A Design-Supporting Tool for Implementing the Learning-Based Approach: Accommodating Users' Domain Knowledge into Design Processes

Jung-Min Choi[1] and Keiichi Sato[2]

[1] Faculty of Crafts and Design, Seoul National University,
599 Gwanak-ro, Gwanak-gu, Seoul 151-742, Korea
[2] Institute of Design, Illinois Institute of Technology,
350 N LaSalle St., Chicago, IL, 60610, USA
jmchoi@snu.ac.kr, sato@id.iit.edu

Abstract. In the current interactive product/system design, while users' acquisition of sufficient knowledge for operating a product or system is increasingly considered important, their acquisition of problem-solving knowledge in the task domain has largely been disregarded. Without enough domain knowledge, users will bot be able to learn how to creatively adjust their product us to produce satisfactory results and better experiences. This research aims to develop a methodology for designing interactive products/systems that can support users' development of domain knowledge through interaction. This new approach to user-product interaction is named the Learning-Based Approach (LBA). Based on the previous theoretical and empirical studies, this paper proposes some mechanisms for implementing the LBA. Then, a computer-based tool is developed in order to support designers' more effective and efficient application of the LBA mechanisms in design processes.

Keywords: users' domain knowledge, Learning-Based Approach, design-supporting system, interaction design methodology.

1 Introduction

Interactive technologies embedded in products and systems have drastically reduced people's efforts in solving problems and achieving goals in everyday practices. In order to improve the quality of interaction with products and systems, designers and researchers has mainly been concerned with how to help users solve problems and achieve goals more easily and efficiently. In the current design process, while users' acquisition of sufficient knowledge for operating a product or system (e.g. how to operate a coffee machine) is increasingly considered important, their acquisition of problem-solving knowledge in the task domain (e.g. how to make delicious coffee) has largely been disregarded. As a result, domain problem-solving knowledge has been hidden behind step-by-step product operational procedures, detached from actual user experiences.

However, people's primary purpose of using a product is not proficient product operation itself, but the acquisition of satisfactory results and experiences through

M. Kurosu (Ed.): Human Centered Design, HCII 2011, LNCS 6776, pp. 369–378, 2011.

interaction. In the authors' previous research, it was found that without sufficient knowledge of problem-solving mechanisms in the task domain (i.e. domain knowledge), users were not able to creatively adjust their product use in order to achieve satisfactory results and generate better experiences [1]. Therefore, when designers seek to support users' interaction with a product, it is not sufficient to consider how to help them obtain sufficient knowledge of operating the product. Rather, designers should also consider how to support users' acquisition of their own problem-solving knowledge in the domain.

The present research aims to develop a methodology for designing interactive products and systems that can help users actively develop their domain problem-solving knowledge through interaction. This new approach to user-product interaction is named the Learning-Based Approach (LBA) [1]. Based on the authors' previous theoretical and empirical studies, this particular paper proposes some mechanisms for implementing the LBA. Then, a computer-based tool is developed by employing the mechanisms in order to support designers' more effective and efficient application of the mechanisms in design practices.

By supporting users' acquisition of domain knowledge – i.e. by adopting the LBA – designers can provide users with more robust and fundamental way for assisting their goal achievement through interaction. In addition, for designers, this new approach provides not only a new understanding of user-system interaction but also applicable methods and tool with which they can effectively develop products that can facilitate users' development of domain knowledge.

2 Learning-Based Approach (LBA) to Interaction

2.1 Perspectives on Users' Knowledge Acquisition in Interaction

In order to achieve satisfactory results while using a system, it is necessary for users to access and apply existing knowledge and obtain new knowledge through interaction. Current approaches to system design tend to consider these kinds of user knowledge development processes mainly as skill and/or knowledge development processes for better *operating* a product/system (e.g.,[2] [3]). Consequently, most designers' primary concern regarding supporting users' knowledge development focuses only on how to help users gain knowledge and skills for operating a product as quickly as possible (e.g., [4] [5]). By obtaining only proficient skills for operating a particular product, however, users may have difficulty transferring the obtained knowledge when they are using a new product or solving different types of problems.

Some researchers, such as [3], argue that product/system designs should support users' generation of appropriate mental models (part of knowledge) because successful user-product interaction depends on whether users can build up proper mental models and apply existing mental models to newly encountered problem-solving situations. However, proper mental models of how to operate a product/system do not completely resolve the problem of users' problem-solving capability limited within the functioning of the machine. Furthermore, proficient product operations may not be users' primary purpose of product use.

Some more recent research, particularly in activity theory approaches, provides an alternative viewpoint by explaining the relationship between users and products from a more fundamental viewpoint (e.g.,[6] [7]). Those approaches consider products/ systems of as tools that mediate inherent mutual knowledge development between a user and an object of activity. In other words, interactive products are viewed as knowledge-mediating tools that help users attain their object of activity.

From this viewpoint, in order to fully support users' object-oriented activities, it is not sufficient for designers to consider how to support users' development of skills and knowledge for operating a product (tool). Rather, they should also consider how to help users obtain knowledge needed for fulfilling the object of activity domain.

2.2 Designing Interaction as a Learning Process

By thinking of interactive products as knowledge-mediating tools and user-product interaction as knowledge development processes, one can say that supporting users' knowledge development (i.e. learning) in interaction can be an effective design strategy for improving interaction quality.

This new approach to user-product interaction is named the Learning-Based Approach (LBA). The LBA emphasizes the significance of users' domain knowledge in interaction; when users have enough knowledge about problem-solving mechanisms in the task domain, they will be able to figure out how to achieve a goal by creatively adjusting their product use to meet their variable needs. Therefore, designers should consider how to help users develop their own domain knowledge through interaction.

In order to empirically investigate the effects of users' operation and domain knowledge, the authors previously conducted observational user studies [1]. The studies compared four different groups' problem-solving behaviors, which were organized according to types of knowledge (domain vs. operation knowledge) and their levels of knowledge (novice vs. expert). From the analysis results, it was clear that in order to achieve quality results in using a product, people need to acquire and effectively apply domain knowledge. For example, participants who were accustomed to operating a product (e.g. coffee maker), but did not have knowledge of the task domain (coffee-making principles), had difficulty creatively adjusting their knowledge in certain unexpected situations. Unlike the group, participants who had enough knowledge of both operation and the task domain were able to flexibly modify their behaviors in different situations.

2.3 Some Terminology: T*ask Domain* and D*omain Knowledge*

Task Domain. A task domain is defined here as the domain *in* which the user is working, but it is also concerned with a user's *reasons for* conducting an activity. The term task domain in this research may seem similar to the notion of domain that is often used in the field of knowledge-based system development, and yet, there are some important distinctions to be made. In the system development field, some researchers emphasize that designers need to understand the "task domain" in order to develop a software system that can better fulfill users' needs when performing tasks in the domain. For example, [8] argues that software designers should analyze the domain by considering "where and for what the system design is used." The

knowledge-based system development field works to provide users with specific knowledge necessary for particular problem-solving. While agreeing with this overall frame, the present research is based on some different viewpoints. This research argues that designers should not think of users' problem-solving methods as fixed or entirely predictable, and that designers should account for users' dynamic needs when developing products and systems. Therefore, while knowledge-based system development often holds that a task domain is stable, this present research focuses on the notion of the task domain as constructive and changing. In other words, a task domain is differently *constructed* based on an individual user's different needs in variable situations, and can also be continuously modified according to the user's changing contexts and needs.

Domain Knowledge. Domain knowledge refers to knowledge necessary for solving problems in the task domain. It is about the kinds of factors that shape the quality of results, and about how to adjust the factors in order to achieve a desired quality of results. It includes conceptual, hierarchical, sequential, and cause-effect knowledge. There is also a distinction between *domain knowledge* as it is used here and the same term used in knowledge-based system development. Knowledge-based system designers are concerned with *designers'* understandings of domain knowledge structures, which allow them to develop systems that *solve* the problems users encounter. By formalizing domain knowledge structures and using the structures to pre-configure *solutions* for problems in the domain, knowledge-based systems usually aim to develop systems that provide users with a set of domain knowledge *pre-selected to solve users' problems*. The present research, on the other hand, focuses on how to support *users' development of problem-solving abilities.*

3 Mechanisms for Implementing the LBA in Design Processes

In order to help designers apply the concept of LBA in their design processes, this research proposes some mechanisms for implementing the LBA. The basic mechanisms for integrating the LBA in product/system designs are depicted in Fig. 1. The LBA mechanisms are divided into two main processes. The first part is for defining *what kinds of* domain knowledge should be represented for users during task performance. This involves such sub-processes as structuring domain knowledge, analyzing tasks, and mapping between domain knowledge and tasks. The second part describes the processes for defining *how to deliver* specific domain knowledge by appropriately representing this knowledge according to users' contextual conditions. This part involves sub-processes such as defining contexts, defining representational factors, and mapping between contexts and representational methods.

Fig. 1 also shows the boundaries of this research, to what extent the present research addresses system design processes and what kinds of external information or mechanisms should be input in order to complete a full system design process. In other words, the LBA methodology is not intended to constitute self-sufficient and independent design processes in and of itself; rather, it is integrated in regular design processes as a new and important constituent. At a more detailed level, the LBA-implementing mechanisms involve the following steps.

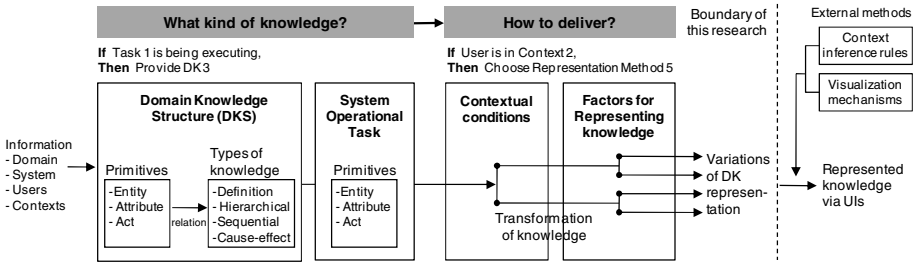

Fig. 1. Mechanisms for implementing the LBA in design processes

Understanding Domain Knowledge Structure (DKS). In the first phase, designers should understand and represent the DKS in a way that a product/system can recognize and process it. In this phase, designers need to define what kinds of particular domain knowledge should be delivered during users' interaction with the product/system. In order to represent domain knowledge, several sets of elements are defined.

1. Ontological elements of knowledge representation: act, entity, attribute, and relation
2. Elements for DKS description: goals, actions, entities and procedures
3. Types of knowledge: definition, hierarchical, sequential, and cause-effect knowledge – by making relations among the DKS elements

The present research describes domain knowledge structure (DKS) by adapting the format of task knowledge structure (TKS), which was proposed by [9]. The purpose of TKS is to represent the structure of knowledge involved in users' task performance. By using a similar format to TKS, the DKS aims to describe the structure of domain knowledge. Fig. 2 describes how domain knowledge can be described differently from operation knowledge in the same format. Although this research adopts the format of [9], the intention of the representation is somewhat different. TKS is described for developing a system that can support users' task execution in using the system, based on *designers' (systems')* understanding of users' task knowledge. The present research, on the other hands, aims to enhance *users'* development of domain problem-solving abilities.

Analysis of System Operational Tasks. In the second phase of the LBA mechanisms, users' system operational tasks are analyzed in order for designers to consider delivering domain knowledge that can support particular stages of system operations. In the system design field, task analysis is typically used to identify users' tasks in their goal-oriented behaviors. Since users need to obtain particular domain knowledge to achieve goals, and because goals can be attained by conducting a series of tasks when using a product, designers should also analyze users' tasks involved in operating the target product.

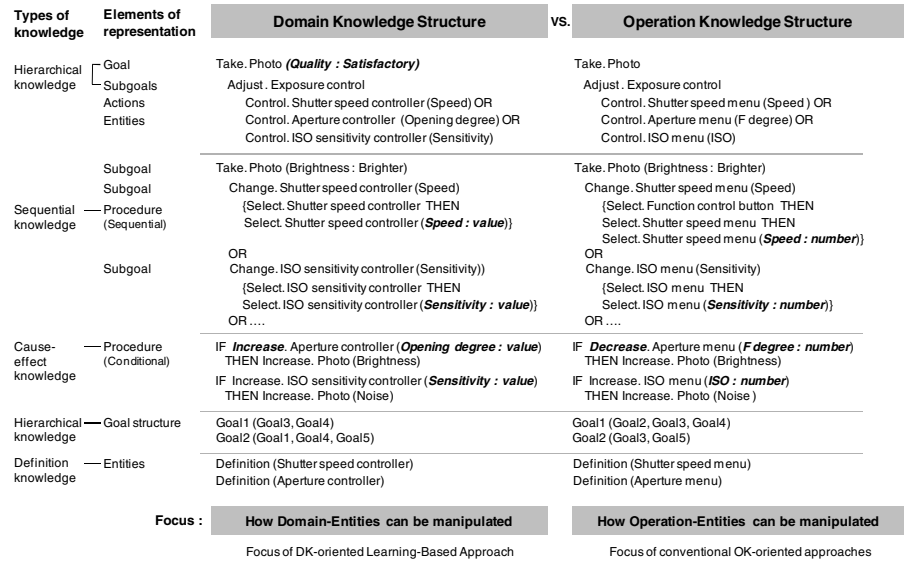

Fig. 2. Comparison the representation of domain knowledge-oriented representation with operation knowledge-oriented representation

Mapping between Tasks and Domain Knowledge. The essence of the mapping mechanism in this phase is based on the idea that a system should help users be aware of what *domain entities (defined in DKS)* they are actually manipulating, how their manipulations relate to the resulting quality, and to reach this awareness while they are controlling a particular *operation entity (defined in task analysis)*. In other words, by using a LBA-applied product, users will be able to continuously learn what kinds of *domain problem-solving mechanisms* they are actually manipulating while they are controlling *the mechanisms of interface elements* to achieve their goals. The consequences of these processes can be described as such:

"If a user is executing task 1, then domain knowledge 3 and 5 should be provided."

Delivering Domain Knowledge through User Interface Designs. The next consideration should be choosing the most effective methods for representing specific domain knowledge objects in a way that can facilitate users' acquisition of the knowledge through interaction. Since the appropriateness of the representation methods of particular knowledge is thought to depend on whether the representation method is matched to the user's contexts, designers need to define particular types of users' contextual conditions. Then, designers should also create variations of representational methods by categorizing factors for representing knowledge and defining the properties for each factor so that a system can infer appropriate representational methods according to recognized contexts. The results of mapping between contextual conditions and representational methods can be described as such:

"If a user is in context 2, then domain knowledge 3 will be represented in method 5."

Developing Prototype Specifications. By selectively employing some part of the design information produced in the prior phases, designers will be able to develop prototypes that reflect particular concerns of the project.

4 Development of Domain Knowledge-Based Tool (DKT)

In order to help designers apply the LBA mechanisms through the design processes, this research develops a computerized design-supporting tool, the DKT (Domain Knowledge-based design Tool).

4.1 Overview of the DKT

The DKT is developed to incorporate the concept of the LBA into product and system design processes. By using the computational tool, designers will be able to apply the LBA mechanisms to their practices more efficiently and effectively. The DKT provides step-by-step functional modules, each of which corresponds to the specific step for implementing the LBA mechanisms described above. The DKT consists of separate but interrelated five-phase functional modules, as shown in Fig. 3: *DK Structuring, Task Analyzing, Task-DK Mapping, Knowledge Delivering, and Prototype Specification.*

Based on the conceptual model shown in Fig. 3, basic components of the tool and the information flow amongst the components are defined and illustrated in a block diagram, as shown in Fig. 4. *Database layer.* Database components in the database layer are classified into two groups: 1) a primitive database group and 2) a composite database group. Database components in the primitive database group are designed to store foundational data such as *DK Entity, Operational Entity, Context, and Factor for Representation*. These types of data cannot be further deconstructed. On the other hand, database components in composite database groups are designed to accumulate data that are generated using the components in the primitive database group and/or other components in the composite database groups.

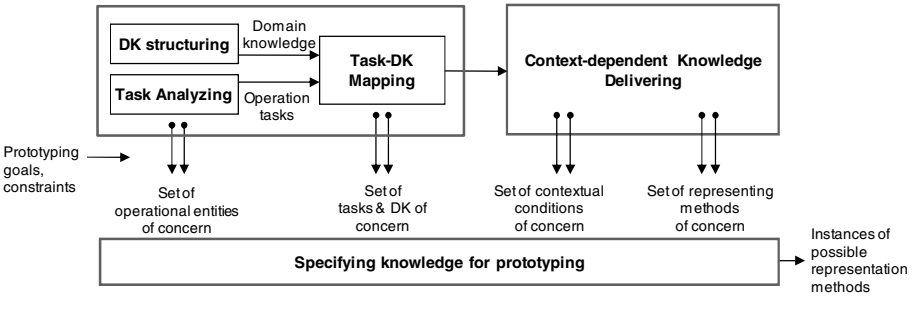

Fig. 3. Conceptual model of the Domain Knowledge-based Design Tool (DKT)

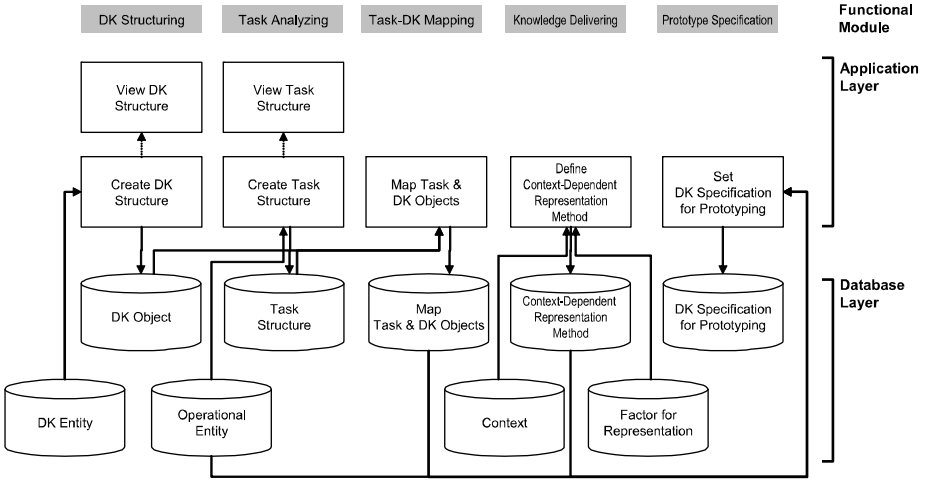

Fig. 4. System architecture of the DKT

4.2 Functional Modules of DKT

In order to more effectively explain the details of using functional modules of the DKT, consistent examples drawn from the task domain of photo-taking are used throughout all main features of the tool.

1) ***DK (domain knowledge) Structuring.*** Fig. 5 shows the full screen of the *DK Structuring* module. The *DK Structuring* module consists of two main features: – in the upper area of the screen – and composing DK objects based on the defined entities – in the lower area of the screen.

2) ***Task Analyzing*****.** This module allows designers to define operational entities of a particular product and the structure of users' tasks that are related to using the entities.

3) ***Task-DK Mapping.*** This module enables designers to specify particular DK corresponding to each operational task. As shown in Fig. 6, the functions of this module are divided into the *Composed Task* section (left) and the *Relevant Domain Knowledge Object (DKO) for Task* section (right).

4) ***Knowledge Delivering.*** In this module, designers can generate variations of context-dependent knowledge representation methods. This consists of three sub-modules: *Creating Context*, *Creating Factors for Representation*, and *Defining Context-Dependent Knowledge Representation Method.*

5) ***Prototype Specification.*** The *Prototype Specifier* allows designers to easily recognize available information elements in the pull-down lists so that they can select specific sets of information necessary for prototyping.

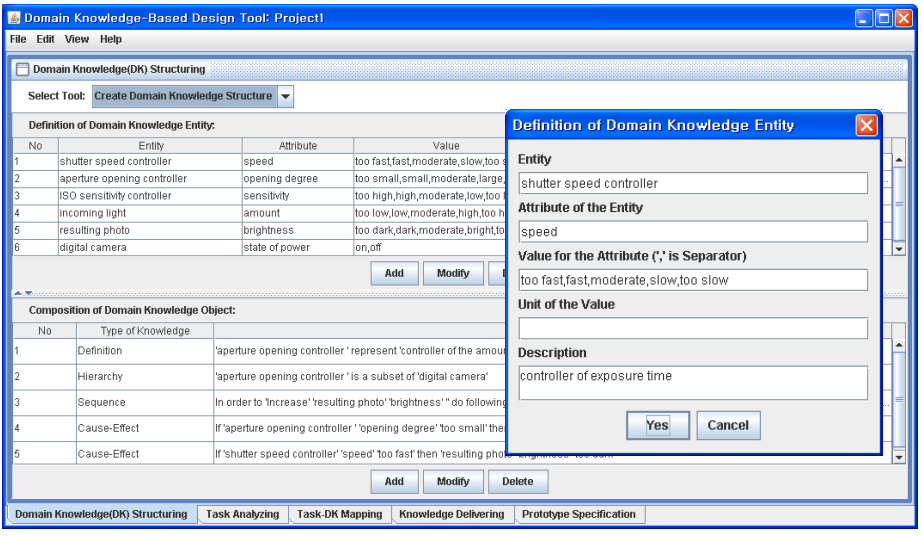

Fig. 5. *Domain Knowledge (DK) Structuring* module of DKT

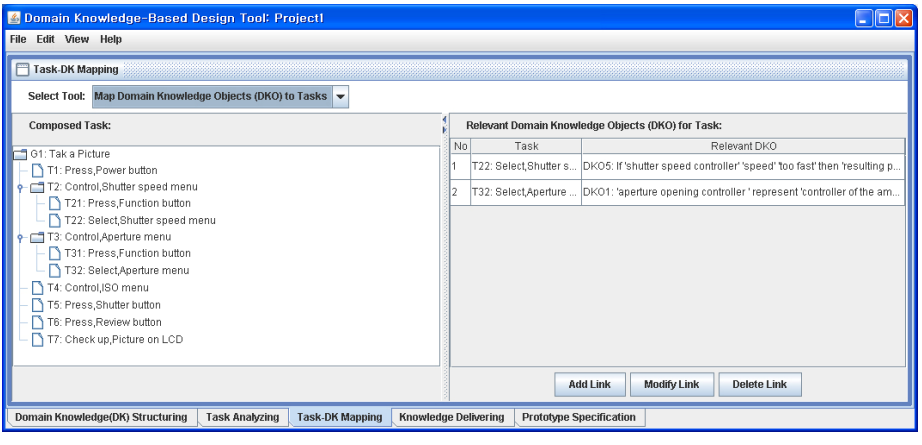

Fig. 6. *Task-DK Mapping* module

5 Conclusions

The Learning-Based Approach (LBA) to interactive product and system design is proposed to provide designers with a new viewpoint of users' knowledge development in interaction, as well as applicable methods and tools for developing a product from this perspective. The LBA emphasizes the significance of users' domain knowledge in using a product/system; when users have enough knowledge about problem-solving mechanisms in the task domain, they will be able to figure out how to achieve a goal by creatively manipulating their interaction behaviors to meet their variable needs. Therefore, system designers should consider how to help users

develop their own domain knowledge through interaction. By obtaining sufficient domain knowledge through the interaction with a system, users will be able to attain high quality results, richer user experiences, and overall higher levels of satisfaction. In this paper, mechanisms for implementing the LBA are proposed, including 1) describing domain knowledge structure in a way that a product/system can recognize and process it, 2) analyzing users' tasks for operating a product, 3) mapping domain knowledge onto relevant particular tasks, 4) delivering domain knowledge to users via user interfaces by flexibly representing domain knowledge according to users' contextual conditions, and 5) developing prototype specifications by selectively employing some part of the design information produced in the prior phases. In order to help designers incorporate the mechanisms in design processes, this research develops a computerized design-supporting tool. The future research will introduce the prototype development of a case product. Through the prototyping process, the user interface outputs that are drawn on the LBA mechanisms will be demonstrated and evaluated.

References

1. Choi, J., Sato, K.: Interaction as Learning Process: Incorporating Domain knowledge into System Use. In: 5th NordiCHI, pp. 73–82. ACM Press, New York (2008)
2. Card, S.K., Moran, T.P., Newell, A.: The Psychology of Human-Computer Interaction. Lawrence Erlbaum Associates, Hillsdale (1983)
3. Norman, D.A.: The Design of Everyday Things. Basic Books, New York (1988)
4. Carroll, J.: The Nurnberg Funnell: Designing Minimalist Instruction for Practical Computer Skill. MIT Press, Boston (1990)
5. Nielsen, J.: A Meta-Model for Interacting with Computers. Interacting with Computers 2, 147–160 (1990)
6. Engeström, Y.: Activity Theory and Individual and Social Transformation. In: Engeström, Y., Miettinen, R., Punamäki, P. (eds.) Perspectives on Activity Theory. Cambridge University Press, Cambridge (1999)
7. Kaptelinin, V.: Activity Theory: Implications for Human-Computer Interaction. In: Nardi, B.A. (ed.) Context and Consciousness: Activity Theory and Human-Computer Interaction, 3rd edn. MIT Press, Boston (2001)
8. Wu, Y.: What else should an HCI pattern language include? In: Pattern Languages for Interaction Design: Building Momentum, CHI 2000, The Hague, The Netherlands (2000)
9. Johnson, P.: Human-Computer Interaction: Psychology, Task Analysis and Software Engineering. McGraw Hill, London (1992)