An Ontology-Based Human Friendly Message Generation in a Multiagent Human Media System for Oil Refinery Plant Operation

Riichiro Mizoguchi, Toshinobu Sano and Yoshinobu Kitamura
ISIR, Osaka University
8-1 Mihogaoka, Ibaraki, Osaka, 567-0047 Japan
{miz, sanop, kita}@ei.sanken.osaka-u.ac.jp

ABSTRACT

The role of an ontology and ontology server in the interface for oil-refinery plant operation is described. It plays a central role as the Knowledge media which is one of the key technologies in Human Media project. Plant ontology as well as operation task ontology has been developed. The main task of ontology server in the current implementation is message generation with appropriate word selection which is done by focused point tracing module and adaptive word selection module both of which consult the model built using the components obtained by instantiating the concepts in the plant ontology. The system has been implemented and preliminary evaluation has been done successfully.

1. INTRODUCTION

The Human Media project is intended to invent an innovative media technology for happier human life in the coming information society in 21st century. It is something an integration of the three representative media such as Knowledge media, Virtual media and Kansei media. “Kansei” is a Japanese term which roughly means the sixth or seventh sense sensing for satisfaction, comfort, beauty, softness, etc. This paper mainly describes the role of knowledge media in the project of “Development of a human interface for the next generation plant operation” running as a subproject of Human Media.

As described in the overview paper[1], the interface for oil-refinery plant operation has been developed intended to establish a sophisticated technology for advanced interface for plant operators and consists of Interface agent: IA, Virtual plant agent: VPA, Semantic information presentation agent: SIA, Ontology server: OS and Distributed collaboration infrastructure: DCI. The last two are mainly for issues related to system building, while the first three are related directly to interface issues.

Knowledge base technology has a long history and has attained an epoch-making result called expert systems. In spite of its great success, it has suffered from serious difficulties such as knowledge acquisition bottleneck, non-reusability and non-sharability of knowledge. Departing from simple rule base technology, the efforts to overcome these difficulties have been made and the resulting new technologies include Case Base Reasoning: CBR which uses cases rather than knowledge, model-based approach which employs runnable model or theory of the target object rather than heuristics of domain experts and ontological engineering. CBR is used in the Interface Agent and Model-based approach is used in Semantic information presentation agent in our project. Ontological engineering is the key topic of this paper.

Ontology server, OS for short, has two major functions in the system:
(1) To standardize each agent’s understanding of the target world, the oil-refinery plant in our case, by providing a plant ontology.
(2) To generate human friendly messages in terms of appropriate words operators use daily.

This paper presents the reasons why we need an ontology, what ontology we built and how the ontology is used in the system. The next section discusses a background information of knowledge processing community followed by a short introduction on ontology. Section 3 explains the role of the plant ontology. Detailed description of the plant ontology we designed is given in Section 4. Section 5 presents the explanation of its use in the entire interface system we are developing followed by implementation and conclusion.

2. WHAT IS ONTOLOGICAL ENGINEERING AND WHAT IS AN ONTOLOGY?

Ontological engineering is a successor of knowledge engineering which has been considered as a technology for building knowledge-intensive systems. Although knowledge engineering has contributed to eliciting expertise, organizing it into a computational structure, and building knowledge bases, AI researchers have noticed the necessity of a more robust and theoretically sound engineering which enables knowledge sharing/reuse and formulation of the problem solving process itself. Knowledge engineering has thus developed into “ontological engineering” where “ontology” is the key concept to investigate. Roughly speaking, ontology consists of **task ontology** which characterizes the computational architecture of a knowledge-based system which performs a task and **domain ontology** which characterizes the domain knowledge where the task is performed. By a task, we mean a process like diagnosis, monitoring, scheduling, design, and so on. In our context, operation is a task. The idea of task ontology which serves as a theory of vocabulary/concepts used as building blocks for knowledge-based systems[2][3][4] might provide us with an effective methodology and vocabulary for both analyzing and synthesizing knowledge-based systems. An ontology is understood to serve as a kernel theory and building blocks for content-oriented research. Definitions of an ontology are presented below:

(1) In philosophy, it means *theory of existence*. It tries to explain what exists in the world and how the world is configured by introducing a system of critical categories to account for things and their intrinsic relations.

(2) From an AI point of view, an ontology is defined as "explicit specification of conceptualization" [5]. "Conceptualization" here should be interpreted as "intensional" rather than "extensional" conceptualization contrary to that defined in [6].
(3) From a knowledge-based systems point of view, it is defined as "a theory (system) of concepts/ vocabulary used as building blocks of information processing systems" [3]. In the context of knowledge-based problem solving, ontologies are divided into two types: Task ontology for problem solving process, and domain ontology for domains where the task is performed.
(4) A compositional definition: An ontology consists of concepts with definitions, hierarchical organization of them (not mandatory), relations among them (more than is-a and part-of), and axioms to formalize the definitions and relations.

Why ontology instead of knowledge? Knowledge is domain-dependent, and hence knowledge engineering which directly investigates such knowledge has been suffering from rather serious difficulties, such as domain-specificity and diversity. Further, much of the knowledge dealt with in expert systems has been heuristics domain experts have, which makes knowledge manipulation more difficult. However, in ontological engineering, we investigate knowledge in terms of its origin and elements from which knowledge is constructed. An ontology reflects what exists out there in the world of interest or represents what we should think exists there. An ontology is essentially designed to be objective and shared by many people. Hierarchical structure of concepts and decomposability of knowledge enable us to identify portions of concepts sharable among people. Exploitation of such characteristics makes it possible to avoid the difficulties knowledge engineering has faced with. The following is an enumeration of the merits we can enjoy from an ontology:

(a) *A common vocabulary*. The description of the target world needs a vocabulary agreed among people involved.
(b) *Explication* of what has been often left implicit. In all of the human activities, we find presuppositions/assumptions which usually are left implicit. Any knowledge base built is based on a conceptualization possessed by the builder and is usually implicit. An ontology is an explication of the very implicit knowledge. Such an explicit representation of assumptions and conceptualization is more than a simple explication.
(c) *Systematization* of knowledge. Knowledge systematization requires well-established vocabulary/concepts in terms people use to describe phenomena, theories and target things under consideration. An ontology thus contributes to providing a backbone for the systematization of knowledge.
(d) *Standardization.* The common vocabulary and knowledge systematization bring us more or less standardized terms/concepts.
(e) *Meta-model functionality.* A model is usually built in the computer as an abstraction of the real target. And, an ontology provides us with concepts and relations among them which are used as building blocks of the model. Thus, an ontology specifies the models to build by giving guidelines and constraints which should be satisfied. This function is viewed as that at the metalevel.

3. THE ROLE OF A PLANT ONTOLOGY

Any intelligent system needs a considerable amount of domain knowledge to be useful in a domain. The amount of knowledge necessary often goes large, which sometimes causes difficulties in the initial construction and maintenance phase. As described above, one of the methods we adopted to cope with such problems is an ontology engineering. The plant ontology makes contributions in our system in many respects described above. Roughly speaking, the essential contribution of an ontology is making shared commitment to the target world explicit, and hence terminology is standardized within the community of agents. By agents, we also mean human agents, operators, to share such a fundamental understanding about the plant. This enables the system to communicate with operators using the terms stored in Ontology server: OS. It is the second major role of OS in the current implementation of the interface system which is discussed below in Section 4.

In message generation, we need to pay maximal attention in word selection to make operators' cognitive load minimum in message understanding. After an intensive interview with domain experts, we found human operators use different terms to denote the same thing depending on context. When we first noticed this fact, domain experts apologized for this seemingly random fluctuation of word usage, since they did not know the reason why they use terms that way and they were used to collaboration with computer engineers who do not like neat adaptation and tend to compel their idea of "this is what a computer can do, so accept it". They kindly declared that they will soon determine a unique label for each thing. But, we were different from such computer engineers. Instead of accepting their proposal, we carefully analyzed the way of their word usage and finally came up with that it is not random except a few cases. Many of the wording have good justifications which have to be taken care of in the message generation. The way of doing so is described in 4.2.

The reasons why we employed distributed collaboration architecture with multiple agents include to make the whole system robust and easy to maintain. As is well known, however, these merits are not free. We need a powerful negotiation and collaboration protocol to realize them. One of the key factors of such an enterprise is an ontology for negotiation. Such an ontology should be domain-independent to make the protocol widely applicable. We could call it "Negotiation task ontology". It is highly possible contrary to the difficulty found at a first glance. It is just the same as the efforts the authors have made in task ontology design. Negotiation is a task like so is diagnosis.

4. PLANT ONTOLOGY

We built a plant ontology which consists of several hierarchical organizations of concepts such as operation task ontology, plant components and plant objects, basic attributes & ordinary attribute. The key issue in design of an ontology is clear distinction essential categories from peripheral or view-dependent concepts.

Operation Task Ontology

The major constituents of a task ontology are concepts of action done by the task performer, operators in our case, and concepts of the role which domain objects play in the task performance. This is the key issue of task ontology. That is, a task ontology reveals the problem solving context in a task of interest to specify the roles the domain objects play. Without this, it is left implicit that how domain concepts should be organized under a specific task.

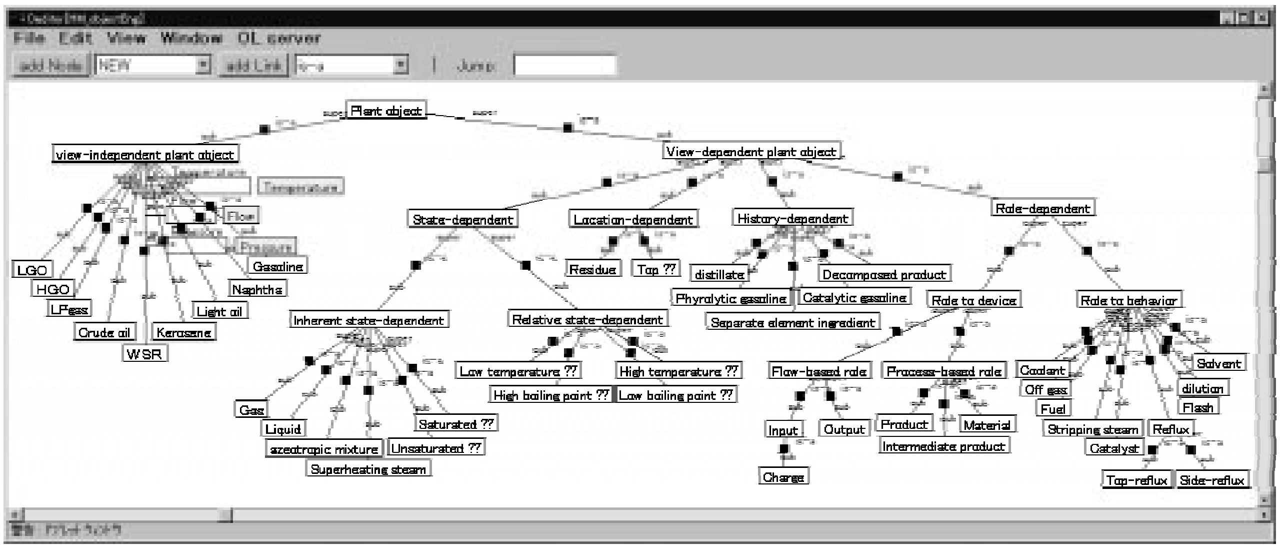


Figure 1: A Portion of Plant Object is-a Hierarchy in the Plant Domain Ontology

Activity Concepts: Operation of a plant consists of monitoring the behavior of the plant, diagnosing abnormal states if any and operating devices to recover from such states. Thus, the three actions, ***monitor***, ***diagnose*** and ***operate*** are the top level category of action part. Under these, we also identified ***enumerate, list up, decide, predict, etc.***

There are two kinds of hierarchies of concepts organization. Is-a hierarchy and part-of hierarchy. For example, sub-concepts of ***operate*** in the part-of hierarchy would be ***recognize***(the state), ***predict***(the near future), ***identify***(the causes), and ***decide*** (operation to take). On the other hand, sub-concepts of ***reason*** in the is-a hierarchy, are ***predict and retrospectively reason*** and its super is ***think.*** In the is-a hierarchy case, properties of the super concepts are inherited to the sub-concepts. In ***reason*** case, its [input : output] roles, [state : state], are inherited by ***predict and retrospectively reason*** but specialized to [current state : future state] and [current state : causal state].

Role Concepts: Major task-specific role concepts include state of operation, ***abnormal state, candidate cause, countermeasure operation*** etc. When a state of operation is recognized as abnormal, then it comes to be called ***abnormal state***. ***Near future state*** is a state predicted from the an ***abnormal state*** as the ***current state***. A ***cause*** of a fault has its sub-states: a ***candidate cause*** which is an inferred causal state and a cause which is a real cause.

Domain Ontology

There exist two major things in the plant domain: ***Plant components(devices)*** and ***plant objects*** to be processed by the ***devices***. Domain concepts also have role concepts like task ontology. To say precisely, many of the domain concepts are role concepts. The first things we have to do when designing a domain ontology is discrimination of roles concepts from essential categories(or basic concepts), i.e., view- or context-independent concepts. Let us take ***plant object***. The top level categories of ***plant object*** are ***view-independent object*** and ***view-dependent object***. The former includes LP gas, gasoline, naphtha, etc. which are categories persistent in any situation. The latter includes ***tower-head ingredient, liquid, distillate, input, intermediate product, raw material, fuel***, etc. All are view- or context-dependent. The major task needed was to categorization of such dependency. Fig. 1 shows a portion of ***plant object*** is-a hierarchy. The top level categories of ***view-dependent plant object*** are ***state-dependent, location-dependent, history-dependent and role-dependent*** objects***.*** ***state-dependent*** objects has ***inherent state-dependent*** and ***relative state-dependent*** objects as its sub-concepts. The former includes ***liquid, gas, super heater*** etc. and the latter ***low temperature ingredient***, ***low boiling point ingredient***, etc.

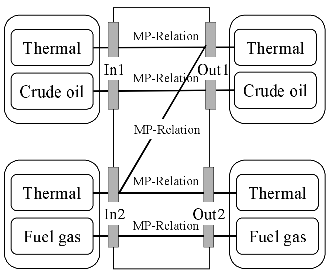
Attribute also needs careful treatment. Most of the attributes people think so are not true attribute but ***role attribute***. Let us take an example of ***height***. It is a ***role attribute*** whose ***basic attribute*** is ***length. Height, depth, width*** and ***distance*** are ***role attributes.*** Just like a man is called a husband when he has got married. The true attribute is called ***basic attribute***. Examples of ***basic attribute*** include ***length, area, mass, temperature, pressure, volt, etc***. Role attribute includes ***height, depth, input pressure, maximum weight, area of cross section, etc.*** Needless to say, these attributes are also decomposed into several sub-concepts.

We build an ontology which contain about 350 concepts which are approved by the domain experts and the coverage is around the normal pressure fractionater of a full-scale refinery plant. So, it is not a toy ontology.

Another kind of domain ontology is necessary to build a model of a plant as an active artifact. That is an ontology of function and behavior which is task-independent. This ontology is used for deeper understanding of the dynamic characteristics of an artifact.

FBRL and Functional Ontology

FBRL: This subsection outlines a functional modeling language named FBRL (abbreviation of Function and Behavior Representation Language)[7]. An FBRL model of a target system consists of component models and structural information among components. An FBRL component model consists of a behavioral model and a set of the mapping primitives called ***Functional Toppings*** (FTs) as shown in Figure 2. The behavioral component model consists of inflow objects, outflow objects, connection ports, and constraints over parameters. There are four types of the functional toppings; (1)O-Focus representing focus on attributes of objects, (2)P-Focus representing focus on ports (interaction to neighboring components), (3)FuncType representing types of



FTset:
O-Focus: Phase
P-Focus: ((In1)(Out1))
FuncType: ToMake
Necessity: NIL

Functional Concept:
"Vaporize"

Figure 2: FBRL model of a Heater

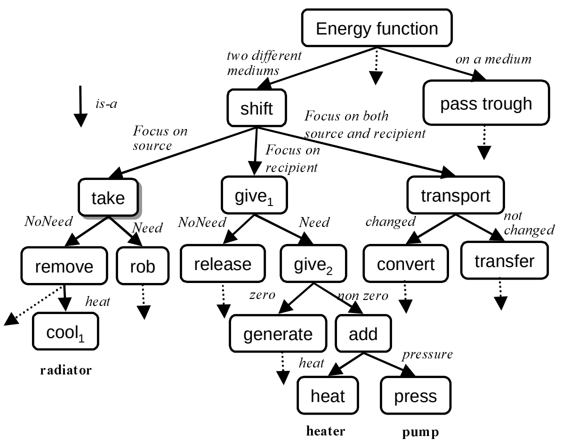


Figure 3: A part of the functional ontology.

contribution (extended one of those defined by Keuneke [8], and (4)Necessity of objects. For more details of FBRL, see [7].

Note that such FTs of a function are highly independent of its implementation, that is, details of behavior and internal structure of the component. For example, P-Focus specifies not concrete location but abstract interaction with the neighboring components. In FBRL, the functional relations are categorized into two types, ***causal-type*** functional relations defined by causal relations in behavior, and ***structural-type*** relations defined by structure. The former has subtypes such as ***providing-***, ***preventing-*** and ***efficiency-type*** and is called Meta-function. The later has subtypes such as ***series, parallel, sequential*** and ***simultaneous***.

Functional Ontology based on FBRL: The functional ontology consists of ***functional concepts*** organized in an ***is-a*** hierarchy with clues of classification shown in Figure 3. The definition of a functional concept consists of a label representing the concept and conditions of behavior and functional toppings. For example, the functional concept ***take energy*** is defined as "an energy flow between two mediums" (a behavioral condition), and "focus on the medium transferring the heat" (a functional condition) as shown in Figure 3b. Moreover, the definition of "remove" is that of ***take*** plus "the heat is unnecessary". Thus, ***take*** is a general (super) concept of ***remove*** as shown in Figure 3.

The functional ontology provides conceptual vocabulary for describing functional knowledge such as the functional decomposition patterns described bellow. Because the ontology makes the knowledge detached from behavior and structure, the knowledge is reusable. And, the functional model with a functional ontology is operational in the sense that computer can

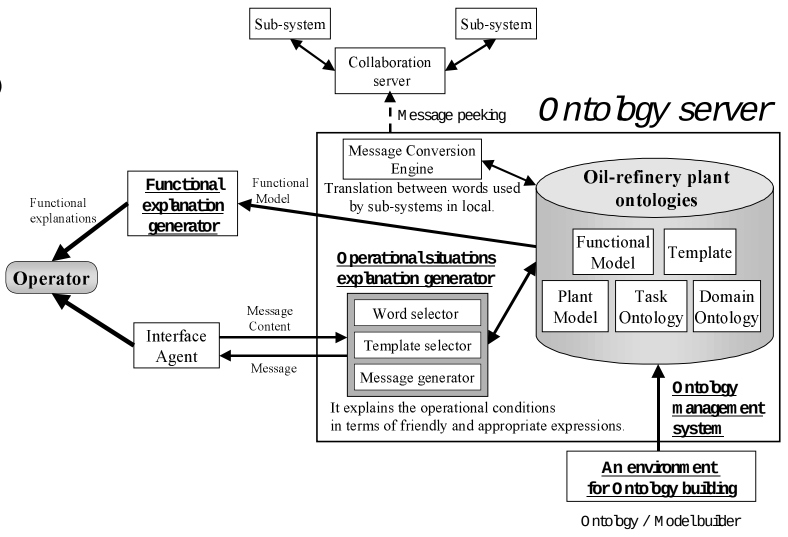

Sub-system
Sub-system
Collaboration server
Message peeking
Ontology server
Message Conversion Engine
Translation between words used by sub-systems in local.
Oil-refinery plant ontologies
Functional Model
Template
Plant Model
Task Ontology
Domain Ontology
Functional explanation generator
Functional Model
Functional explanations
Operator
Operational situations explanation generator
Word selector
Template selector
Message generator
Interface Agent
Message Content
Message
It explains the operational conditions in terms of friendly and appropriate expressions.
Ontology management system
An environment for Ontology building
Ontology / Modelbuilder

Figure 4. Block diagram of Ontology server

manipulate the functional terms which are grounded in computational behavioral model.

5. USE OF ONTOLOGY

System Overview of Ontology Server: OS

Ontology server has several functions in the interface system.
(1)To store the plant ontology for use of representing message contents.
(2)To build a model of the target plant shared by other agents
(3)To generate natural language messages presented to operators according to appropriate word selection
(4)To generate explanation about behavior and function of sub-components specified.
(5)To provide a negotiation ontology for smooth collaboration among agents.

In general, the major contribution of OS to the whole system is to standardize concepts about the target plant each agent has as well as vocabulary used by them. Fig. 4 shows the block diagram of OS where several types of plant ontologies are stored and two functional modules such as explanation generator and message generator are also shown. OS has a stylized application protocol interface supported by DCI. In our group according which any agent can communicate with OS and can inspect the ontology in it.

Message generation with appropriate word selection

After intensive discussion with domain experts, we found a remarkable fact that they sometimes use multiple names to denote the same entity. Let us take an example shown in Fig. 5 in which two controllers exist: Level controller(LC29) and flow controller (FC29). Both controllers use the same control valve as an actuator. It is a typical example of cascaded control. LC29 takes care of the liquid level of the overhead drum which contains reflux(Naphtha). And FC29 is in charge of controlling the flow of Naphtha coming out of the overhead drum. The control valve is called "Level adjustment control valve" and "Naphtha extraction flow control valve" depending on which controller the operator focuses on. The problem, however, is that the focused controller is seldom explicit. So, the system has to infer where is the focus on and how the focus shifts during the course of operation. Further, the system has to know the term generation mechanism to attain the maximal flexibility of the system. Concerning the labels of

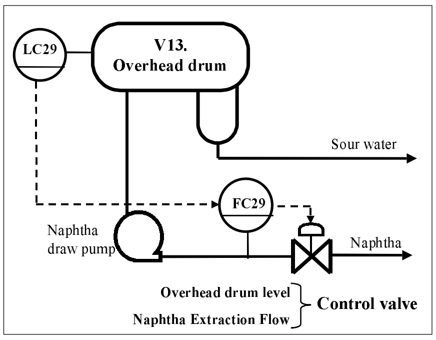


Figure 5. Cascaded control of LC and FC

each concept, we only ask the model builder of the target plant to write the default name of things appearing in the plant. The role name which will be attached to things are basically generated by the system according to the context identified. This makes system building easier. Especially, plant objects have various role names which the objects play from the viewpoint of the device which processes them. For example, ***input material*** is a typical role name which many of the plant objects could play.

Focus tracing rules: Assume the system is asked to choose an appropriate term for things(attributes, name, etc.) of an object identified by its unique identifier. Let us call the object CO: Current Object. The rule are divided into four cases according to what things of CO is:

Case 1: An attribute of a plant object(not a device)
(a) If CO is input or output of the device focused previously, then the focused device is the same as the last focused device.
(b) If the last focused device is either a controller or a control valve and CO is equal to that the controller measures, then the device the controller measures is the new focused device.
(c) If the task is decision of countermeasure, then the new focused device is the entire plant.
(d) Otherwise, no focused device exists.

Case 2: An attribute of a device
(a) If a controller which measures the attributes exists, then the new focused device is the controller.
(b) Otherwise, the new focused device is the same as the last focused device.

Case 3: A control valve(CO is a control valve itself)
(a) If CO is the same as the target object of the last focused device, the focused device is the same as the last focused device.
(b) Otherwise, the device which operates CO is the new focused device.

Case 4: others
(a) In all cases where device names are concerned other than the cases (2) and (3), CO is the new focused device.

Role name generation rules: The philosophy is to associate a default name with each thing and rules are invoked when names other than the default one should be used.

(1) For plant objects
 (a) If there is a focused device, then
  (a-1) if the focused device has role names of its input/output objects, then CO name is the role name of the input/output object of the focused device.
  (a-2) otherwise, the CO name is "<focused device> input/output".
 (b) Otherwise, default name of CO is used.

(2) For a controller and a control valve
 (a) If the focused device measures an attribute of another device, then the name of the CO is "<the device which focused device controls>" + "<the attribute>" + ["control valve" or "controller"](e.g., overhead drum level controller).
 (b) If the focused device measures an attribute of a plant object and if the device controlled by the focused device has a role name of its input/output object, then CO name is "<the role name>" + "<the attribute>" + ["control valve" or "controller"]. If there is no role name, then the CO name is "<the controlled device>" + ["inlet" or "outlet"] + "<the attribute>" + ["control valve" or "controller"](e.g., desalter inlet temperature controller).

(3) For a thing other than a control valve or a controller
 (a) If an attribute of CO is concerned, then return "<the object which the focused device measures>" + "<the attribute>".
 (b) Otherwise, CO name is the name of the focused device.

While the rule application, knowledge about each concept/term is obtained by consulting the ontology and the plant model which has been built by connecting components generated by instantiation of corresponding concepts in the ontology.

Message Generation

Classification of messages: A message should be easily understood by operators in any situation. So, its syntactic form should be highly stylized and cannot be long. We collected a lot of sample messages with the help of domain experts and classified the syntactic forms into the following seven types:

(1) Warning
(2) Near future prediction
(3) Candidate cause presentation
(4) Diagnosis result presentation
(5) Justification of decisions
(6) Countermeasure presentation
(7) Justification of countermeasures

Each type has a few templates for covering a small amount of variations. Each template is specified using concepts in the ontology. Message construction algorithm can be simple for the above reason. In fact, a simple blank filling method is employed.

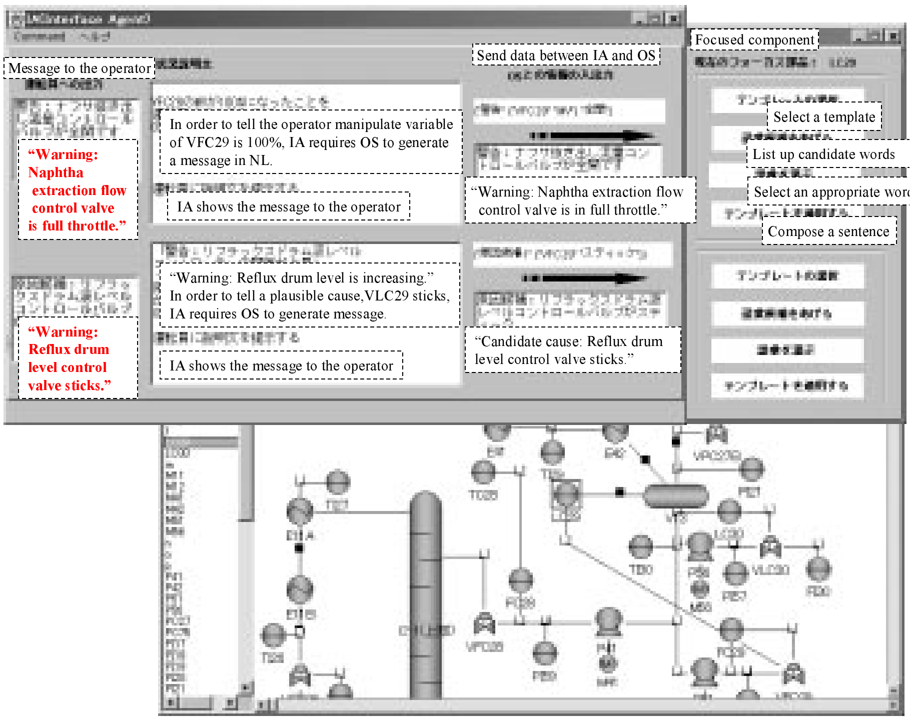
Example: OS receives a message content from Interface Agent which monitors the plant. A message content is represented in a list as follows: (<Template type> <non-terminal symbols>*)

J: [警告 [VFC29, MV] 全開]
E: [Warning [VFC29, MV] full throttle]

J: 警告:ナフサ抜き出し流量コントロールバルブが全開です
E: Warning: Naphtha extraction flow control valve is in full throttle.

J: [原因候補1 [VFC29] スティック]
E: [Candidate cause1 [VFC29] stick]

J: 原因候補：リフラックスドラム液レベルコントロールバルブがスティック
E: Plausible cause: Reflux drum level control valve sticks.



"Warning: Naphtha extraction flow control valve is full throttle."

In order to tell the operator manipulate variable of VFC29 is 100%, IA requires OS to generate a message in NL.

IA shows the message to the operator

"Warning: Naphtha extraction flow control valve is in full throttle."

"Warning: Reflux drum level control valve sticks."

"Warning: Reflux drum level is increasing." In order to tell a plausible cause, VLC29 sticks, IA requires OS to generate message.

IA shows the message to the operator

"Candidate cause: Reflux drum level control valve sticks."

Figure 6. Screen dump of a message generation demonstration

6. IMPLEMENTATION AND EVALUATION

OS and functional explanation module have been implemented in Java and Lisp. The ontology developed has been implemented in Ontology editor also developed by Mizlab. The number of concepts in the ontology is about 350. We did a small experiment to evaluate the system. Domain experts developed seven scenarios of various faults with countermeasures for them and many types of messages to operators. The templates we built cover all the messages. We picked up eight typical messages to evaluate the word selection and message generation functions of OS and confirmed all of the eight message contents sent from IA through LAN were successfully translated into the desirable messages in terms of appropriate terms. Fig. 6 shows an example screen dump of the prototype system . The lower half is a portion of the plant model we built and the left upper window is a window of IA and right one is of OS. The window configuration is arranged for demonstration.

7. CONCLUSION

Plant ontology and its roles in the interface system for oil-refinery plant operation has been discussed. The plant ontology design has been almost completed and its utility has been demonstrated in message generation with appropriate word selection. The future work until the end of the project, March in 2001 includes design of a negotiation ontology for use of flexible and powerful collaboration among agents through sophisticated negotiation. Task ontology will be used for specifying functionality of each agent and hence for helping negotiation task ontology design. This topic is one of the main topics in our research plan.

8. ACKNOWLEDGMENTS

The authors are grateful to Mr. Kouji Nanba for his initial efforts in designing the plant ontology.

This paper was prepared under an Entrustment Contract with the Laboratories of Image Information Science and Technology (LIST) from the New Energy and Industrial Technology Development Organization (NEDO) in concern with the Human Media Research and Development Project under the Industrial Science and Technology Frontier (ISTF) program of the Ministry of International Trade and Industry (MITI) of Japan.

9. REFERENCES

[1] R. Mizoguchi, A. Gofuku, Y. Matsuura, Y. Sakashita, M. Tokunaga, Human Media Interface System for the Next Generation Plant Operation, Proc. of 1999 IEEE Int'l Conf. On SMC, Tokyo, October, 1999(to appear).

[2] Mizoguchi, R. *A Step towards Ontological Engineering.* http://www.ei.sanken.osaka-u.ac.jp/english/step-onteng.html

[3] R. Mizoguchi, et al., Task Ontology for Reuse of Problem Solving Knowledge. KB&KS '95, pp.46-59, 1995.

[4] B. Chandrasekaran, J. R. Josephson, and R. Benjamins, What are ontologies, and why do we need them?, IEEE Intelligent Systems, Vol.14, No.1, pp.20-26, 1999

[5] T. Gruber, What-is-an-ontology?, http://www-ksl.stanford.edu/kst/waht-is-an-ontology.html

[6] Genesereth and Nilsson. Foundation of Artificial Intelligence, Morgan Kaufmann, 1987.

[7] M. Sasajima, Y. Kitamura, M. Ikeda, and R. Mizoguchi, FBRL:A Function and Behavior Representation Language, Proc. of IJCAI'95, pp.1830-1836, 1995

[8] A. M. Keuneke, A. device representation: the significance of functional knowledge. IEEE Expert, 24:22-25, 1991

[9] Y. Kitamura and R. Mizoguchi, Meta-Functions of Artifacts, The Thirteenth International Workshop on Qualitative Reasoning (QR-99), Scotland, June 6-9 1999.

[10] M.Ikeda, K.Seta, O.Kakusho and R.Mizoguchi, An ontology for building a conceptual problem solving model, Workshop Note of Applications of ontologies and problem solving methods, ECAI98, Brighton, UK, pp.126-13, 1998

[11] Riichiro Mizoguchi, A step towards ontological engineering, National Conference on AI of JSAI, AI-L13, 1998