of the documents are not analyzed) although it is constrained by material reality and cannot be changed at will. In view of this, the current study does not focus on user actual interactions with concrete material objects. Nonetheless, implications of the resulting theoretical insights could be discussed in relation to the recent debates on materiality.

The theoretical framework of historical materialism is shown in Figure 1. The material condition is mapped in the lower half and the idealist domain, i.e., knowledge, in the upper half. This figure portrays tandem change in knowledge and material condition; knowledge is constituted through material design, and contradictions in the material condition then lead to a crisis and engender alteration of existing knowledge. In the following, this historical materialism is explicated.

Insert Figure 1 around here.

Designing as Interplay of Knowledge and Material Condition

A key insight of Marxian historical materialism is the reversal of Hegelian idealism; social consciousness is shaped by the material conditions. Within the material conditions, Marx particularly focused on modes of production. Modes of production are not specific to manufacturing; they refer to the way in which people "produce their means of subsistence" (Marx & Engels, 1976, p. 37). This subsistence is about elements of everyday human life that that are not abstract but material, and the subsistence is more than simple physical survival on a minimum of food and drink; the mode of production is "a definite form of expressing their life, a definite mode of life on their part" (p. 37). Marx and Engels continued, "As individuals express their life, so they are. What they are, therefore, coincides with their production, both with what they produce and with how they produce. Hence what individuals are depends on the material conditions of their production" (p. 37). Furthermore, "consciousness can never be anything else than conscious being, and the being of men is their actual life-process" (p. 42). The actual life-process rooted in the material condition is emphasized.

In the context of this study, the term knowledge is used in place of consciousness. Knowledge that users and IT developers have about a particular sphere of work, the organization of the workplace, and the information system used is created and validated in relation to the mode of production. The mode of production in this context is the mode by which users and IT developers design the users' work processes and the information system needed to do the work. It encompasses the forces of production (production technologies) and relations of production (ownership of means of production and division of labor). In this study, forces of production include not only design tools and methodology but also the base system that is applied to this client—a base system that had been developed for another organization. The relations of production are based on a contractual relationship between the customer and the IT vendor and entail division of labor between users in the customer organization and the IT vendor's developers. Produced results (commodities) also constitute the material condition; for Marx, commodity had a defining characteristic in society and for the present study, the produced results, i.e., IT designs, specify users' new work processes and thereby shape and are shaped by knowledge regarding the work.

In the figure, the arrow labeled "designing" indicates that by designing the material reality of the work, IT developers come to attain certain knowledge of the work, the organization, and the world. As users and IT developers successfully specify a design, they validate their knowledge. The relationship between knowledge and design is reflexive. We are influenced by the material conditions but we also change and create the material conditions as well. Marx and Engels (1976) wrote, "Men, developing their material production and their material intercourse, alter, along with this their actual world, also their thinking and the products of their thinking" (p. 42). Therefore, the arrow labeled "designing" in Figure 1 is bi-directional.

Designing is then bound to the knowledge at a particular moment in history. This is why Marx (2008, p. 9) wrote, "Man makes his own history, but he does not make it out of the whole cloth; he does not make it out of conditions chosen by himself, but out of such as he finds close at hand. The tradition of all past generations weighs like an all upon the brain of the living." This nuanced characterization of the reflexive relationship between knowledge and material conditions is key when we seek to understand the design process of a new information system. Material conditions cannot be created optimally; they are tied to the historical moment and change only gradually. New knowledge is created and validated in relation to the material conditions, which in turn are designed based on existing knowledge. Therefore, some designs may be devised while knowledge is largely unchanged and many contradictions in the material conditions, as described below, remain.

Contradictions, Conflicts and Crises

Historical materialism suggests that contradictions inherent in material conditions are drivers of social change. Contradictions consist of two opposing forces and the struggles between them (Benson, 1977; Carlo et al., 2012). For instance, Marx (1992) showed that the more the capitalist mode of production succeeds, the more inner contradictions within the mode of production deteriorate. Contradictions are not logical aporia that can somehow be overcome in idea, as portrayed in Hegelian idealism (Hegel, 1977). Contradictions are rooted in material conditions and therefore cannot be revolved once for all (Althusser & Balibar, 1997). Althusser claimed that there are a number of inter-related contradictions. He proposed the notion of "overdetermination" to suggest that it is not a single contradiction that directly determines social change, but rather the structure of the inter-related contradictions have this effect. Therefore, no material condition can be expected to be free of contradictions. In a recent study, Carlo, Lyytinen and Boland (2012, p. 1104) also found that contradictions do not disappear through sublation or supersession (*Aufhebung*):

> Sometimes, the negation can result in something new—a novel synthesis, as in a classic Hegelian dialectic with thesis, antithesis, and synthesis, moving toward an ideal state. We did not see this pattern in our study. Instead, we saw a continuous interplay of contradictions creating ever-new conditions, which open up new possibilities, but do not necessarily move toward a final synthesis.

Although their study does not take the Marxian perspective, the insight is important. To this insight, we need to add historical perspective. We similarly cannot expect that knowledge is transformed in one fell swoop through idealistic resolution. We need to investigate knowledge transformation where new knowledge is only partially transformed and some, if not most, contradictions remain. Materialism helps us to be sensitive to the difficulty of knowledge transformation rooted in the material condition.

In the context of this study, one key contradiction is two opposing logics of the users' work. While users want to create particular work practices, a new system's underlying logic may be contradictory to the work practices. For instance, new systems are designed to integrate data in a centralized manner; but users may desire to have flexible data management at the local level to make their work easier. Contradictions also appear in the way users and developers work together. In particular, the power relation between these two parties can be contradictory. Users may think that they can simply demand certain features and expect developers to create suitable designs that realize these features. This working relationship is often a result of an economic arrangement in which the users' organization is a client for the developers' organization; a client has a certain power to dictate the design. This kind of one-way communication is contradictory in that designing requires a greater level of active participation to derive workable designs. Less participation tends to result in inferior designs and user dissatisfaction.

Contradictions engender conflicts. The opposing logics mentioned above reflect the assumptions of the different groups; users hold one logic and developers another. The economic relation of users and developers also produces conflicts in this interaction. Users have resources, i.e., access to budget and knowledge on the accounting work, and hold power over the IT developers they hire to develop their system. The developers on the other hand hold expertise in designing and implementing the system on which the users rely. A higher-level manager who controls budgets and authorization may exert power over both users and developers. Therefore, designing entails negotiations. Some compromised designs may result while many contradictions remain unresolved.

In discussing conflict, we should be careful not to attribute these conflicts to intrinsic interests of these groups and portray conflicts as political struggles to advance interests. Conflicts stem from contradictions in the material conditions and as much as material conditions change, power relations change. This is why Marx (1992), finally, in *Capital* began his work with a structural analysis of the capitalist mode of production as opposed to an earlier emphasis of emancipation of exploited laborers (Althusser & Balibar, 1997). Therefore, we need to trace changes in power relations in interaction with changes in knowledge and material conditions.

In historical materialism, contradictions are drivers of social change as they surface during crises. Marx's theory was oriented to praxis of overcoming crisis by revealing contradictions and the mechanism by which the society is led to a crisis—his "critique" and "crisis" reflect the same thing (Habermas, 1988, pp. 212-219). Marx tries to show that the material conditions people take to be givens are in fact results of their own production and therefore can be changed—altering their knowledge. In the context of this study, we maintain the crises that emerge out of contradictions trigger alternation of knowledge, although we do not need to privilege intellectual theory for guiding praxis (Habermas, 1988; Jay, 1984). This alteration of knowledge is then a condition for the revolutionary praxis to alter the material conditions. "Revolutionary" only means that the material conditions, the basis of the society, are transformed; thus it is not necessarily a disruptive societal change. In information system design, crises often surface in the form of development costs. For instance, when two logics are unresolved and compromise into a design, the design needs to fulfill two contradictory goals and thereby requires more development work. The estimated development cost then likely significantly exceeds the budget.

Summary

Through the application of historical materialism, I seek to advance our understanding of knowledge transformation in information system design. The historical nature of knowledge transformation is explicated. Knowledge is not transformed altogether at one point in time. Often, new knowledge is created and validated as a new material condition is specified, but as the material condition still contains many contradictions. These contradictions emerge as crises, typically in the form of budget overrun, and trigger alteration of existing knowledge. Knowledge, material design and power relations are reflexively intertwined and the relationships among these factors are historical, i.e., subject to change. This also means that resulting designs always embody contradictions. For instance, some contradictions may be resolved but others may remain, making the design only a partial solution that could be altered even further. Contradictions then are drivers of knowledge transformation and information system design.

METHOD

An ethnographic method was employed to elucidate individuals' actions, and their knowledge as exhibited by these actions. To understand what users know and how they alter this knowledge, knowledge cannot be predefined. An individual's knowledge tends to be unclear even to the individual, particularly when it becomes a taken-for-granted reality. Thus, we cannot simply ask them about what they know. To track transformation, I followed the same team over the course of the design phase.

As part of a collaborative research project between my (author) team and an IT vendor, I had the opportunity to observe the upstream design activities of a system development project. System development projects involve the gathering of confidential or proprietary data related to the client. Few clients are willing to accept a researcher who will study the details of such projects. Therefore, empirical analyses of recorded user–developer conversations in naturalistic, industrial settings are rare. This project took place in Japan and all communication was in Japanese; the data presented below were translated into English.

A client hired an IT vendor to replace an existing accounting system. The IT vendor proposed implementing an innovative system that had been developed for another company. Therefore, the system already existed when the project began. Nonetheless, as shown in many package implementation projects, it is not easy for the organization to adjust to the given system. Much work is needed to alter the features of the system as well as the work practices of the organization. The term "design" includes the modification and construction of not only the system features but also the work and organizational practices in which the features are used. Such design is often as substantial as custom system development for which the work may require less alteration. Furthermore, unlike the implementation of a full-feature package solution that can be customized using parameters, the system deployed in this case needed substantial modification.

I studied the accounting team responsible for the subsystem related to the accounting department. This team comprised five developers from the IT vendor, four user representatives from the client's accounting department, and a member of the client's information system (IS) department (Table 1). The developers had one team leader (hereinafter "lead developer"). The junior accountant was regarded as knowledgeable about most areas of accounting work. The manager of the accounting department (accounting manager) also occasionally participated. One IS member worked with the team and participated in some of the meetings; he was not trained as a system developer but was quite knowledgeable about accounting systems. He liaised between the client and the IT vendor. The client's senior executive, who approved the project, also played an important role; he intervened when it became apparent that the project was not proceeding as planned (specifically, when the estimated development cost exceeded the budget).

Insert Table 1 around here

Developers did not use a specific design method (e.g., object-oriented design, rapid prototyping, or agile), but used general tools such as workflow diagrams, fit/gap analysis, and a type of function point cost estimation. In many cases, users and developers interacted in meetings that were observed and recorded. During meetings, users and developers communicated using printed documents or projections on a screen, in addition to drawings on a whiteboard. I obtained copies of these documents. Interactions between users and developers also occurred outside of meetings, particularly immediately before and after meetings, (e.g., in elevators and hallways). I captured these interactions, although audio recording was not always possible. Social activities, such as after-hours parties with users and developers, were also observed.

The first basic design phase spanned three months, during which the author observed and recorded audio between users and developers at 30 meetings, each lasting between 30 minutes to 4 hours and 10 minutes, with an average duration of 1 hour and 33 minutes. The developers alone attended thirteen meetings. I began participating in the middle of the basic design phase, in the fifth week. As the participants spent time setting up the infrastructure and specifying processes the design would be based on, the actual design began only one week prior to my observation. To capture what happened during the previous week, an audio recording of the meetings by participants themselves was provided, along with copies of the documents used in those meetings.

The analyses proceeded by examining the recorded interactions and collected documents. The data were organized by grouping interactions according to a particular feature or set of features (e.g., “one-time payment” and “payment slip approval”). The first part of the analysis was dedicated to understanding the discussion of the users and developers; this was important because of the technical nature of the accounting work. Audio recordings were reviewed repeatedly; to elucidate what participants were saying, the recordings were transcribed and analyzed along with the documents used by the participants. I presented many questions to participants, and consulted many accounting textbooks. The second part was devoted to making theoretical sense of the data. As I began to understand the data, it became clear that the way in which users altered their knowledge and gained new knowledge about their work was critical. They initially insisted on certain features based on existing knowledge of the work but later reconsidered these features once an alternative way of doing the work was understood. This led me to focus on knowledge transformation, largely inspired by Carlile (2004). The analysis began with a description of how knowledge transformation unfolded, based on paying attention to the details being discussed between users and developers. Soon, however, I realized that Carlile’s model, which was used to explain boundary spanning and not information system design, could not account for the whole process over which knowledge was only gradually transformed. The dialectical framework appeared to be useful to explain this process. At the same time, materialism was needed to account for this dynamic because material constraints were salient. Eventually, I settled on Marxian historical materialism as a way to frame the analysis by clarifying how knowledge, material conditions and power relations were related.

I now describe the design of four major features to explain how the users transformed their knowledge. Features are used as units because developers and users organized their meetings according to features; each meeting was divided into a series of segments focusing on one specific feature. Several features were observed throughout, although some data for other features were missing. Not all the meetings were observed. Often, a design was determined through informal communications that took place outside of meetings. In such cases, participants were interviewed to help understand developments. Among the features that were thoroughly observed, four were chosen to portray the extent to which users transformed their knowledge. One feature was determined and implemented with little transformation and another was determined and implemented with some transformation. Two features were chosen to show more thorough transformations. For other features, in which some data was missing, fragmentary data are reported.

FINDINGS

Case Background

The existing system used by the client ran on a mainframe computer with text-based terminals. The new system was built using client–server architecture with a graphical front end. Therefore, the two systems' underlying design concepts were markedly different. The main users of the old system were accounting personnel, but the new system was designed for use throughout the company. In the old system, employees filled out slips and forms, and sent them to the accounting department, which the accounting department staff then entered in the system. In the new system, all employees directly input their information into the system.

The project commenced after the client's senior executive approved the project and allocated the budget. However, end users who participated in the design discussions were unaware of the details in the RFP and the vendor's proposal. When talking to several users, developers were surprised that users demanded features not included in the RFP while the vendor estimated the development cost based on the RFP, and the project budget was further negotiated down from the estimate, thinning the vendor's profit margin. The vendor agreed to the contract based on the calculation that if users accepted their design proposals the development cost could be maintained at a reasonable level. Nonetheless, the users did not even know how much time they would need to spend on system development. The developers convinced the accounting department manager that at least one user representative ("junior accountant") spends most of her time on system development; she did not expect to participate in the actual design activities.

The timeline is presented in Figure 2. At the top are the rough processes that the project followed. At the bottom are the specific meetings described below. During this three-month phase, the basic design was refined into more detailed technical and programming specifications including data types, class structures, exception handling, and communication protocols. Users were less involved in these technical designs. In this sense, the project employed a typical waterfall process, through which design is refined gradually into technical details.

Insert Figure 2 around here

The summary of the findings is shown in Table 2. The processes of knowledge transformation are summarized in three broad steps although actual processes were highly complicated. Initially, users rejected developers' initial design proposals. As the project started, the developers walked the users through the new system's typical usage with screen capture images and workflow diagrams. At this time, the users pointed out several issues with the new system as they perceived them. The users then explained the features they wanted by showing their existing workflow using screen captures and the existing user manual. The developers learned users' work processes and system requirements in detail. Users' feature requests mostly coincided with the existing system's familiar features. When the developers proposed alternative designs that required little modification to the new system, the users refused them. At this time, users did not consider IT design to be part of their job and simply insisted on certain requirements they expected the IT developers to implement.

Insert Table 2 around here

The users and developers then scheduled a series of meetings to discuss each issue in detail. Developers repeatedly explained that the features they were requesting would incur considerable cost to implement and proposed alternative designs. This discussion reached a deadlock. Then, the junior accountant and other users began to explore compromises. Beyond demanding requirements, the users started to actively participate in concrete design. Users, however, sought to work around the new system's constraints and still replicate existing work processes, as described below in detail. As a result, the estimated system development cost remained significantly above the client's budget.

This situation was reported to the steering committee, which comprised the client's executives and the vendor's general managers, in the eighth week. The client's senior executive considered the situation to be problematic and intervened by pressuring the users to reduce the cost. Both the users and the developers then began reconsidering many of the requirements. Users themselves proposed significant changes to work processes and radically innovative features to support the new work processes. Eventually, the design was finalized. The final designs still contained some contradictions. Yet, the cost estimate was reduced to a reasonable range. The project was then approved to move into the next phase: detailed technical design.

Feature 1: Dropdown Comments

One particular requirement was related to the data fields on several types of accounting slips, such as payment slips. This discussion started in the fourth week and concluded in the seventh. In the existing system, slips contained a text comment field for each dropdown list on the slip. If someone purchased a personal computer with an optional memory module and a software program, then the purchase slip contained three dropdown lists: one for the computer, one for the memory module, and one for the program. The accountants demanded a comment field for each of the dropdown lists (hereinafter “dropdown comments”). The new system included only one comment field for the entire slip and not for each dropdown list.

The developers then began working on a proposal that would be acceptable to the users. Developers attempted to align the solutions with the system’s built-in functions so that limited modification would be necessary. In the seventh week (Meeting 2), the developers explained their proposal for working around dropdown comments, which was nothing more than using the comment field for the entire slip. Nevertheless, users considered the gap to be a system inadequacy and rejected the proposed design, insisting on their original requirement. The junior accountant said, “No way. Please add it [the feature]. If they buy each with its own product code, it might work. But they won’t.” Eventually, developers gave in and agreed to modify the system to accommodate this requirement. These strong reactions on the part of the junior accountant did not allow the developers to propose an alternative. As we can see from the manner in which the junior accountant spoke, users basically demanded a feature and expected the IT developers to implement it. The contractual relationship whereby the users’ organization paid the IT developers to implement the system was critical; from the users' point of view, the designers were hired professionals doing a job, and the users could simply give input to be implemented in the design. The developers had to acquiesce as a result. Yet, the budget determined in the contract was based on the IT developers' assumption that users would be more flexible in accepting changes to their work processes. This would lead to a contradiction between what the user wanted and the client's budget. In this case, however, the design was finalized with the contradiction unresolved.

In this example, the user's knowledge was fixed despite her being aware of the investment made to develop a new information system. Her mode of design was simply a replication of the same features as those in the existing system. On the other hand, the developers thought that the users' work processes should be altered to use the system's built-in features as much as possible. The primary contradiction was that the new system was based on the assumption that all employees would use the accounting system and take responsibility for the data they entered, while the existing system had been used only by accountants. Accountants had verified and modified accounting slips sent from employees, correcting errors in journal titles using dropdown comments. In principle, however, real-time accounting required that accountants had increased trust in the data from other departments, in addition to performing less extensive verification. Otherwise, the data would become buffered in accounting and would not be processed in real time. The point is not to suggest which accounting policy is better. Instead, the fact that the users did not entertain possibilities other than the features they were using indicates the taken-for-granted characteristics of their knowledge.

Feature 2: Key

Although dropdown comments were eventually added to the system at high cost, other features were developed further. The users had used a unique identification number, called a key, to locate accounting slips. They could type the key into the system if they needed to refer to a slip. They used the key to open a particular accounting slip and modify the data, such as journal entries, before posting it to the general ledger (GL). Users soon found that the new system did not offer the same identification numbers.

The users and developers began by mapping the familiar work practices and the new system (Meeting 3a). Later, the developers created a design proposal. In the seventh week, the developers explained the new system's numbering process and suggested that the journal entry number was probably the closest to the key (Meeting 3b). Yet, when the discussion continued, they realized that the journal entry number was issued only when the entry was posted to the GL at the end of the workflow. Therefore, the journal entry number was issued too late and could not be printed on the documents that accountants wanted to use to verify slips before posting them to the GL. Once the problem was identified, the developers considered using the "process number" instead. However, they found cases in which the same process number could be attached to multiple slips. Furthermore, the process number lacked digits that represented certain information, such as the payment type, which was required to group the slips. Although they sought a solution, they came to a dead end. The junior accountant stepped back and reiterated the necessity of the key.

```
JA: To realize the key, what can we use? With the key we can
know the type of the slip, when it was issued. [omitted] We
can know from the slip type whether it is purchasing, lease,
salary, and so forth. [omitted]
LD: In that sense, we need to create a new number. The numbers
we currently have will not satisfy your request.
```

The next day, the lead developer indicated to the junior accountant that they would create the key in the new system, at an added cost. Here, we can see, as in the previous case, that the users imposed their requirements and expected IT developers to create a design satisfying these requirements.

The discussion, nevertheless, continued. In the eighth week, the client's senior executive, who learned that the project had greatly exceeded the budget, instructed the users to reconsider their requirements to reduce development costs. This prompted the junior accountant to consider radical changes and induced the accounting manager, who had ceased participating in meetings, to return to the discussion. In the ninth week, the accountants began to reconsider the key (Meeting 3c). The junior accountant proposed ideas for combining the process number with other data fields to avoid duplication and to limit multiple payments during the same month. The accounting manager proposed a high-level policy change of having other departments group slips together according to payment types, eliminating the problem of losing digits in the number representing the payment type information. Eventually, these ideas constituted a solution that worked around the issues. They decided to use a combination of the process number and other data fields as a substitute for the key.

This example shows that users became quite active in design activities, more flexible with their requirement of the key, and willing to accept certain changes. Initially, the users rejected the developer's proposal, leading to a contradiction regarding the user's requests being over their budget, resulting in pressure for the senior executive to reduce costs. The senior executive's intervention constituted a crisis suddenly altering the relationship between the users and the IT developers. This event forced the users to reconsider their requirements, altering existing knowledge. Based on the developers' proposal, the user found a compromise by combining features. The relationship shifted from a one-way request to collaboration. The accounting manager was self-reflective; during a meeting he said, "We need to think about a way to do without the number." Nonetheless, the users aimed for something they were familiar with by seeking an equivalent to the key. The mode of design was to replicate the same work process with minimum modifications to the system. Developers became satisfied by the fact that users reached a compromise.

The remaining contradiction was that the new base system was designed such that users did not have to consider the numbers, while accountants wanted to use numbers to manage slips. The numbers in the new base system were complicated because they were used only for internal design rather than being shown to users. The accountants had been using the key because of the character-based terminal of the mainframe system; they did not use a mouse to click on slips, but typed in a code to access slips. This contradiction was recognized and discussed explicitly. The lead developer commented to the user, "In a so-called mainframe system [omitted] you basically type in the code. In new systems developed today, you can type in the code but in most cases you can choose among predefined options. Maybe you don't mistype, but the whole policy of the new system is to eliminate codes." The junior accountant responded, "We need to see all the journal entries…We know which accounting title to choose but it is difficult to select. There are too many of them. It depends on the person but most of us remember the code." The junior accountant refused to reconsider existing work practices. Consequently, a number based on the process number was implemented at an additional cost.

Feature 3: Journal Table Sheet

Accountants wanted to be able to print the table with all the slip data (hereinafter “journal table sheet”) at certain intervals so that they could quickly review all the slips to check for incorrect information. The new system was incapable of printing this table. The developers proposed a work process that did not use a printed table. The users, however, repeatedly clarified that this table was necessary; for instance, in the fifth week (Meeting 4a), the junior accountant commented, “I don’t know when we print the table, but in any case we need the function to print it for the person approving slips.” The junior accountant’s insistence is another example of rejection. The feature she demanded would result in a higher development cost. As in the previous cases, the negotiation here was unilateral; a user demanded a certain feature and refused to consider any alternative.

The accountants considered the journal table sheets necessary because they wanted to verify and correct all accounting slips in the accounting department. This feature was connected to dropdown comments and the key, which were also required for verification. Moreover, related to this feature was the feature that allowed the accounting department to correct slips. With the new system, a slip that contained an error was to be returned to the person who submitted it. Adding a feature to allow the accounting department to correct the errors was not simply a matter of giving the accounting department access to the screen for correcting entries. The user interface needed to be altered because desirable information (the name of the person who submitted the slip) was not displayed, whereas undesirable information (the person’s personal bank account) was, because the screen was intended for use only by the person that submitted the slip, not by accountants.

In the eighth week, after the senior executive’s mandate to reduce development costs, the junior accountant said that they had changed their minds after some internal discussions, and were now willing to disregard this requirement (Meeting 4b). However, she immediately said that they wanted to print out another list instead by stating, “Actually, now we are thinking of changing the policy. We are considering giving up the journal table sheets. Then, we had the cover sheet, right? We wonder how much we can tune up the cover sheet." She proposed repurposing the cover sheet, which was printed for a different purpose. She further explained the rationale behind this:

> Basically we cannot check the journaling on the display so give up…We will make [a sheet] come out daily, listing all that have been approved or all entries of that day. That way, we can prevent [wrong entries from remaining in GL]. Checking afterwards as opposed to checking beforehand using the journal table sheets.

Although users had been verifying and modifying slips before posting them to the GL, they here proposed checking the slips after the entries were posted to the GL, fixing any incorrect data at that point. A minor modification would enable the system to print the cover sheet. This was a major policy change because the users initially disliked any incorrect data in the GL. For accountants, maintaining clean GL was important; they could not accept the idea that errors were inserted into the GL. The compromise was therefore significant, as earlier in the process the users had no method to accept such a compromise.

The junior accountant's knowledge was, however, not completely transformed. In a subsequent discussion, she suggested including a number of fields in the list (cover sheets) so that she could examine the journal entries in detail. She said, "…we [need many fields] if the assumption is that we check slips." However, the accounting manager counter-argued, saying, "If we do not approve slips in the accounting department, we don't need that many fields." The accounting manager attempted to change the work practice such that the accountants would verify the slips on screen, keeping verification to a minimum. He also said, "Because the journal entry is approved at each department, the applicant should be responsible for entering the data correctly. Then we just look at the data on the system." He suggested that the responsibility for accounting data be shifted to the departments. The junior accountant disagreed stating, "Whether we do extensive verification or not, if we want to be ready to do minimum verification, this much data should be printed." As a result of the junior accountant's insistence, they finally decided to implement the detailed list for after-the-fact verification. The junior accountant wanted to keep the previous method used to verify and correct data in GL.

The argument is not a criticism of the junior accountant for not furthering the transformation; she had a legitimate reason to defend her perspective. As an accountant, getting the numbers right is essential (Boland, 1999). She made sufficiently radical changes to the requirements. The users' central requirement had been the need for verification to keep the GL data clean; to alter the GL, new modification slips would have to be issued. They no longer sought to replicate the same work, but began to construct a new work process. The relation between users and developers changed. While the user previously considered the design proposed by developers, she led the design with her own design proposal, which developers then examined and elaborated upon. Thus, the relationship between users and IT developers was reversed.

The design was creative so far as the user pioneered the idea of using a feature that was designed for a different purpose. If a design like this had not been created, an agreeable design could not have been developed; i.e., the design materialized and validated new knowledge. At the same time, because the modification to add extra fields to the cover sheets was not expensive, the design and knowledge were settled without further transformation.

Feature 4: Special Payment Function

We can consider another example to explain this pattern. In the fourth week (Meeting 1), users requested a special payment function that deviated from the regular workflow. Although regular payments were automatically tied to fund transfers, this special payment function allowed manual fund transfers. The new base system did not allow this—ordinary payment functions were tied to fund transfer. Although the new base system had a built in fund-transfer system, the users needed to use their own transfer system due to software license constraints. Eventually they would build their own fund transfer system, but at the moment they needed to find a way to do without it. The users wanted to make a large set of fund transfers using a different fund transfer system and to enter the journal entries at a later time with this special payment function. A more extensive discussion was held in the ninth week, after the senior executive's intervention (Meeting 5). Developers proposed using a completely different function to realize this payment. This function satisfied users' requirements but had a few drawbacks. For instance, this function did not record the payment in the “payment” category and did not include the date field to record the payment date. The users refused this design.

When the discussion reached a dead end, the accounting manager proposed revisiting some of the constraints. He suggested adopting the distributed accounting policy: Other departments, not the accounting department, could enter payment data. This shift significantly altered the constraints on design. The users wanted this special payment function because they needed to enter a number of payment data at one time. If the accounting department did not have to enter a huge volume of data on behalf of other departments, they would not need this function. This policy change was significant because it meant that accountants would no longer take direct responsibility for the data but instead instruct other departments on proper methods to handle their own data. A change of this type would have been impossible to consider in earlier phases.

An issue persisted because payment functions were still automatically tied to fund transfers. The junior accountant proposed using a payment function called a “payment form,” which was used to handle paper payment forms and would, therefore, not involve fund transfers. That is to say, they would just trash the generated paper forms. She said, “If we pay with a payment form, this may be a stupid idea, then outstanding payment is marked. However, because it does not involve money transfer, the payment is not done.” This idea was to creatively repurpose the existing feature.

A problem still existed as to how to reconcile journal entries with actual payments. Unless reconciled, the journal entries showed outstanding payments in the ledger. Furthermore, multiple transfers needed to be aggregated and reconciled with a single journal entry. However, a feature to do so did not exist. Again, the junior accountant proposed a solution to these issues, suggesting a grouping of entries by the “budget center number” from which the payment was made. Even if multiple transfers were made, they originated from a single budget center; it was easy to search for payments of the same budget center and reconcile them. The developers agreed that the junior accountant’s idea was viable and started writing the specification for this feature. Here, we see that the user proposed an innovative, concrete design and IT developers only refined this design. We should note, however, that this whole design was a compromise due to the technical constraint of software licensing.

Summary of the Findings

These findings help to specify the historical, dialectical process of user knowledge transformation. The findings are summarized in Figure 3. The users initially resisted the idea of altering system features and work processes. Users maintained their work practices and demanded those features with which they were familiar. There was a lack of participation and delegation of design to the developers. Developers, on the other hand, sought to lead the users to accept as many features of the new system as possible. Contradictions were apparent. Users sought to construct the same work with the same system features, while the new system was based on a different model of work altogether. To implement the same set of features, many modifications were necessary.

Insert Figure 3 around here

The contradictions manifested themselves as crises when IT developers explained that it was too costly to accommodate users' requests and the discussion reached the dead-end. The users then recognized the need to alter their existing knowledge and actively examined and extended the IT developers' design proposal. The analysis revealed, however, that the junior accountant maintained much of her existing knowledge, and continued to take work processes for granted. She sought to use the compromised features to accomplish the same work and to replicate the features that were needed for existing work processes. Although the users and developers were able to tweak the new system to replicate the existing work, the work that users wanted to design and the new system's design basis were contradictory. Although the users participated in design by examining and modifying the design proposals by developers, the developers' proposals were constrained by the requirements that users had previously demanded. Modifications were necessary and the resulting cost exceeded the budget.

The contradiction then led to a crisis: the client's senior executive gave a top–down directive to maintain the cost within budget. This crisis urged users to consider altering what they took for granted. The junior accountant came to propose creative designs. IT developers could not produce such innovative designs themselves because they could not alter or abandon many of the constraints initially imposed by the user. The IT developers sought an appropriate solution within most of these constraints. Once the user began to explore alternative work processes, she made radical changes to the constraints she had initially deemed necessary. The resulting design was, however, not free of contradiction. The design was fixed as the estimated cost was brought to a low enough level, although it remained above the initial budget.

DISCUSSION

This study applied knowledge transformation theory to information system design. An important focus of this study was to examine the way in which knowledge transformation relates to the entire design process. In particular, the relationship between design and knowledge has remained unclear; therefore the current study sought to offer a holistic view of materiality and knowledge drawing on historical materialism. Empirical analysis revealed first that knowledge transformation is driven by contradictions stemming from material conditions, and second that new knowledge is created and validated through designing in which material designs are conceived, negotiated, and detailed in conjunction with knowledge. In this view, knowledge, material condition, and power relations are all reflexively related and information system design unfolds through the interactions among these factors. From this we gain a historical perspective that knowledge, material conditions, and power relations that operate at one point in time are in fact products of previous practices and subject to change. Contradictions inherent in material conditions are drivers of this change. In this section, we discuss each of these elements in light of the empirical findings.

Materiality as a Source of Contradictions

Knowledge transformation is inherently tied to material conditions in that knowledge transformation is a process driven by contradictions stemming from the material environment. In the current study, there were contradictions both in resulting designs and in the working relationships between users and developers. Resulting designs embodied the existing model of work while the new system was based on a completely new work model. One feature was only a surface manifestation of the underlying logic. For instance, key and dropdown comments first appeared to be trivial features that could be worked around with small tweaks. Yet, they were tied to the underlying logic of centralized accounting where accountants take responsibility for journal entries. Because contradictions such as this are not a single gap in materiality but a systemic opposition of different logics, we need to examine materiality as a complex network of contradictions rather than as independent contradictions (Althusser & Balibar, 1997).

The previously existing system had been based on centralized accounting whereby accountants entered and controlled the accounting data, and the new base system was based on distributed accounting whereby users in other departments entered the data. The new base system was also premised to process accounting data in real-time; the data entered by departments are automatically committed to the ledger so that managers can know the accounting situation in real-time. On the other hand, the previously existing work processes had operated in a batch-mode so that the data were accumulated in the accounting department and approved before being committed to a ledger. Therefore, designing system features for batch accounting with the new system resulted in a contradiction. It would be still possible to implement this, although the cost would be higher than previously expected. Therefore the contradictions manifested themselves in the form of high development costs. Conflicts were inevitable when users and developers operated with these opposing logics.

Furthermore, the relationship between users and developers was contradictory. Users could demand features they believed to be necessary and expected developers to implement these features, although they would be costly to implement and exceed the agreed upon budget. In contract negotiations, the vendor reduced the quote based on the assumption that there would be a minimum of customer requests. Although users generally do not consider designing to be part of their job, without active user participation, no design that entails a reimagining of user work is possible. These contradictions engender conflicts in the interactions between users and developers. In the case examined in the current study, although users initially controlled the critical resource, i.e., money for hiring the IT vendor, the IT developers' expertise and knowledge of IT design became critical in the design process. Then, the users while maintaining their initial power, found they had to compromise and rely on the IT developers in order to receive a system with the desired features.. Subsequently, the relationship changed from one based on unilateral user demands to one in which the users participated in design. Furthermore, the senior executive who authorized and controlled the budget became involved due to the contradiction, i.e., budget overrun. His involvement changed the power relationship and triggered further knowledge transformation on the part of the users.

Therefore, the conflicts stemmed not from intrinsic interests of groups but from contradictions in the material condition. This study showed that even when external individuals intervene, the source of that event lies with the contradiction in the material condition. As Althusser (1997) discussed, while contradictions in abstract ideas can often be quickly dialectically sublated and resolved, contradictions in material conditions are complexly inter-related and difficult to disentangle. This is similar to what Carlo, Lyytinen and Boland (Carlo et al., 2012) documented, e.g., constant interplay without synthesis. To this, the current study adds that system design is also a historical process that moves the transformation of knowledge and material conditions, and contradictions then manifest themselves as crises. This finding therefore validates Carlile's (2002; 2004) view that novelty triggers knowledge transformation. Moreover, materiality can be a source of conflicts (Engeström, 1987; Nicolini, Mengis, & Swan, 2012) in addition to being a tool for resolving conflicts, e.g., boundary objects for facilitating communication across boundaries.

Knowledge Creation and Validation through Material Designing

In addition to altering existing knowledge, materiality is central to the creation and validation of new knowledge. Even if users know that they need to revise their existing knowledge, they cannot create and validate new knowledge unless they arrive at a satisfactory material design. Even if a suitable idea is proposed, a myriad of details need to be worked out. In the current study, the junior accountant and the accounting manager sometimes reverted to the existing work model even after they had begun to break out from it. For instance, the journal entry number appeared to be a good option for replacing the key in all respects except for timing: the number would be issued too late. As they were unable to find an alternative design, users and IT developers had no choice to revert back to the original work and system model. In another case, the users adopted the real-time accounting practice of automatically committing the data from other departments to the ledger. Yet, one user wanted to maintain the practice of verifying and correcting data on printed paper.

Even if existing knowledge is altered, new knowledge cannot suddenly be created; it needs to be created and validated in relation to the material design. While knowledge transformation, e.g., revisiting assumptions and constructively solving problems (Boland, 1978; Bostrom, 1989; Majchrzak et al., 2005; Urquhart, 2001), has been considered a prerequisite of design; material design is a constitutive part of that very knowledge transformation. In the design process, both knowledge and materiality develop together. Taking this viewpoint, we should not assume it a logical necessity that all contradictions be resolved. Faced with contradictions, users and developers seek to settle on a feasible design. Yet, there is no guarantee that such a new design can be created. Resulting designs that are reached at the end of the design phase are not free of contradictions. In some cases, e.g., dropdown comments, a design was determined while the key contradiction remained; the existing model of work persisted while the feature was not consistent with the adopted new model.

On the other hand, it is equally possible that adept users and developers can come up with a creative design that resolves a contradiction so quickly or easily that transformation does not unfold further. The above analysis showed that users and developers settled on designs that only worked around contradictions rather than overcoming them. The junior accountant's idea of using a cover sheet in place of the journal table sheets still contained the elements of existing work practices verifying slips at the accounting department although she was willing to alter the policy of keeping the GL clean. No further transformation occurred after this design was set because adding fields to the cover sheets happened to cost little and developers were satisfied with the compromise. Therefore, we need to acknowledge that when knowledge is transformed, much existing knowledge may remain while a new material condition is successfully specified. Studies have documented creative strategies on the part of users, such as workarounds and tweaks (Gasser, 1986; Orlikowski, 1993). In addition to celebrating these practices, we should also examine the larger historical process in which these practices may suppress rather than to resolve contradictions.

Historical Processes of Knowledge Transformation

This study describes the historical view of knowledge transformation, which Carlile (2004) outlined as an iterative process of transfer, translation, and transformation, in the context of IT design.

Historical materialism emphasizes that a given reality is only a reality at that moment and is in fact the result of the work done by people involved. We were struck by the fact that features users had adamantly insisted on were sometimes completely changed in a later phase. Many of the constraints that users took for granted as a solid reality were eventually abandoned. Yet, it is a mistake to view this change as having happened through simple learning on the part of users. The knowledge transformation was a non-linear process in which crises occasioned alteration of knowledge that entailed conflicts. At various moments in the process, the users and designers negotiated and settled on a certain design and specific knowledge associated with that design. For them, such a negotiated design might have appeared to be appropriate, but often they later realized that it did not work due to contradictions in the new design framework.

This renewed model implies that it may not be appropriate to talk about knowledge transformation as the term transformation gives an impression that knowledge is transformed conclusively. At best, transformation is only for the moment; design is bound to one’s knowledge, which, in turn, is bound to a material condition and a previous design. That is to say, knowledge transformation and design are historically related. As knowledge is transformed, design is transformed. In fact, a design that is set at the end of a design phase should also be considered as historical. The design contained contradictions. The historical perspective is required to understand a particular design.

Knowledge, material conditions and power relations are reflexively tied. To view IT design as a political struggle is insufficient because knowledge and material conditions are also involved. For the same reason, we should not view IT design as a process of knowledge transformation without power and materiality. As much as knowledge is altered in relation to the power relation of the particular time and material design is done within a certain relationship, knowledge and material conditions are reflexively shaped in relation to the power relations. While Marx is seen to have privileged material conditions as the final cause of the social change, we should maintain that knowledge, material conditions, and power relations are reflexively tied and form a structural whole (Althusser & Balibar, 1997; Jameson, 2013). This dialectical development of knowledge, material design, and power relations unfolds gradually through iterations. Existing knowledge cannot simply be negated because it is rooted in a particular and momentary material condition. Furthermore, because creation and validation of new knowledge is tied to materiality, which cannot be shaped at will, the change is not completely free from existing material order.

Therefore, it is not appropriate to discuss power relations in isolation. Instead, we should examine power relations at one moment in the historical development through which they are tied to a particular material condition and a certain knowledge set. In the current study, the initial unilateral relation is conditioned by the users' knowledge, which was limited to the existing work processes and system, and the contractual relation was part of the material condition that led to users' a priori power over IT developers. Power relations changed not simply because of political struggle but largely because of changes in material conditions and knowledge. Although it could be considered that the IT developers gradually gained power by means of some political maneuvering, their power was mediated by the material condition. At first, they could only attempt to persuade the users to reconsider design constraints; such persuasion alone did not work. The relationship changed as users and developers overcame the deadlock that resulted from the previous power relationship, and the users allowed greater flexibility regarding alternative designs and took a slightly more active role in the process. The eventual power relationship was based on a significant overhaul of the existing knowledge triggered by the crisis-like event stemming from contradictions in the previous material condition. Therefore, contradictions in material conditions were both sources of conflicts and occasions for change in power relations.

Finally, prior studies have tried to explain how organizations change in relation to material technologies. Structuration theory has been applied to explain how technologies and practices are intertwined (Barley, 1986; DeSanctis, 1994; Orlikowski, 1992). This steam of research has advanced our understanding of materiality and its relation to practice. We should be careful, however, because the notion of material condition, which encompasses relations, technologies, processes, and outcomes of production, includes both human and material agencies. Nonetheless, there are some important implications that merit discussion. Recent debates on materiality emphasize material agency; what materiality does and does not do to make things happen. This stream of research, however, tends to put more weight on inter-relations as well as fusion and reconciliation of materiality and practice than on contradictions between these factors. Orlikowski's notion of technology-in-practice (Orlikowski, 2000) and her recent notion of sociomateriality (Orlikowski & Scott, 2008) tend to focus on fusion of materiality in practice. Leonardi's (2011; 2012) discussion of imbrication of human and material agencies is more nuanced and focuses on both the separation and inter-relatedness of materiality and practice. The reconciliation of human and material agencies can be driven by contradictions. Therefore, it is helpful to view imbrication being as contradictory as it is reconciliatory. Although Leonardi (2011) emphasized that prior imbrications become "black-boxed" and "transparent," this study shows that such imbrications are not necessarily closed; human and material agencies cannot be completely reconciled even if the outcome is black-boxed. The remaining contradictions then lead to further imbrications. This view does not obviously contradict these previous discussions of materiality; rather it makes materiality even more central and strengthens these prior arguments.

CONCLUSION

This study investigated how knowledge is transformed in information system design. The historical interplay of knowledge and material conditions was empirically described and theorized based on historical materialism. The key findings include that contradictions in material conditions engender crises, which then produce conflicts and trigger knowledge transformation, that new knowledge is created and validated in interaction with materiality but contradictions cannot be eradicated, and that knowledge, material design, and power relations are reflexively settled but are always subject to change due to contradictions. All these findings are integrated into the framework of historical materialism. These findings clarify how knowledge transformation and information system design are related.

Questions unanswered by this study include how knowledge transformation would proceed even after a design was set and the information was implemented. This later phase could not be observed. We can expect that once an information system is implemented, it would not be as easy to change as it was in the design phase. Yet, we know that use of a technology is not determined by the technology itself and quite flexible. The way in which a proposed model is applied in the use phase is a question to explore. In particular, how contradictions inherent in the resulting material condition would be understood and worked out merits further examination. This study had an opportunity to examine design processes that were relatively confined in a particular accounting department. No interaction was observed across department boundaries, e.g., interactions between the accounting department and other departments. If such interactions were involved, the process of knowledge transformation would be much more complex. Future research can examine how this study's findings could be upheld or modified in such intricate situations.

REFERENCES

Althusser, L., & Balibar, É. (1997). *Reading Capital*. (Ben Brewster, Trans.). Brooklyn: Verso.

Barki, H., & Hartwick, J. (1994). User participation, conflict, and conflict resolution: The mediating roles of influence. *Information Systems Research*, *5*(4), 422–438.

Barki, H., & Hartwick, J. (2001). Interpersonal conflict and its management in information system development. *MIS Quarterly*, *25*(2), 195–228.

Barley, S. R. (1986). Technology as an occasion for structuring: Evidence from observations of CT scanners and the social order of radiology departments. *Administrative Science Quarterly, 31*(1), 78–108.

Bechky, B. (2003). Sharing meaning across occupational communities: The transformation of understanding on a production floor. *Organization Science, 14*(3), 312–330.

Benson, J. K. (1977). Organizations: A dialectical view. *Administrative Science Quarterly*, *22*(1), 1–21.

Bjerknes, G. (1991). Dialectical reflection in information systems development. *Scandinavian Journal of Information Systems*, *3*(1), 55–77.

Boland, R. (1978). The process and product of system design. *Management Science*, *24*(9), 887–898.

Boland, R. (1999). Accounting as a representational craft: Lessons for research on information systems. In W. Currie & B. Galliers (Eds.), *Rethinking Management Information Systems : an Interdisciplinary Perspective: An Interdiscipdelinary Perspective* (pp. 229–244). Oxford: Oxford University Press.

Bostrom, R. (1989). Successful application of communication techniques to improve the systems development process. *Information and Management*, *16*, 279–295.

Bresser, R. K., & Bishop, R. C. (1983). Dysfunctional effects of formal planning: Two theoretical explanations. *Academy of Management Review*, *8*(4), 588–599.

Carlile, P. (2002). A pragmatic view of knowledge and boundaries: Boundary objects in new product development. *Organization Science*, *13*(4), 442–455.

Carlile, P. (2004). Transferring, translating, and transforming: An integrative framework for managing knowledge across boundaries. *Organization Science*, *15*(5), 555–568.

Carlo, J. L., Lyytinen, K., & Boland, R. J. (2012). Dialectics of collective minding: Contradictory appropriations of information technology in a high-risk project. *MIS Quarterly*.

Carr, A. (2000). Critical theory and the management of change in organizations. *Journal of Organizational Change Management, 13*(3), 208–220.

Cho, S., Mathiassen, L., & Robey, D. (2006). Dialectics of resilience: A multi-level analysis of a telehealth innovation. *Journal of Information Technology, 22*(1), 24–35.

DeSanctis, G. (1994). Capturing the complexity in advanced technology use: Adaptive structuration theory. *Organization Science*, *5*(2), 121–147.

Ehn, P. (1993). Scandinavian design: On participation and skill. In D. Schuler & A. Namioka (Eds.), *Participatory Design: Principles and Practices*. Hillsdale, NJ: Erlbaum.

Engeström, Y. (1987). *Learning by Expanding. An Activity-Theoretical Approach to Developmental Research*. Orienta-Konsultit Oy.

Gasser, L. (1986). The integration of computing and routine work. *ACM Transactions on Office Information Systems*, *4*(3), 205–225.

Greenbaum, J. M., & Kyng, M. (1991). Epilogue: Design by doing. In *Design at Work* (pp. 269–279). New York: CRC.

Habermas, J. (1988). *Theory and Practice*. Boston: Beacon Press.

Hegel, G. W. F. (1977). *Phenomenology of Spirit*. (J. N. Findlay & A. V. Miller, Trans.). Oxford University Press, USA.

Jameson, F. (2013). *The Political Unconscious*. New York: Routledge.

Jay, M. (1984). *Marxism and Totality*. Berkeley: University of California Press.

Leonardi, P. M. (2011). When flexible routines meet flexible technologies: Affordance, constraint, and the imbrication of human and material agencies. *MIS Quarterly*, *35*(1), 147–167.

Leonardi, P. M. (2012). Materiality, sociomateriality, and socio-technical systems: What do these terms mean? How are they related? Do we need them? In P. M. Leonardi, B. A. Nardi, & J. Kallinikos (Eds.), *Materiality and Organizing: Social Interaction in a Technological World* (pp. 25–48). Oxford: Oxford University Press.

Levina, N. (2005). Collaborating on multiparty information systems development projects: A collective reflection-in-action view. *Information Systems Research*, *16*(2), 109–130.

Levina, N., & Vaast, E. (2005). The emergence of boundary spanning competence in practice: Implications for implementation and use of information systems. *MIS Quarterly*, *29*(2), 335–363.

Majchrzak, A., Lim, R., & Chin, W. (2005). Managing client dialogues during information systems design to facilitate client learning. *MIS Quarterly*, *29*(4), 653–2672.

Marx, K. (1992). *Capital: A Critique of Political Economy. Volume One*. (Ben Fowkes, Trans.) (Reprint. Vol. 1). New York: Penguine Books.

Marx, K. (2008). *The Eighteenth Brumaire of Louis Bonaparte*. Rockville, MD: Serenity Publishers.

Marx, K., & Engels, F. (1976). *The German Ideology*. Amherst, NY: Prometheus Books.

Mathiassen, L. (1998). Reflective systems development. *Scandinavian Journal of Information Systems*, *10*(1&2), 67–118.

Nicolini, D., Mengis, J., & Swan, J. (2012). Understanding the role of objects in cross-disciplinary collaboration. *Organization Science*, *23*(3), 612–629.

Orlikowski, W. (1992). The duality of technology: Rethinking the concept of technology in organizations. *Organization Science*, *3*(3), 398–427.

Orlikowski, W. (2000). Using technology and constituting structures: A practice lens for studying technology in organizations. *Organization Science*, *11*(4), 404–428.

Orlikowski, W. J. (1993). CASE tools as organizational change: Investigating incremental and radical changes in systems development. *MIS Quarterly*, *17*(3), 309–340.

Orlikowski, W. J., & Scott, S. V. (2008). Sociomateriality: Challenging the separation of technology, work and organization. *The Academy of Management Annals*, *2*(1), 433–474.

Pawlowski, S., & Robey, D. (2004). Bridging user organizations: Knowledge brokering and the work of information technology professionals. *MIS Quarterly*, *28*(4), 645–672.

Robey, D., & Boudreau, M.-C. (1999). Accounting for the contradictory organizational consequences of information technology: Theoretical directions and methodological implications. *Information Systems Research*, *10*(2), 167–185.

Robey, D., & Farrow, D. (1982). User involvement in information system development: A conflict model and empirical test. *Management Science*, *28*(1), 73–85.

Robey, D., Farrow, D., & Franz, C. (1989). Group process and conflict in system development. *Management Science*, *35*(10), 1172–1191.

Robey, D., Ross, J., & Boudreau, M.-C. (2002). Learning to implement enterprise systems: An exploratory study of the dialectics of change. *Journal of Management Information Systems*, *19*(1), 17–46.

Salaway, G. (1987). An organizational learning approach to information systems development. *MIS Quarterly*, *11*(2), 244.

Seo, M.-G., & Creed, W. D. (2002). Institutional contradictions, praxis, and institutional change: A dialectical perspective. *Academy of Management Review*, 222–247.

Swedberg, R. (2008). The centrality of materiality: Economic theorizing from Xenophon to home economics and beyond. In T. J. Pinch & R. Swedberg (Eds.), *Living in a Material World* (pp. 57–87). Cambridge, MA: The MIT Press.

Urquhart, C. (2001). Analysts and clients in organisational contexts: a conversational perspective. *The Journal of Strategic Information Systems*, *10*(3), 243–262.

Van de Ven, A. H., & Poole, M. S. (1995). Explaining development and change in organizations. *Academy of Management Review*, 510–540.

Table 1: Study Participants

Participants	Organization	Roles
Junior accountant (JA)	Client	The main representative from the accounting department; participated in all the meetings with developers.
Accounting manager (AM)	Client	The accounting department manager; participated in several meetings initially and after the seventh week.
Accountants 1, 2 (A1, A2)	Client	Accounting department members; participated in only two meetings when their expertise was needed.
Senior executive	Client	The manager who had initially approved the project and intervened in the middle of the project.
Information system department member (IS)	Client	A liaison from the information system department; participated in almost all meetings.
Lead developer (LD)	IT Vendor	Leader of the design team; participated in all meetings.
Developers 1, 2, 3, and 4 (D1, D2, D3, and D4)	IT Vendor	Developers working under LD; specialized in a subarea of the system; participated only in the meetings on the subarea.

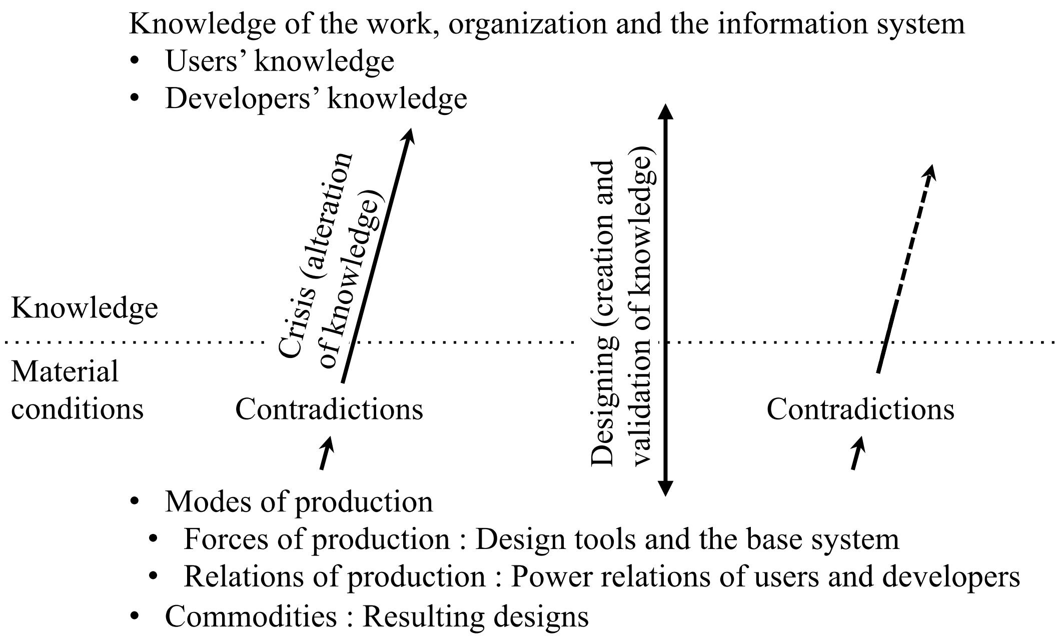

Knowledge of the work, organization and the information system
• Users' knowledge
• Developers' knowledge
Crisis (alteration of knowledge)
Designing (creation and validation of knowledge)
Knowledge
Material conditions
Contradictions
Contradictions
• Modes of production
• Forces of production : Design tools and the base system
• Relations of production : Power relations of users and developers
• Commodities : Resulting designs

Figure 1: Historical materialism process of knowledge transformation

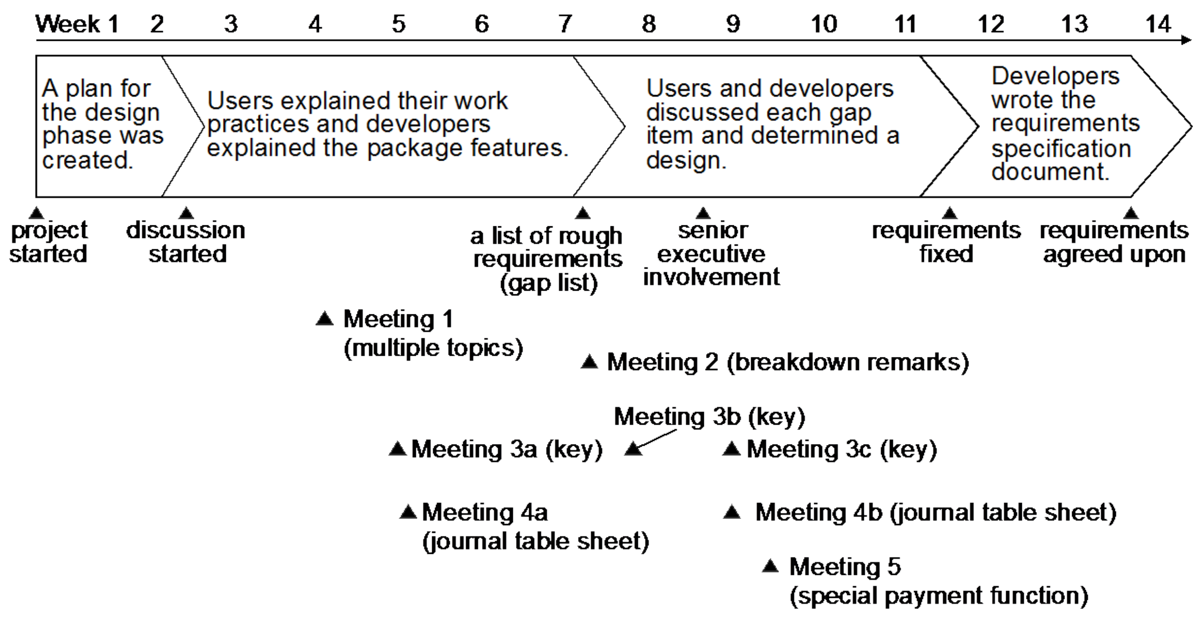

Week 1
2
3
4
5
6
7
8
9
10
11
12
13
14
A plan for the design phase was created.
Users explained their work practices and developers explained the package features.
Users and developers discussed each gap item and determined a design.
Developers wrote the requirements specification document.
project started
discussion started
a list of rough requirements (gap list)
senior executive involvement
requirements fixed
requirements agreed upon
Meeting 1 (multiple topics)
Meeting 2 (breakdown remarks)
Meeting 3b (key)
Meeting 3a (key)
Meeting 3c (key)
Meeting 4a (journal table sheet)
Meeting 4b (journal table sheet)
Meeting 5 (special payment function)

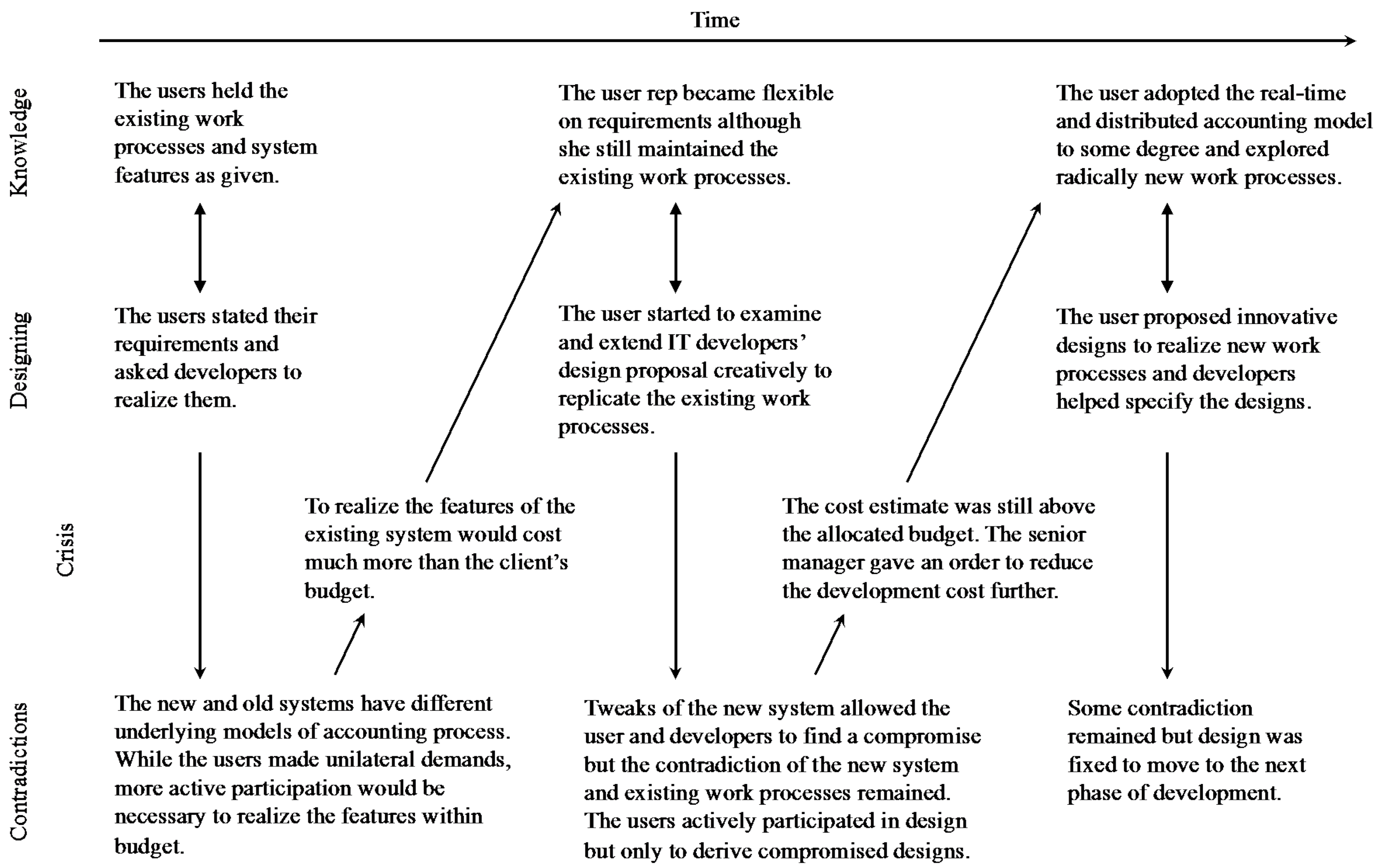
Figure 2: Case timeline

Table 2: Summary of findings

Features	Initial interactions: Users rejected the new system's features	Subsequent interactions: Users explored workarounds.	Further interactions: Users explored a new model of work.
Dropdown Comments	User Knowledge: Users in the accounting department needed dropdown comments to verify accounting slips. The junior accountant thought that they "cannot do without" a comment field for each dropdown. Developer Knowledge: Developers thought that users could do the work only with the overall comment field. Designing: The accountant insisted that a comment field be added—"No way. Please add it." Developers agreed to add the fields.		
Key	User Knowledge: The users organized slips using a unique ID called the key. The key contained various information items indicating what department issued the slip, when it was issued, and so on. Developer Knowledge: Developers thought about mapping the key with the numbers in the new system to realize the same work. Designing: The accountant demanded the key, asking "To realize the key, what can we use?"	User Knowledge: The users needed features to perform the same work practices, such as storing paper invoices and receipts and communicating with other departments about specific slips. Developer Knowledge: Developers learned several difficulties with the new system's features for the users and explored compromises. Designing: A developer proposed combining multiple variables to simulate the key, and the accountant explored and settled on the idea.	
Journal table sheet	User Knowledge: The users needed printed journal table sheets to verify and approve all the slips quickly. Developer Knowledge: When developers learned about this requirement, they thought users would not need the sheets if they changed the work process. Designing: When the users demanded the journal table sheets,	User Knowledge: The users maintained the same need for the printed journal table sheets. Developer Knowledge: Developers could not accept the requirement due to the budget constraint but had no viable solution. Designing: The developers proposed "paperless"	User Knowledge: The user abandoned the intensive verification at the accounting department and proposed the idea of modifying the data after journal entries are directly committed to the GL. Developer Knowledge: Developers agreed to the user's proposal as a feasible solution. Designing:

	the developers postponed the discussion.	journal table sheets, but the junior accountant rejected this: “We need the function to print it.”	After the senior executive’s directive to reduce development costs, the users proposed dropping the feature and proposed an alternative feature, which they refined with developers.
Special payment function	User Knowledge: The users wanted to enter payment data while handling the fund transfer with a different system. Developer Knowledge: Developers learned about this special way of making payments, which the new system did not support. Designing: The developers suggested that the new system did not have the same payment function and set aside time for detailed discussions.	User Knowledge: The users wanted to enter a large amount of data at one time. Developer Knowledge: Given the users’ requirements, developers sought a function of the new system that they could use instead. Designing: The developers proposed using a different function. The users identified a problem with the proposal.	User Knowledge: The users realized that each department could take responsibility for entering its own data—no need for batch import at accounting department. Developer Knowledge: Developers found the user’s proposal as technically feasible. Designing: The user proposed using an unused feature of the new system to enter data. Although this idea raised another problem, she also found a way to overcome the problem using the new system’s features. The developers helped refine the idea.
Fragmentary evidence	Accounting approval The users modified and approved the accounting slips by looking through a list of slips. The new system required each slip to be sent back to the person who issued it. Users repeatedly suggested that they would need to modify the data: “Oh, we cannot modify it. It is difficult, then.” Batch approval The accounting department wanted batch approval to approve a number of slips at the same time; however, the new system did not have this feature. Users suggested, “We need the batch approval at least.” Tax withholding Users wanted to list all the taxes withheld so that they could be paid later. The users rejected the features of the new system, which required users to process one entry at a time: “The feature is meaningless if you can only add data one by one.” Duplicate payments Users wanted to be notified when two payments	Accounting approval Because the users insisted that they needed to be able to modify data in the accounting department, the developers proposed tweaking the system to allow it, but then it would no longer be possible to send the slips back to the people who issued them. Tax withholding An accountant and an IS member suggested using a separate data warehouse system for managing tax data. Users determined that the data reported by the data warehouse system were inappropriate. Cash management While the new system required the accounting department to close cashbooks every day, the accounting department did not want to do the work every day. Developers proposed modification to allow them to close the books less frequently than every day, which the users accepted.	Accounting approval Users agreed that they would send the slips back to those who submitted them when there were errors. Users suggested that they needed to “educate the other departments” to “take responsibility for their data.” Cash payments to employees Users decided to get rid of workflow that involved cash payment to employees (e.g., travel advance) and instead sought features to achieve an alternative work process, including transferring money in a shorter cycle and handling the unused amount. The users proposed an alteration of the work to drop the cash operation.

	were made mistakenly for a single invoice. Users suggested that if no notification could be issued automatically, “It is difficult to use the features [of the new system].” Error check for budget overrun Users wanted a fine-grained error check to monitor budget overrun. Developers suggested that it was difficult to implement the check unless the users specified the detailed logical rules to implement.	Cash flow statements Users produced a separate cash flow statement at the end of each fiscal quarter, apart from those specific to each month. The new system produced accumulated statements every month. The developers proposed extracting monthly data into Excel files with a small modification and using the existing cash flow statements for fiscal quarters. The users agreed.	



Time
Knowledge
Designing
Crisis
Contradictions
The users held the existing work processes and system features as given.
The users stated their requirements and asked developers to realize them.
The new and old systems have different underlying models of accounting process. While the users made unilateral demands, more active participation would be necessary to realize the features within budget.
To realize the features of the existing system would cost much more than the client's budget.
The user rep became flexible on requirements although she still maintained the existing work processes.
The user started to examine and extend IT developers' design proposal creatively to replicate the existing work processes.
Tweaks of the new system allowed the user and developers to find a compromise but the contradiction of the new system and existing work processes remained. The users actively participated in design but only to derive compromised designs.
The cost estimate was still above the allocated budget. The senior manager gave an order to reduce the development cost further.
The user adopted the real-time and distributed accounting model to some degree and explored radically new work processes.
The user proposed innovative designs to realize new work processes and developers helped specify the designs.
Some contradiction remained but design was fixed to move to the next phase of development.

Figure 3: Summary of the dialectical process