REPRESENTING THE COLLABORATIVE DESIGN PROCESS: A PRODUCT MODEL-ORIENTED APPROACH

BANGYU LEI, TOSHIHARU TAURA
The University of Tokyo, Japan

AND

JUN NUMATA
SONY Systems Design Co., Ltd, Japan

Abstract. The collaborative design process can be viewed essentially as the evolution of product data, which is the results of a series of decisions. Contemporary product data representations in describing design deliberations are either informal with plain text or insufficient using some simple data structures. Explicit modeling of the evolution, alternatives and constraints of the product data in a large design space is crucial for capturing the process information at any a state of its recorded history. This paper develops a product data model on the basis of the integrated generic resources from STEP[1], which is used to formally describe the objective of a task, assignments and alternatives etc. of a decision, and in particulars, the constraints among decisions. A product model-oriented representation of the collaborative design process is proposed to develop a database which addresses the archiving of design history. The proposed representation focuses on the formal specifications of the process ingredients and the dependencies among these ingredients, such as product-data-model, assignment, activity, task, negotiation, and agent. Accordingly, a data model is developed in EXPRESS[2]. An object-oriented database is under development to implement the data model .

1. Introduction

The attitude of designers describing a design process is similar to that of mathematicians describing a theorem-proving process. In the formal design documentation such as drawings and design reports, the painstaking processes of trial and error, revision and conflict adjustment are all invisible. Designs are revamped and polished until all traces of how they were developed are completely hidden (Banares-Alcantara, 1991). For this reason, it may be more productive to focus on the results of the designer's thinking process rather than to address the thinking process itself. In other words, it is

[1]Standard for the Exchange of Product Model Data, -- ISO 10303.
[2]A formal data specification language, specified in ISO 10303-11.

J. S. Gero et al. (eds.), *Advances in Formal Design Methods for CAD*
© Springer Science+Business Media Dordrecht 1996

much easier to observe the events that transpired during the process. The design activities which take place when a state transition of the artifact being designed is made will provide important information. Recording all possible solutions and the argumentation for each decision-making provides access to the alternatives that were identified and the reasons for selecting or rejecting them. By making this information and its evolution explicit, other designers can avoid considering the same unfruitful areas or find some new chances when modification, redesign, or new design in the future. A number of researchers have hypothesized that capture and reuse of this evolving information has the potential for improving the design process and reusing of design information (Ullman, 1994).

On the other hand, the complexity of modern designing often demands integrated design teams. In industry, frequently, the domain knowledge required to develop a product is available; the requirements, constraints and primitive design components are usually established over decades of practices; however, the limiting factor determining the speed of product development is the efficiency with which the information environment can be coordinated to develop the product. This results in growing demands on capture and representation of not only the rationale behind design decisions of individual designers for a shared understanding, but also the coordination activities of multiple members of a design team within multiple tasks, which bind the information accessed, shared and generated, during the collaborative design process.

Recently, there is increasing interest in capturing information about design processes and recording the rationale behind the decisions affecting the evolution of the information. Published work has used the terms "design history", "design rationale", and "design intent" that manage the capture, storage and query of the evolving information (Ullman, 1994; Brown, 1994; Ganeshan et al., 1994; Chung and Goodwin, 1994). However, this effort is still in its infancy and gives rise to many questions on modeling and controlling design information in the design process, especially in the collaborative design process.

From the viewpoint of the artificial intelligence community, design has the following properties. (1) Design is an opportunistic activity. It is not performed using a fixed set of operators applied in an ordered way. Design is a process wherein various design activities occur in an opportunistic manner, either top-down or bottom-up. (2) Design is an exploration activity. It is classified as exploration (Smithers et al., 1989) rather than search, because knowledge about the space of possible solutions has to be obtained before goals can be well formulated. Typically, the initial description of the solution is incomplete and/or ambiguous and/or inconsistent. Design has a large space of possible solutions. Complexity arises from the very large number of alternatives of intermediate and final designs. (3) Problems of consistency are inevitable. Since design is an incremental activity, it is certain to reverse some of the previous decisions either for refinement, or for resolution of conflicts with other decisions self-made or made by other team members. Maintaining consistency among multiple decisions is one of the characteristics of the collaborative design process.

Taking these properties into full account, we present a product model-oriented representation of the collaborative design process, to build a data model for a database which addresses the archiving of design history and serves as the information infrastructure of a concurrent design environment in the context of an industry project at Sony Corporation. The proposed representation regards the collaborative design process as the evolution of product data. The product evolution can be viewed as the result of a series of decisions, i.e., *activities* and *negotiation*s, each of which is one step in the transformation of the *product*, made by multiple *design agent*s. A STEP-based (ISO, 1993a) *product data model* is developed to support the description of *objective*s of a task, *assignment*s and *alternative*s of a decision, and so on. Design *activities* and *negotiation*s are organized into *task*s which are charged by *agent*s. The *product data model* plays a key role in task decomposition and the interactions between the agents by providing a common ontology on product data access and consistency maintenance.

In the next section, previous approaches taken for design history representation are reviewed. The third section introduces a product data model on the basis of the integrated generic resources from STEP. Then, ingredients representing the collaborative design process are formalized and specified in EXPRESS (ISO, 1993b) using the product data model. The discussion of the current application state and desired future developments of the proposed approach, and conclusions follow.

2. Related Work

Since Mostow (1985) stated that there is a growing consensus in the artificial intelligence community that "An idealized design history is a useful abstraction of the design process," both the artificial intelligence and design science communities have been active in developing the concept of the design history as records of the rationale behind design decisions and of the intent of the designers. The root of most of the previous work lies in the IBIS[3] method (Rittel et al., 1973) for policy decision-making in the domain of government administration and planning where the deliberation process for complex problems is viewed as a process of negotiation among different groups with different stakes in the problem in terms of *issue*s (tasks,

[3]Issue Based Information System.

questions or problems), *alternatives* (proposals or concepts), *arguments* (evaluations) and *decisions*.

In the artificial intelligence community, previous work focused on the development of IBIS-based tools, e.g., gIBIS, Potts and Brun model, DRL[4], and DRCS. gIBIS is a computer tool to capture design histories and support computer-mediated teamwork (Conklin and Begeman, 1988). Potts and Brun (1988) distinguish between two types of design information: the process of design and the product of design. Designers work from an initial design problem to the final design by identifying alternatives, exploring them and then selecting one that satisfies or moves towards satisfying the design objectives. Lee (1991) extended Potts and Brun's model to develop DRL. In most embodiments of IBIS, artifact information such as goals, alternatives and specifications has been left informal. Plain textual representations are used for describing design deliberations. These efforts provide a means of organizing these deliberations in the form of nodes and links within the computer. In contrast to the natural language text representation for the contents of the network's nodes, a structured language, DRCS, attempts to represent the product and the process (Klein 1993). However, it is quite general and still under development and untested (Ullman, 1994).

The design science community has introduced the IBIS method to address the capture of design histories in different fields such as mechanical design (Ullman, 1994; Brown, 1994; Nagy and Ullman, 1992; Thompson and Lu, 1990; Chen et al., 1990), civil design (Ganeshan et al., 1994; Rosenman et al., 1994) and chemical plant design (Chung and Goodwin, 1994). Researchers in this community observe the limitations of IBIS-based approaches developed by the artificial intelligence people, that is, informal plain text representation for product, and implicit description of the constraints among decisions. To overcome these shortcomings, researchers at Oregon State University (Nagy and Ullman, 1992: Chen et al., 1990) used the decision network to index the changing state of the evolving artifacts, so that the sequence, composition and dependence between decisions are described. It further describes the constraint development and propagation and the dependence on the design specifications. Information on the product is represented using two basic structures, the features of objects i.e., "object-attribute-value", and features of relations between objects, that is, "object1-object2-relationship-attribute-value", (Ullman et al., 1994). Here an "object" is defined as an assembly, a component, a feature, a human or other identifiable physical thing that is used to describe some physical aspect of the product being designed. Around the same time, Ganeshan et al. (1994) proposed a framework to capture design history of a design process,

[4]Decision Representation Language.

taking an example in the domain of spatial layout of small buildings. They represented the product being designed with an "objective-variable-value" structure over the design space. In their approach, the 'objective' is used to define design problems and to represent intermediate stages leading to the final solution; the decision-making process is an iteration of "focus-refine-evaluate-select-resolve" with an explicit linkage to each state (objective, variable, alternative) of the product. The design reasons are represented implicitly in the iteration.

Both of the above efforts have resulted in their significant progress in introducing the formal product data into the IBIS-based representation of design processes. However, as pointed out earlier, design has a large state space, where each state corresponds to a possible solution, either intermediate or final. The design process can be seen as a navigation from an initial state, the specification of the problem, to a final state, the proposed solution. Neither the "objective-variable-value" nor the "object-attribute-value" structure is sufficient for representing all product data in the whole state space explicitly. This results in the mappings between product data and process information not explicit and even a design process that is ephemeral and difficult to manage. Therefore, we attempt to capture the design history by introducing the concept of the *product data model* to represent the product data. Few related work records the histories of information on design teams. By contrast, we introduce the concept *agent* and *task* from the research (Jin and Levitt, 1993) on distributed artificial intelligence and organization theory to represent the *negotiation* activities in the collaborative design process.

3. Product Data Model

Product modeling technologies attempt to generate an information reservoir of complete product data to support various activities at different product development phases (Krause et al., 1993). The term product model can be interpreted as the logical accumulation of all relevant information concerning a given product during the product life cycle. Although a clear trend toward a wider usage of product models and a strong emphasis on product modeling processes is observed, no definite and commonly agreed product modeling approaches exist to date. To be effective and efficient, we focus on the modeling of product information on a smaller scale, namely, we address only a formal description of the issues, alternatives, and assignments in the design decisions concerning a product.

First, let us distinguish two basic concepts: *product data* and *product data model*. The term *product data* in this paper refers to the facts, concepts, or instructions about a product or set of products in a formal manner suitable for communication, interpretation, or processing by a human being or by automatic means (ISO, 1993a); *product data model* is an information model which provides an abstract description of the product data. The evolution of product data usually begins with an ill-defined need for a product and ends with exact specifications for production, use and retirement or recycling. Designers need to make decisions about product data at different levels of abstraction. The consequences of the presence of different levels of abstraction on product data must be explored. The product data are evaluated and changed through the different stages of a design process. The forward and feedback links among the various stages of the design process imply certain mappings and feedback among various parts of product data. Constraints play a crucial role in this regard. Product data must keep all constraints and differentiate between constructed versus derived geometric and non-geometric features. In addition, several design activities may have concurrence of accesses to product data. It is necessary that all activities performed during different phases of a process chain have identical data available to them concerning a particular subject. In this sense, this paper represents the design process on the basis of two assumptions, that is, each product datum, once it is created by any a design activity, is the same to all the design activities in which it is present; the constraints among various design activities imply the constraints among various elements of the product data.

The above requirements on product data for representing design processes and recording design histories lead to challenges in the development of the product data model in this paper. The most useful basis for developing the product data model is the resource constructs from STEP (ISO, 1993a), an international standard for the computer-interpretable representation and exchange of product data. Its objective is to provide a neutral mechanism capable of describing product data throughout the life cycle of a product, independent of any particular system. The nature of this description makes it suitable not only for neutral file exchange, but also as a basis for implementing and sharing product databases, and archiving. To specify and develop the STEP information models, a variety of tools for information modeling have been used. One of the tools, the data specification language EXPRESS (ISO, 1993b), focuses on the definition of entities, which are the *object*s of interest. The definition of an entity is in terms of its properties (attributes), which are characterized by specification of a domain and the constraints on that domain. The term resource construct refers to the collection of EXPRESS language entities, types, functions, rules and references that together define a valid description of product data. In this paper, all the text descriptions on data modeling are based on the terminology of EXPRESS.

As shown in Figure 1, the developed product data model includes five partial models. The requirement model supports the specifications of a product derived from an analysis of customer needs for the product. The entity *product-concept* from STEP part 44 is used in this model. A *product-concept* is a set of product features identified by the customers or derived from customers' needs. The *product-concept* is customer-oriented, while the *product* is engineering-design or manufacturing-design oriented. A product concept is essentially a marketing idea and includes customer-driven inputs. Therefore, the market-context is also defined as an attribute of the entity *product-concept*.

The function model defines the function structure of a product. No entity from STEP can be interpreted into this model. The function specifications are expressed in plain text in the description attribute of the *function* entity. The entity *function-relationship* defines the hierarchy between *functions*, while the entity *alternative-function-relationship* defines the relationship between base function and its alternatives. More detailed discussion about function analysis can be found in another paper (Taura, 1995) by one of the authors. In the solution principle model, the entities such as *physical-law*, *physical-phenomenon*, *physical-quantity*, and *working-domain,* have been formalized to represent physical effects, for example, the friction effect described by Coulomb's law ($F_F = \mu F_N$). The entity *physical-principle* defines the relationship between a *function* and the corresponding *physical-law* selected to fulfill the *function*, e.g., the friction effect used to fulfill the function 'transfer torque'. The entity *solution-principle* can then be specified by associated *parts* selected to fulfill the function and the parts' key features selected to define the *working-domain* of the corresponding *physical-law*. The entity *alternative-principle-relationship* is formalized to describe the alternatives of a *solution-principle*. However, the relationships between *solution-principles* are not defined explicitly, since the *function-relationship* and/or *product-structure* imply them.

The development of the product structure model is based on the resource constructs from STEP part 44: 'product structure configuration' and STEP part 41: 'product description and support.' The product structure defines the different methods by which a product can be represented, as being made up of constituents. Product structure relationships are established among the assemblies and constituents that make up a product. The product structure (i.e., composition relationships) may be modeled mathematically by nodes representing assembly products and by directed links representing the "composed-of" relationship. Usually, two major data structures are used to represent product structure: bill-of-material and parts list. The bill-of-material structure is a structural description of a product in terms of its nested constituents, while the parts list structure is a structural description of a product in terms of a hierarchy of all distinct usages of its constituents. The product structure model in this paper supports both. The entities *product-definition*, *assembly-component-usage* and *alternative-product-relationship* are selected from STEP part 41 or part 44.

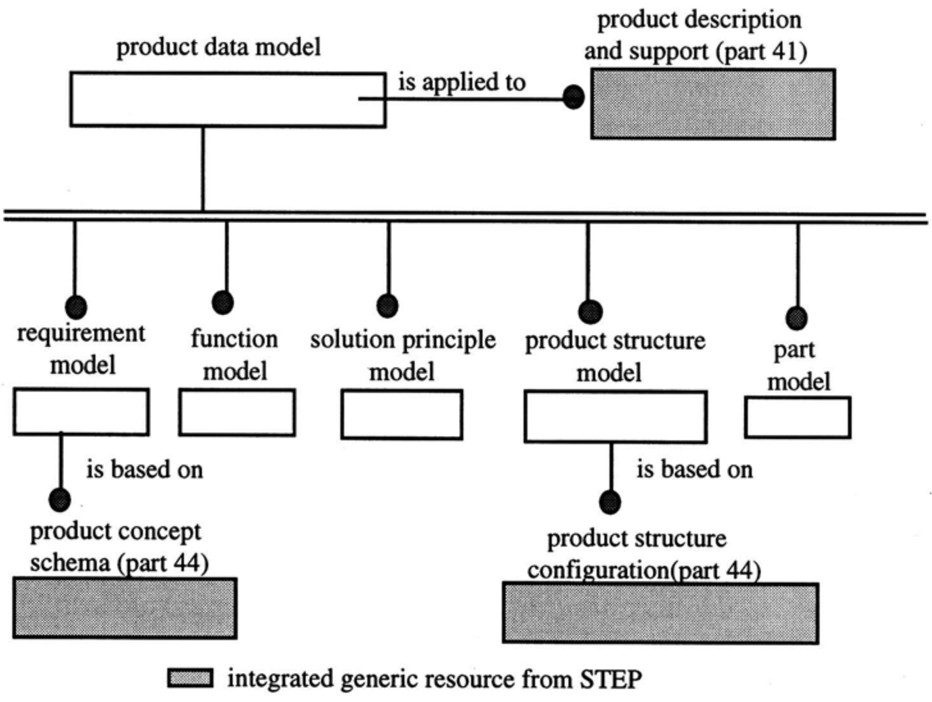

product data model
is applied to
product description and support (part 41)
requirement model
function model
solution principle model
product structure model
part model
is based on
is based on
product concept schema (part 44)
product structure configuration(part 44)
integrated generic resource from STEP

Figure 1: An overview of the product data model.

Figure 2 shows an IDEF1x graphic (NIST, 1992) of the main components for modeling the product data of a part. The resource constructs from STEP concerning the representation of material, tolerances, geometric model, and form features are interpreted into this model. This model supports the detailed design phase of a part. For an exact description of all the individual modules except the "parametric shape model," the readers are referred to the literature (Lei, 1994) written by one of the authors.

Contemporary shape representations are either geometric by CSG/B-reps or incompletely parametric by implicit form feature representations. A parametric shape model is desirable for the definition of all shape aspects into which the shape is divided, their configurations, and the constraints among them. Therefore, a generalized topology schema for parametric shape modeling has been developed. The proposed approach addresses the shape definition only with finite primitives and in a completely parametric manner. Introducing a new topological entity standing for the relationships among entities at different lower topological levels makes available a concise schema for the parametrization of both the shape aspects sharable for design and manufacture features and their dimensional relations rather than only dependency relations. It also ensures the universality of the schema for the parametrization of all shapes with or without free surfaces and makes the evaluation of such a parametric shape model into a B-rep model easy and unique. We have submitted a paper (Lei and Taura, 1995) about this parametric shape modeling to *CAD Journal.*

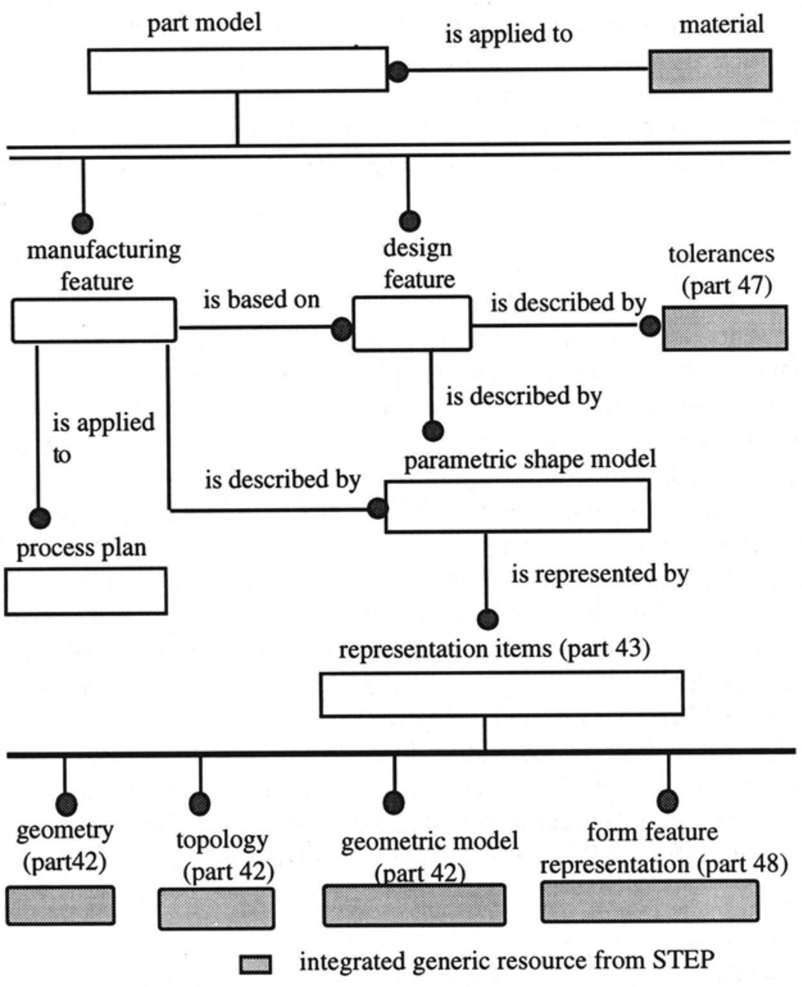


Figure 2. An IDEF1x graphic of the main components in the part model.

3. Formal Process Ingredients

To abstract and formalize the basic process ingredients, a collaborative design is generally regarded as an opportunistic process with a network of tasks. Each task records, in a temporal order, all the decision-making and conflict-resolution activities that fulfill itself. For the design deliberations of one decision, a process ingredient *activity* will be abstracted to accumulate the assertion, alternatives and assumptions of the decisions made on product data, the agent that made them, and the rationale used to make them. The mappings between process ingredients and product data take place only through three channels, that is, generations of the objectives of tasks via the ingredient *task*, determinations of the attribute values of a task's objectives via the ingredient *assignment*, and detection of conflict sources and constraints among tasks or among activities via the ingredient *negotiation*. All three channels use the product data model directly. Let us make an exact distinction between these ingredients.

3.1. ASSIGNMENT

Assignments are decisions on product data. As mentioned earlier, the collaborative design process is viewed here as the evolution of product data, which is the result of a series of decisions. Each decision is one step in the transformation of the product. In an assignment, the value of a single attribute, or values of coherent pieces of attributes if necessary, of an *entity* in the *product-data-model* is determined and recorded as assigned-data. These assigned data may be used as a proposed solution, an alternatives, an assumption, or a preference. In this sense, the ingredient *assignment* is the major channel through which process information interrelates and interacts with product data. The rationale behind these value determinations can be expressed formally by the entity *physical-law* defined in the above section, or PROCEDUREs specified in EXPRESS (ISO, 1993b) or informally by STRINGs, i.e., plain text. The rationale illustrates how and why the values are derived or determined. Once *assignment*s are committed, all assigned data will be recorded as product data that can be shared by other activities and by other agents. The EXPRESS specifications of the entity *assignment* are as follows.

```
ENTITY  assignment;
        assigned-data                 : product-data-model;
        rationale                     : SET [1:?] OF rational;
        created-by                    : (INV) activity;
        maker                         : agent;
        status                        : role;
END_ENTITY;
```

```
ENTITY   product-data-model;
         SUPERTYPE OF (requirement-model, function-model,
                        solution-principle, product-structure,
                        part-model):
END_ENTITY;

TYPE  rational = SELECT (physical-law, PROCEDURE, STRING) ;
END_TYPE;

TYPE  role = ENUMERATION OF (proposed, ready-for-alternative,
                              assumed, expected) ;
END_TYPE;
```

3.2. ACTIVITY

Activities are efforts that take place during a process. An *activity* aggregates the information generated by a single action during a process, including its objective, the decision, alternatives, and argument. There are different levels of granularity to aggregate the action information. On the finest level, information on micro decisions made at the rate of about one per minute is tracked. This shows that the fineness is necessary for completeness of information capture but is very difficult to implement and is even unrealistic for an operational system (Ullman, 1994). On the coarser project/program level of granularity, projects are defined as design activities performed by single-discipline teams whereas programs require team members from diverse disciplines. The design effort is normally seen in corporate product development plans. This is the level of information often handled by commercial systems such as IBM's Product Manager™/6000 and SDRC's DMCS[5]. Such systems may miss much important information concerning the alternatives considered, evaluations completed, and assumptions made. In this paper, action information is aggregated as activity at the attribute level of granularity. That means an activity is defined here as the design effort of a single agent on a single attribute, or coherent pieces of attributes of an *entity* in the *product-data-model*. Concretely, the *EXPRESS* specification of the entity *activity* is given below.

```
TYPE  criteria = ENUMERATION OF (cost, time, behavior, trade-off) ;
END_TYPE;

TYPE  state = ENUMERATION OF (admissible, optimal, rejected,
                               committed, retracted) ;
END_TYPE;
```

[5]Data Management and Control System.

```
ENTITY  activity;
        identification      : INTEGER;
        name                : STRING;
        goal                : task;
        assertion           : assignment;
        alternative         : OPTIONAL SET [1:?] OF assignment;
        argument            : criteria;
        maker               : agent;
        opportunity         : OPTIONAL SET [1:?] OF assignment;
        assumption          : OPTIONAL SET [1:?] OF assignment;
        status              : state;
END_ENTITY;
```

In the above specification, the attribute goal refers to the task whose objective is to be designed or determined in current activity. The task's objective will be defined in the next sub-section using any subentity of the *product-data-model.* The assertion slot records the proposed *assignment* on focused attributes of the current task's objective. The alternative slot corresponds to all the possible solutions to replace the assignment stored in the assertion slot. The attribute argument is described by the *criteria*, which explains why the *assignment* in the assertion slot is selected from all alternatives. The opportunity slot refers to the preference decision and the associated conflicts that make it unavailable. The attribute assumption assigns virtually all product data needed by executing current activity but which do not exist. The status of an *activity* can be one of the following: admissible, optimal, rejected, committed, or retracted.

3.3. TASK

A task is a set of pre-determined actions such as activities, negotiations, subtasks, and assignment-committing. Obviously, a collaborative design is not carried out by one designer in one session as a single task. In other words, collaborative design activities may occur in parallel. In this sense, the collaborative design scenario can be described by a network of tasks. *Activities* do not stand alone but are grouped into tasks. The task is determined both by the temporal ordering of *activities* and the interaction between grouped *activities.* The entity *task* in EXPRESS is specified below.

```
ENTITY  task;
        identification              : INTEGER;
        name                        : STRING;
        super-task                  : OPTIONAL SET [1:?] OF task;
        objective                   : product-data-model;
        working-activity            : LIST [1:?] OF  activity;
        charged-by                  : agent;
        conflict-resolution         : OPTIONAL SET [1:?] OF negotiation;
        subtask                     : OPTIONAL SET [1:?] OF task;
        committed-activity          : SET [1:?] OF  activity;
END_ENTITY;
```

It has been pointed out in the discussion of the level of granularity that the objective of a task is to instantiate an entity in the *product-data-model*. Hence, here is the second channel through which a process ingredient interrelates with the product data model. The attributes of the instantiated object will be determined by a list of activities which are recorded in the working-activity slot as LIST instead of SET, since the temporal order of the activities is also useful information for the design history. A task consists of at least one working activity. If multiple working activities exist in a task, the conflict-resolution slot can optionally store all possible actions to resolve the possible conflicts among the working activities using the entity *negotiation* formalized in the next section. After conflict resolutions, the resulting new set of consistent activities can be recorded in the committed-activity slot. All assigned data created in each *assignment* of the committed activities, either as assertion or as alternatives, can be committed into product data for further common sharing by the responsible agent of the current task, who is appointed in the charged-by slot. Only this agent has access to commit *assignment*s into product data in the task level, although multiple agents can submit different *assignment*s in the activity level to fulfill the objective of the current task.

3.4. NEGOTIATION

Negotiations here refer to the actions initiated to resolve conflicting activities. There are three kinds of conflicts. The simplest one is the serial collaboration between activities, that is, one activity can take place only after the decision of another activity is ascertained. In such a case, if both activities are within the same task, they can be arranged in a corresponding serial manner by the responsible agent. Otherwise, either assumptions or negotiations can be raised by the suspended activity. The second one is the conflict assignments for the same attribute from different perspectives, i.e., a single attribute of the objective of a task may be determined by multiple agents in different disciplines in multiple distinct *activities*. The last and most complex one is the dependent assignments, viz., some constraints representing the relationship among the attributes of different task objectives are invalid according to the current assignments by all corresponding activities. Once the identity of the conflicting perspectives is known, the negotiation process must be initiated to resolve conflicts.

The product data model plays a key role in such situations. As pointed out earlier, we represent the design process on the basis of two assumptions, that is, each product datum, once created by any design activity, is the same to all design activities in which it is present; and the constraints among various design activities imply the same constraints among various *object*s of the product data. If the data assigned by an activity are committed into a shared product database supported by the product data model described above, the two assumptions are admissible. The product data model written in EXPRESS is certainly object-oriented and EXPRESS supports the definition of constraints among the attributes of different entities by RULEs and PROCEDUREs. Therefore, the first assumption, together with the uniqueness of an *object,* assure the detection of the first and second kinds of conflicts. Usually, the dependence among the attributes in the same entity is defined by defining 'DERIVE' attributes in an entity of the *product-data-model,* while the attribute dependencies among multiple entities are defined as associated constraints in the form of RULEs including PROCEDUREs. This assures the detection of the third kind of conflict. In a word, conflict detection and constraints propagation can be controlled through product data, because conflict sources and/or constraints can be formally and explicitly expressed by the *product-data -model* .

```
ENTITY  negotiation;
        identification          : INTEGER;
        name                    : STRING;
        conflict-source         : SET [1:?] OF product-data-model;
        conflict-activity       : SET [2:?] OF activity;
        conflict-resolution     : SET [1:?] OF activity;
        negotiator              : SET [2:?] OF agent;
        note                    : OPTIONAL STRING;
END_ENTITY;
```

As shown in the above specification, at least two *agent*s recorded in the negotiator slot take part in negotiation. In addition, at least two conflicting activities are involved to be resolved in a negotiation and can be stored in the conflict-activity slot. The conflict-source slot records either invalid constraints or conflicting attributes of an *object* defined by the *product-data-model.* The result of a negotiation for conflicting activities may be one or a set of new *assignment*s accepted by all of the negotiators and can be represented explicitly by a set of *activities* in the conflict-resolution slot, since the new *assignment*s in these situations also involves the same properties of an *activity* such as alternatives, and assumptions. The note slot is ready as an optional property to record some other information that must be recorded but cannot be formally represented in the above-mentioned slots, because negotiations may consist of some very complex deliberations.

3.5. AGENT

An agent here refers to a combination of human and software information storage and processing. The combination ranges from an agent that is human with a software interface to interact with other agents, or an agent completely implemented in software. The agent description includes role characteristics such as position in the team hierarchy; authority for design, approval, and coordination tasks. A concrete description of the ingredient *agent* is shown in the following EXPRESS specifications.

```
ENTITY  agent;
        identification          : INTEGER;
        name                    : STRING;
        role                    : organization-role;
        in-charge-of            : OPTIONAL SET [1:?] OF  task;
        proposed-activity       : OPTIONAL SET [1:?] OF  activity;
END_ENTITY;
```

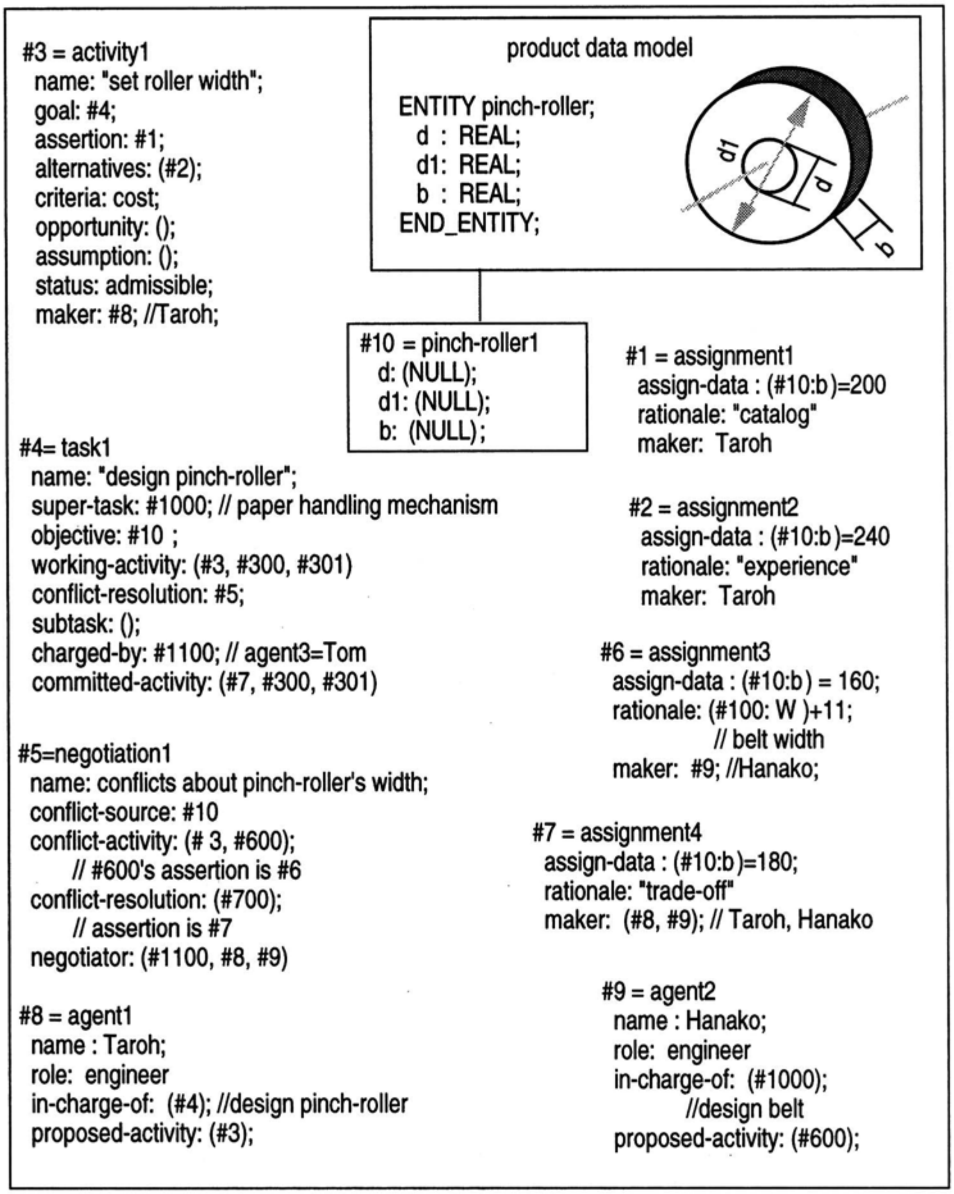
Here, the role characteristics are expressed by the entity *organization-role*, which is selected from the management_resources_schema in STEP part 41. The other two attributes define the related tasks and activities charged by the current agent.

4. Example

The above-mentioned product data model is developed in EXPRESS as a STEP application protocol. It includes the detailed specifications of schemata of the five partial models discussed above. The normative references for these schemata are as follows: STEP part 41, 42, 44, 45, 47 and 48. The data model is implemented on an object-oriented database called ONTOS in C++. All product data concerning an example shaft are instantiated on the corresponding database schema (Lei, 1994).

Based on the above-mentioned formal process ingredients, a design history base, as an extension of the product database, is under development to record the design history of Sony's color video printer UP-5000. Two of the authors have analyzed in detail the Sony's current product planning and design process, taking the development of this printer as an example (Numata and Taura, 1995). The first assembly selected to be implemented is the paper-handling system of the printer. We have focused on one particular kind of paper transport, namely, that which uses pinch rolls to isolate one sheet from the other sheets piled at the paper entrance. Figure 3 shows the example instances of the process ingredients.

The design of the isolation mechanism in the paper-handing system involves six tasks: the design of pinch roller, belt, rubber roller, gate roller, press plate and floor guide. Within the *task* for pinch roller design, task1 (#4 in Figure 3), three independent activities are planning for the inner-diameter, for the outer-diameter, and for the width. Its objective is to instantiate the entity *pinch-roller* in product data model with an empty instance, pinch-roller1 (#10 in Figure 3). Within the *activity* for determining the width of the roller (#3), two assignments, assignment1 and assignment2, are possible.



```
#3 = activity1
  name: "set roller width";
  goal: #4;
  assertion: #1;
  alternatives: (#2);
  criteria: cost;
  opportunity: ();
  assumption: ();
  status: admissible;
  maker: #8; //Taroh;
```

```
product data model

ENTITY pinch-roller;
  d  : REAL;
  d1: REAL;
  b  : REAL;
END_ENTITY;
```

```
#10 = pinch-roller1
  d: (NULL);
  d1: (NULL);
  b: (NULL);
```

```
#1 = assignment1
  assign-data : (#10:b)=200
  rationale: "catalog"
  maker:  Taroh
```

```
#4= task1
  name: "design pinch-roller";
  super-task: #1000; // paper handling mechanism
  objective: #10 ;
  working-activity: (#3, #300, #301)
  conflict-resolution: #5;
  subtask: ();
  charged-by: #1100; // agent3=Tom
  committed-activity: (#7, #300, #301)
```

```
#2 = assignment2
  assign-data : (#10:b )=240
  rationale: "experience"
  maker:  Taroh
```

```
#6 = assignment3
  assign-data : (#10:b) = 160;
  rationale: (#100: W )+11;
                 // belt width
  maker:  #9; //Hanako;
```

```
#5=negotiation1
  name: conflicts about pinch-roller's width;
  conflict-source: #10
  conflict-activity: (# 3, #600);
      // #600's assertion is #6
  conflict-resolution: (#700);
      // assertion is #7
  negotiator: (#1100, #8, #9)
```

```
#7 = assignment4
  assign-data : (#10:b )=180;
  rationale: "trade-off"
  maker:  (#8, #9); // Taroh, Hanako
```

```
#8 = agent1
  name : Taroh;
  role:  engineer
  in-charge-of:  (#4); //design pinch-roller
  proposed-activity: (#3);
```

```
#9 = agent2
  name : Hanako;
  role:  engineer
  in-charge-of:  (#1000);
         //design belt
  proposed-activity: (#600);
```

Figure 3. Example instances of the process ingredients.

The agent responsible for roller design, Taroh, asserts assignment1 in consideration of the cost criteria. However, the agent in charge of belt design, Hanako, suggests another assignment3 (#6), because of the constraint between the roller width and belt width. Then negotiation1 (#5) occurs between Taroh and Hanako and assignment4 (#7) is decided as a "trade-off". The product data model in Figure 3 is provided only for easier understanding of the design process entities. The real product data model in the database is more abstract.

5. Conclusions

A product model-oriented approach for formal representation of the collaborative design processes has been proposed to capture design histories. One of the important features of this proposed approach is introduction of the product data model to control the explicit mappings between process ingredients and product data. To be more precise, the product data model supports the generation, exploration and navigation of the objective in each task/activity in a formal manner; the product data model also explicitly represents the evolution and alternatives of product data from objectives focused through all intermediate assignments to final specifications at any state of its recorded history, beginning with the general definition of customer's requirements for a product and ending with the exact specifications for production, use, and retirement or recycling; in particular, constraints among product data defined by the product data model imply constraints among tasks , and the product data model facilitates the detection of conflict sources and propagation of the constraints among activities.

In addition, the proposed representation regards a collaborative design as an opportunistic process with a network of tasks. In each task the temporal order of the decision-making and conflict-resolution activities has been recorded. For the design deliberations on one decision, the process ingredient *activity* has been specified to record the assertion, alternatives and assumptions of the decisions made on product data, the agent that made them, and the rationale used to make them.

However, all the data models are assumed to be statically defined. There are no STEP mechanisms for modifying a product model while it is in use. Further research will focus on dynamic definition of the data model to reflect the dynamic nature of design. Another future work will be a more formal representation of the design rationale in the framework of the proposed approach by means of a formal description of the agent's knowledge and communication patterns. A concurrent design environment will be developed taking the developed data model and database as the information infrastructure.

Acknowledgments

The work reported in this paper is supported partly by Japan Society for Promotion of Science. Any opinions, findings, conclusions and recommendations expressed are those of the authors, and do not necessarily reflect the views of the sponsoring agencies.

References

Banares-Alcantara, R.: 1991, Representing the engineering design process: two hypotheses, *Computer-Aided Design*, **23** (9), 595-603.

Brown, D. C.: 1994, Rationale in design, *in* P. W. H Chung and R. Banares-Alcantara (eds), *AID'94Workshop on Representing and Using Design Rationale*, pp. 1-3.

Chen, A., McGinnis, B. and Ullman, D. G.: 1990, Design history knowledge representation and its basic computer implementation, *Proceedings of the Second International ASME Conference on Design Theory and Methodology, ASME DE,* 27, pp. 157-184.

Chung, P. W. H. and Goodwin, R.: 1994, Representing design history, *in* J. S. Gero, and F. Sudweeks (eds), *Artificial Intelligence in Design '94,* Kluwer, Dordrecht, pp. 735-752.

Conklin, J. and Begeman M. L.: 1988, gIBIS: a hypertext tool for exploratory policy discussion. *ACM Transactions on Office Information Systems,* **6**(4), 303-331.

Ganeshan, R., Garrett, J. and Finger, S.: 1994, A frramework for representing design intent, *Design Studies,* **15**(1), 59-84.

ISO: 1993a, *Standard for the Exchange of Product Model Data, ISO DIS 10303*, International Standards Organization.

ISO: 1993b, *Description Methods: the EXPRESS Language Reference Manual, ISO DIS 10303 Part 11* International Standards Organization.

Jin, Y. and Levitt R.: 1993, i-AGENTS: modeling organizational problem solving in multiagent teams, *Intelligent System in Accounting, Finance and Management,* **2**, 247-270.

Klein, M.: 1993, Capture design rationale in concurrent engineering, *IEEE Computer*, **1**, 39-47.

Krause, F-L., Kimura, F., Kjellberg, T. and Lu, S.: 1993, Product modeling, *Annals of the CIRP,* **42.**

Lee, J.: 1991, Extending the Potts and Burn model for recording design rationale, *Proceedings of the 13th International Conference on Software Engineering.*

Lei, B.: 1994, A data model for the integration of CAD/CAPP/NC, *Technical Report*, Laboratory for Machine Tools and Production Engineering (WZL), Aachen University of Technology, Germany.

Lei, B. and Taura, T.: 1995, Parametric shape modeling of mechanical parts: A generalized topology approach, *CAD* (submitted).

Mostow, J.: 1985, Toward better models of the design process, *AI Magazine,* **6**(1), 44-57.

Nagy, G. L. and Ullman D. G.: 1992, A data representation for collaborative mechanical design, *Research in Engineering Design,* **3**(4), 595-603.

NIST: 1992, *DEF1X (ICAM Definition Language 1 Extended) Integration Definition for Information modeling*, FIPS PUB XXX, NIST.

Numata, J. and Taura, T.: 1995, A network system for knowledge amplification in the product development process, *IEEE Engineering Management* (submitted).

Potts, C. and Brun, G.: 1988, Recording the reasons for design decisions, *Proceedings of the 10th International Conference on Software Engineering,* CS Press, pp. 418-427.

Rittel, H. W. J. and Webber M. M. J.: 1973, Dilemmas in a general theory of planning, *Policy Sciences,* **4**, 155-169.

Rosenman, M. A., Gero, J. S. and Maher, M. L.: 1994, Design intent and multiple abstraction, Knowledge-Based Research at the Key Centre of Design Computing, University of Sydney, *Working Paper*, Key Centre of Design Computing, University of Sydney, pp. 20-24.

Smithers, T., Conkie, A., Doheny, J., Logan, B., Mollington, K. and Tang, M. X.: 1989, Design as intelligent behavior: An AI in design research programme, *Research Paper DAI 426*, University of Edinburgh, UK.

Taura, T.: 1995, Design science for functional design process modeling, *Proceedings of the International Conference on Engineering Design, ICED'95* .

Thompson, J. and Lu, S.: 1990, Design evolution management: a methodology for representing and using design rationale, *Proceedings of Second International ASME Conference on Design Theory and Methodology*

Ullman, D. G.: 1994, Issues critical to the development of design history, design rationale and design intent systems, *in* T. K. Night and F. Mistree (eds), *Proceedings of the Sixth International ASME Conference on Design Theory and Methodology*, pp. 249-258.

Ullman, D. G., Herling D. and Sinton, A.: 1994, Analysis of protocol data to identify product information evolution and decision making process, *Analyzing Design Activity Delft Protocol Workshop*, pp. 67-82.