PICCSS: Problem Interactive Clarification and Concurrent Solving System

Y. Nishioka

*Department of Industrial Administration,
Science University of Tokyo,
2641 Yamazaki, Noda-shi, Chiba 278, Japan,
e-mail: nishioka@iaws-20.ia.noda.sut.ac.jp*

S. Nakasuka and K. Hori

*Research Center for Advanced Science and Technology,
The University of Tokyo,
4-6-1 Komaba, Meguro-ku, Tokyo 153, Japan,
e-mail: [nakasuka,hori]@ai.rcast.u-tokyo.ac.jp*

Abstract

This paper proposes the system named as 'PICCSS' (Problem Interactive Clarification and Concurrent Solving System). PICCSS is developed for application of the Concurrent Problem Solving (CPS) framework. The CPS framework provides a new problem solving paradigm. The main features of CPS are 1) connection-oriented knowledge representation, 2) top-down function refinement, and 3) integration of spatial and temporal structures. PICCSS can support a designer on conceptual design problems where a product and manufacturing processes should be considered. After illustrating PICCSS architecture, the sequence of using PICCSS to design a product and make its manufacturing plan is described using an example of designing a certain mechanical pump. This experimental case study shows that PICCSS can deal with the conceptual design and provide not only a rough product draft but also a rough manufacturing plan very efficiently. This paper demonstrates these results with the reusable knowledge for mechanical design. Finally, our system is evaluated from some viewpoints and future enhancements to be made are given.

Keywords

conceptual design, feature based modelling, knowledge representation, ill-defined problem, CAD, CAPP, concurrent engineering

1 INTRODUCTION

Product development in manufacturers can be regarded as a problem solving process, in which both design knowledge and manufacturing knowledge are used to satisfy the

various requirements in the particular manufacturing circumstances. Although this kind of problem is asked to be supported by computers, designers and planners usually carry out most of their tasks without computers. One reason is that the product development is an ill-defined problem, and there are few problem solving frameworks for such ill-defined problems, particularly using an appropriate knowledge representation schemes (Ohsuga 1993) (Kimura 1993).

The purpose of this paper is to demonstrate applicability of our problem solving framework named CPS (Concurrent Problem Solving) to conceptual design problems in the early stage of the product development. Generally in conceptual design, the problem itself has not yet clarified until the solution of the problem is obtained. Furthermore, designers should consider the problem using their knowledge about both the product and the manufacturing processes. In order to early detect inconsistency in the problem, these different types of knowledge have to be treated concurrently. Therefore, the required framework should have capability to deal with ill-defined problems and to present a common platform for knowledge about products and manufacturing processes to be employed.

In manufacturers, the term 'product design' and 'manufacturing process planning' have many implications. This paper emphasizes that product design determines a spatial structure of the problem elements, while manufacturing process planning determines a temporal structure of the problem elements. Conventional product design and conventional manufacturing process planning are dealt with sequentially. However, supposing spatial and temporal aspects of the problem, these two types of problems cannot completely be separated in the conceptual design. Thus, the aim of our research is to integrate a problem designing a spatial structure and a problem designing a temporal structure, that is, to integrate a product design and manufacturing planning.

The rest of this paper is organized as follows: the next section introduces the new problem solving framework underlying our system, and Section 3 shows the overview of the proposed system PICCSS (Problem Interactive Clarification and Concurrent Solving System). Section 4 addresses the practical knowledge necessary for the mechanical design in our representation scheme, and then, an experimental case study of a product development is demonstrated in Section 5. Section 6 discusses the result of the experiment, and finally in Section 7, concluding remarks are given.

2 CONCURRENT PROBLEM SOLVING FRAMEWORK

Problem solving is one of the most basic activities of human beings. There are many problem solving frameworks proposed in the past researches. Most of the problem solving frameworks effective for the industrial applications are domain dependent, whereas a few frameworks such as GPS (Newell and Simon 1963) and STRIPS (Fikes and Nilsson, 1971) are domain independent. Our problem solving framework used in this paper is classified into the latter, that is, the framework is independent of the domain knowledge.

In this paper, problem solving is defined as a process which is 'filling up gaps detected between requirements and facts.' When both requirements and facts are represented by structures of elements, the gaps correspond to topological and parametric differences between the two structures. To fill the gaps, the problem solver has to modify and maintain the structures. The main features of CPS are the following three (Nishioka, 1996a):

- *Connection-oriented knowledge representation* — Knowledge is characterized depending on how the elements of knowledge are connected with each other. The elements themselves are less important than the connections of them. In the representation scheme, elements called primitives have a parameter and constraint/causality prototypes which will compose a computational model.
- *Top-down function refinement* — Problem solving begins with rough and partial knowledge, and then proceeds with more detailed knowledge which is obtained through the problem solving. Functions, which associate requirements with facts, are refined in a top-down manner, so that granularity of the represented knowledge can vary according to this top-down process.
- *Integration of spatial and temporal structures* — The problem is treated as both spatial structures and temporal structures. These two types of structures are defined using the common representation scheme, where spatial elements and temporal elements are closely connected in a primitive level. These integrated structures can be visualized from one of some spatial aspects and a temporal aspect.

2.1 Knowledge Representation

Knowledge representation on the CPS framework consists of two specification forms: *primitive* and *template*. Primitives are building blocks of knowledge, so that they compose a problem by connected with each other. These primitives are independent of particular situations, and they are not affected by experineces. Thus, primitives can be defined in advance without information about particular problems.

On the other hand, templates are pieces of knowledge, which represent various empirical situations. Knowledge called know-how and knowledge about spatial patterns such as products and parts can be represented by templates. Many conventional representation schemata such as *frame* (Minsky 1981) and *script* (Schank 1982) are similar to templates in their representation target. However, their specification looks different. A template consists of primitives in a graph structure, where nodes correspond to primitives and arcs correspond to their connections.

Primitives can be divided into four classes: *entity*, *process*, *relation*, and *operator*. Entities and relations represent a spatial structure of the problem, whereas processes and operators represent a temporal structure. On the other point of view, entities and processes are conceptual elements, whereas relations and operators are computational elements. The designer directly perceives some substances as conceptual elements, while computational elements are calculated by the computer. Primitives of each class are characterized as follows:

Entities are primitives representing physical substances. All objects visually perceived by designers can be classified into entities. In other words, objects represented by entities spread over the spatial world. Connected with relations which have position parameters, entities can be drawn on a 2-D drafting area.

Processes are temporal primitives representing activities, methods or procedures. They represent temporal substances which can be associated with various events and their pre/post-conditions. Processes can be drawn on a temporal chart if they have connection with some operators. This chart shows a process plan of the problem.

Relations are primitives which represent static relationships between more than two

primitives. Some relations connected with one primitive represent its attributes. Each relation has a parameter and some constraints. Some relations in the problem structure will be invalid depending on the parameter.

Operators are primitives representing events with their causal relationships. These causal relationships are used to make state transitions. Each operator can modify a relation parameter of its post-condition, if all of the preconditions are valid. In our scheme, a 'time' is defined by state transition, thus each operator will correspond to a 'time.'

Most of the conventional knowledge representations such as object-oriented approach relatively focus on conceptual elements. In other words, represented elements have many information about the target substances independently. In contrast to these approach, our representation scheme is connection-oriented. All the data, which the computer needs to calculate the problem, are included in the computational elements, i.e., relations and operators. Conceptual elements such as entities and processes are defined by connecting these computational elements in order to represent themselves.

In our research, the data structures have been prescribed for entities, processes, relations and operators. First, each primitive has a class name, a super class name and indexes. Second, each of entities and processes has properties which suggest sufficient connections for the class. Furthermore, each relation has a parameter and constraints, while each operator has a temporal parameter (time) and precedence and causality. More details of the representation scheme are discussed in somewhere else (Nishioka 1996b).

2.2 Main Flow of CPS

To deal with an ill-defined problem, the main flow of CPS performs in a top-down refinement loop, where the problem is gradually refined with additional requirements and facts. During the refinement loop, the human or the knowledge-base gives some functions which associate the requirements with the facts. When the problem structure has no gap between the requirements and the facts, this top-down refinement loop is terminated. There are three main steps in CPS as follows:

Problem Clarification: To begin with, requirements of the designer and facts of the circumstance are clarified in the computer according to the knowledge representation scheme. This step articulates the problem itself, so as to represent the problem in the computer in the form of a structure of primitives. Functional sentences are used for articulating the problem and transforming it into the computer.

Consistency Management: The clarified structure may initially have some inconsistencies in terms of constraints and causality. The consistency management deals with this inconsistencies. The constraint satisfaction is performed to establish a spatial structure, while the state transition is executed to establish a temporal structure. Precedence is created dynamically on operators through the state transition. This two calculation modules are executed complementarily.

Problem Evaluation: In this step, the results of the consistency management are evaluated and visualized for the designer. Even if the problem represented in the computer has no gap between the requirements and the facts, it may not commit to the real world problem. Therefore, the problem is visualized using some significant viewpoints,

so that the designer evaluates these solution candidates with his/her own knowledge. The main loop is terminated depending on the evaluation.

3 PICCSS IMPLEMENTATION

PICCSS is a prototype system where the CPS framework is underlying. It is implemented by C++ and Motif on an UNIX workstation. This prototype system works for the user's problem which is not clear in the beginning and has spatial and temporal aspects. PICCSS supports the user to solve the problem interactively, by clarifying the problem itself and making a consistent structure in which the requirements and the facts are definitely associated.

PICCSS prescribes a guideline of problem solving and a knowledge representation scheme. The CPS framework doesn't have any knowledge about particular problem solving. Thus, in PICCSS, all kinds of knowledge are detachable in the knowledge-base, the database and the case-base. Generally, problem solving performs well entirely depending on the stored knowledge, so that the knowledge in PICCSS takes the most important role. The three storage areas for knowledge are characterized as follows:

Knowledge-base: Knowledge represented by primitives or templates is stored in the knowledge-base. Primitives are prepared in advance by knowledge engineers to sufficiently represent various problems in the domain. Templates are extracted from the past cases considering efficiency of reuse of the knowledge. Both primitives and template are retrieved using indexes.

Database: In the real world, facts of the circumstance consist of various numerical data. These data are prepared in the database and will be substituted for the parameters of the primitives. The database manages the data associated with the fact of the problem by monitoring circumstances or allowing the user to input significant data.

Case-base: The case-base has various instances of the problem solving in the past. These instances contain not only final solutions of the past problems but also an incomplete part of the problem structure through the problem solving processes. These instances are called cases. The stored cases are used to extract new templates or retrieved directly for similar problems.

PICCSS has several modules as shown in Fig. 1. All of the modules are associated with the problem structure, which the system is clarifying through the problem solving. Each module has a graphical user interface to execute its functions cooperatively with the user. Functions of each module are briefly described as follows:

Requirement Editor supports articulating the users' requirements using functional sentences, and then the terms obtained here are translated into primitives. These primitives are connected considering the semantics of the sentence.

Structure Editor displays the problem structure and allows the user directly to make or modify the connections of the primitives. According to the level of the problem refinement, this editor controls the granularity of the displayed primitives.

Parameter Editor substitutes values of requirements or facts for parameters in the

Figure 1 System Architecture

problem structure. This editor also changes attributes 'fixed/free' and 'valid/invalid' of each parameter.

Constraint Manager calculates parameters to satisfy the constraints and causality of the problem. The constraint satisfaction module performs to calculate a spatial structure, whereas the state transition module provides a temporal structure.

Problem Visualizer visualizes the problem from some significant viewpoints. In visualizing, some primitives of the problem are filtered, and then the remainder is transformed on a selected coordinate system.

Knowledge-base Manager retrieves primitives or templates using indexes. If an appropriate knowledge is retrieved, it is applied to the corresponding part of the problem structure. Similarity between a template and its target is considered there.

Fact Emulator monitors the circumstances and prepares data of the facts necessary for constructing or solving the problem. Fact data stored in the database are substituted for the corresponding part of the structure.

Case Manager registers the new case of the problem solving, and retrieves an appropriate knowledge from various problem structures in the past. This manages the cases in order to efficiently reuse the stored knowledge.

The problem structure obtained as a result can be translated into some data formats and output to the other systems such as CAD/CAM and CAPP. For example, CAD system needs spatial aspects of the problem, thus 2-D or 3-D shapes of the product are composed from two or three views of the single dimensional drafts respectively. On the

other hand, temporal aspects of the problem are used for CAM or CAPP systems, which deal with sequences of operations, manufacturing processes and so on.

According to the CPS framework, PICCSS performs with the user to solve his/her problem. First of all, the problem clarification step is executed on Requirement Editor and Fact Emulator, both of which are associated with Knowledge-base Manager. Secondly, Constraint Manager performs by means of constraint satisfaction and state transition. Finally, in the problem evaluation step, Problem Visualizer shows the structure in significant ways to evaluate the problem. Structure Editor and Parameter Editor are frequently employed through the problem solving process.

4 KNOWLEDGE FOR MECHANICAL DESIGN

In the CPS framework, primitives are ontology of the problem domain. Here, ontology is a system of vocabulary used as a fundamental concepts in building an artificial system (Mizoguchi 1995). Thus, it is necessary to prepare the primitives sufficient for the problem domain in advance. Once primitives are registered, they determine the problem domain where the designer can deal with the problem. This section shows some usable primitives for the mechanical design field. PICCSS currently has more than 850 primitives for the mechanical design.

4.1 Primitives for Products

The registered primitives in PICCSS include entities which compose a spatial structure of a problem. In the mechanical design, a spatial structure contains a shape of a product, a shape of a part, an assembly relation of parts, and so on. In order to efficiently represent the spatial structure, primitives listed in Table 1 are prepared. Table 1 shows some typical entities, where various relations and operators are listed as their properties. These entities almost correspond to form features of the feature-based design paradigm (Salomons 1993).

The representation of the entities takes a single-dimensional coordinate system, so that geometric features consist of three types of entities: x-coordinate entities, y-coordinate entities and z-coordinate entities. Moreover, each coordinate entity may be divided into two directions such as right direction and left direction, when the feature is not symmetrical. For example, the entity *step_circ*, which represents a circular step on a cylinder, has six sub-classes: *step_circ_r*, *step_circ_l*, *step_circ_u*, *step_circ_d*, *step_circ_f*, *step_circ_b*.

Each entity connects with relations or operators. These connections sufficient for representing the target element are specified in the property list of the entity. For example, *cylinder_h* has 15 properties: *org_x*, *org_y*, *org_center_h*, *org_center_v*, *diam_h*, *mirror_face_v*, *length_h*, *center_h*, *center_v*, *cface_u*, *cface_d*, *pface_r*, *pface_l*, *cylinder_exist*, and *making_cylinder*. In this property list, *making_cylinder* is an operator, and the others are relations.

Some of relations connecting the entities have constraints. These constraints affect to characterize each entity. For example, *length_h* of *cylinder_h* has a constraint which defines the length of the cylinder. The expression shows that the distance between *pface_r* and *pface_l* is equal to the parameter of *length_h*, i.e., $pface_r - pface_l = length_h$. In some cases that relations connect two different entities, the constraints might describe binary

Table 1 Entities for Product Model

entity name	super class	list of properties
block_v	block	org_x org_y org_center_h org_center_v mirror_face_h mirror_face_v length_h length_v center_v center_h pface_r pface_l pface_u pface_d block_exist making_block*
cylinder_h	cylinder	org_x org_y org_center_h org_center_v diam_h mirror_face_v length_h center_h center_v cface_u cface_d pface_r pface_l cylinder_exist making_cylinder*
tube_h	tube	org_x org_y org_center_h diam_h length_h diam_in_h thick_circ_u thick_circ_d center_h cface_u cface_d pface_r pface_l cface_ccv_u cface_ccv_d tube_exist making_tube*
step_circ_r	step_circ	org_x org_y org_pface_r diam_ppd_h diam_h depth_circ_d depth_circ_u center_h cedge_d cedge_u cface_d cface_u pface_r has_step_circ cutting_step_circ*
slot_circ_h	step_circ	org_x org_y diam_ppd_h diam_h gap_face_h depth_circ_u depth_circ_d center_h cedge_d cedge_u cface_d cface_u pface_l pface_r has_slot_circ cutting_slot_circ*
hole_through_h	hole_through	org_x org_y diam_in_h length_h center_h pface_r pface_l cface_ccv_u cface_ccv_d has_hole_through cutting_hole_through*
hole_blind_d	hole_blind	org_x org_y diam_in_v depth_cedge_d center_v cedge_d pface_d cface_ccv_r cface_ccv_l has_hole_blind cutting_hole_blind*
pocket_d	pocket	org_x org_y org_center_v mirror_face_v gap_face_h depth_d center_v edge_d pface_d pface_l pface_r has_pocket cutting_pocket*
thread_cvx_r	thread_cvx	org_x org_y diam_h thread_length_r center_h cface_d cface_u pface_r cedge_r has_thread_cvx cutting_thread_cvx*
thread_ccv_r	thread_ccv	org_x org_y diam_in_h thread_length_r center_h cface_ccv_d cface_ccv_u pface_r has_thread_ccv cedge_r cutting_thread_ccv*

The properties with * mark are operators, the others are relations.

relationships such as mating relations, assembly relations and some other manufacturing relations.

4.2 Primitives for Manufacturing

Most of the other registered primitives in PICCSS are relative to the temporal structure of a problem. Here, temporal structures correspond to manufacturing processes which have various causality. Although entities corresponding to machines, tools, materials and other resources are important in detail, manufacturing processes are mainly represented by processes.

Processes for manufacturing are briefly listed in Table 2. The name of super class and the list of properties for each process can be shown there. For example, cut_step_circ is a sub-class of cut_step and has six properties: cutting_step_circ, cutting_step_circ_dur, chucking, chucking_dur, tool_prepared and has_step_circ. In this list, cutting_step_circ and chucking are operators which will compose a temporal structure. The others in the list are relations, which represent preconditions, post-conditions or constraints associated with

Table 2 Processes for Manufacturing

process name	super class	list of properties
cut_step_circ	cut_step	cutting_step_circ* cutting_step_circ_dur chucking* chucking_dur tool_prepared has_step_circ
cut_off_bar	cut_off	cutting_off_bar* cutting_off_bar_dur chucking* chucking_dur tool_prepared cylinder_exist
cut_off_block	cut_off	cutting_off_block* cutting_off_block_dur chucking* chucking_dur tool_prepared block_exist
cut_hole_through	cut_hole	cutting_hole_through* cutting_hole_through_dur chucking* chucking_dur marking_center* marking_center_dur tool_prepared has_hole_through
cut_hole_blind	cut_hole	cutting_hole_blind* cutting_hole_blind_dur chucking* chucking_dur marking_center* marking_center_dur tool_prepared has_hole_blind
cut_pocket	cut	cutting_pocket* cutting_pocket_dur has_pocket
cut_thread	cut	cutting_thread_cvx* cutting_thread_cvx_dur chucking* chucking_dur tool_prepared has_thread_cvx
tap	cut	cutting_thread_ccv* cutting_thread_ccv_dur chucking* chucking_dur tool_prepared has_thread_ccv
insert	insert	labor_exist inserting* inserting_dur inserted
fix_by_thread	fix	fixing_by_thread* fixing_by_thread_dur labor_exist fixed_by_thread
press_round	press	pressing_round* pressing_round_dur holding* holding_dur tool_prepared plate_round_exist
take_from_stock	take	exist_in_stock taking_from_stock* transferring* transferring_dur exist

The properties with * mark are operators, the others are relations.

the operators. In the case of cut_step_circ, cutting_step_circ_dur has a parameter of the precedence of cutting_step_circ. The relation chucking_dur has a constraint and a parameter which define the duration between two operators: chucking and cutting_step_circ. Finally, tool_prepared is the precondition and has_step_circ is the post-condition of cutting_step_circ.

It is important to consider whether a constraint or precedence the designer chooses to represent as a relationship between two operators. A permanent relationship between two operators might be generally represented by a constraint of a relation. On the other hand, where the target operator (successor) of the relationship is determined dynamically, it should be represented by precedence of the target operator (predecessor) with other relations such as a duration parameter.

4.3 Product and Process Correspondence

In the mechanical design, correspondence of the product shape to the manufacturing processes in the aggregate level is very difficult to find out because of its combinatorial complexity. However, in the primitive level, each feature of the product shape can be associated with some certain manufacturing processes. Consequently, it is possible to prepare various typical correspondence between a product and processes in advance.

(a) conceptual model

(b) connections of computational elements

Figure 2 Example of product and process correspondence

For example, Fig. 2 shows a template which represents correspondence between entities of a product and processes of manufacturing, where nodes with a ellipse correspond to entities, nodes with a rectangle correspond to processes, nodes with a small square correspond to relations, and nodes with a small circle correspond to operators. Dashed lines represent preconditions of operators. In Fig. 2 (a), *cylinder_h* and *step_circ_r* are entities representing a feature of a mechanical part. This shows that the part has a circular step. *Nc_machine* and *cut_circ_tool* are also entities, but these are for manufacturing circumstances.

On the other hand, processes such as *cut_step_circ* and *prepare_cut_tool* are connected with these entities through relations and operators. For example, the relation *has_step_circ_r*, which represents that the cylinder has a circular step, is connected by the operator *cut-*

ting_step_circ, so that the relationship between cut_step_circ, cylinder_h and step_circ_r is established. Because of understandability, Fig. 2 (a) omits primitives connecting with only one primitive. This diagram is a portion of a conceptual model of a problem.

Fig. 2 (b) represents the same knowledge as (a), but it shows connections of all the computational elements, where entities and processes are eliminated. This diagram is used to automatically generate numerical expressions of constraints and causality to calculate the solutions. In Fig. 2 (b), each node has a parameter, while connections represent constraints or causality. For example, has_step_circ_r has one Boolean parameter and three constraints: $cface_u = cedge_u$, $cface_d = cedge_u$, and $center_h = center_h$. The Boolean parameter is changed into true by the operator cutting_step_circ, if all its three preconditions: cylinder_exist, cutting_step_circ_dur and tool_prepared are valid.

5 EXPERIMENTAL RESULT

This section reports an experimental case study of a conceptual design. An interview with a designer of a target mechanical product is held to investigate what functions he considers during the conceptual design processes. After preparing primitives which can sufficiently represent the problem as vocabulary, an experiment of a problem solving is executed on PICCSS. According to the functional guideline described by the designer, a knowledge engineer execute the problem solving with the designer until a rough design and a rough manufacturing plan are obtained.

5.1 Functional Sentences

The product for the experiment is an aired pump, which drives using energy of compressed air. First of all, functions of the pump, in which various requirements and facts are associated, are described by the designer in Functional Sentences (FSs). These FSs were obtained through the top-down design processes. Some of FSs early in the refinement steps are shown in Table 3.

In the conceptual design stage, the designer recognizes various elements of the problem by means of articulation in his/her mental world (Hori 1994). This articulation process occurs with terms in FSs, since any drafts of the product have not been obtained. Table 3 describes that, the aired pump basically should be a machine which transfers slurry from the inlet of the manifold to the outlet of the manifold by compressed air. This FS is subject to the problem structure. Thus, the FS is not only a result of articulation, but also a result of problem clarification on PICCSS.

To accomplish the top level function of the aired pump, sub-functions, which will satisfy the upper function, are considered. This top-down function refinement provides various FSs one after another. Sometimes, appropriate functions are retrieved from the knowledge-base, and sometimes they are given by the designer. After completing the experiment, the structure of FSs are established hierarchically following after Table 3.

A rough sketch of the product is usually drawn by designers. This information is one of important results of conceptual design. Fig. 3 is a rough sketch of the case study in which various spatial information about the product is represented. This sketch was drawn by the designer during the interview. Note that these FSs and the rough sketch have not been

Table 3 Functional Sentences in the Conceptual Design

level	no	upr	functional sentences
1	01	-	Transfer slurry from the inlet of the manifold to the outlet of the manifold by compressed air.
1	02	-	Control the flow of slurry.
2	03	01	Transform compressed air into the flow of slurry by the diaphragm.
3	04	03	Get outlet pressure by the forward motion of the diaphragm.
3	05	03	Get inlet pressure by the backward motion of the diaphragm.
2	06	01	Make reciprocating motion of the diaphragm.
2	07	01	Keep the direction of the flow in the manifold.
3	08	06	Put two diaphragms symmetrically.
3	09	06	Supply compressed air to the air room when the diaphragm moves forward.
3	10	06	Release compressed air from the air room when the diaphragm moves backward.
3	11	06	Connect the two diaphragms by a shaft.
4	12	11	Transmit the forward motion of one diaphragm to the backward motion of another diaphragm.
3	13	06	Change the destination of the compressed air when the forward motion is finished.
4	14	13	...

Figure 3 Rough Sketch of the Product

completed before the problem solving is finished. They were actually obtained through the problem solving in step by step as results of the problem clarification.

5.2 Design Parameters

The quantitative requirements of the designer are given into the problem structure by substituting for the parameters. Some of the parameters are described definitely in FSs. The others are described implicitly in qualitative expressions. The values substituted for the parameters affect as maximum or minimum bounds through the constraint satisfaction. Of course, these design parameters may be changed with constraint relaxation, by modifying the requirements associated with these parameters. The constraint relaxation is performed considering the priority values of the parameters.

The design parameters and their values are listed in Table 4. In this experimental case, the final level of the problem structure has 12 parameters which are necessary to calculate all of the spatial relationships on the draft. These parameters are already associated with the upper level of the problem. Thus, these values are regarded as not only the requirement of the designer but also the result parameters of the design problem, because they might be changed. For the present, default parameters shown in Table 5 also need to be defined. However, these default parameters will be omitted using some empirical knowledge on each particular problem.

Table 4 Design Parameters

ID.	description	value
A	distance between diaphragms	140
B	diameter of plates	56
C	thickness of plates	5
D	stroke	12.5
E	body width	76
F	body height	52
G	body inside width	46
H	diameter of air hole	3
I	distance of air holes	12
J	diameter of shaft	10
K	diameter of thread	6
L	height of change block	5

Table. 5 Default Parameters

ID.	description	value
M	slot depth of shaft	3
N	thickness of slider	2
O	length of slider step	3
P	depth of slider slot	4
Q	depth of change block pocket	2
R	gap of change block and slider	2
S	length of thread	12
T	intersection of thread	6
U	diameter of nut	12
V	thickness of nut	10
W	depth of nut thread	8

5.3 Top-down Refinement

FSs are described by the designer in a top-down manner. This top-down problem refinement is allowed by PICCSS, where the FSs can be replaced by primitives and their connections. Translating a FS to the representation form interactively, terms in the FS are selected in turn, and then, PICCSS retrieves and generates primitives which accurately

correspond to the terms. The indexes defined for each primitive are used to retrieve them. After generating the primitives in the computer, they are connected each other.

In the refinement process, some primitives can be replaced by more detailed primitives, if the functions of these detailed primitives satisfy the functions of the upper primitives. Between these two refinement levels, the granularity of the representation and applicability to the real world are different. PICCSS keeps the relationships over the two different refinement levels. Consequently, this top-down refinement process is performed supervised by the designer and the knowledge engineer.

Fig.4 illustrates the problem structure in PICCSS in the early stage of the function refinement. This structure is corresponding to FSs from no. 01 to no. 07 of Table 3. First of all, the top level of the FSs: ‘Transfer slurry from the inlet of the manifold to the outlet of the manifold by compressed air’ is represented by ten hatched primitives: manifold_1, inlet_1, outlet_1, slurry_1, compressed_air_1, at_1, at_2, exist_1, transfer_1, and transferring_1. According to the semantics of the FS, these primitives are connected to each other.

The first level of the functional representation drives the first loop of the main flow of CPS. Here, if there are some constraints or causality in the problem structure, the consistency management is executed. Then, the problem solving of the next level is carried out iteratively in the same way, until enough requirements and facts are clarified and there are no conflict nor inconsistency in the problem structure. In this experiment, the depth

Figure 4 Initial Problem Structure

Figure 5 Problem Structure of the Pump (Final Level)

of the refinement level to clarify the problem is same as the level of the rough sketch shown in Fig. 3.

When the refinement of the problem is terminated, most of the primitives in the final level are the registered primitives described in the previous section. Thus, the problem structure with these primitives can be calculated precisely to produce the spatial and the temporal structures. The final level of the problem structure of the pump is shown in Fig. 5. This level has more than 620 instances and more than 160 classes of primitives. The diagram of Fig. 5 is displayed by the Structure Editor, where relations and operators are drawn by their plain name and hatched name respectively. In this diagram, most of the primitives are omitted to display because of understandability.

Figure 6 Spatial Chart (1-D Draft of Product)

5.4 Problem Visualization

The problem structure shown in Fig. 5 has various information about both a product shape and manufacturing processes. However, the information is contained by the unified structure of entities, processes, relations and operators, so that the designer hardly comprehends the information. Thus, it is necessary to visualize the information through the PICCSS graphical user interfaces. Using this, the designer can evaluate the problem by investigating the solution candidates from some significant viewpoints.

Figure 7 Temporal Chart (Manufacturing Process Plan)

Some visualized charts are obtained in the case study. Each of the charts is on a different viewpoint. Two of these charts are illustrated in Fig. 6 and Fig. 7. First of all, Fig. 6 is a result of visualizing the problem structure from one of spatial viewpoints. This chart can be regarded as a single-dimensional draft of the product. Fig. 6 represents horizontal positions of some parts, e.g., the shaft (cylinder_h_1, tube_h_2, tube_h_3), the slider (tube_h_1, slot_circ_h_1), the changing block (block_v_2, pocket_u_1), and the body (block_v_1, hole_through_v_1, hole_blind_d_1, hole_bline_d_2, pocket_d_1). There are another spatial charts, one of which represents vertical positions of these parts. Combining the horizontal chart and the vertical chart, 2-D product shape can be generated in order to send data to the conventional CAD systems.

On the other hand, Fig. 7 is a temporal chart which represents a manufacturing process plan. Primitives listed on the left are processes, while the plots on the chart are operators. Dashed lines connecting different processes represent their precedence. For example, inserting_1 is a predecessor of inserting_2 and inserting_3, which are again predecessors of inserting_4 and inserting_6 respectively. Precedence between two operators is generated by Constraint Manager in accordance with the causality of the operators.

Using a common algorithm on Problem Visualizer, these various charts are visualized in order to evaluate the solution. In this case study, Fig. 6 and Fig. 7 were accepted for the final solution of the conceptual design.

6 DISCUSSIONS

The proposed system and the underlying framework don't provide knowledge of problem solving. They only provide a platform for problem solving. Thus, it is necessary to develop appropriate knowledge for particular problem domain. Because the evaluation for this kind of system is difficult, we discuss from the following five criteria. On each of these criteria, we evaluate our system to be one of four subjective levels made of excellent, good, minor and questionable.

First, friendliness of the problem specification is minor. It is because the designer is not familiar with the connection-oriented knowledge representation. In particular, primitives without any instructions to use are too hard to be selected. Second, understandability of the display information is questionable. A problem, which consists of primitives and their connections, is generally difficult to understand. Thus, visualization technique should be applied more flexibly. Third, reusability of the registered knowledge is good. This characteristic is remarkable in lower level of the problem structure. In the experiment, 75.0 % of the primitives in the final level are reusable.

Fourth, correctness of the calculated results is excellent. Since the computer calculates accurately depending on the stored knowledge, capability of representing knowledge is much important. Thus, it is significant point that the proposed system allows the designer to represent knowledge from various aspects. Finally, appropriateness of the system scope is good. Particularly, it is remarkable that the early stage of product design is supported by the computer where top-down function refinement is performed. Even if the knowledge has different aspects and has not completely clarified yet, the proposed system can deal with the problem.

In spite of informality of the knowledge used in the conceptual design, PICCSS efficiently dealt with the knowledge and allowed the designer to synthesize it. Consequently, it can be argued that PICCSS totally performs well in the conceptual design problem.

There are some practical research for the conceptual design. However, the application of these research to the conceptual design in the real industry has not been popular yet. Thus, we discussed comparing the proposed system with one of the conventional representation methods, rough sketching. The comparison between the rough sketch of Fig. 3 and the solution provided indicates the following advantages of PICCSS.

- PICCSS can explicitly deal with causality which produces behavior of the product. For example, the reciprocating motion of the shaft can be represented, where two states of the limit positions and the transition rules between them are defined. In the sketch, it is difficult to represent the state transition or the sequences of such a motion.
- Manufacturing processes associated with products or parts are represented in PICCSS. Moreover, these manufacturing processes may not be fixed for each feature of the product, that is, alternatives can be described in the problem structure. This ambiguous representation gives capability to reorganize the plan in the following stages.
- Functional descriptions are also put into the problem structure. In contrast to write them on the rough sketch, PICCSS keeps the relationship between the descriptions and some spatial or temporal features. These functional relationships represent some information about the refinement processes so that the design process knowledge can be reused as well.

In conceptual design, the hierarchy of FSs is an important result. In PICCSS, each FS is given by the designer, but it is strongly affected by PICCSS in concerning with various knowledge about a product and manufacturing. Capability for representing knowledge and efficiency of reusing the knowledge should be focused in evaluating the knowledge intensive CAD.

7 CONCLUSION

This paper, first, introduced the CPS framework developed for ill-defined problems such as conceptual design. To clarify the problem, the CPS framework has capability to assist a designer to articulate knowledge about requirements and facts. The main flow of CPS performs in a top-down manner so that functions, which associate the requirements with facts, are described hierarchically. There, this paper also showed that a spatial structure and a temporal structure of the problem are treated in a common platform.

A conceptual design support system named PICCSS was then proposed. This system was implemented in the computer according to the CPS framework. In the experimental case study, requirements on the mechanical product were fragmentarily represented, and then, the problem to be solved was clarified. During the clarification of the problem, various constraints and causality were calculated until all the requirements and facts were satisfied. Finally, PICCSS provided a rough product draft and a rough manufacturing plan as a solution of the problem.

In this problem solving process, PICCSS allowed the designer to synthesize his/her knowledge in the computer using prepared primitives. During this, the various knowledge was communicated between the designer and the computer. The experimental results showed that PICCSS performs well in the conceptual design in terms of knowledge intensive CAD. Furthermore, it can be argued that the CPS framework is much appropriate to this kind of problems because of its representation scheme and the calculation algorithm.

In terms of knowledge intensive CAD, the advantages of the proposed approach provide some industrial expectations. The first is to shorten time of the design process by reusing valuable knowledge in the past. The second is to detect design failure by producing a rough manufacturing plan in advance, and the third is to take massive information for the conceptual design using the computer power.

Many future works can be listed such as semi-automatic function refinement, acquisition of template knowledge from the case-base, interfaces to CAD/CAM and CAPP, and support of collaborative product development.

REFERENCES

- Finger, S. and Dixon, J.R. (1989) A Review of Research in Mechanical Engineering Design. Part I: Descriptive, Prescriptive, and Computer-Based Models of Design Process. *Research in Engineering Design*, **1**, 51-67.
- Fikes, R.E. and Nilsson, N.J. (1971) STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, **2**, 189-208
- Hori, K. (1994) A System for Aiding Creative Concept Formation. *IEEE Transactions on Systems, Man, and Cybernetics*, **24**, 882-894.

- Kimura, F. (1993) Product and Process Modelling as a Kernel for Virtual Manufacturing Environment. *Annals of the CIRP*, **42**, 147–150.
- Minsky, M. (1981) A Framework for Representing Knowledge. *Mind Design* (Haugeland, J. ed.), MIT Press.
- Mizoguchi, R., Ikeda, M., Seta, K., et al. (1995) Ontology for Modeling the world from problem solving perspectives. *Proc. of IJCAI Workshop on Basic Ontological Issues in Knowledge Sharing*.
- Newell, A. and Simon, H.A. (1963) GPS: A Program That Simulates Human Thought. *Computers and Thought* (Feigenbaum, E.A. and Feldman, J. eds.), McGraw-Hill.
- Nishioka, Y. (1996a) A Study for Concurrent Problem Solving on Design and Planning (in Japanese). *Ph.D. Thesis*, The University of Tokyo.
- Nishioka, Y. (1996b) Knowledge Representation for Conceptual Design Concerning with a Product and Manufacturing Processes. *Advances in Production Management Systems* (Okino, N. et al. eds.), 199–204, IFIP.
- Ohsuga, S. (1993) How can knowledge-based systems solve large-scale problems?: model-based decomposition and problem solving. *Knowledge-Based Systems*, **6**, 38–62.
- Salomons, O.W., van Houten, F.J.A.M. and Kalas, J.J.J (1993) Review of Research in Feature-Based Design. *Journal of Manufacturing Systems*, **12**, 113–132.
- Schank, R.C. (1982) *Dynamic Memory*, Cambridge University Press.

BIOGRAPHY

Dr. Yasuyuki Nishioka has been a research associate at the Department of Industrial Administration, Science University of Tokyo, since 1996. From 1985 to 1994, he worked for a computer software company as a systems engineer. He received his doctor's degree in Interdisciplinary Course on Advanced Science and Technology, from the Graduate School of the University of Tokyo in 1996. His research interest includes planning and scheduling, concurrent engineering, human and computer interaction, activity modelling, organization intelligence, intelligent CAD, and future manufacturing systems.

Dr. Shinichi Nakasuka received his doctor's degree from University of Tokyo in Aeronautics in 1988. He joined IBM Tokyo Research Laboratory from 1988 to 1990 and has been an Associate Professor in Department of Aeronautics and Astronautics as well as Research Center of Advanced Science and Technology, University of Tokyo since 1990. Now he is a visiting researcher at Department of Computer Science, University of Maryland, USA. His current interest includes machine learning, autonomous systems' architecture, design of space systems, and planning and scheduling.

Dr. Koichi Hori received B. Eng, M. Eng, and Dr. Eng. degrees in electronic engineering from the University of Tokyo, Japan, in 1979, 1981, and 1984, respectively. In 1984, he joined National Institute of Japanese Literature, Japan, where he developed AI systems for literature studies. Since 1988, he has been an associate professor in Research Center for Advanced Science and Technology (RCAST), the University of Tokyo, Japan. From September 1989 to January 1990, he also held a visiting position at University of Compiegne, France. His current research interests include AI technology for supporting human creative activities, cognitive engineering, and Intelligent CAD systems.