ABSTRACT
In information system development, end-users often participate in design and in many cases learn to design their own system. Design, however, requires a distinct approach that users typically are not familiar with. The unique position in which users find themselves makes users’ learning even more complicated: They have no interest in designing and becoming designers. To understand how users learn to design despite such difficulties, longitudinal ethnography was conducted in an accounting system development project. The analysis reveals that users progressively acquired the design capability in two months. In the beginning, the user treated the problem as given and rejected design proposed by designers that did not solve the given problems. The user then gradually learned the details of the system and started to explore various configurations of features; however, this design focused on parts, with the problem remaining fixed. The user finally demonstrated her design capability by constructing creative designs both in parts and on the whole. Drawing on situated learning theory, the notion of the peripheral designer is used to illustrate this type of design practice. The user became a peripheral designer in that she could design in a competent but peripheral manner without becoming a full designer. Power relationships posed challenges to this learning. The notion of the peripheral designer clarifies what designing is, beyond active participation in discussion and configuration of parts, and delineates a realistic picture of how a user learns to design in the real world.

Categories and Subject Descriptors
H.5.2 [Information Interfaces and Presentation]: User Interfaces – user centered design; K.6.1 [Computing Milieux]: Project and People Management – Systems analysis and design

General Terms
Design, Human Factors, Management

Keywords
User participation, design learning, design skills, design practices, situated learning, periphery

1. INTRODUCTION
Research on information systems, as well as that on human-computer interaction and participatory design, has advocated end-users’ participation in the design process. Participation is, however, not a straightforward phenomenon [19] and can mean many different things [12, 15, 18, 32]. Users can simply participate in design review meetings whether or not they have an actual say in the design [1]. Users can also make decisions to select or approve the design that designers have created. Users can also take control of the design process [21, 22] and form high-level policies while somebody else develops the actual design. There is yet another crucial form of participation: Users themselves can design [27, 45]. Although users’ participation in design has been widely discussed, it is not clear specifically how users design and how they learn to do so.

Design is an epistemologically distinct activity that requires unique skills that typically differ from the skills that users need everyday [48]. Studies suggest that there are certain orientations—designerly ways of knowing [8]—people need to acquire. This is not something that can be taught systematically in a classroom; learning through participation in actual design is required [37]. In the case of system users, the situation is more complicated. Unlike typical design education, users have no intention to become designers. They often have no idea that they even need to learn to design. Furthermore, users undertaking design need to learn within a short time, typically in a few months.

To understand how users learn to design, I draw on situated learning theory [24, 49] that emphasizes participative learning through actual engagement in practice. Particularly, the access and the power relationships that situated learning requires are relevant to system design in peculiar ways. The analysis reveals the peripheral nature of users’ design practices. In a short period of time, a user learns to design competently without acquiring full knowledge of technical design skills. Power relationships make this learning particularly problematic.

2. LEARNING TO DESIGN
2.1 Design Skills
Design practice has been studied extensively in design studies [8, 26, 35, 38, 40]. In this paper, design refers not only to creation of a technical artifact but also to the redefinition of work processes, roles and responsibilities, policies and other organizational configurations. Information systems (IS) research has a long history of using the design approach. Design research in IS
stresses the design science nature of research itself, rather than the nature of design [48]. Researchers discuss epistemological differences between behavioral research and design research [14, 16, 17, 30]. Some researchers have tried to develop prescriptive theories to guide design of a particular class of systems [31, 47]. Organizational theorists have also discussed design modes of research inquiry [4, 34, 42, 43]. On the other hand, a few researchers have discussed actual design practices. Yoo et al. [50], for instance, described how the architect Frank Gehry and his colleagues designed buildings. Stolterman [41] borrowed concepts from design research to discuss information system design. As an engineering discipline, Human-Computer Interaction has adopted design as its core theme [10, 28].

Concerned with invention and creation of artifacts, design is epistemologically distinct. Design as both practice and research is often contrasted with natural, social, and behavioral sciences that are concerned with explanation and discovery of given phenomena [6, 13, 30, 34, 43]. Buchanan [6, 24] suggested that “The subject matter of design is radically indeterminate.” The subject matter is not given to designers; they create it. As a result, “The essential nature of design calls for both the process and the results of designing to be open to debate and disagreement” [6]. This contrast between design and science can be carried over to the contrast between the design of an information system and its use. Boland and Collopy [4] contrasted the “design attitude” with the “decision attitude.” They suggest that current education emphasizes the decision attitude so that managers can make a right decision among given alternatives. Typically, users, like accountants using an accounting system and product managers using a manufacturing system, conduct analytical work to solve problems within a predefined boundary. In contrast, design attitude emphasizes the creation of alternatives in the first place [see also 38]. In each design circumstance, a designer must “ask oneself anew what is the real problem being faced and what is the best solution?” [4, 9]. I will discuss key features of the design attitude in detail.

One distinct feature of design practice is the exploration of problem and solution together, as opposed to deriving a solution from the understanding of the problem. Designers are solution-focused, rather than problem-focused [8, 26, 34, 35; 77-78]. Cross [8] wrote, “Designers tend to use solution conjectures as the means of developing their understanding of the problem” (P. 102). Lawson [25] found in experiments that architectural students in their final year of study explored the problem by trying various solutions, whereas postgraduate science students tried to discover rules behind the problem. Some researchers have characterized this practices as “abductive” [8:37, 35:102]. Instead of deriving a solution from the problem inductively or deductively, designers typically try out conjectures and see how they fit the situation. Because typical users of information systems such as accountants, sales reps, and production managers do not design artifacts, they likely lack this solution-focused and abductive skill. In learning to design, users need to acquire this solution-focused orientation.

Because there is no single correct design, designing is a constructive activity that aims to produce a satisfactory result among many possibilities. As a result, designers often create a frame that guides the design in a particular direction rather than seeking the optimal solution freely without any constraints [8:94, 35:79, 44]. Designers are capable of not only working around constraints but also introducing new constraints to derive creative designs. Studies have shown that early in the design process, designers create a set of core ideas, or the “primary generator” [9], that produce various conjectures. It is often observed that early on, designers are fixated on ideas [35]. Yet, designers can also keep ideas open [4, 46]. Designers are also more reflective on their own design practices [33, 37, 38]. This practice of constructing constraints to shape the problem is not easy to learn, as in the typical workplace, people are used to working around constraints, not creating them [44].

Designers also achieve both a coherent whole and concrete parts. Romme [34] emphasized systems thinking to conceive a problem within the larger context. Stolterman [41] suggested that designers transform initial key principles into an overall structure that is implemented (p. 7). A successful transformation leads to an “architectonic” design (p. 8), or a sense of being designed as a whole, with each detail relating to the whole. A system without an overall structure becomes “a simple cause-effect system” without organizing principles and is characterized as “tectonic” (p. 8). Similarly, Yoo et al. [50] described how Frank Gehry’s architectural vision played a crucial role in determining his designs. This issue has been widely noted in system design. Brooks [5] stressed the importance of system integrity or consistent implementation of a key design theme. However, the vision does not itself achieve the design. Designers still need to consider detailed constraints. Schön [38] explained that design problems are “figurally complex” (p. 120) in that when one part is changed, the whole is changed. To achieve designs in parts and as a whole, designers oscillate between local and global and between parts and whole [26, 50]. Designers often create a single design that can overcome multiple distinct problems all together [26]. Designers constantly reflect on the changes they have made and listen to what the change creates as a whole, and often reflect on their own reflections to explore different wholes [37].

In sum, design skills are distinct from those skills used by typical system users in their work. We cannot abstractly state that users participate in design, but need to explicate the distinct design practices that users demonstrate and do not demonstrate.

2.2 Situated Learning

Many scholars argue that design is part of any human activity [8, 37, 40]. In discussing design education, however, most scholars discuss design students who seek to become professional designers. Even when Cross [8] acknowledged the fact that students in general education do not intend to become professional designers, he discussed students who want to learn. If design is a pervasive phenomenon, we need to understand how non-designers learn to design.

Design skills are not amenable to classroom and discursive learning. Design practices are “learnable, coachable but not teachable” [37:158] and as such, actual participation in practice is essential. Situated learning theory [24, 49] can help explain this kind of participative learning. In this theory, learning is viewed as a trajectory of participation from periphery to full membership in a community of practice. There are two important components to situated learning: legitimate access and periphery.

Lave and Wenger [24:100-105] argued that in order to learn to become full members, newcomers need to have legitimate access to technologies, artifacts and practices. When sequestered, newcomers cannot learn. George et al. [11] reported that those in a low occupational status were expected to follow canonical processes in using information systems and were prohibited from
becoming insiders of the community of practice. Contu and Willmott [7] took this notion of access seriously and emphasized power relationships. Power relationships are particularly crucial in design. Through building material artifacts that affect human life, design, both as product and as practice, is inherently ethical [17] and political [23]. Buchanan [6] claimed that design is rhetorical, as “the art of conceiving and planning products,” which are “vehicles of argument and persuasion about the desirable qualities of private and public life” (p. 26). Designers tend to have a certain power to bring their own argument and persuasion forward. Others affected by the design, namely users, need to fight against such an attempt. Participatory design researchers have tried to give disadvantaged users access to design [12, 39].

What has not been studied much is the reverse of a typical client-vendor relationship. When a client hires an IT vendor to develop a system, the client has a certain power over the vendor. As a consequence, the client users often have more power, in a way, as they expect the vendor’s designers to design as an exchange for payment. Users often take a stance of “We tell you what we want. Please build it.” Projects often fail because users refuse to use the resulting system. The vendor designers often yield to the users. Designers certainly exhibit power based on their knowledge of technical design, on which users need to rely, and also try to acquire some leverage by bypassing the users to senior managers of the client. Nonetheless, the client-vendor relationship is not an equal one.

This reversed power relationship renders participative learning problematic. Schön [37] wrote that “A student cannot at first understand what he needs to learn, can learn it only by educating himself, and can educate himself only by beginning to do what he does not yet understand” (p. 95). Because design skills cannot be articulated enough to be taught, “willing suspension of disbelief” is required on the part of the student (p. 94). The student then becomes dependent on her instructor, compromises her own freedom, and inevitably feels anxiety. In addition, users often do not have any intention to learn. On the contrary, users typically intend to teach designers about their work and requirements. Because users are typically busy with their own everyday work, their participation in system development is often seen as extra work. Participation is therefore not a privilege and it may be difficult to get users to learn when they do not want to. Jordan [20], cited by Lave and Wenger [24], similarly showed that a young girl learned the trade of midwifery by living in the household of a midwife, without intention to become one. The girl, however, is clearly in the situation to learn as a young person vis-à-vis an older, more knowledgeable and powerful person.

Next, periphery is of special importance in system design. Whereas situated learning theory emphasizes the movement from peripheral participation to full membership, users ultimately have no intention to become professional designers. After the system is implemented, they go back to their group and continue their work as users. Moreover, users need to design while concurrently learning and utilizing what they have learned. Therefore, they typically have no background experience to bring to bear, for instance, by seeing the unique problem as a familiar one from their experience [38]. Furthermore, system design is a time-bound activity. The upstream basic design is sometimes done in as short as three months. The learning ceases when the time is up, not when the person has learned the job. Because no user has free time and no project has ample time, this remains a problem, although less so, even when iterative design methods are used.

Wenger [49] has discussed the subtle position of periphery. He wrote, “Communities of practice can connect with the rest of the world by providing peripheral experiences—of the kind I argued newcomers need—to people who are not on a trajectory to become full members” (p. 117). The periphery is “neither fully inside nor fully outside” (p. 117). Wenger [49:120-121] gave an example of insurance claims processors’ learning of medical jargons through peripheral participation in medical practices. The claims processors will not become full members of the medical profession. Periphery can not only give outsiders access to a community of practice but also enable one to relate to other communities of practice. Moreover, Wenger [49:165] emphasized that a person is constantly passing boundaries and obtains “multimembership” in multiple communities of practice. When a person participates in one community of practice, he or she brings a certain identity and reconciles it with his or her current participation. Because design research assumes students, those who learn to design are seen as starting more or less as tabula rasa. This assumption is unrealistic in system design because users bring a whole range of issues. Design is even more challenging to users who have to reexamine and transform their own problems. Therefore, it is not easy to learn to take a solution-focused approach.

These aspects of power relationships and peripheral participation illustrate the challenging nature of design learning. I will document a case of such learning and discuss its key aspects.

3. METHOD

3.1 Ethnographic Research

Drawing on situated learning theory, ethnography was a natural choice. Participation in practice is not a factual process that can be captured by external measures. We need to understand the lived experience of those who participate. On top of this, as Lave and Wenger [24] discussed, there is no one way to participate in practice; participation is always contextual. To understand context, naturalistic observation is the most appropriate method.

In system design, there is often no clear design studio in which design happens. In many cases users and designers talked in meetings and I observed these meetings and audio-recorded the discussions. In meetings, users and designers communicated using documents that were either printed on paper or projected on screen and drawings on a whiteboard. I obtained copies of these documents and took pictures of any other documents that were not on paper. Interactions between users and designers also happened outside meetings, particularly right before and after meetings, in elevators, hallways, and so on. I made sure that I could capture these, although audio recording was often not possible. I also participated in social activities such as an after-hours party.

In the beginning of the study I conducted formal but open-ended interviews with key members. Through these interviews I learned the general characteristics of the project, such as the size and the scope, the customer, and the team characteristics. I also conducted in-situ interviews whenever possible during the fieldwork. Because accounting terminology is technical, I had to constantly ask questions to clarify meaning. The whole basic design had to be determined within three months and therefore participants were extremely busy. After writing up the analysis, I presented it to the designers. This discussion clarified and validated my analysis. After the observational period, the project moved to the development, testing and deployment phases. I maintained
occasional interactions with the members, about once every two months.

The analysis proceeded mainly by examining recorded interactions and documents. The first part of the analysis was dedicated to understand what was going on. This was not trivial because of the technical nature of accounting work, with which I was not familiar. I listened to the audio recordings many times and transcribed them to understand what participants were saying. Then, I located parts of documents that the participants were talking about and juxtaposed the talk and the document. I also asked many questions of participants and consulted accounting textbooks. Once I understood what the participants were talking about, I organized the data by grouping interactions on a particular function or a set of functions (e.g., “one-time payment” and “payment slip approval”) together. Participants had multiple meetings over time to discuss a particular function. This enabled me to follow the creation, evaluation and transformation of the design. Patterns emerged through this analysis. Eventually I arrived at the three overall design phases. I also coded utterances in design meetings using a simple, objective scheme (see APPENDIX). The purpose is not to do a quantitative analysis to test hypotheses but to provide some overall picture of the data in a succinct manner.

3.2 Research Site

I studied a project that designed, developed and deployed an enterprise system. A rapidly growing service company (hereafter the customer) issued a request for proposal (RFP) to replace an existing accounting, leasing and procurement systems. An IT vendor that proposed a packaged system with the latest technology and sophisticated accounting processes won the bid. This was a new customer for the vendor; no prior development or maintenance contracts existed. As the vendor’s management and I were looking for a project to study in order to understand and improve system development practices, this project was selected as a sample. The choice was opportunistic as there were few projects we could study in depth; system development projects gather confidential data about the customer and therefore few customers were willing to accept a researcher that would study the design practices in detail.

The project consisted of several teams, each of which was responsible for one or two sub-systems (e.g., purchasing sub-system and customer data management sub-system). I studied the accounting team responsible for the sub-system for work related to the accounting department. This team consisted of five designers from the IT vendor, four user representatives from the customer’s accounting department, and a member from the customer’s information systems (IS) department (Table 1). Among the designers was one team leader (design leader). One accountant (junior accountant) was chosen as a focal participant. Although relatively young, she was considered knowledgeable in most areas of the accounting work. A manager of the accounting department (accounting manager) also participated occasionally, particularly in the later stages. One IS member mainly worked with the team. He was not trained as a system designer, although he was quite knowledgeable about accounting systems; he played a liaison role.

<table>
<thead>
<tr>
<th>Table 1: Study participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participants (abbr.)</td>
</tr>
<tr>
<td>------------------------</td>
</tr>
<tr>
<td>Junior accountant (JA)</td>
</tr>
<tr>
<td>Accounting manager (AM)</td>
</tr>
<tr>
<td>Accountants 1, 2 (A1, A2)</td>
</tr>
<tr>
<td>Information system dept member (IS)</td>
</tr>
<tr>
<td>Design leader (DL)</td>
</tr>
<tr>
<td>Designers 1, 2, 3, 4 (D1, D2, D3, D4)</td>
</tr>
</tbody>
</table>

The existing system used by the customer ran on a mainframe computer with text-based terminals. The new system is built on client-server architecture with a graphical front-end. The two systems’ underlying design concepts were significantly different. The main users of the old system were accounting personnel, while the new system was designed for use throughout the company. With the old system, employees filled out slips and forms and sent them to the accounting department; accounting department staff then put the information into the system. In the new system, all employees put their information directly into the system.

I studied the first basic design phase that spanned three months. I observed 42 meetings between the users and the designers and audio-recorded 56 hours and 57 minutes of talk. I also observed 21 meetings among designers only and recorded 20 hours and 55 minutes. The timeline is described in Figure 1. On the top of the figure are the rough processes through which the project proceeded. On the bottom are particular meetings that I will describe below. Prior to this phase, the users did not have any contact with the designers. After this phase, the basic design was refined into more detailed technical and programming specifications including data types, class structures, exception handling, and communication protocols. In this sense, the project employed a typical waterfall process in which design is gradually refined into technical details, as is still common in large-scale enterprise system development. Designers used no specific design method (e.g., object-oriented design, rapid prototyping and Agile) but used such general tools as workflow diagrams, fit/gap analysis, and a type of function-point cost estimation. Although I participated in the middle of the basic design phase, in the fifth week, the actual design started only one week prior; the participants had to spend some time to set up the infrastructure and specify the processes by which they were going to design. To capture what had happened the week before, I obtained audio recording of the meetings and copies of documents.
4. FINDINGS
I will focus on design of four specific features. These features were selected to show the progress of users’ learning.

4.1 Feature 1: Breakdown Remarks
One particular requirement was concerned with the data fields on several types of accounting slips, e.g., payment slips. This discussion started in the fourth week and concluded in the seventh week. In the existing system, slips contained a free-text, remark field for each breakdown on the slip. If someone purchased a personal computer with optional hardware and software, the purchase slip contained an item for the computer and other items for the options as breakdowns. The users used a remark field for each of the breakdowns (hereafter breakdown remarks). The package on which the new system was based contained only one remark field for the entire slip (slip remarks) and not for each of the breakdowns.

In the beginning, the accounting manager, junior accountant and two other accountants started by explaining the existing system. After a few meetings, the junior accountant became a main participant. She used screen captures of the old system and walked the designers through its uses. The meeting proceeded as she gave explanations; designers worked to understand the explanations by asking questions. I coded all the utterances of a 54-minute meeting in the fourth week using a simple scheme (Meeting 1 in Figure 1). Among 207 utterances identified, the junior accountant produced 94 utterances (45.4%) and the design leader produced 73 (35.3%). The junior accountant explained 17 topics of the users’ work processes, 9 of which were followed by designers’ questions to clarify the details (the other 8 topics were understood without clarification). The design leader produced 35 question utterances. Among the 207 utterances, neither side produced any design proposals. The consistent pattern shows that the discussion was highly structured such that each side had its own roles.

In this interaction, the users and the designers tried to map features that the users wanted and features that the package offered. Many of the features the users wanted could be directly mapped to the package’s features. Inevitably, however, designers identified some “gaps.” When designers had analyzed the accounting slips that users presented, they discovered the gap of breakdown remarks. The designers explained to the user representatives that the package did not have breakdown remarks. The users stated their need for breakdown remarks. The design leader suggested that they would propose alternatives because it was not easy to add breakdown remarks into the system.

The designers then started to create design proposals. For the design proposals, designers tried to align with the package’s built-in functions so that little modification would be necessary. Modifying the package or adding new features would require costly impact analysis and system testing. In the seventh week (Meeting 2), the designers explained their proposal to do without breakdown remarks. Yet, the users considered the gap as the package’s limitation and rejected the design proposal by insisting on their original requirement. A short segment of this discussion is shown below. Words in square brackets [] were added by the author to aid interpretation. Some words are emphasized in bold to show the analytical point.

Designer 1 (D1): Do you split journal titles?
Junior accountant (JA): Yeah, right now we attach one for each product code but of course we also split.

D1: Right now, the system has only one journal title.
JA: We can’t do without it. [omitted] No way. Please add it. If they buy each with its own product code, it may work. But they don’t. [omitted]

Information system member (IS): Here it’s the matter of principle. For each breakdown of the each slip, at least we need the project number, journal title, department and all that, and for accounting slips, a remark field.

Eventually, designers gave in and agreed that they would modify the package to accommodate this requirement. These strong reactions on the part of the user and the IS member allowed no room for designers to propose any other alternatives. The user treated the requirement as given. The users verified all the slips extensively in accounting before posting them to the general ledger. The accounting department had taken responsibility for the data. Decentralized accounting was adopted for the new system, however. With decentralized accounting, accountants trust data from other departments more—although not completely—and do less extensive verification. When the junior accountant claimed that accountants had to do verification using the breakdown remarks and correct the journal titles, no discussion was held in
terms of the underlying difference between existing practices and the newly envisioned ones. The package being introduced was also designed on the real-time accounting model. Because slips were frequently sent to accounting in real-time, most slips would contain only one breakdown whereas in the existing model, departments sent a paper slip to accounting at certain intervals—many items were put on each slip as a result. Therefore, it is likely that the overall remarks field for the slip was sufficient. The point is not to suggest which accounting policy is good but to highlight the difficulty in redefining the problem that users take for granted.

### 4.2 Feature 2: Key

Whereas the design of breakdown remarks was constructed based on the users’ rejection, some other features were developed further. The users had used a unique identification number called key to locate accounting slips. If they needed to refer to a slip, they could type the key in the system. Designers and users soon found that the package did not have the same identification number. I will describe this process in chronological order.

The users and the designers began by mapping between the work practices that the users were familiar with and the packaged system that the designers were introducing. One example of the interaction is reported here from a meeting in the fourth week (Meeting 3a). In this example, designers tried to understand the key by mapping it to numbers they knew. This discussion continued and explored how the number was issued and used.

Design Leader (DL): The number to identify slips uniquely is often called a slip number or a journal entry number. Here can we say that the key is that number?

JA: Yes, it is.

DL: Okay. Could you explain rules, if any, for issuing this number?

[continues]

After this, the designers constructed a design proposal. In the seventh week, the designers explained the package’s numbering system and suggested that the journal entry number was probably the closest to the key (Meeting 3b). Yet, when the discussion continued, they came to realize that the journal entry number was issued only when the entry was posted to the general ledger, 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. Once the problem was identified, the junior accountant and designers started wondering if they could use the “process number” instead. However, they found that there were cases where 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. Although they tried to find 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]

DL: In that sense, we need to create a new number. The numbers we currently have won’t satisfy your request.

The junior accountant kept demanding the key and rejected any of the alternatives that the designers proposed. On the next day, the design leader indicated to her that they would create a new number. Designers had no power to overthrow the client’s persistent demand. The junior accountant treated this requirement as given and only sought the solution to realize it. This is a problem-oriented approach in which the problem is understood first and a solution is sought to solve the problem.

The discussion, however, continued. In the seventh week, after the gap items were listed, designers provided the client with rough cost estimates. The total cost exceeded the client’s budget significantly. This prompted users to become more flexible on certain requirements to reduce the cost, although the client exhibited some distrust against the vendor. The client was informed about the cost increase only at this time, when they had no choice but to rely on the vendor. Despite the increased flexibility, users tried to replicate existing work practices using the package’s features and focused only on the parts.

In the ninth week the accountants started to reconsider the key (Meeting 3c). The accounting manager joined the discussion. The accountants suggested that they could use the process number despite its problems.

JA: We can use the process number to label slips. It works as the standard code. But the downside is the duplication for continued transactions. [omitted] But for continued transactions there is only one transaction per month, right?

DL: It is possible to enter two payments a month.

JA: But twice a month is [not likely]- Well I am not sure. I don’t know. No no.

They sought a way to use the process number. In terms of the number of utterances, the users dominated the discussion. Among 112 utterances identified, users—the junior accountant and the accounting manager—produced 93 (71.5%), the designers 35 (26.9%) and the IS member 2 (1.5%). There were 5 utterances that proposed new designs. The junior accountant and the accounting manager both produced 2 proposals while the design leader produced 1. The junior accountant proposed ideas of combining the process number with other data fields to avoid duplication and limiting multiple payments in 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. Users became quite active in designing, more flexible with their requirement of the key, and became willing to accept some changes. If we simply focus on how actively users participate and create IT artifacts, this may be considered a successful user design.

Nonetheless, the resulting design did not integrate the parts and the whole. Users were still trying to find an equivalent to the key. In contrast, the package is designed in such a way that users do not even have to think about the numbers; they can locate slips by means of various search criteria such as employee name, date, slip type, and so on. This is why the numbers in the package were so complicated; they were not supposed to be exposed to users. What became clear is that users had been using the key because of the character-based terminal of the mainframe system. Users did not use a mouse to click on a slip but typed in a code to bring up a certain slip. Yet, the new system would be equipped with an advanced graphical user interface in which users can simply click.

With the new package, “users can use the system without
consciously thinking of the numbers,” as one of the designers later put it. Furthermore, the existing system had a small display on which only limited amount of information could be displayed. Therefore, most detailed information (particularly, lists and tables) was printed on paper. A key was needed to link the information on paper to information on the system (because users could not click on paper). The new package was based on the concept of working without paper.

The designers had explained this design policy of the package system, although in this case the discussion was about journal titles, not the key.

**JA:** The items are different. Everything will change.

**DL:** Right. In a so-called mainframe system 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.

**JA:** One way to think about it is to make it easier for other departments [to enter data]. But for accounting we need to see all the journal entries. We know which journal 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.

Accountants remembered the codes, whereas the new system was supposed to be used by other departments, the members of which were not familiar with the accounting codes. Note that the junior accountant understood the difference between the old and the new system. She nonetheless refused to give up what she had been using. As a result, a similar number based on the process number was implemented for additional cost.

### 4.3 Feature 3: Journal Table Sheet

Whereas some designs focused only on parts and not on the whole, a few others requirements developed even further. Accountants wanted to print the table of all the slip data (hereafter journal table sheet) at certain intervals so that they could quickly go through the slips to check for incorrect information. The package could not print this table. The users, however, repeatedly made it clear that this table was necessary, for instance, in the fifth week (Meeting 4a).

**IS:** You can bring it to the cashier [omitted] and get cash, also pay cash.

**JA:** How about sending it with the journal table?

**IS:** [omitted] I still think that we can use a paperless journal table. [omitted]

**JA:** Okay 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 IS member’s suggestion of using the table on the screen, not printing it, does not change the practice because users still use the table. In this sense, IS focused on the parts.

In the eighth week, the junior accountant, however, suddenly said that they had changed their minds after some internal discussions, and now were willing to drop this requirement (Meeting 4b). Yet, she immediately said that they wanted to print out another list instead.

**JA:** Actually now we are thinking of changing the policy, we are considering about giving up the journal table sheets. Then, we had the cover sheet, right? We wonder how much we can tune up the cover sheet.

**DL:** The one to attach vouchers, right?

**JA:** Yes, that function [requires] not too much [modification]. I mean you can print the screen itself.

**JA:** Basically we cannot check the journaling on the display so give up, and I think you remember the journaling summary list or daily summary journaling list. That thing that comes out in batch. We will make it 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 general ledger, they conceded and suggested that they could check the slips after the entries were posted to the ledger, modifying any incorrect data at that point. A minor modification enables the system to print the cover sheet. The change was crucial for the realization of real-time accounting; it was a “policy” change. Because the junior accountant gave up the before-the-fact verification, the data is processed in real-time. The junior accountant presented not only her grasp of the high-level policy but also the technical solution. She obtained this idea by seeing the problem as a familiar one [38]. She had learned about the cover sheet before and tried to use them for a different purpose, as a “daily journaling summary list.” She took a solution-focused approach. This idea of cover sheets was a primary generator that more or less determined the design and details were worked out to fit in this design.

Nonetheless, she did not know all the technical details of the system and in that regard had to rely on the designers. Note that her design proposal was in a semi-question form: how easy it was to tune up the cover sheet. She made an assumption that it was technically feasible and easy to do so. In this way, her design proposal was contingent on her various assumptions which needed to be verified by the designers. As a result, her proposal was accompanied by a number of technical questions. She encountered some details on which she made incorrect assumptions. For instance, she did not know that the accountants could not modify the slips and had to return them to the users who submitted the slips. Designers clarified these details and helped refine the design. Her design embodied both the parts and the whole, but only contingently on her collaboration with the designers. This design was concluded in a short discussion, which contained 37 utterances. The junior accountant proposed 3 design proposals and the IS member proposed 2. The designer produced no proposals. Out of the total 8 questions identified, 4 were by the junior accountant. Among 12 utterances that offered factual information, the designers produced 9. Roughly speaking, users offered design ideas and the designers provided factual information about the system to refine the ideas.

This design of a whole and its parts is not a clear-cut phenomenon, however. In a subsequent discussion the junior
accountant 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." Yet, the accounting manager, who joined the meeting in the middle, presented counter-arguments, saying "If we don't approve slips in the accounting, we don't need that many [fields]." The accounting manager tried to change the work practice in such a way that accountants would verify the slips on screen, keeping the 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." The junior accountant disagreed. "Whether [we do extensive verification] or not, if we want to be ready to do minimum verification, this much data should be printed." However, they finally decided to implement the detailed list for the after-the-fact verification, largely at the junior accountant's insistence.

This argument shows the multi-layered nature of designing the whole and the parts, and is not included to criticize the junior accountant for not taking the policy further. The junior accountant had a legitimate reason to defend her perspective. As an accountant, getting the numbers right is the center of her work [2]. Nonetheless, we need to acknowledge that she had difficulty in reflecting on constraints. Because the new system employed an easy-to-use graphical user interface (GUI), there were likely to be fewer mistakes. In the old system, users had to type numeric codes to enter journal titles and other data. With GUI, users would not be even conscious of journal titles, since the GUI concealed the esoteric accounting terminology. No discussion like this was observed. I later heard from the designers that only when they touched the system (in the later testing phase) did the users suggest that they did not need even the table for after-the-fact verification.

4.4 Feature 4: Special Payment Function

We can see another example to explicate this pattern more thoroughly. In the forth week (Meeting 1) users requested a special payment function that deviated from the regular workflow. Whereas payments were automatically tied to fund transfers in the regular workflow, this function entered journal entries but fund transfers would be done separately. The package did not provide this function. A more extensive discussion was held in the ninth week (Meeting 5). Designers proposed a journal entry adjustment function, which was designed only to manipulate journal entries for exceptional cases such as error correction and cancelation, because this function was not tied to fund transfer. Furthermore, users demanded a batch data import feature to make a huge amount of entries at one time. Whereas none of the payment functions had this feature, the journal entry adjustment function did. This proposal, however, was rejected for various reasons. First, accountants did not feel comfortable using the adjustment function for payment, which did not require accountants' formal approval. Second, the adjustment function did not have a date field, which users found necessary.

Then, the accounting manager proposed a design that required significant modification to the system and was determined to be unfeasible. When the discussion came to a dead end, the accounting manager offered another proposal by revisiting some of the constraints. He suggested that other departments, not the accounting department, could enter the payment data. This shift to decentralized accounting made the batch data import unnecessary because the accounting department would not have to enter the huge amount of data on behalf of other departments. Yet the majority of the problems persisted because payment functions were still automatically tied to fund transfer.

The junior accountant again proposed a design that solved many of these small problems altogether.

JA: If we pay with a payment form, this may be a stupid idea, outstanding payment is marked. But because it does not involve money transfer, the payment is not done. In the journal, account payable and bank deposit are entered. Can this be matched with fund transfer already executed?

Instead of using the adjustment function, they could use the "payment forms" which is a function specifically designed to process a fund transfer separately. Instead, a paper form is printed and taken to a bank for payment. The payment form had a date field as well. They could simply forego taking the printed form to the bank, and instead reconcile the payment form with the bank statement at some later date. The junior accountant could not explain this idea clearly and therefore the designers initially did not see why payment forms were relevant in this case.

When she gradually made herself understood, two problems became clear. First, the payment form entries (journal entries) and the bank statements (actual transfers) were not automatically reconciled. Second, whereas multiple transfers needed to be reconciled with a single journal entry, there was no way to do so. As it happened, the junior accountant came up with a solution for these as well. She suggested that grouping entries by the 'budget center' number from which the payment was made should make it easy to reconcile, because even if multiple transfers were made they would be from a single budget center. This idea stimulated designers. One designer suggested using a "transfer forms" payment function instead of payment forms. The database for payment forms was already used while the one for transfer forms was available. Designers concluded that the payment could be realized with little cost.

Simple and straightforward design resulted. We can see again that the junior accountant had learned about the payment form and proposed using it for an unintended purpose. She again understood the problem by "seeing as" [38:133] and examining it in a solution-focused manner. The idea of payment forms served as a primary generator to which other details (e.g., entry reconciliation) become adjusted [9]. Because the destination accounts to which fund was transferred were typically used for reconciliation, use of the source accounts, i.e., budget centers, was creative. Nonetheless, her proposal was again contingent on her assumptions that were validated by the designers. Among 163 utterances identified within this discussion, the junior accountant produced 14 questions, 10 of which were used to test her assumptions with the designers. The designers took the idea and refined it into a further technical level. Therefore the design was a collaborative effort among multiple people. Out of the 163 utterances, 10 were utterances that offered design proposals. Four were from the designers. The junior accountant and the accounting manager produced 3 each. Whereas 5 of the ideas were rejected, the other 5 developed on top of each other.
5. DISCUSSION

5.1 The User’s Learning Process

The key contribution of this study is to document what specifically a user does when she learns to design. Design is not just provision of information, a formation of high-level policies, or choice among design alternatives. Nor is design limited to active participation in design discussion and collaborative exploration of alternatives. Competent design requires distinct skills such as a solution-focused, abductive approach, reflective and productive use of constraints, and integration of parts and the whole. This analysis revealed how the user in the above example learned and demonstrated these skills.

In the first phase of the learning, the users had no intention to design and expected the designers to design for them. The junior accountant was looking at design practices from outside the designers’ community of practice and as such blackboxed whole design practices. Therefore, the idea that design was needed was not obvious to her. Consequently, when she found that her requirements were not realized, she simply rejected the design. This practice can be described as problem-focused as opposed to solution-focused [8, 26]; she examined the problem and expected a solution to be derived from the problem. This one-way process has been shown to happen rarely in design, while it is common in other disciplines like science [26]. In this phase of design, the problem and the solution were separated between the users and the designers. The users brought more or less fixed problems whereas the designers brought solutions for these problems. Such mapping did not work in many cases.

In the second phase, we saw users became more flexible on constraints in that they did not insist the exactly same features or functions as initially desired. Then the junior accountant actively explored combinations of the system features to realize the requirements. She started to experience the designers’ thought process as designers came back with design alternatives and shared the technical difficulty in realizing exactly what the users wanted. In this sense, she participated in the designers’ community, although the participation was not deep. While she was engaged in solution creation by tinkering the package features, she still took the problem-focused approach with the problem fixed. She considered the problem in isolation and tried to derive the best possible solution for it without referencing a larger whole. This is in contrast to the accounting manager who tried to push the new policy of distributed accounting but could not suggest any concrete design; the whole was referenced without the parts.

In the final phase, the junior accountant started to design more creatively with an eye on the whole. To do this, she exhibited a “creative leap” [8:65], i.e., a change of policy. She revisited her initial constraints and derived solutions, not as a solution to one constraint at a time but holistically, to overcome various constraints altogether. She took the solution-focused approach and accomplished design in parts and as the whole. Not only was the holistic concept of real-time and decentralized accounting realized, but the simpler design also eliminated the costs of modifying the package. Through this learning, we can say that the junior accountant came to acquire “designerly ways of knowing” [8]. Yet, we should also note that integration of whole and parts is not a clear-cut phenomenon. While the junior accountant undertook the holistic perspective at the core of the design, she was still trying to replicate the existing practices in some areas.

If we simply talk about participation in design, we do not know what it is that users do to design. Active participation and exploration of design alternatives do not necessarily mean the user is designing in the full sense. When users learn, we need to examine specifically how they design the whole as well as the parts, employing a solution-focused approach. The revealed learning process helps distinguish these subtle differences. It is also important to acknowledge that design was being determined while the user was learning. Users do not have the luxury of learning how to design before beginning to actively design and as a result, some functions were designed before the user learned to design. The process number and breakdown remarks were chosen even though it is possible that the user could have learned to arrive at more creative, simple design. As a result, the customer had to pay for the modifications to the package. Furthermore, the user learned to design only in two months, starting with the complete disinterest in design. The short, one-path learning did not enable users to construct ideal designs. Whereas designers typically see each unique problem as something familiar [38], users have only one path and will not design another system at least for a while. This “seeing as” [38:133] has to happen within the short, single path.

5.2 A Peripheral Designer

Retrospectively it seems remarkable that the junior accountant could achieve this learning only in two months. On the other hand, she was by no means a fully competent professional designer. Borrowing the notion of legitimate peripheral participation from Lave and Wenger [24] and the notion of periphery from Wenger [49], I call this level of design expertise a peripheral designer. The term “peripheral” has two aspects. First, peripheral participation in the designers’ community of practice suggests that users do not become a full member of the designers’ community but still exhibit design capabilities. Second, users become peripheral to their own users’ community of practice by learning to maintain some distance from their problems and are able to revisit constraints that members typically conceive as fixed.

Periphery in the designers’ community of practice is a complicated phenomenon. The junior accountant learned detailed functions of the package, not only the workflow but also the data fields. By seeing the problem as something she had already learned [38] she could appropriate a function for an unintended purpose. She also demonstrated her knowledge of the system details and thought through possible scenarios and potential problems. She could, however, only elaborate her ideas in interaction with the designers because she lacked the knowledge about the specific technical details required in refining the design. Therefore, she did not completely learn the details, but enough to make assumptions. Peripheral designers are therefore contingent, social and interactive. This periphery largely comes from the necessity that users need to learn to design within a short period of time. They also have a little interest in becoming a full member of the designers’ community of practices as they are and will be users and not designers. This type of designer is clearly different from those that the design education literature portrays, and illustrates a more realistic model of designers for many professional activities that comprise designing.

Periphery in the users’ community of practice enables users to reflect on constraints that users typically take for granted. Yet, the users do not become non-users by this move to periphery. The
peripheries is not as distant from the center as the periphery that newcomers experience in participating in a community. Newcomers take a fresh perspective on constraints but have yet to understand the constraints properly; both the detailed understanding and reflective examination of the constraints need to be realized. Wenger [49:175] suggested that “imagination” was a way of belonging to a community. By imagining, members can create “new images of the world and themselves” (p. 176) by extrapolating from their own experiences. Combined with engagement in actual practices, imagination results in reflective practices and “the ability both to engage and to distance” (p. 217). This distance is crucial for designing the whole, whereas designing parts can be achieved without such a distance.

Researchers have discussed the importance of reflection on assumptions [e.g., 3, 29, 36]. Once users learn to revisit their assumptions they can design a system from a fresh perspective. Reflection, however, does not necessarily lead to design; the accounting manager was quite self-reflective but did not know the details of the system enough to create any design that made sense. Furthermore, reflection is not simply done by means of interaction methods that encourage reflection. More than cognitive and communication guidance, participation is needed. Specifically, multimechanism fosters alignment of different perspectives [49]. Reflection is therefore an integral part of design practices. By learning how to take the solution-focused, abductive approach, the user could reframe the problem in light of a solution, which could vanish various constraints all together.

Users can obtain peripheral participation in design while still deeply rooted in their own community. For instance, the junior accountant tried to replicate the existing work practices using the new system features. She was trying to design in the sense that she understood details of the system features and attempted to reconfigure them for a specific goal. Nonetheless, she took the problem as given and focused only on the parts. On the other hand, users can also take a reflective stance on their own community without participating fully in design. Note that the accounting manager could revisit the users’ constraints reflectively and abstractly without participating in the designers’ community of practice. Although he could offer some design proposals, most of them were unrealistic and costly. When users learn to design, their learning needs to be pushed all the way so that they can design both the parts and the whole. A peripheral designer can achieve this by maintaining a delicate distance from the two communities of practice.

5.3 Power of a Peripheral Designer

Power relationships are relevant in the peripheries of both users’ and designers’ communities of practices. First, the power relationships make it more challenging for a user to join the designers’ community of practice. Situated learning theory has emphasized the difficulty that disadvantaged newcomers face in gaining access [24]. Yet powerful newcomers experience their own difficulties. In particular, the fact that the client is paying the vendor for services seems to lead users to expect that the design is solely the vendor’s job. We therefore need to understand that users do not simply learn to design; they need to learn to learn, and often they implicitly learn to learn without knowing they are learning. Designers did not make it explicit that the users needed to design, although they stressed the importance of users’ participation in design. The designers could show their thought processes and share difficulties with the users. In so doing, designers showed that it was costly to realize all that users were asking for. Neither situated learning theory nor design education literature gives much attention to this reverse power relationship. Masters are always seen to have power. Once we understand the reverse power relationship, we can see that being able to tell learners how to learn should not be taken for granted. When parents lose leverage over their children, a child’s learning may well be lost. We need to rebalance the assumption of power relationships in situated learning theory.

Second, power relationships affect users’ move to the periphery in their own community. Users in powerful positions can easily refuse to revisit their constraints. In this case, however, the designers simply did not listen to the users. They tried to convince users by providing various logics and design alternatives. Yet, they could do so only so many times. On the other hand, once the user learned to take the solution-focused approach, she transformed her initial constraints more radically than designers. Designers could not create the designs that the junior accountant could, though this does not necessarily prove the designers’ incompetence. Power relationships made it difficult for designers to free themselves from the constraints that users took for granted. For instance, the designers chose to use the journal entry adjustment function because the users emphasized the need for a batch data import and separation of fund transfers from journal entries. Treating these constraints as given, designers’ design options were limited. To be sure, “good” designers could still convince users otherwise. Yet, this is largely a power problem, not a problem of technical design skills. As it is often said, good designers should also be good at politics.

It is often the case that users are disadvantaged and sequestered from design, particularly those users who work on the shop floor or in the field and have relatively little power. Managers can also dictate design according to their own priorities. In these situations the design practices and the learning processes will be somewhat different. Yet, in real-world enterprise system development, there are many cases where vendors have limited leverage over clients. Unlike experimental studies or research projects for which funding comes not from the client but from an external source, users of this study invested in this project and were accountable for the results. It is important to understand what happens in a real-world situation.

6. CONCLUSION

This study documented how users learned to design a new enterprise system. The first contribution of this study is to show the subtle learning processes and reveal design practices as something more than configuration of parts. Scholars have discussed design, both in research and in practice, as a distinct activity in contrast to natural and behavioral sciences [6, 13, 30, 34, 43] and decision-making [4]. Yet, this categorical contrast makes us blind to the variety and the subtlety within design practices. Even when various design alternatives are actively explored, problem-focused, non-holistic design is different from solution-focused, holistic design. We need to be specific as to how it is so; otherwise, we may be talking about different things using the same label. In the same way, users’ participation in design process should be discussed in a specific manner. Active participation in design and collaborative exploration of design alternatives may not constitute the kind of design practices that competent designers exhibit. Users may still take problems as fixed and focus only on parts. By explicating the practices in
detail, this study could reveal the learning process in a precise manner.

The second contribution is to portray a realistic model of design practices rather than assuming fully trained professional designers. Specifically, I highlighted the notion of power and periphery inherent in the user’s design learning by drawing on situated learning theory. Theories of learning tend to make an implicit assumption that learners will learn to the full extent. This study pushed Wenger’s [49] notion of periphery further and highlighted it as a key aspect of users’ learning. Yet, periphery should not be confused with halfway, unfinished or tepid learning; we saw the junior accountant exhibit significant design competence. Because this type of learning is pervasive in the workplace, theories of learning should expand their focus from full-fledged learning to more nuanced peripheral learning. The power relationships also demand careful examination. Unlike typical learning situations where the knowledgeable master is powerful, clients learn from less powerful vendors. This reverse relationship is consequential for design learning. Given that design is pervasive and everybody can design, we need to anticipate various types of power relationships.

The notion of peripheral designers is useful for practical purposes. The fact that users can learn to design within a short period of time should encourage practitioners. The detailed description of the learning process can guide practitioners to assess how much their users have learned and what more the users need to learn. Practitioners should not be satisfied even when users participate in design discussion actively and explore alternatives. We can explore ways to facilitate users’ learning. Because users can take a solution-focused approach by seeing a new situation creatively as something they already know, designers should familiarize users with a number of features. The power relationship needs to be managed as well. Designers can bring to bear the power of high-level managers, although this is a delicate issue as we saw above.

Peripheral designers are crucial in many different contexts. von Hippel [45] showed that users, not just vendors, innovate new products and services. Toolkits enable users to design what they want. Unlike the full designers and innovators portrayed in von Hippel’s [45] studies, however, many more users, not just lead users but also ordinary ones, can learn to design in a more nuanced manner. Given that design is part of many professions [8, 37, 40], we need to understand how those who are not professionally trained designers can design. Particularly, they may well need to learn to design in a short period of time and have little interest in the learning. This study’s findings help understand design practices in a broader context.

This analysis from a single case provides deeper understanding of the subtle nature of learning. Based on this understanding we need to explore how people learn to design in other contexts. One interesting question is: how do users learn to design in a custom-made system development as opposed to deploying packaged applications? Although in the present case the package was used flexibility with many modifications, the package presented a number of constraints. In designing from scratch, design practices and the learning process may be different. Furthermore, when multiple user groups are involved and there is some conflict between users groups, the learning would be more complicated. Future studies need to investigate the various other shapes of the interactions between users and designers.

7. ACKNOWLEDGMENTS

I am grateful for my advisors Profs. Burt Swanson, Barbara Lawrence and Corinne Bendersky of UCLA as well as my colleagues at PARC. I also owe a great debt to members of our client who helped me in this research.

8. REFERENCES


[33] Olsson, E. What active users and designers contribute in the design process. *Interacting with Computers*, 16 (2). 377-401.


9. APPENDIX

I used the following coding scheme.

Description (documentary): Statement of factual information using a document (factual information that the speaker has on a document); e.g., a user or a designer goes over a prepared document

Description: Statement of factual information (factual information that the speaker has); e.g., a response to a question and an addition to what has been said

Question: Solicitation of factual information (factual information that the addressee has); e.g., a user asking about the system and a designer asking about the work

Contention: Statement of feeling and demands (what the speaker wants to do or wants the addressee to do); e.g., a user stating requirements and a designer stating system’s limitations

Design proposal: Statement of design proposal; e.g., alternative workflow, system features, policies and technical implementation

Other: All other utterances that contain no new information; e.g., acknowledging, appreciating, summarizing, and confirming