

An Ontology-Aware Authoring Tool

- Functional structure and guidance generation -

Lai JIN, Weiqin CHEN, Yusuke HAYASHI, Mitsuru IKEDA, Riichiro MIZOGUCHI
ISIR, Osaka University

Email: {kin, wendy, hayashi, ikeda, miz}@ei.sanken.osaka-u.ac.jp
8-1, Mihogaoka, Ibaraki, 567-0047, Osaka, JAPAN

Yoshiyuki TAKAOKA, Mamoru OHTA
TOKO SEIKI Company

Email: {charly, ohta}@kodama.cdd.toko-s.co.jp
3-14-40, Senrioka, Setushi, 566, Osaka, JAPAN

Abstract: This paper describes an ontology-aware authoring tool for intelligent training systems. The major advantages of the authoring tool are: (1) To provide human-friendly primitives in terms of which users can easily describe their own models of a task (descriptiveness, readability). (2) To simulate the abstract behavior of the model in terms of conceptual level primitives (conceptual level operability). (3) To verify the consistency of models. In this paper, we show the total image of an ontology-aware authoring system, describe the functional structure and the guidance generation mechanism in detail with an emphasis on the roles which the ontology plays. Finally, we introduce the intelligent training system SmartTrainer's Authoring Tools (SmartTrainer/AT) and its realization as an example.

1. Introduction

The development of the intelligent educational system(IES) using an authoring tool is a collaboration process between an author and a tool. The tool should be transparent to make him/her to understand what and why things are going on. It should be rich in functionality to enable for him/her to realize his/her goals but cannot be too complex. It should know what it is going to build, that is, it should have a model of the IES to behave intelligently.

An intelligent authoring tool sometimes cares about the author too much. If the guidance is too strong and persistent, the author feels uncomfortable. He/she usually needs to take the initiative in the authoring process. However, he/she sometimes needs timely help when he/she gets stuck. Such help is in principle given by the knowledge about what an IES should be.

Then, how can we build such an authoring tool? The authors have been involved in the development of such a tool believing an ontology is a key to the success of such an enterprise[2][3][8]. We have employed two kinds of ontologies: One is "task ontology"[7][11] which is one for representing problem solving process domain-independently and the other is "domain ontology" which corresponds to ordinary one. In the substation operator training case, a task is "training" and a domain is "Substation in the electric power network".

Training task ontology provides us with unit activities and task-dependent concepts(role concepts) which collectively specify the appropriate context where each domain concept should be put. T-D binding explained later plays a critical role to combine the role concepts in the task ontology and domain concepts in the domain ontology. The separation between task and domain enables the reuse of the knowledge.

Our tool is said to be "ontology-aware" in the sense that it knows the above two kinds of ontologies which are the source of its intelligence. The two ontologies play the role of a

meta-model which helps users build a model, that is, a training system in a certain domain. They are not a hard-wired model but a set of building blocks with formal semantics, therefore the tool will never be inflexible but can be intelligent. They make the tool friendly and helpful, since they have vocabulary at the conceptual level at which authors communicate with it, and the axioms or semantic constraints associated with the ontologies can generate timely messages of various kinds when authors violate them.

We have discussed these properties of our authoring tool [2][3] [8], but the image of the total system with how ontologies are used have not been described clearly. In this paper, we will show the whole image of an ontology-aware authoring system, discuss (1) the functional structure of the whole system, and (2) portion of the performance of the tool with an emphasis on the guidance messages generation using axioms defined in ontology. In session 2, we will describe the functional structure of the ontology-aware authoring system and the workflow from ontology building to model design, then in session 3, we will discuss the guidance needed in authoring process and the way to generate the guidance for the intelligent training system SmartTrainer's authoring tool based on training ontology. Finally, in session 4, we will show the image of the intelligent training system SmartTrainer with the emphasis on the guidance supporting of its ontology-aware authoring tools as an example.

2. Functional structure of an ontology-aware authoring tool

2.1 Workflow from ontology building to model design

In the knowledge base community, ontology is defined as "a system of primitive vocabulary/concepts used for building artificial systems"[6]. Fig.1 shows the functional structure and workflow from ontology building to model design, where there are three activities of an author such as building an ontology, using the ontology and designing a model. First, the ontology author builds an ontology, then the teaching material author (we call them the end author) designs model using the ontology built by the ontology author. Those activities are supported by ontology server, ontology editor, and ontology-aware authoring tool, respectively.

Ontology editor provides a tool for building ontology. Ontology authors can use it to define terms and axioms for representing the knowledge. Those terms and axioms constitute ontology library which includes several kinds of ontologies, such as task ontology and domain ontology. Ontology-aware authoring tool provides the author with vocabularies to represent the design of a training system model, "easy-to-use" interface for model design and design guidance during the course of authoring. Ontology server supports the process of ontology building, ontology using and model designing. In the process of ontology building, it provides guidance for ontology configuration, checks the consistency of the ontology, manages the viewpoint of task-domain ontology. In the processes of both using an ontology and designing a model, it provides design guidance of the training material, checks the consistency between the model and the ontology, and manages the viewpoints. Here we list the main functions in ontology server (a portion):

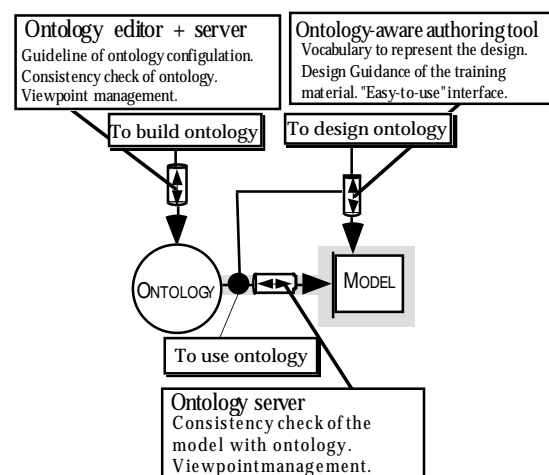


Fig.1 Overview of function structure

- define/modify/delete a class
- define/modify/delete an instance of a class
- define/modify/delete an axiom
- get the information about a class/an instance/an axiom.
- check the consistency of models based on axiom

Both ontology editor and ontology server have been implemented with Java and

Common Lisp, respectively, on the platform of PC. They will be published very soon.

2.2 Authoring process

Creation of a training material requires the following three procedures:

- (a) Building training scenarios
- (b) Developing a simulator of the target object
- (c) Deciding the learning items appearing the scenarios

Due to the procedures listed above, the authoring process of our ontology-based intelligent training system should:

- (1) Make the concepts for creating teaching materials and the interface clear. (Done by the ontology author with the ontology editor)
- (2) Define the axioms of concepts. (Done by the ontology author)
- (3) Make the models of training scenarios (Done by the end author) at the conceptual level based on the teaching material ontology, and then compile those models so that the authoring system can check the coordination of the models according to the axioms defined by the ontology author.
- (4) Execute those models at the conceptual level to see the result.
- (5) After the repetition of (3)&(4) write the final teaching materials in details by creating simulation cards and learning item cards based on the conceptual level models, and compile the final teaching materials to check if there is an undefined necessary card. All the unnecessary information for executing will be deleted at that time.

According to the authoring process shown above, the construction of our intelligent authoring tool consists of authoring interface, model compiler, conceptual level execution module, teaching materials compiler, and ontology construction environment. In these components, ontology construction environment contributes to (1)&(2), model compiler contributes to (3), conceptual level execution module contributes to (4), and teaching materials compiler contributes to (5).

2.3 Provide guidance in authoring process

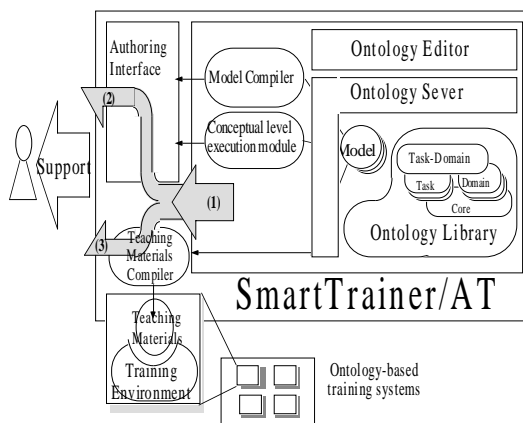


Fig.2 Draw up guidance in authoring process

Different guidance will be provided to the end author in different authoring stages, and ontology plays important roles on it. Fig.2 shows it in detail.

In Fig.2, arrow (1) represents the roles the ontology plays, we will explain it in detail focusing on the flows shown by arrow (2) & (3). Arrow (2) shows the contributions of ontology through interface, and the guidance from the process of model creating, model compiling and model execution. Arrow(3) shows the contributions through teaching materials compiler and the guidance from the process of it. In fact, lots of high quality authoring tools with convenient guidance have been developed to date without ontology

supporting, so what is the difference?

The answers existing in arrow (2) are:

- A. Our tool provides friendly interfaces: because the ontology define the concepts which are very familiar to humans to express the model, the end author can select and combine the concepts which are organized in different layers through graphical interface.
- B. Our tool verifies the consistency of model at the conceptual level and give some warning messages or error messages to the end author if there is any inconsistency in the end author's model: because the guidance has been made clear as an axiom of ontology.

1	(Teaching Materials	24	(question intention	45	(equipment
2	d*@n backbone stream: backbone stream	25	p+learning content: A	46	p+joining terminal: equipment
3	d*@n rib stream: rib stream)	26	p+thinking procedure: thinking)	47	p+operation: operation
4	(backbone stream	27	(question	48	p+parameter: parameter set))
5	p+teaching content: A	28	p+question content: A	49	- main circuit equipment
6	p+question list: series of questions	29	p+correct answer: A	50	-(power cable
7	{+corresponding path[?P2]: path }	30	p+incorrect answer:	51	p+joining terminal: equipment))
8	p+corresponding path[?P1]: path	31	p+standard: difficult/easy	52	-bus
9	p+object accident: accident)	32	p*@n error pattern: error pattern))	53	-(transformer
10	~[?p1]=[?p2]	33	-(multiple-choice question	54	p+joining terminal: connecting line))
11	(rib stream	34	p*@n multiple-choice option [?ASI-i]:	55	-(protection relay
12	p+knowledge: learning item		multiple choice option	56	p+joining terminal: connecting line
13	p+corresponding label: learner's label	35	p+correct answer[?ASI]:	57	p+adjacent equipment:circuit breaker/
14	p*@n teaching behaviors: teaching strategy))		multiple choice option))		dsiconnecting switch))
15	(questions	36	error pattern	58	(operation
16	p+question: question	37	- multiple-choice question error pattern	59	p+subject:
17	p+question intention: question intention	38	-(lack of correct answer	60	p+object:equipment/media
18	p+object scope: workflow	39	p+correct answer:	61	p+application condition:
19	p*@n treatment: treatment))		multiple choice option))	62	p+function:)
20	(series of questions	40	-(insertion of incorrect answer	63	-(confirm
21	p+corresponding path[?P2]: path	41	p+incorrect answer:	64	p+subject: operator
22	p+questions: questions))		multiple choice option))	65	p+subject: 30F display
23	{+object scope[?P-n]: workflow }	42	(Teaching Strategy	66	p+application condition: alarming
		43	p+goal goal	67	p+function: object goes into the head of
		44	p*@sub goal:goal/teaching behavior))		subject)

Note: “p+”: Part-of “d+”: Division-of “a+”:Attribute-of “r+”:Other relations “*@n”: the plural “p+”: Inherited slot
“(“: Beginning of the class definition “)“: End of the class definition “-“: Is a sub class
“x:y”: “x” represents the slot name, “y” represents the class constraint “{...}”: Explanation

Is-a: Describe the relations between general concept and specific concepts.
Part-of: Describe the relations between whole and part. Attribute-of: Means in the slot is the value of the attribute.
Division-of: Means in the slot are the divisions of the class. Compared with the part-of relation, the content in the slot of division-of relation has to be mutually exclusive.

Fig. 3 Training Task Ontology and Domain Ontology (A portion)

The answer existing in arrow (3) is:

C. The end author fills a conceptual level model with concrete contents in the form of card, and the authoring tool will give an error message to him/her when a necessary card has not been defined.

SmartTrainer/AT is an ontology based intelligent authoring tool. Next, we will introduce the guidance on SmartTrainer/AT and its realization in detail.

3. Guidance on SmartTrainer/AT

Fig. 3 shows a portion of training task ontology on teaching material construction(Fig. 3, line1~44) and a portion of domain ontology on substation in the electric power network(Fig. 3, line45~67). Training task ontology provides us with unit activities and task-dependent concepts(role concepts) which collectively specify the appropriate context where each domain concept should be put. Teaching materials of SmartTrainer is composed of backbone streams and rib streams. A backbone stream is a sequence of questions for the learner to master the learning items included in an accident recovering process. A rib stream consists of a series of teaching behaviors for instructing each piece of the knowledge corresponding to the question in the backbone stream. When the learner makes a mistake in the backbone stream, a suitable rib stream will be selected to help him/her to learn according to the learner model and the intention behind the question. Ontology can be used to check the rationality of the models of teaching materials constructed by the end author, and to give some helpful guidance to him/her if there is any inconsistency. In this section, we will discuss the roles of axioms in authoring process and examples of guidance messages generated from axioms.

3.1 Roles of axioms in the authoring process

According to the axioms defined in training task ontology, there are three kinds of merits which can be offered to the end author during the authoring process:

(1) An appropriate menu can be provided to the end author according to the axioms on class constraint of a slot, and this can help the author to select the alternatives easily.

For example, the end author is writing a multiple-choice question (Fig. 3, line 33). He/She has to specify the correct answer slot. At this time, a menu which contains the instances of multiple-choice option class will be provided by SmartTrainer/AT, so that the end author can choose one from the menu items.

(2) The rationality of the models will be verified, and warnings or explanations will be given to the end author when he/she made inappropriate selections in building models.

For example, the end author specifies a teaching strategy corresponding to a certain bug the learner may have. The teaching strategy includes a teaching goal and a series of subgoals and teaching behaviors (Fig. 3, line 42). SmartTrainer/AT will check whether the teaching strategy specified by the end author is consistent with the axioms. If not, SmartTrainer/AT will give the end author a helpful message, so that he/she can understand why it is inappropriate and know what is the good way to attain the teaching goal. The different kinds of messages will be discussed in details later in this paper.

(3) The view can be changed from task model to domain model or from domain model to task model automatically. According to the need of the end author during the authoring process, SmartTrainer/AT pops up the corresponding window for editing task model or domain model.

As we have written in Section 2, when the end author tries to create training material, he/she needs to perform three procedures explained earlier. He/She can begin with either building training process or building domain model. He/She can also reuse the task model and domain model built by other end authors. In SmartTrainer/AT, the configuration of substation is the domain model. Suppose an end author begins with adding a training scenario in training process. He/She will define the corresponding items such as backbone stream and rib stream, according to the training task ontology shown in Fig. 3. During this procedure, He/She needs to refer to domain model. For example, when defining a backbone stream (Fig. 3, line 4), a corresponding path needs to be specified (Fig. 3, line 8). Path is defined in domain ontology as a sequence of operations beginning with the situation when an accident happens, and ending with the status when the accident has been recovered from. Because at this time the part of domain model needed hasn't been built, SmartTrainer/AT will change the view from task model to domain model, so that the end author can begin to build domain model. This view change is supported by the T-D Binding mechanism. SmartTrainer/AT pops up the window for domain model editing and prompts the end author to build domain model. This is a guidance for directing the end author what to do based on authoring process guideline.

(4) The function of automatic-model-linking can be provided: The equalities between the instances of the slots in different classes have been made clear, so if one instance of a slot has been specified in other classes before, the content will automatically be transferred to this slot.

3.2 Examples of Guidance Messages Generated from Axioms

Guidance messages provided by SmartTrainer/AT is classified into three categories, those are, error, warning, and suggestion.

An error is detected by the ontology server, whenever a model violates the regulation specified by the strong axiom. SmartTrainer/AT generates the guidance message that explains why the model must be revised and how to revise it based on the axiom violated. Most of the strong axioms are described in terms of conceptual relations, such as, is-a or part-of. In this sense, those intuitively seem to be close to syntax rules or type constraints of a programming language. As we know, the error messages of conventional programming environment are not so much helpful, because they do not show us useful information about why the program should be revised or how to revise it, which programmers want to know. An ontology-aware authoring environment do provide such kind of information as guidance messages because the ontology is built based on how the target should be modeled from not computational viewpoint but end author's viewpoint. In other words, ontological constraint is more helpful than the simple type constraint because it captures inherent meaning of a task and a domain at hand. For example, "The **learner** we are supposing now is **novice**, it is unsuitable to give a **question** with a **expert-level** to him/her" is a typical error message. (Note: The bold characters represent the concepts defined in training task ontology).

Guidance messages are generated by a simple template-based natural language generator. Each axiom has a set of message templates in its definition. The axiom from which the above message is generated is shown below.

```
(def-axiom not-to-give-expert-level-question-to-novice
  (participants
    (?learner LEARNER))
  (axiom-body
    (is ?(level-of ?learner) novice) &
    (not-exist ?questions in ?(questions-given ?learner))&
    (expert-level ?question))
  (axiom-type strong))
```

The definition of an axiom is composed of four portions, those are axiom name, participants, axiom body and axiom type. Participant shows what kind of conceptual entity constitutes the axiom, it includes the role name of the entity and the class constraint. Restricted form of logical expression can be described in the axiom-body. Ontology server checks whether the condition has been satisfied or not and notifies client, SmartTrainer/AT in our case, of the result.

Warning guidance is given to the end author when SmartTrainer/AT is notified of a weak axiom violation by ontology server. The major reason why an ontology author specifies a certain axiom as weak is he/she thinks there could be some exceptional modeling viewpoints. An ontology author often faces hard dilemma when maintaining its generality and flexibility. Weak axiom is a way to describe the axiom that usually holds but sometimes could be violated with exceptional but right intention. This means, of course, that the ontology author leaves the right intention outside of ontology. For example, most of training strategies are classified as weak axioms, because we not only have general principles on what should be done for learners in what situation but also know there could be a variety of deviations from the principle to cope with a dynamic and complex training context. As such too complex contexts cannot be fully captured in any modeling paradigm, it is reasonable an ontology author puts them outside of ontology formulation.

SmartTrainer/AT provides the end author with a warning message to let him/her know it violates a general guideline of modeling and how the target should be modeled in most cases as just for his/her information. Basically, whether he/she follows the guideline or not is up to him/her. For example, "The **learning-style** of the **learner** we are supposing now is **example-oriented**, the **bug** of the **learner** is **missing-knowledge**, and the knowledge is **difficult** to understand. There is a good **example** to **explain** the meaning of the knowledge. In order to **help** the **learner** to reach **deep-understanding**, it is better to **give-the-example** to him/her" is a typical warning message provided by SmartTrainer/AT.

Again we should note that the difference between the quality of the guidance provided by the ontology-aware authoring tool and the one of warning message provided by programming environment. The difference might come from the difference of depth of the conceptualization. They are same in the sense that both messages are generated based on the guidelines for intellectual work. The only difference is how deeply the tool knows the contents of the work. In case of programming tool, the warning message is generated from very surface-level rule, for example syntax rule, which is specified in terms of computational concept such as variable, operator, type and so on. On the other hand, the warning guidance is generated from the ontology axioms that are described in terms of the task and the domain concepts with richer meaning.

The violation of the weak message is delivered to the end author as a warning or a suggestion message. When missing part of the model is detected based on the weak axiom, for example, the violation is detected and then the following suggestion message is generated: "The **mistake** made by the **learner** is incorrect-operation, the **operation** is very **important** and **critical**, and the **learner** seems to be too **rash** to do the operation. So, in order to **leave** the **mistake** in his/her a **long term memory**, it is better to **show-result-caused-by-incorrect-operation** to him/her." as a typical suggestive guidance.

Most of authoring process guidelines on what to be done next are provided as suggestive guidance. In SmartTrainer/AT, the authoring process is guided by focus control mechanism that keeps the author's focus of attention to appropriate workplace based on the structure of the ontology.

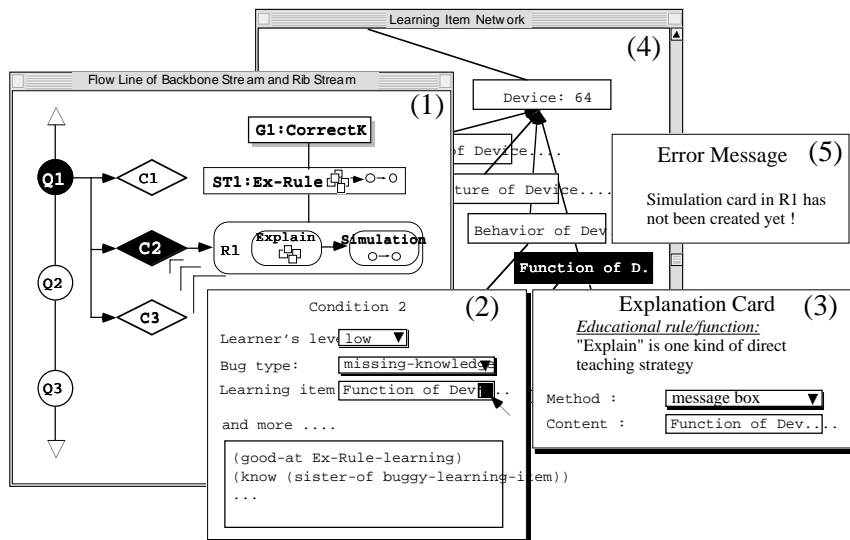


Fig.4 Flow line editor & Teaching strategy card

4. Example of guidance message in SmartTrainer/AT

We have discussed the generating guidance of the intelligent authoring tool based ontology in session 3. In this session, we will show some images of the guidance supporting through the introduction of SmartTrainer/AT. SmartTrainer/AT is composed of four parts, those are work flow editor, backbone stream editor, flow line editor and teaching strategy card.

Because target task is very large and very complicated, changing it into teaching materials directly is very difficult for the end author, we present a work flow edit tool for the end author to write a work flow first which can be used as a reference to write backbone streams. Work flow is a sequence of recognition, judgment and practice for recovering an accident in the electric power system. Initial state expresses the situation when the accident happens and goal state expresses the status when the accident has been covered.

Referring to the workflow, the end author can select a path as a backbone stream, system will check all the paths in the work flow so that no routine will be forgotten by the end author.

After making a workflow, the end author will edit the backbone streams and the rib streams in them one by one. The model of a backbone stream and its rib streams built at conceptual level should be compiled and executed, and the end author will re-build the model if necessary, according to the result of the consistency checking and the execution of the model.

Fig.4 shows the flow line editor and teaching strategy card. Window(1) shows a backbone stream model made at concept level. The line on the left side represents a sequence of questions in that backbone stream, and c1~c3 show the control of the rib streams. The detail information of condition2 is shown in window(2). Each teaching behavior in R1(rib stream1) is just a conceptual behavior, the end author should enrich it by teaching strategy card which provides the educational rule/function(window 3). Learning items menu can be given to the end author according to the class constraint automatically, and the end author uses it to select the contents of the explanation(window 4). If the simulation card of R1 does not exist, the teaching material compiler will give an error message (window 5). We adopted a conventional simulator code used in the target domain as our simulator. What our ontology does for the simulation is not composing the code but setting up parameters of the simulator which specify what to simulate in what conditions. These should be carefully set according to the training goal, that is, the training context. The ontology specify the general constraints between the training context and such parameters.

5. Concluding Remarks

The task ontology and domain ontology acting as core of SmartTrainer/AT are not hard-wired but a set of building blocks with formal semantics, therefore the tool can be flexible

enough for end authors to feel free when composing teaching material. They make the tool friendly and helpful, since they have vocabulary at the conceptual level at which the ontology aware-authoring tools can generate some helpful guidance to the end authors. Appropriate guidance can be provided to IES authors by the ontology aware authoring tool. The major characteristic of ontology-aware authoring tool is that it can provide not only the surface-level guidance but also content-oriented guidance as we have seen in this paper. The prototype of guidance generation mechanism have been integrated into SmartTrainer/AT.

In our ontology-related project, an integrated ontology development environment has been developed and acts as a key technology to build high-quality, reusable ontology. It lays the foundation for separation/integration of task ontology and domain ontology based on task-domain binding mechanism. The separation of task and domain can be key to build sophisticated ontology because each ontology can be built systematically from each own viewpoint and the integration of them provides end author with easy-to-use representation scheme. As a result, we can improve the generality of ontology without losing its facility. The environment built with JAVA will be open to the public in the near future. The current version of SmartTrainer/AT is not tightly connected to the environment, because the interface is built with CLOS. In the development of the next version, we will re-implement the total system with JAVA and CLOS and evaluate the usability of the tool.

The ontology-aware authoring tool supports the authoring process with a moderate intelligence. Although one might think that such a tool may generate a learning environment with a little too much care for the learner, it is not true. We can design an ontology for learner-centered learning environments. The issue here is that how carefully we design an ontology for various learning paradigms such as discovery learning, collaborative learning, exploration-based learning as well as simulation-based skill training to reflect their inherent characteristics. Each paradigm has its own conceptualization of what is learning which can be explicated as an ontology. It is crucial to collect such ontologies to make ontology-aware authoring tools richer and more effective. Future work also includes to formulate modeling guidelines in the form of Design patterns in software engineering[12].

References

- [1] Chandrasekaran, B.: Generic tasks for knowledge-based reasoning: the right level of abstraction for knowledge acquisition, *IEEE Expert*, 1, pp.23-30, 1986.
- [2] M. Ikeda, K. Seta, R. Mizoguchi: Task Ontology Makes It Easier To Use Authoring Tools, *IJCAI'97*, Nagoya Japan, pp.342~347, 1997.
- [3] M. Ikeda, L. Jin, Y. Hayashi, K. Seta, R. Mizoguchi, Y. Takaoka, M. Ohta; Task Ontology Makes Authoring Easier -Training Task Ontology and Authoring Support Function -, *SIG-J-9801-11(12/10)*, 1998.
- [4] Jin, L, T. Sano, M. Ikeda, R. Mizoguchi, K. Hirobe, Y. Takaoka, Design and Prototyping of Smart Trainer: A Training System Dealing with Fault Recovery Operation in Electric Power Systems, *Proceedings of the 21th Annual Conference of JSISE*, 1996.
- [5] Leila A., Richard K., Intelligent Simulation Environment an Application to Air Traffic Control Training, *Sintect'96 Nelbourne Narch96*, Australia, pp.31-38, 1996.
- [6] Mizoguchi, R.: Knowledge acquisition and ontology, *Proc. of the KB&KS'93*, Tokyo, pp. 121-128, 1993.
- [7] Mizoguchi, R, M. Ikeda, K. Seta, et al.: Ontology for modeling the world from problem solving perspectives, *Proc. of IJCAI Workshop on Basic Ontological Issues in Knowledge Sharing*, Montreal, 1995.
- [8] R. Mizoguchi, M. Ikeda, K. Sinitsa: Roles of Shared Ontology in AI-ED Research, *AIED97*, Kobe, Japan. pp. 537-544, 1997.
- [9] Tom Murray: Authoring Knowledge-Based Tutors: Tools for Content, Instructional, Strategy, Student Model, and Interface Design; *The journal of the learning sciences*, 7(1). pp.5~64, 1998.
- [10] Rasmussen, J.: Skills, Rules, and Knowledge; Signals, Signs, and Symbols, and Other Distinctions in Human Performance Models, *IEEE Trans. SMC*, SMC-13 [4] , pp.257-278, 1983.
- [11] K. Seta, M. Ikeda, T. Shima, O. Kakusho, R. Mizoguchi: A Task Ontology Based Environment for Building Problem Solving Systems -Causality in Problem Solving Processes-, *AI97-83, KBSE97-42*, pp.65~72, 1997.
- [12] H. Shimiz, K. Seta, Y. Hayashi, M. Ikeda, R. Mizoguchi: A Basic Consideration on Design Patterns for Ontology Building, *Proceeding of the 58th Annual Conference of IPSJ*, pp.175~176, 1999.