Product Knowledge Modelling and Management

Y. Zhang, K.J. MacCallum and A.H.B. Duffy[1]

CAD Centre, DMEM
University of Strathclyde
75 Montrose Street
Glasgow, G1 1XJ, UK
Email: yan{ken, alex}@cad.strath.ac.uk

29 May 1996

Abstract

The term, Product Knowledge is used to refer to two related but distinct concepts; the knowledge of a specific product (Specific Product Knowledge) and the knowledge of a product domain (Product Domain Knowledge). Modelling and managing Product Knowledge is an essential part of carrying out design.

A scheme is presented in this paper to model, i.e. classify, formalise and structure the product knowledge for the purpose of supporting function-oriented design. The Product Design Specification (PDS) and four types of required attributes of a specific product have been identified to form the Specific Product Knowledge involved in the product generation process and have been modelled as a PDS unit and four viewpoint models. Further, as far as Product Domain Knowledge is concerned, seven categories of that and their relationships are considered as necessary and sufficient to support function-oriented design. Thus they are modelled in our scheme. Both Specific Product Knowledge and Product Domain Knowledge are modelled at two levels, a meta-model and an information-level.

Following that, a computational implementation scheme to manage, i.e. represent and control the proposed product knowledge models within a dynamically changing environment is presented.

1. Introduction

Design is a transformation process which may be thought to start from the requirements for a new product and to move to a *product definition* which meets the requirements. The requirements for a new product can consist of function requirements (by function, we mean the qualitative description of intended purpose of a product), behaviour requirements (by behaviour, we mean the performance, working state of a product which could be described quantitatively or qualitatively by a set of variables) and constraints imposed on the design. The Product Definition is constructed from all the constituent parts, which have been fully defined by their characteristics, and the assembly relationships between these parts.

The design solution, in which the knowledge of the product being designed is encapsulated, continuously evolves during the design process as a result of carrying out design activities. This evolution is achieved by employing knowledge of the design process, the design domain and the current design solution in order to better meet the design requirements. Modelling and managing Product Knowledge, i.e. the knowledge of a specific product and the knowledge of a product domain is an essential part of carrying out design.

A product knowledge modelling and management scheme is proposed in this paper to support the function-oriented product generation process.

---

[1] Visiting Professor, Institute of Control and Engineering Design, Technical University of Denmark, 2800 Lyngby, Denmark

To understand the knowledge involved during design, issues concerned with a product and its design are discussed in Section 2. Following that, a scheme to model both the knowledge of a specific product and the knowledge of a product domain is presented in Section 3. In Section 4, a computational implementation scheme to manage, i.e. represent and control the proposed product knowledge models is presented. Finally, the main ideas proposed in this paper are summarised in Section 5.

2. Product Knowledge in Design

A model, showing the relationships between design, the product being designed and the knowledge employed in design, is shown in Figure 1. Relevant concepts are explained as follows.

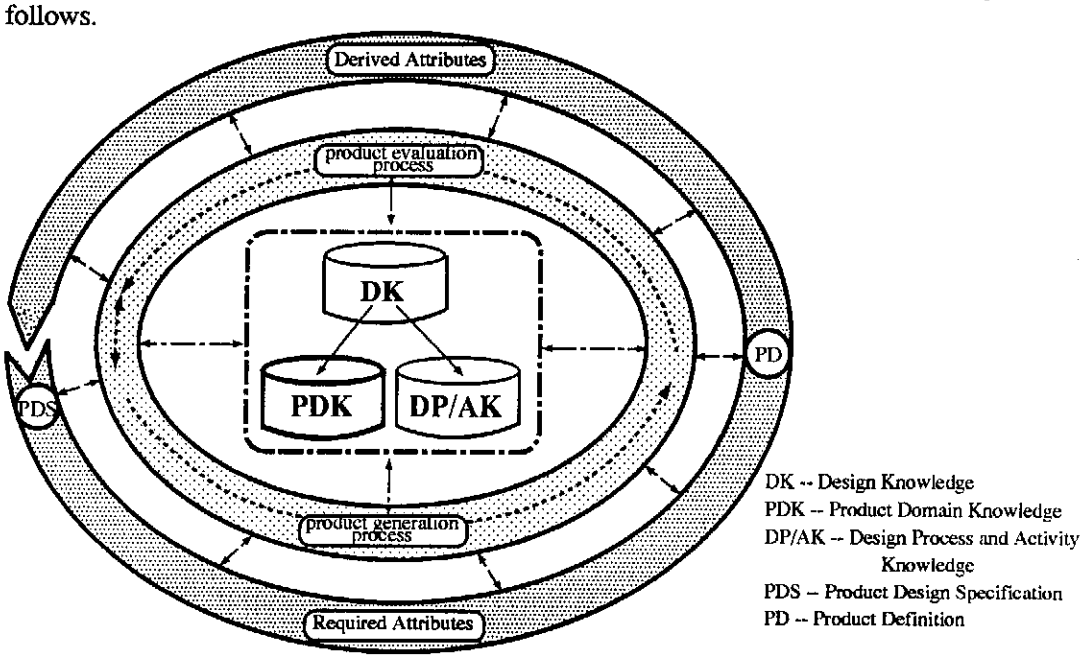

Figure1: Product and Its Design

Design of a product can be generally thought of as comprising of two kinds of interconnected processes, i.e. the *Product Generation Process* and the *Product Evaluation Process*. The product generation process starts from *initial requirements (PDS)*, which might change with the progress of design, and proceeds to generate a *Product Definition (PD)*. *Required attributes*, such as required function, behaviour, working principles, etc., are generated during the product generation process. On the other hand, the product evaluation process deals with analysing a product definition to determine *derived attributes* of the product, e.g. derived function, behaviour, etc., and then comparing them with the requirements to evaluate the design. Obviously, the derived attributes needed for evaluating a product are dictated by the required attributes.

The knowledge of a specific product is termed ***Specific Product Knowledge (SPK)***. It consists of the *PDS, required attributes, PD* and *derived attributes*.

The knowledge employed in design (Design Knowledge) can be divided into two types,

- ***Product Domain Knowledge (PDK)***: The knowledge of a product domain concerned with the nature (e.g. function, working principle, etc.) of the products in this domain.
- ***Design Process/Activity Knowledge***: The general and domain specific knowledge about design process and design activity.

In our work, ***Product Knowledge*** is used to refer to the two related but distinct concepts; ***Specific Product Knowledge*** and ***Product Domain Knowledge***, as defined above.

Our work focuses on the product generation process from the viewpoint of function-oriented design to investigate issues on modelling and managing product knowledge to facilitate the design of a product; for example, "What knowledge should be modelled?" "Why?" "How to model and manage it?", etc. A product knowledge modelling scheme is presented in the following section.

3. Product Knowledge Modelling Scheme

3.1 Classification of Product Knowledge

Design is a complex, knowledge intensive process. Numerous models have been proposed which describe the design process with different emphases [1, 2, 6, 7, 8]. A product definition (solution) can be generated gradually through several stages; for example *Clarification of the Task, Conceptual Design, Embodiment Design* and *Detail Design* in [7]. Each of these stages could be decomposed further into a sequence of operations with specific objectives.

It is an information flow of the product being designed behind the process of its design. This information flow indicates that design is a gradual process of making design decisions in order to evolve design. These decisions correspond to different types of the knowledge of the product being designed from different viewpoints. Dynamic design solution is being constructed from these decisions and their relationships as a result of carrying out design. The knowledge of the product being designed is encapsulated in the dynamic design solution. The information of the product involved during its design can be classified into several classes, for example *Task, Function, Solution Principle, Solution Concept, Assembly* and *Production Documents* in [7]. Although the design process models may differ, their information needs are very similar [9].

Based on the analysis of design process, the *PDS, four kinds of required attributes*, which describe the Specific Product Knowledge from four viewpoints respectively, and *their relationships* are generalised as the necessary and sufficient elements of Specific Product Knowledge involved in product generation process from the viewpoint of function-oriented design. Thus they are modelled in our product knowledge modelling scheme. The four viewpoints are defined as follows.

- ***Function***: the qualitative representation of the intended purpose of a design, e.g. to reduce speed, to provide rotation, etc.
- ***Mode of action***: the way by which a complete or part product fulfils its function, namely the working principle of a complete or part product, e.g. transmission, relative movement, etc.
- ***Functional-carrier***: a complete or part product which can implement some functions is called a functional-carrier, e.g. lathe, gear-box, shaft-system, etc.
- ***Construction***: all the constituent parts, which have been fully defined by their characteristics (e.g. shape, dimension, material, surface quality, etc.), and the assembly relationships between these parts. That is the *Product Definition.*

As far as the product domain knowledge is concerned, seven categories of that and their relationships are considered as necessary and sufficient to support the function-oriented product generation process. They are modelled in our product knowledge modelling scheme. The seven categories of the product domain knowledge are,

- Four types of basic product domain knowledge. They are ***Function, Mode of Action, Functional-carrier*** and ***Part Knowledge***. The definitions of function, mode of action

and functional-carrier are the same as above. The emphasis here is on the knowledge in a product domain rather than the knowledge of a specific product. A physical component which exists as a whole object and could not be decomposed physically into sub-parts is called a part, e.g. shaft, gear, bearing, etc.

- Two types of product domain knowledge. They are needed to describe the above four types of basic product domain knowledge and build the links between them. They are,
  - ***Physical Link Knowledge***: Physical dependency between functional-carriers and/or parts is described by physical link knowledge, e.g. the physical link between a gear and a shaft could *be 'cylinder abut with one translational freedom'*, the physical link between a pair of gears is *'gear mesh'*, etc.
  - ***Numerical Knowledge***: Quantitative *characteristics*, the *relations* between characteristics and the *rules* used to handle the update of the values of characteristics and to manipulate the use of relations all together form numerical knowledge [10].
- Also, ***Product Family Knowledge***, i.e. the knowledge of product families and the Specific Product Knowledge of the existing products, is identified as another type of product domain knowledge.

3.2 Specific Product Knowledge Modelling

Based on the classification of specific product knowledge introduced above, ***one PDS unit*** and ***four viewpoint models*** are used to model corresponding specific product knowledge as shown in Figure 2.

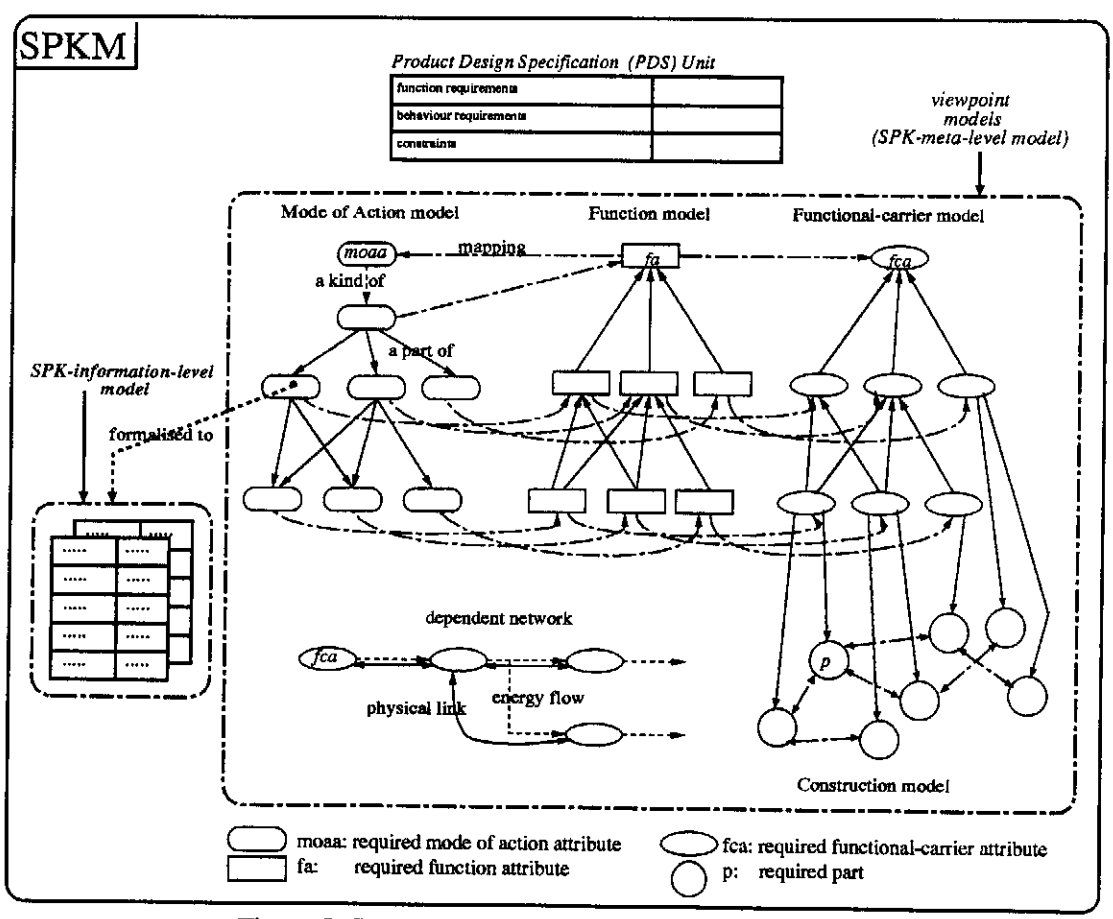

Figure 2: Specific Product Knowledge Model (SPKM)

- ***PDS unit***: All the requirements for a product to be designed form the PDS unit. It consists of *function requirements, behaviour requirements* and *constraints*. The contents of the PDS might change with the progress of design.
- ***Viewpoint model***: Each viewpoint model is a collection of one kind of required attribute. Required attributes are modelled at two levels, *a SPK-meta-level* and *a SPK-information-level*.
  - ***SPK-meta-level model***: It consists of *SPK-meta-concepts* and *SPK-meta-relations*.
    - ***SPK-meta-concept***: A SPK-meta-concept is a generalised, conceptual representation of a required attribute generated during product generation process as a decision, either temporary or permanent, made by a designer. It could be created as an instance of a piece of product domain knowledge or just as the conceptual representation of a piece of the designer's mental concept. For example, 'to reduce speed', as a required function attribute, is a SPK-meta-concept. A SPK-meta-concept is represented as a node in Figure 2.
    - ***SPK-meta-relation***: SPK-meta-relation refers to the relationship between SPK-meta-concepts. SPK-meta-relations, e.g. 'a part of', 'a kind of', certain kinds of physical links, etc., organise the attributes of the same viewpoint into structures, either a hierarchical structure and/or a network. SPK-meta-relations also build the links between the attributes of different viewpoints. SPK-meta-relations reflect the progress of design. For example, a SPK-meta-relation *'mapping'* exists between the required function attribute *'to reduce speed'* and the required functional-carrier attribute *'gear-box'*. It indicates that *'gear-box'* has been selected to fulfil the function requirement *'to reduce speed'*. A SPK-meta-relation is represented as an arrow line in Figure 2.
  - ***SPK-information-level model***: The SPK-information-level model of a SPK-meta-concept (a required attribute) consists of the formalization of the declarative and procedure knowledge of the SPK-meta-concept. *Declarative knowledge* of a meta-concept deals with the knowledge which represents the meta-concept and the relations between the meta-concept and the others. *Procedure knowledge* of a meta-concept describes the reactions to the operations happened to the meta-concept. Procedure knowledge of a meta-concept is formalised in terms of the declarative knowledge of the meta-concept, especially that part of declarative knowledge representing the meta-relations between this meta-concept and the others. It is the procedure knowledge that provides the base through which the consistency of the product knowledge models is maintained. The SPK-information-level model of a SPK-meta-concept changes with the type of the SPK-meta-concept.

3.3 Product Domain Knowledge Modelling

Figure 3 shows the Product Domain Knowledge Model (PDKM). Seven types of product domain knowledge is also modelled at two levels, *a PDK-meta-level* and *a PDK-information-level*.

- ***PDK-meta-level model***. It consists of *PDK-meta-concepts* and *PDK-meta-relations*.
  - ***PDK-meta-concept***: A generalised, conceptual representation of a piece of product domain knowledge is a PDK-meta-concept, which in our scheme is called a primitive. For example, *'to reduce speed'* is a function primitive, *'gear-box'* is a functional-carrier primitive. PDK-meta-concept (primitive) is represented as a node in Figure 3.

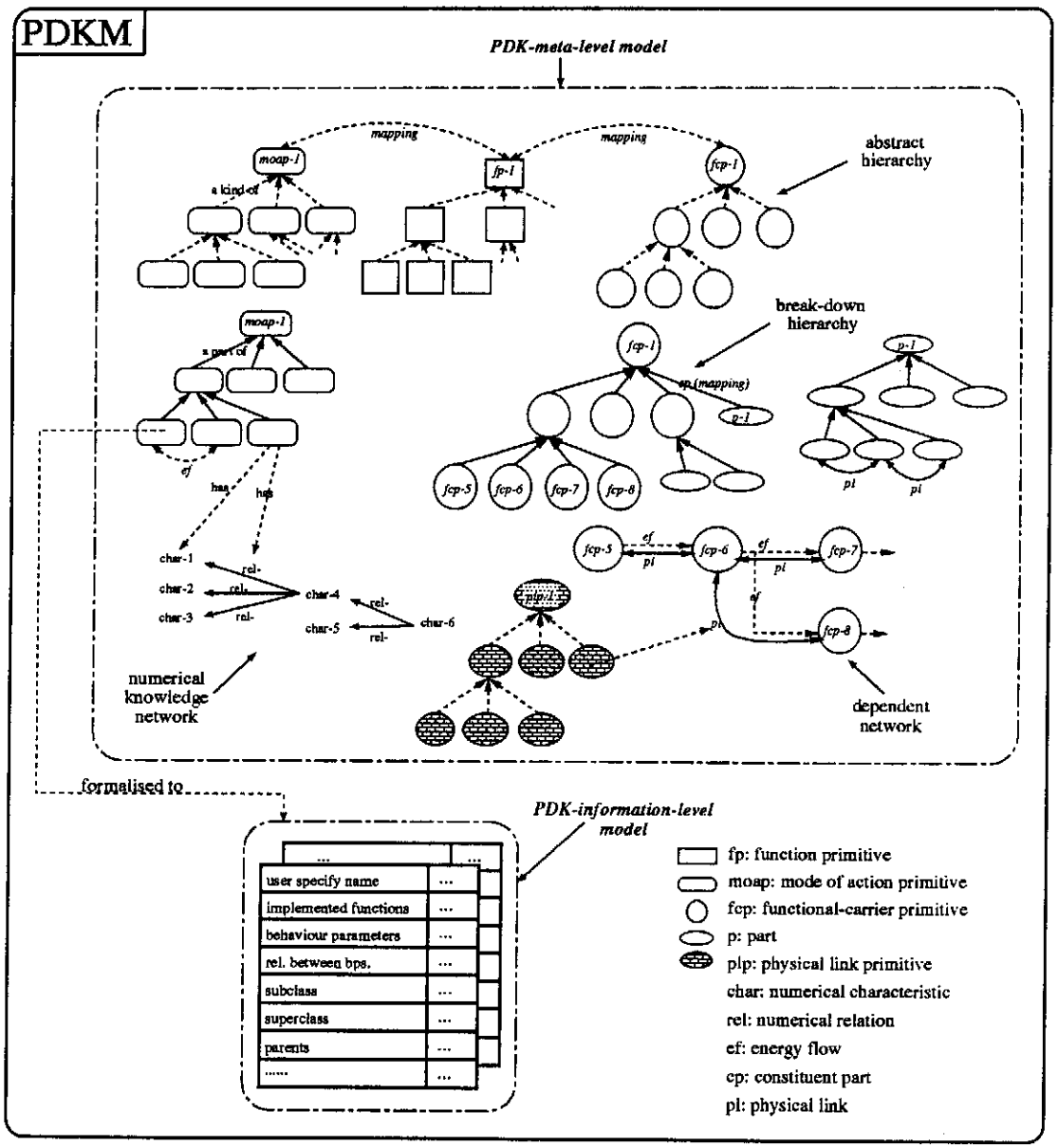

Figure 3: Product Domain Knowledge Model (PDKM)

- ***PDK-meta-relation***: PDK-meta-relation refers to the relationship between primitives. PDK-meta-relations organise the primitives of the same category into structures, e.g. an abstraction hierarchy, a break-down hierarchy, a dependent network, etc. PDK-meta-relations also link the primitives of different categorises. The mapping between primitives of different categories is facilitated through '*mapping*' PDK-meta-relations. For example, a PDK-meta-relation '*mapping*' exists between the function primitive '*to reduce speed*' and the functional-carrier primitive '*gear-box*'. It indicates that when a function '*to reduce speed*' is required, a functional-carrier '*gear-box*' could be an alternative solution to fulfil this required function. A PDK-meta-relation is represented as an arrow line in Figure 3. Note that some PDK-meta-relations themselves could also be modelled as PDK-meta-concepts, e.g. a physical link between parts is referred to as a PDK-meta-relation in regard of the whole product domain knowledge model, meanwhile it is also modelled as a PDK-meta-concept as far as this certain piece of knowledge is concerned.

- ***PDK-information-level model.*** The PDK-information-level model of a primitive is the formalization of the declarative and procedure knowledge of the primitive. The PDK-information-level model of a primitive changes with the type of the primitive.

4. Computer Supported Management of Product Knowledge Models

A computer supported management of product knowledge models is needed for the purpose of providing users with a modelling shell and building facilities to build and employ specific product knowledge model (i.e. design of a product) and the product domain knowledge model within a dynamically changing environment effectively and efficiently.

The Object-Oriented method has been adopted as the basis of our product knowledge management scheme. Issues considered in the management scheme, i.e. *representation* and *control* of the product knowledge models are presented as follows.

- ***Representation***: Frame (Unit-Slot-Facet) representation method is used to represent the declarative knowledge of the meta-concepts (the knowledge of the meta-concepts themselves and the relationships between them) based on the modelling scheme proposed in Section 3. A meta-concept (either a required attribute of a specific product or a primitive of a product domain) is represented as an unit.
  - ***General classes***: Units in product knowledge models are grouped into classes according to the classification of Specific Product Knowledge and Product Domain Knowledge in the proposed modelling scheme. Table 1 gives all the general classes. They are defined to represent different types of product knowledge respectively.

Class Name	Represented Knowledge Type
FA#	Function Knowledge of SPK
MOAA#	Mode of Action Knowledge of SPK
FCA#	Functional-carrier Knowledge of SPK
CA#	Construction Knowledge of SPK
FP#	Function Knowledge of PDK
MOAP#	Mode of Action Knowledge of PDK
FCP#	Functional-carrier Knowledge of PDK
PART#	Part Knowledge of PDK
PLP#	Physical Link Knowledge of PDK
PRODUCT#	Product Family Knowledge of PDK
CHAR#	Numerical Characteristic Knowledge of SPK and PDK
REL#	Numerical Relation Knowledge of SPK and PDK
RULE#	Numerical Rule Knowledge of SPK and PDK
GOAL#	Numerical Goal Knowledge of SPK and PDK
DEF#	Numerical Goal Definition Knowledge of SPK and PDK

Table 1: General Classes

  - ***Class definition***: Each class is defined by a set of generalised overall properties and slots of the units in this class in terms of the formalization of the declarative knowledge of the corresponding type of meta-concept. As an example, Table 2 gives a part of the FCP# class definition.

Property or Slot of Unit		Meaning	Value(s)	Example
Property	*super.class*	superclass of the Functional-carrier	list of functional-carrier primitives	shaft-system
	sub.class	subclass of the Functional-carrier	list of functional-carrier primitives	

Slot	*user.specify .name*	name of the Functional-carrier specified by user	any symbol	shaft-system-with-bevel-gear
	implemented.functions	functions of this Functional-carrier	list of function primitives	reduce-speed change-speed-direction
	parents	parents of the Functional-carrier	list of functional-carrier primitives	gear-box, etc.
	...	...	...	...

Table 2: Funcational-carrier Knowledge of PDK (FCP#) Class Definition

- ***Inheritance Mechanism***: An inheritance mechanism has been developed to ensure the correctness of abstraction hierarchies of PDKM and to allow the user to build the SPKM and PDKM in an effective and efficient way. The inheritance mechanism includes the control of the inheritance of the slot and that of the slot value.
  - ***Inheritance of the slots***: two types of slots; *Member Slot* and *Own Slot* are used to control the inheritance of the slot.
    - ***Member Slot***: used to represent the knowledge common to all the same type of meta-concept (represented as unit). A member slot of an unit is inherited to its subclass units as a member slot and to its instance units as an own slot. For example, user.specify.name slot, behaviour.parameters slot, etc.
    - ***Own Unit***: used to represent the information item which is owned by this meta-concept only. Own slot is not inherited to any of its subclass and instance units. For example, all slots of attribute units in a specific product knowledge model are own units because attributes are instances.
  - ***Inheritance of the slot value***: three inheritance roles are used to control the inheritance of the slot values of the superclass unit to its subclass and instance units. Table 3 gives the inheritance roles used in the management scheme.

Inheritance Role	Meaning	Example Slot
variable.values	the value of this slot is not inherited to the corresponding slots of the subclass and instance units regardless whether or not the corresponding slots of the subclass and instance units have their local values	user.specify.name slot, etc.
override.values	the value of this slot is inherited to the corresponding slots of the subclass and instance units until the corresponding slots of the subclass and instance units have their local values; once the corresponding slots have their own values, the slot value of superclass unit is no longer inherited anymore regardless the modification made to the local values, e.g. the local values have been deleted later	comment slot, parametric.drawing slot, etc.
union.values	the slot values of the subclass and instance units are union of the local value of the units and the slot value of their superclass unit	behaviour.parameters slot, physical.parameters slot, etc.

Table 3: Inheritance Roles Used in the management scheme

A primitive (a piece of product domain knowledge) is created either as a subclass of the corresponding general class only if the primitive to be created has no superclass or as a subclass of the same type of a/some primitive(s) if the primitive to be created has superclass. The slot values of the superclass are inherited to the new created primitive in terms of the inheritance roles of the slots.

An attribute (a piece of specific product knowledge) is created either as an instance of corresponding general class only or as an instance of both the corresponding general class and a primitive in the product domain knowledge model. In the former case, the attribute represents merely a mental concept of the designer, the knowledge of this attribute comes from the designer's mind; on the other hand, the later case indicates that the attribute to be created is an application of a certain piece of product domain knowledge (a primitive) in the current design case. Thus, the knowledge of this attribute comes from the product domain knowledge model initially. The slot values of the general class and (if any) the superclass primitive are inherited to the new created attribute in terms of the inheritance roles of the slots.

- ***Control of SPKM and PDKM***
  - ***Building facilities***: Based on the representation scheme introduced above, users are provided with some facilities to build their product knowledge models under the control of computer supported management system. These facilities can be grouped into *Add a meta-concept; Delete a meta-concept* and *Modify the knowledge of a meta-concept.*
  - ***Consistency Maintenance***: Meta-concepts are linked together by the meta-relations between them, such as p.part.of, a.kind.of, mapping, physical link relations, etc. Consequently, the change of some knowledge of a meta-concept might cause the change of the relevant knowledge of some meta-concepts; which can be the same meta-concept, other meta-concepts of the same type and/or other types. Such influence are formalised into the procedure knowledge of the meta-concept in the product knowledge modelling scheme. In order to provide a modelling shell and building facilities to allow users to build their SPKM and PDKM in a dynamically changing environment, run-time modification of SPKM and PDKM is indispensable. Therefore, maintaining the consistency of the SPKM and PDKM is a great concern regarding to the management of the SPKM and PDKM. Formalization of the product knowledge, especially the procedure knowledge, provides a base through which the consistency of the product knowledge models can be maintained. The concept of message passing in Object-Oriented method is used to maintain the consistency of the product knowledge models. Once a change of a certain slot value (knowledge) occurs, the computer supported management system will handle the propagation of the change to all the relevant slots of all relevant units in SPKM and PDKM automatically according to the meta-relations currently existing in the SPKM and PDKM.

The product knowledge modelling and management schemes proposed in Section 3.2, 3.3 and 4 are being implemented and tested in a computer-based Product Knowledge Modelling and Management System (PKM&MS).

5. Summary

In this paper, the term, *Product Knowledge* is used to refer to the two related but distinct concepts; *Specific Product Knowledge* (the knowledge of a specific product) and *Product Domain Knowledge* (the knowledge of a product domain).

A scheme is proposed to model, i.e. classify, formalise and structure both Specific Product Knowledge and Product Domain Knowledge for the purpose of supporting function-oriented product generation process. The PDS and four types (viewpoints) of required attributes (function, mode of action, functional-carrier and construction) are identified as the Specific

Product Knowledge involved in product generation process and modelled in One PDS unit and four viewpoint models respectively. Product Domain Knowledge is classified into seven categories; function, mode of action, functional-carrier, part, physical link, numerical and product family knowledge. The seven categories of product domain knowledge and their relationships are considered as necessary and sufficient to support function-oriented design. Thus they are modelled in the proposed modelling scheme. The product knowledge is modelled at two levels, a meta-level and an information-level. Meta-level model consists of meta-concepts and meta-relations. A meta-concept is a generalised conceptual representation of a piece of product knowledge (either a required attribute of a specific product or a primitive of a product domain). A meta-relation refers to the relationship between attributes or primitives. Meta-relations organise and link meta-concepts. The information-level model of a meta-concept is the formalization of the declarative and procedure knowledge of the meta-concept.

A scheme to manage, i.e. represent and control the proposed product knowledge models in the computer environment is presented in Section 4. Object-Oriented method is adopted to manage the product knowledge models. General class classification, class definition and inheritance mechanism form the core of the representation scheme. Building facilities and consistency maintenance mechanism are designed to control the product knowledge models within a dynamically changing environment.

It is believed that modelling and managing the product knowledge in such a way could assist designers in the product generation process from various aspects.

References

1. E. Tjalve, 1979, *"A Short Course in Industrial Design"*, Borough Green, Severoaks, keut, TN15, 8PH, ISBN 0-408-00388 X Newnes-Butter Worths
2. V. Hubka and W.E. Eder, 1988, *"Theory of Technical Systems"*, Berlin, Springer-Verlag
3. J.S. Gero and M.A. Roseman, 1990, *"A Conceptual Framework for Knowledge-Based Design Research at Sydney University's Design Computing Unit"*, Artificial Intelligence in Engineering, Vol. 5, No. 2
4. Y. Umeda, H. Takeda, T. Tomiyama and H. Yoshikawa, 1990, *"Function Behaviour, and Structure"*, Applications of Artificial Intelligence in Engineering V, Vol. 1 Design pp. 177-193
5. M.M. Andreasen, 1991, *"Design Methodology"*, Journal of Engineering Design, Vol. 2, No. 4
6. G. Pahl and W. Beitz, 1988, *"Engineering Design, a Systematic Approach"*, Springer-Verlag, The Design Council, London, ISBN 0-85072-239
7. S. Pugh, 1990, *"Total Design"*, Addisn-Wesley Publishing Company, ISBN 0-201-4635-5
8. T. Smithers, A. Conkie, J. Doheny, B. Logan, K. Millington and M.X. Tang, 1990, *"Design as Intelligent Behaviour: an AI in Design Research Programme"*, Artificial Intelligence in Engineering, Vol. 5, No. 2
9. M.R. Henderson and L.E. Taylor, 1993, *"A Meta-Model for Mechanical Products Based Upon the Mechanical Design Process"*, Research in Engineering Design, Vol. 5, pp. 140-160
10. A.H.B. Duffy and K.J. MacCallum, 1989, *"Computer Representation of Numerical Expertise for Preliminary Ship Design"*, Marine Technology, Vol. 26, No. 4, pp. 289-302, October, 1989