Task Ontology Makes It Easier To Use Authoring Tools

Mitsuru Ikeda Kazuhisa Seta Riichiro Mizoguchi
The Institute of Scientific and Industrial Research
Osaka University
8-1, Mihogaoka, Ibaraki, Osaka, 567,
Japan

Abstract

The main purpose of this paper is to illustrate the characteristics of ontology-based authoring tools for Computer Based Training (CBT) systems. It has two major advantages as follows. (A) It provides human-friendly primitives in terms of which users can easily describe their own model of a task(descriptiveness, readability). (B) It can simulate the abstract behavior of the model in terms of conceptual level primitives (conceptual level operationality). In this paper, we will discuss the basic issues on the concept of task ontology and then describe the design principle of an ontology-based authoring tool for Computer Based Training (CBT) systems.

1. Introduction

Recently much attention has been paid to the notion of "ontology" in the expectation that it can serve as the new, strong foundation of knowledge engineering. In the conventional approach to theory of knowledge, to give the operational semantics of knowledge representation has been regarded as of major importance and the analysis of contents of knowledge has been considered to be subordinate to it. To solidify the foundation of knowledge engineering, however, many researchers , especially in the field of knowledge sharing and reuse, has strongly felt necessity of the change of such a way of thinking. The key to the problem is to understand the essential interaction between "form" and "contents" on equal importance. This implies that deep understanding of "content" will give us new insight into design of knowledge representation. The notion of "ontology" can be key to this issue.

The ultimate goal of research on ontology is to give the full picture of theory of knowledge. To make improvements in the study of this difficult issue, of course, it is important to accumulate huge amount of "contents", and develop sophisticated ontology representation language as fundamental "form" of knowledge.

The same thing applies to the field of intelligent educational systems (IES). Building an IES requires a lot of work. At the present situation, however, it is always built from scratch. Few functional components are reusable and we cannot compare or assess the existing systems. Only existing contribution to the solution of the problem can be found in study of the authoring tools for educational systems. However, it is considered questionable whether substantial benefit for the authors engaged in the complex task may be expected or not, since most of existing authoring tools do not satisfy the requirements for the authoring tools as shown below.

- To provide human-friendly primitives in terms of which authors can easily describe define their own skeleton of IES.
- To give appropriate guidance to authors based on the established principle of the educational task by checking the rationality of the skeleton of IES.
- To show the dynamic behavior of the IES in conceptual level by which the authors can examine its validity.

We think the key to the solution of the problem is intelligent support based on "task ontology" which serves as a theory of vocabulary/concepts used as building blocks for knowledge-based systems. The issues here also include how to represent what we know about the fundamental characteristics of an IES as "task ontology" and how to integrate it into intelligent authoring tools. Our solution is integration of an ontology construction environment CLEPE as a part of the authoring tool we have developed. CLEPE provides us with all the functions needed to satisfy the requirements shown above.

The most important role of CLEPE is to lay the theoretical foundation for IES development process. It maintains continuity from author's conceptual understanding of educational task to the computational semantics of IESs. It provides human friendly vocabulary for authors to describe the educational task. For the authoring tools, on the other hand, it specifies the computational semantics of vocabulary and also provides a set of components represented in terms of both conceptual primitives and object-oriented code fragments.

The goals of our research on task ontology are to exemplify the benefits of task ontology through the development of an ontology based authoring tool for Computer Based Training (CBT) systems. In this paper, we will discuss the basic issues on the concept of task ontology and then describe the design principle of an ontology-based authoring tool for Computer Based Training (CBT) systems.

2. What is task ontology

Now let us go into the detail of task ontology. Generally ontology is composed of two parts, that is, taxonomy and axioms. Taxonomy is a hierarchical system of concepts and axioms are rules, principles, or constraints among the concepts. From the viewpoint of the ontology use, axioms specify the competence of an ontology. In other words, a class of the questions to which the answers can be derived from the axiom specify the competence of the ontology.

Following the analogy to natural language processing, we can easily understand the role of task ontology as a system of semantic features to represent the meaning of the problem solving description. The advantages of the integration of task ontology is as follows:

A. Task ontology provides human-friendly primitives in terms of which users can easily describe their own task (descriptiveness, readability).
B. The system can simulate the problem solving processes at the conceptual level and show users the execution process in terms of conceptual level primitives (conceptual level operationality).
C. The system makes the task description runnable by translating it into symbol level code (symbol level operationality).

For the moment, it may be useful to look more closely at the functional feature of task ontology. Here, let us introduce three models M(A), M(B), and M(C), which embody the functions A, B, and C listed above, respectively. According to the analogy of natural language again, M(A) corresponds to sentences of natural language, M(B) is an internal model of intended meaning represented by the sentences, and M(C) has a capability to simulate the dynamic, concrete story implied by the sentences.

From now on, M(A), M(B) and M(C) are called "lexical level model", "conceptual level model", and "symbol level model", respectively. Lexical level model mainly deals with the syntactic aspect of the problem solving description, and conceptual level model captures conceptual level meaning of the description. Symbol level model corresponds to runnable program and specifies the computational semantics of the problem solving. Ontology at the top layer is called lexical level ontology in terms of which M(A) is represented. Ontology at the middle layer is called conceptual level ontology which specifies computational semantics of M(B). Lexical level ontology specifies the language in terms of which users externalize their own knowledge of the target task, while conceptual level ontology is an ontology which represents the contents of knowledge in their minds.

Figure 1 shows a taxonomy of terms included in lexical level ontology. Because of the space limitation, here we just show a part of the CBT ontology which we are building now (see [Mizoguchi et al., 1996] for a more detailed account of IES ontology). All the terms of lexical level ontology are organized into word classes, such as, generic verb, generic noun, etc. :

- Generic nouns[N] representing objects reflecting their roles appearing in the problem solving process,
- Generic verbs[V] representing unit activities appearing in the problem solving process,
- Generic adjectives modifying the objects.

Task ontology for computer-based training, for example, looks as follows:

Nouns: "Problem", "Scenario", "Answer", "Example", "Accident", "Operation", "Hint", etc.
Verbs: "Provide", "Show", "Ask", "Simulate", "Give", etc.
Adjectives: "Unsolved", "Easy", "Correct", "Counter" etc.

Verbs are defined as a set of procedures representing its operational meaning. So, they collectively serve as a set of reusable components for building IESs.

In the conceptual level ontology, the concepts to represent our perception of problem solving are organized into generic concepts, such as, activity, object, status, and so on. On top of that, a collection of symbol level CLOS code fragments are organized in symbol level ontology. There are some prerequisite relations among these three levels. Intuitively generic verb, generic noun, and generic adjective in lexical level ontology correspond to activity, object, and status in the conceptual level one, respectively. For each activity, at least one code fragment is prepared in the symbol level.

Thus, task ontology provides primitives in terms of which authors can describe their own educational task model. Using their own model, authors can easily put a piece of teaching material into the appropriate educational context, since it provides authors with abstract educational roles of various objects which could be easily instantiated to domain-specific objects. In the above examples, "problems", "examples", and "hints" represent such objects with educational roles.

One of the most important characteristics of task ontology is that meanings of verbs are also defined at the symbol level, that is, at least one executable code is associated with each verb to enable semiautomatic generation of runnable educational application.

Thus, roles of IES task ontology can be summarized as follows:

(1) To provide vocabulary/concepts in terms of which one can compare and assess existing IESs.
(2) To formalize training tasks.
(3) To specify the tutoring/training context which contributes to making it easy to put domain knowledge into a right context, since it provides us with abstract roles of various objects which could be instantiated to domain-specific objects.
(4) To provide reusable components for IES design and development.
(5) To enable translation of the knowledge-level description of the skeleton of IES into symbol-level executable code.

3. An Authoring tool based on CBT task Ontology.

Authors' work of IES systems can be roughly sketched out in the following three stages.

1. Design the skeleton of the CBT process of interest.
2. Compile training material or training aids, for example, notes, pictures, sounds, simulator, etc. into the skeleton.
3. Adjust the control of CBT process in detail.

Of course, the distinction among the stages is not necessarily important for authors. Rather strict distinction sometimes becomes disadvantage of authoring tools. The reason for making the distinction is to focus our discussion to the author's work which enjoy the ontology support.

The existing authoring tools in the literature [Inui et al, 1990], [Major, 1995], [Murray and Woolf, 1992] seems to cover all the three stages. Especially, some authoring tools for ITSs(for example, [Van Marcke and Vedelaar, 1995]) provide sophisticated set of descriptive primitives and corresponding computational components of IES for authors. When looking carefully at the support functions provided by them, however,

Tutoring Actions
- Help[V]
  - Answer (+ to the questions/requests)[V]
  - Help (+learner to overcome impasses)[V]
- Getting learners motivated[V]
  - Encourage[V]
  - Compliment[V]
- Exercise
  - Select (+problems) [V]
  - Give (+problems) [V]
- Guide
  - Show/Prvesent[V]
  - Explain [V]
  - Simulate [V]
  - Suggest(+Hints[N]) [V]
  - Asking [V]
- Evaluate/assess[V]

Objects of Training task
- Scenario[N]
  - process-oriented scenario[N]
  - topic-oriented scenario[N]
- Problems
  - Similar problems[N]
  - Easier problems[N]
  - More difficult problems[N]
- Explanation
  - Deep/Shallow knowledge level[N]
  - Knowledge necessary for solving the problem[N]
  - Prerequisite knowledge[N]
  - Simulation result[N]
- Hints
  - Deductive hints
    - Verification process[N]
    - Intermediate solution[N]
    - Abstract examples[N]
  - Inductive hints
    - Kinds of bugs[N]
    - Counter examples[N]
    - Extreme examples [N]

Learner's state
- Phase in learning process
  - Phase of concepts learning
    - Learning of new concepts[N]
    - In-depth understanding of the new concepts[N]
    - Assimilation of the concept[N]
  - Phase of problem solving learning
    - basic schema[N]
    - Learning advanced schema[N]
    - Learning solving a problem by using a known schema[N]
    - Creating a new problem solving schema[N]
- Knowledge state
  - Coverage of concepts/skills mastered[N]
  - Degree of mastery[N]
  - Belief[N]
  - Connection strength of relations[N]
  - Location of bugs
    - Conditions/Premises/Factors/etc.
    - Actions/Conclusions/Hypotheses/etc.
    - Order of rules[N]
  - Types of bugs
    - Lack of concepts/relations/rules[N]
    - Insertion of concepts/relations[N]
    - Incorrect concepts/relations[N]
    - Incorrect order of operations[N]

Teaching material knowledge
- Domain knowledge
  - Nodes(category)
    - Concepts[N]
    - Rules/Problem solving schema[N]
    - Principles[N]
    - Strategy[N]

....

Figure 1. Lexical level CBT task ontology (Partial)

we found that little effort has been done to give active guidance based on the explicit first principle on education. The reason of this could be lack of explicit "ontology" as design rationale of IESs. To embody active support of authoring process, the authoring tool should know the rationale in the form of meta description of the design.

Based on the above consideration, we have tried to build IES ontology. Although the step we have made is not large, we believe that its implications to the future research in IESs area is not small. CBT ontology we discuss here uses the IES ontology as its core. We organize a set of concepts specific to training system under the IES ontology. In the following sections, we try to show the significance of the CBT ontology. First, we present brief overview of *SmartTrainer* which have been developed as a prototype of Intelligent CBT system, then we will show the image of the authoring tool based on the CBT ontology built based on the analysis of *SmartTrainer*'s task.

3.1 *SmartTrainer*

SmartTrainer is a CBT System in the area of electric power systems. The target task of *SmartTrainer* is mainly to recover the accidents of substations in the electric power systems. When an accident happens, the electric power transmission will be interrupted, and the operators should recover it as quickly as possible. The operators should find the spot of the accident, continue to supply the electric power to some special places such as hospital, police station at once by borrowing some power from the other substations, find the causes of the accident and recover it within the limited time.

The goal of the training conducted by *SmartTrainer* is to improve capability of not only skill-based or rule-based reasoning but also knowledge-based reasoning. The set of the scenarios incorporated into *SmartTrainer* has been designed by the experienced trainers. In order to let the trainee master the principled knowledge, *SmartTrainer* let them do practice first and then teach them the first principle behind it adaptively to their mistakes, and finally, check their learning result by practice again. This is a form of "learning by doing."

SmartTrainer is basically composed of three models, that is, training process model, teaching materials model, learner model, which are part of our CBT task ontology.

3.2 An Ontology-based Authoring Tool

Among three stages of authors' work listed above, we have concentrated on the first one, that is, design process of the skeleton of CBT process. Hereinafter, we sometimes call the skeleton of the CBT as the *model* of a CBT system to emphasize the importance of existence the ontology which governs it. Therefore, the term *model* implies two ideas, one is that it has a set of general axioms to follow and the other is, in an ordinary sense, it abstracts the skeleton from the real CBT systems.

Basic teaching material prepared by the authors of CBT are the following

A) knowledge needed for target operation,

B) training scenario, and

C) simulator of a target system.

The authors compile these three items into the skeleton of a CBT system at the first stage. The typical work of the compilation is to put a fragment of knowledge(A) into the target operation context(B) appropriate for training. The most desired thing for authors is the established guideline on the compilation task, for example, the fundamental structure of the scenario, the typical style of questions, appropriate timing and method of learner modeling, and so on. As we discussed in the previous section, we have defined a set of terms to express the training task of the electric power system operation in the lexical level task ontology and the meaning of the terms in conceptual level task ontology. The author's task is to build a model of the intended training task in terms of the lexical level task ontology as a result of the compilation. Such terms and concepts are provided by an ontology maintenance system, which maintains the consistency between the task ontology

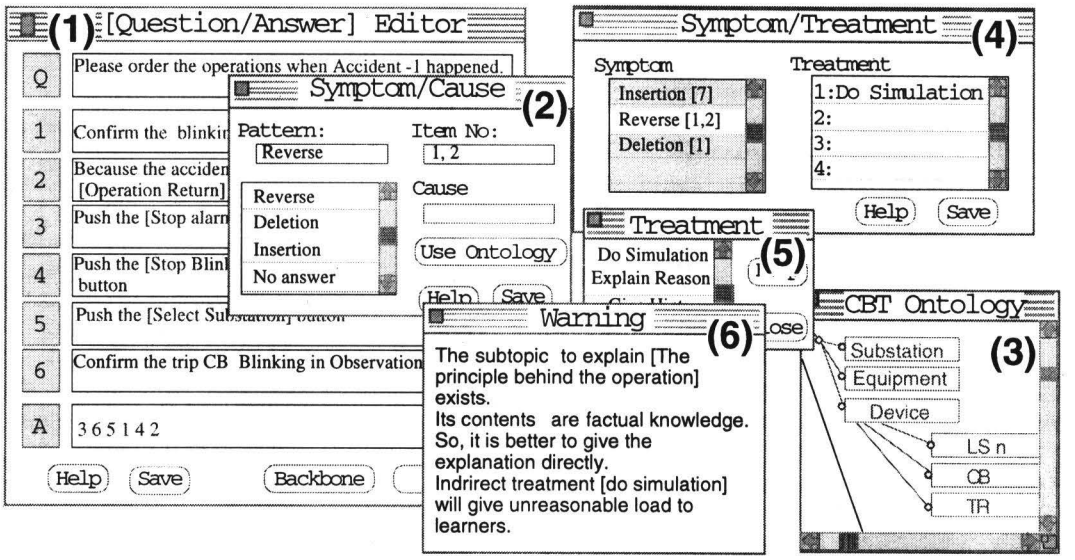

Figure 2 Interface of an Ontology based Authoring Tool

and the model built based on the ontology. As a part of our ontology engineering project, an ontology maintenance environment, called CLEPE, has been developed. Further details of CLEPE will be described later.

On the top of CLEPE, we have been developing an authoring tool for CBT systems. Figure 2 shows the image of the authoring tool interface. The interface is designed in full consideration of the intuitively clear way to express the skeleton of training tasks. In other words, the structure of the interface windows reflects the training task ontology and acts as the guideline on the author's task.

Figure 2 shows the snapshot of interface of the authoring tools we have developed. When the authoring tool is activated, the author is asked to create a training scenario which is a series of question and answer sessions. The window (1) is for editing a question/answer session. Before the window appears, the author was asked to select one from several question/answer types defined by CBT task ontology. In this case, the author selected "ordering question." And then he/she inputs the components of the question/answer, that is, "question", "items to be ordered" and "correct answer." After inputting the question, the author edits a set of learner's errors , called "symptom" in the window (2). The author inputs the possible symptoms as one of "reversing", "missing", "superfluity", which are defined as subcategories of the symptom for ordering question in CBT task ontology. In addition, the author is asked to clarify the cause of the symptom by selecting one node from the domain knowledge shown in the window (3). Since *SmartTrainer* adopts overlay learner model, the cause is formalized as missing of the selected node. Although we assume that domain knowledge is prepared in advance, the author can modify it at anytime when needed. Once the author complete inputting a symptom, the symptom/treatment window (4) will appear. In this window he/she specifies the treatment for the symptom. The possible treatments are listed in the window (5) based on tutoring strategy concept defined in CBT task ontology. The treatment selected by the author is "let him/her do operation". However, the selection violates the constraint on symptom/treatment rationality. It says that direct explanation is better than indirect treatment when the question asked to the student is not very difficult. The window (6) shows the rationality and recommends the "explain the principle" treatment instead.

From figure 2 one might get impression that the description of the question is too detail for the first stage of the author's work. Of course, the boundary between the first two stages of author's work is gray. Our intention of this is, however, that the textual information acts as just labels of the entities of questions. Without such labels, authors cannot structure the skeleton in their mind and also may lack the consistency of their design. Note that the textual information in the question frame is meaningless for ontology management environment at this stage.

3.3 Layers of Ontology

As we have discussed in section 2, task ontology is divided into three complementary partitions. From now on, we will focus on the two of them, that is, lexical level and conceptual level, which are the most important for authoring support function at the first stage of author's work. In figure 3, the two partitions are arranged in depth. The two partitions in depth are allotted different functions to each. Lexical level ontology specifies syntactical aspects and conceptual level specifies semantics of vocabulary.

In addition to that, task ontology has three layers arranged in vertical direction in order of the degree of task-domain dependency. That is, core task ontology with the least dependency and arranged at the lowest layer, task-type specific ontology at middle, and task-domain ontology at the highest.

Core task ontology lays foundation for upper layers by defining inherent concept needed for modeling all the types of problem solving. Task-type specific ontology is a system/theory of vocabulary/concepts for describing task models of a certain type of task, and task-domain ontology is one for de-

scribing domain models from the task-type viewpoint. We have defined the CBT(Computer Based Training) ontology which organizes a set of concepts specific to training system as a task-specific ontology. Based on the top-most layer ontology, authors build the model of task of interest. In our case, the model means a skeleton of an CBT system(*SmartTrainer*).

The relationship between ontology and model is not absolute one but relative one. For example, when building a task-specific ontology, it can be regarded as a model, while the core task ontology as an ontology for the world. That is to say, an object can be considered as an ontology, meanwhile it can also be considered as a model based on the relatively lower layer. In principle, a descriptive primitive of upper layer has richer meaning in the sense of human understanding than that of lower layer. Needless to say, the rigid computational semantics defined for the primitives at the lowest layer.

Thus, we could say an intended model of CBT systems is described based on the upper , front part of task ontology and the computational semantics of it is finally defined at the lowest, rear part.

To characterize each layer from the above viewpoint, we can ideologically divide the authors concerned with the ontology into four types.

(1) the authors who develop training systems (end authors).
(2) The authors who build task-domain ontology (task-domain ontology authors).
(3) The authors who build task-specific ontology (task-specific ontology authors)
(4) The authors who build core task ontology-based on core-task ontology (Core task ontology authors).

In our case,

(1) End author: the instructor of electric power system operation (instructor)
(2) Task-domain ontology author: staff of the software development division of electric power company.
(3) Task-specific ontology author: IES/CBT ontology researcher.
(4) Core task ontology author: general task ontology researcher.

Note that each type of author needs appropriate guidelines corresponding to their jobs. Intuitively, the authors working in the upper layer are more constrained than those in the lower layer. Thus, the authoring tools is desired to have the capability to switch the ontology layer to provide the appropriate guidelines for each type of the authors. The capability is realized by integration of CLEPE into the authoring tool. In the next section, we will look into the role of CLEPE in detail.

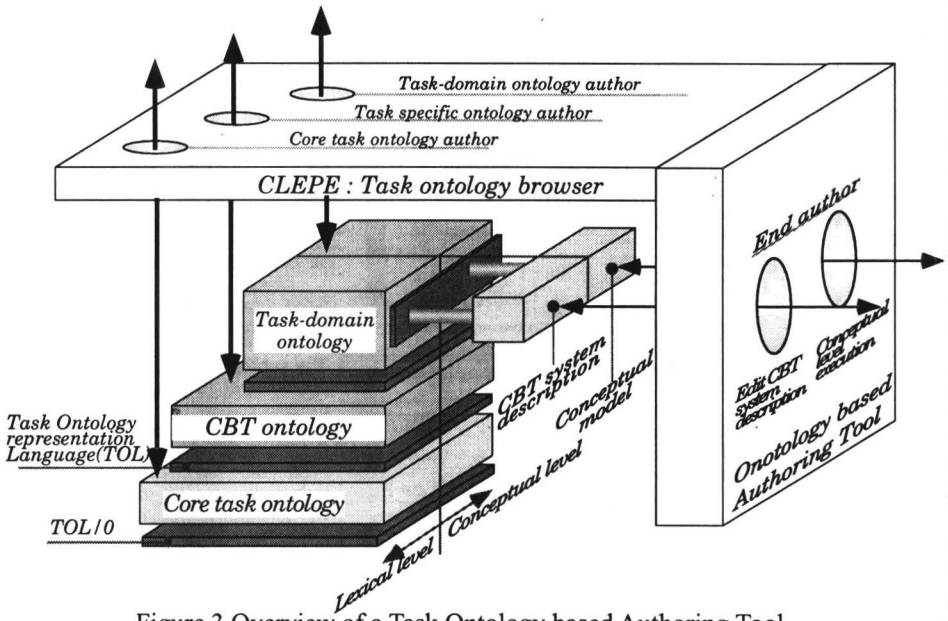

Figure 3 Overview of a Task Ontology based Authoring Tool

4. Task Ontology Management : CLEPE

CLEPE supports both ontology authors who construct task ontology and end authors who develop application, CBT systems in our case. Figure 3 shows the overview of CLEPE and the ontology-based authoring tool we have developed. Ontology authors are arranged above side and end authors right side.

Thin planes stand for languages. The language for task specific ontology author is called Task Ontology representation Language(TOL), whose semantics is specified by core task ontology and the base language TOL/0. TOL/0 provides description primitives for ontology authors and defines semantics of upper-layer languages. Therefore all the semantics of task ontology described with TOL is specified ultimately at the level of TOL/0. An ontology author of core task ontology specifies the lexical entities and conceptual ones using TOL/0 primitives, e.g. Define-Core-Lexical, Define-Core-Concept, etc. By reading the specification of core task ontology into CLEPE, a set of conceptual primitives at the TOL level is introduced. Task type specific ontology author specifies the concepts appearing in the target task type with TOL.

The main work of an end author are as follows:

(1) To describe an intended model of a training task in terms of CBT task ontology (edit CBT system description),
(2) to make sure that his/her description is correctly interpreted by the system (trace the conceptual level execution process),
(3) to modify the description if necessary (debug the description).

In preparation for interpretation of the CBT task description written by an end author, CLEPE reads task ontology description represented with TOL. The task ontology description is translated into internal form by TOL-parser and stored into ontology base. Ontology Manager manages the ontology base and deals with the requests related to the ontology made by other modules, for example, inquiries for class information,

creation of a class instance and so on. Once an end author completes editing his/her own description, CLEPE initiates the interpretation process. The model compiler compiles an internal form of the description and generates the conceptual model. Finally, he/she can *run* the conceptual model with executor.

4.1 Ontology Management

As we have discussed in 3.3, the relatively lower ontology provides the guideline on the construction of the relatively upper ontology. For example, the core task ontology act as a specification for the task-specific ontology. Definition of each concepts appearing in task-specific ontology layer should follow the regulation enforced by core task ontology. "Verb word class" definition in the core task ontology (meta-layer), for example, specifies that all the verbs defined in task-specific ontology layer (base-layer) have "input-slot", "output-slot" and "effect slot" in its definition. Furthermore, a verb definition of "give" in task specific ontology layer specifies that the input slot of each instance of "give" appearing in the model layer for end authors should be filled with limited noun instances such as, "Hint", "Problem", and so on.

Ontology management function of CLEPE is to support the authors work in a certain layer based on the next lower ontology. When it detects the inconsistency between model and ontology, it gives the authors some warning messages along with suggestions to encourage them to revise their models based on the ontology. The major merit of this function is that the authors could be guided kindly in their model building and the models is guaranteed to be consistent with the established ontology.

4.2 Conceptual level execution

Once CLEPE generates a conceptual model, an end author can inquire some questions about the dynamic behavior of the model, for example, "how student model updated", or "then what kind of tutoring action will be taken place." CLEPE can produce answers based on the conceptual model, for example, "the student model will be updated based on the buggy knowledge attached to the learner's answer to the question," or "system will show visual simulation to make learner notice the crucial breakdown." Since the answer to the inquiry is described at conceptual level, he/she could easily understand it. By keeping the continuity from the symbol level program code to conceptual level model, CLEPE can explain the symbol level execution result with conceptual level annotation.

Current implementation of conceptual level execution is limited to static analysis of the task model. The plan for the future development includes dynamic analysis of the task model. The basic idea is to introduce the concept of pseudo-learner. Training task ontology includes a set of terms and concepts to specify the status of the learner using vocabulary shown in figure 1. By specifying certain property of learner, for example, "a learner who prefers deductive way of thinking to inductive one" or "a learner who is good at skill-based operation", the author can examines whether the training model behave well for the learner or not by monitoring the results of conceptual level execution. The executor of the training model will assume the learner's responses to the questions along the training scenario, and activate the tutoring strategies based on the leaner model. Showing the dynamic behavior of the model to the author, the executor can check the validity of the training model based on the task ontology. When some defects or shortcoming of the model are identified, the executor informs the authors and suggest a desirable way to make alternation to the model.

5. Conclusion

We have investigated the inherent characteristic of CBT through the development of *SmartTrainer* and tried to build CBT task ontology. The main purpose of this paper is to illustrate the characteristics of ontology-based authoring tools. It provides two major advantages as follows. (A) It provides human-friendly primitives in terms of which users can easily describe their own model of a task (descriptiveness, readability). (B) It can simulate the abstract behavior of the model in terms of conceptual level primitives (conceptual level operationality).

We have implemented a prototype of ontology-based authoring tool and systematically accumulating CBT ontology and training scenarios. The most important, interesting part of CBT ontology is explicit description of the first principle of training such as "learning by doing." Since the accumulation of the ontology has been in steady progress, it is planned to pursue a close investigation of this issue in the near future.

References

[Inui et al. ,1990] Inui, M., Miyasaka, N., Matsubara, A., Fujita, M. Development of A Model-based Intelligent Training System for Plant Operations. *In Proceedings of International Conference on Advanced Research on Computers in Education, ARCE90,* pages 89-94 , Tokyo, July 1990.

[Major, 1995] Major,M.P. REDEEM: Creating Reusable Intelligent Courseware, *In Proceedings of International Conference on Artificial Intelligence in Education, AI-ED95*, pages 75-82, Washington, DC, August1995.

[Mizoguchi et al., 1996] Mizoguchi, R. , Sinitsa,K., Ikeda,M. Knowledge Engineering of Educational Systems for Authoring System Design — A preliminary results of task ontology design —, *In Proceedings. of EuroAI-ED'96*, pages.329-335, Lisbon, 1996.

[Murray and Woolf, 1992] Murray, T, Woolf,B : Tools for Teacher Participation in ITS Design, *In Proceedings of Intelligent Tutoring Systems ITS'92 , Lecture Notes in Computer Science, 608.. Springer Verlag*, pages.593-600, Montreal, June 1992.

[Van Marcke and Vedelaar, 1995] Van Marcke, K. and Vedelaar, H. Learner adaptivity in generic instructional strategies. *In Proceedings of International Conference on Artificial Intelligence in Education, AI-ED95*, pages 323-333, Washington, DC, August1995.