Conceptual Modeling for Interaction Design

Qingyi Hua, Hui Wang, and Matthias Hemmje

Fraunhofer – IPSI
Dolivostr. 15, D-64293 Darmstadt, Germany
{hua, hwang, hemmje}@ipsi.fhg.de

Abstract

A usable interactive system provides its users with presentation and manipulation of useful concepts for solving their problems at hand without becoming bogged down in accidental features of user interface. It implicates that interaction design is directed by the acquisition and representation of knowledge about the context of use in a way that can be traced back to the users' problem-solving activity. In this paper we focus on conceptualization of that activity. The conceptualization is characterized by a set of ontological terms that capture a continuum between user tasks and problem domain in a declarative way, and by a framework of conceptual models that describe a system on a very abstract level without being limited to particular design models.

1 Introduction

In HCI a task is defined as an activity performed to achieve a goal (Welie, Veer & Elens, 1998). However, there is at least one other more or less parallel activity during the performance, that is, the mental representation and processing of the problem to be solved at hand. Based on the beliefs on the problem domain, the problem-solving activity generates the goal and intentions performing the task in terms of Norman's action theory (Norman, 1986). Users could distract their mental effort from their working situations to the manipulation of the user interface if a system fails to provide the relevant concepts to the problem-solving context. As a result, usability relies on the system provides appropriate content at appropriate time during the process of problem solving.

The problem-solving activity relies on the knowledge structures supporting understanding (schemas) and the mechanisms used to organize that knowledge (plans) (see (Robilland, 1999) for a cognitive discuss). In AI terms, a problem solver requires two types of knowledge (Chandrasekaran, Josephson, & Benjamins, 1998): domain factual knowledge (objects, events, relations, processes, etc) and problem-solving knowledge about how to achieve various goals, such as problem-solving methods. However, mental processing and representation differs from a system description in many ways. For example, human can have semantically much richer, and more complex schemas than the system, whereas the amount of knowledge that can be handled by a human mind at any given time is limited to 5±2 chunks according to psychologists. As a result, it is fundamental to develop a shared understanding between the users and designers according to the context of use, such as the intended users, their tasks and environments.

Contextual development has been recognized to be crucial for meeting the demands of user-centred systems design (Stary, 2000). Two categories of approaches can be found in HCI: task-based and context-oriented. Task-based approaches, e.g. (Welie, et al., 1998), capture only the hierarchical properties of tasks without memory reminding why the tasks are executed or decomposed, whereas context-oriented ones, e.g. (Maguire, 2001), describe working situations

without mental abstraction. However, users guide their tasks to be done by making plans. The properties of plans are anticipation and simplification (Robilland, 1999): a heuristic nature without a detail analysis of the situation, optimal use of memory by keeping only critical properties of objects and events, and higher control level without details of the activity being processed.
In this paper, we present our initial results on conceptualization of knowledge about the context of use. The purpose of the conceptualization is to develop a shared understanding and representation of that knowledge between stakeholders. Section 2 introduces a set of ontological terms that capture the conceptualization and that act as meta-knowledge for the knowledge acquisition from a user's perspective. Section 3 describes the conceptual models and demonstrates a case study. The description of the system is done with the conceptual models on a very abstract level. On this level, we talk of mental states instead of system states, of intentions instead of task decompositions. Consequently, this level allows us to represent knowledge about the context of use in a way that can be traced back to the users' problem-solving ability. On the other hand, it makes possible provide opportunities and constraints for interaction and system design without being limited to a special set of design models. Finally the conclusions are presented in section 4.

2 Ontological assumptions of problem-solving activity

The notion of ontology represents knowledge shared and reused in some domain of interest. The essence of the notion is in (Gruber, 1993): an ontology is an explicit specification of a conceptualization. A conceptualization makes some assumptions about concepts and their relationships in the domain to be modelled. In AI and software engineering, for instance, objects, events, processes, and relations are general terms for modelling domain knowledge. Recently ontological assumptions have been also proposed for modelling problem-solving methods in AI. For instance, (Chandrasekaran, et al., 1998) defines an ontology of problem solver and an ontology of methods for establishing assumptions of the knowledge structures used by a problem solver and of the control mechanisms to use those structures.
However, the difference between the human mind and the system lies in the capability of knowledge processing and representation. To make the system usable, the phenomenon of knowledge chunking has to be taken into account. Chunks are general and do not refer to the information content of knowledge. This implicates that they are a measure of the unrelated knowledge that can be processed naturally (Robilland, 1999). The phenomenon illustrates that the problem-solving knowledge used by the human mind is organized on a more abstract level than the level of domain knowledge. For the mediation of the two different levels, the system has to organize contextual information about task-performing states, or domain states to meet the users' needs. In the remainder of this section, we define a set of primitive terms in order to match the elicitation and representation of the needs.
Term 2.1. Domain object/action/state. We apply the traditional OOA definition for the three terms in general. In particular, we define that a (problem) domain state is a set of values of state variables representing objects in the problem domain.
Term 2.2. Task object. A task object is supposed to match a mental representation describing a problem domain state that a user believes or pursues. Task objects can have the same abstract mechanism and form as domain objects. For example, a task object can be specialized as an entity, relationship, or event depending on whether it is autonomous, subordinate, or instantaneous, respectively. Task objects are essential for the problem-solving activity because they are a way of representing the user's knowledge about the problem domain from a view of task. In other words, they represent the content of tasks to be undertaken. In the domain of a hotel, for example, domain objects can be room, guest list and so on, whereas a guest may have particular concepts about the domain states, such as availability, preference and so on, when she wants to make a reservation. It

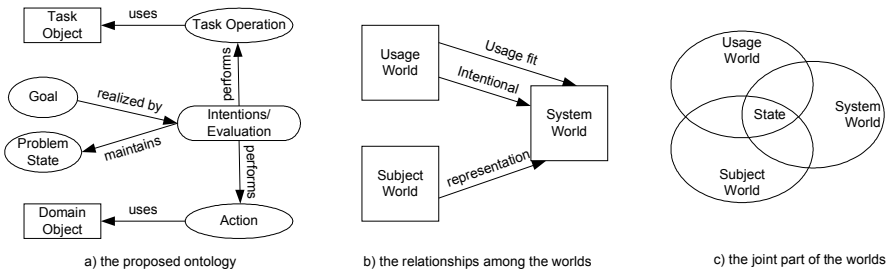

Figure 1: the proposed ontology and its relationship with the three worlds

is also desirable to anticipate possibilities, to determine the current situation, and to remember working history by use of task objects. As a result, task objects provide information about how to represent knowledge about problem domain on the mental level.

Term 2.3. Task operation. A task operation is supposed to match a mental operation representing an input-output relation over task objects. Task operation applications determine state transitions in the domain of task objects. Like a domain action, task operations can be characterized by pre/post conditions, and trigger conditions that capture the elementary state transitions. A task operation can be specialized as *required* or *requested* dependent on if it changes domain states of task objects and generates an event or only queries the task domain, respectively.

Term 2.4. Problem state. The process of problem solving creates, uses and changes a number of task objects referring the states of the affairs. A problem state is a set of values of state variables (e.g. knowledge chunks) representing these task objects. Problem states include information about current goals. Problem states also include all information generated during the process of problem solving, such as beliefs, desires and so on.

Term2.5. Goal. A goal is supposed to match a mental representation that the user has an attitude describing an expected problem state (e.g. 'Make reservation' is the goal in the above example). Goals are realized by intentions and evaluations. The important point is that a goal is some desired end state to be wished by the user, rather than a state to be reached after successful execution of a task in traditional task analysis. As a result, traditional analysis cannot answer if a goal can be achieved, and how failures can be recovered.

Term 2.6. Intention/Evaluation. An intention (or evaluation) is supposed to match a mental thread of problem solving and of maintaining problem states. A mental thread performs a set of required or requested task operations dependent on whether it is an intention realizing a current goal or an evaluation establishing a belief of current domain state, respectively. In general, mental threads change problem states, which in turn, invoke a set of domain actions to complete intentions or evaluations. Intentions (or evaluation) are characterized by trigger and stop conditions. As shown in Fig. 1a, intention/evaluation is significant in that it represents a basic unit of problem-solving knowledge, and establishes a continuum between the different views of task and domain.

(Jarke, Pohl & Rolland, 1994) identifies three worlds and the possible relationships that need to be understood and modelled (Fig. 1b). For the sake of usability, we argue that the joint part of the three worlds has to be taken into account, that is, the states of affairs among the worlds (Fig. 1c). We have identified the joint part between the usage world and the subject world by the definition of the term *problem state*.

3 Conceptual models on the knowledge level

A conceptual model is a set of concepts and their relationships, which embodies the view captured by a set of ontological terms with respect to some domain of interest. The view of the proposed

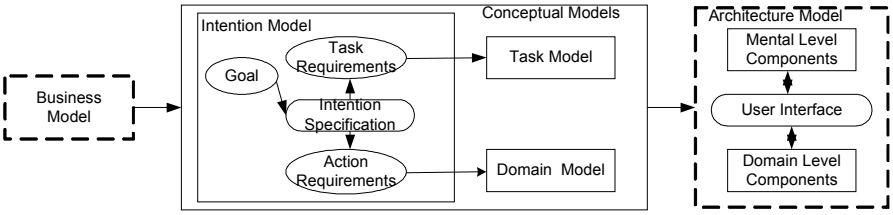

Figure 2: the proposed framework of conceptual models

framework is intention-dominated, that is, it is concerned with what goals to achieve instead of how to achieve these goals.

As shown in Fig. 2, the business and architectural models (the dashed-line boxes) do not belong to the framework. The business process model provides contextual information for the conceptualization, whereas the architecture model is used for the realization of the conceptual modelling. Because of the limited space, in the remainder we will not mention them. The models in the framework are described as follows.

Model 3.1. Intention model. An intentional model consist of a goal the model intends to achieve and three interrelated parts:

- Intention specification. Intention specification is a cluster of intentions/evaluations without sequencing, in which intentions and evaluations realize the goal, and validate whether the goal is achieved, respectively. For interaction design, it provides information how to organize dialogue and presentation of user interface towards the current goal it realized.
- Task requirements. Task requirements are a collection of task operations invoked by the intention specification without sequencing. Task sequencing is determined by intention specification. For interaction design, task requirements determine the actions that a user may initiate.
- Domain requirements. Domain requirements are a collection of actions invoked by the intention specification without sequencing. Domain requirements in general are independent of task requirements, although they are indispensable for system design.

Model 3.2. Task model. A task model is a collection of task objects that represent the content of tasks, and that represent all information that needs to be presented. Computationally, the model represents a working memory that could be seen as the extension of the users' working memory.

Model 3.3. Domain mode. A domain model is a collection of domain objects that represent domain factual knowledge, and it is therefore independent of the task model. In our framework, the model represents only the underlying information that a system should maintain for the purpose of functionality, and it is therefore, usually unperceivable by the users.

In remainder of this section we demonstrate a case study of hotel reservation. For simplicity, we assume that the users' needs are to make a room reservation. The modelling process starts from the artefacts of business process modelling. In general, the artefacts include a domain process model and a domain model. For the sake of the limited space, only the domain model in our example is show in Fig. 3a.

The modelling process includes two complementary steps of intention/evaluation analysis and task elicitation by analysis of the artefacts of business modelling and by asking the users questions repeatedly: *What* do you *intend* to *do* when doing this thing? *What* do you *expect* to *get* from doing this thing? The *what* in general means the content of task (i.e. task objects) and the *do* (and the *get*) means task operations. The users can usually answer such questions because these questions are on the same level as their plans of guiding their tasks to be done. For example, a user may answer that 'I want to know if a room is available in this period as I am making a call to this

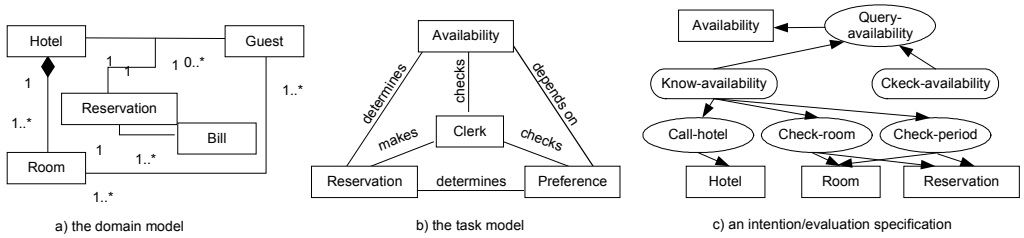

Figure 3: a part of conceptual models in the case study

hotel'. From this answer we can find a task object 'availability' with properties 'room', 'hotel' and 'period', and an operation 'query-availability'. The properties represent a set of state variables relevant to the domain knowledge in Fig. 3a. It is notable that we aim at eliciting which intentions should be realized, rather than exploring how these intentions are realized by control knowledge as it does in traditional task analysis.

Fig. 3b depicts all the task objects and their relationships by repeating the steps of the process. It is not too difficult to establish the relationships between the identified intentions/evaluations (and the operations), and domain actions because each task object, more precisely, its properties make explicit references to domain objects, or their properties. For example, fig. 3c shows a specification for the intention of knowing availability and the evaluation of checking availability.

4 Conclusions

We are not developing a brand-new method, but reusing the traditional OOA technology on the knowledge level. The ontological terms we defined in this paper emphasize acquisition of knowledge about the content of task domain, rather than about control mechanisms in that domain. The continuum defined by these terms represents a declarative specification to mediate activities of problem solving and task performing. On the knowledge level, the proposed conceptual models can represent the requirements for a system in the same state space as the task knowledge, which makes possible build the system to delegate user tasks on the same level.

References

Chandrasekaran, B. Josephson, J. & Benjamins, R. (1998). Ontology of Tasks and Methods. In the Proceedings of KAW'98, Voyager Inn, Banff, Alberta, Canada.

Gruber, T. R., (1993). A translation approach to portable ontology specification. Knowledge Acquisition, 6(2), 199-221

Jarke, M. Pohl, K. and Rolland, C. (1993). Establishing visions in context: towards a model of requirements processes, Proc. 12th Intl. Conf. Information Systems, Orlando, Fl,

Maguire, M. (2001). Methods to support human-centred design. Int. J. Human-Computer Studies, 55, 587- 634

Norman, D. (1986). Cognitive engineering. In: Norman, D. and Draper, S (Ed.): User centered system design (pp. 31-61), Lawrence Erlbaum Associates, Hillsdale, NJ.

Robilland, N. (1999). The role of knowledge in software development. Communication of the ACM, 42 (1), 87-92

Stary, C. (2000). Contextual prototyping of user interfaces. In the Proceedings of ACM DIS'00, 388-395

Welie, M., Veer, G., & Elens, A. (1998). An ontology for task world models. In the proceedings of DSV-IS'98, Spriger Verlag, 57-70