PERGAMON

Robotics and Computer Integrated Manufacturing 15 (1999) 341–352

Robotics
and
Computer Integrated
Manufacturing

www.elsevier.com/locate/rcim

A science-based approach to product design theory Part II: formulation of design requirements and products

Y. Zeng, P. Gu*

Department of Mechanical and Manufacturing Engineering, The University of Calgary, 2500 University Drive, Calgary, Canada, AB T2N 2N4

Abstract

The Design Process begins with design requirements and ends with product descriptions. The design requirements include both structural and performance aspects. The product descriptions deal with the structural aspect of the design requirements while the product performances describe the performance aspect of design requirements. In this part of the paper, a set theory-based representation scheme is proposed to represent design objects in the design process, including design requirements, product descriptions, and product performances. This representation scheme can represent the design objects that evolve in dynamic design processes. The entire mathematical scheme is defined based on structural and behavioral properties. Within one uniform scheme, the design objects are represented at different levels of complexity and abstraction. Several examples are included to explain the scheme and its mathematical formulations. The proposed scheme can be used for science-based studies of product design.

Keywords: Design requirement; Design description; Product performance; Complexity; Abstraction; Set theory

1. Introduction

Design is a basic human activity. During a long period, *design* had been considered as an 'art' which was taught through a 'master–student' model. Only after 1960s, intensive studies into the design activity began as many researchers attempted to propose better design methods to improve design and design education. Consequently, the research has spanned a broad range of fields from philosophy, psychology, and engineering to computer applications. The 'art' of design has been gradually replaced by the 'art and science' of design [1]. The science aspect allows people to better understand design processes while the art aspect allows designers to keep their creativity in rationalized design processes resulting from the science aspect. However, only a few systematic design theories have been established and the research in this field is still in pre-theory stage [2], although there have been a rich and varied body of knowledge including theories, methodologies and applications. The design discipline has reached the point in its evolution where a science-based design theory should be established. The motivations for this kind of research are many. One of the most important comes from the development of computer-aided design systems. Better design support systems can only come after better design theories are developed. The success of AutoCAD, Pro-Engineer, and other CAD systems have been largely dependent on the development of fundamental geometric modeling theories [3]. The development of more advanced computer-aided systems calls for better product and design process models. But design studies have been influenced by engineering, computer science (in particular, software engineering and artificial intelligence), information processing theory, cognitive science, psychology, and philosophy among others. The representation schemes of designs and design processes bear the characteristics of different fields. There are some confusions and vagueness in the representation and communication of ideas. The science-based design theories will certainly improve the development.

The main aim of scientific design theory is to discover and disclose the underlying order of design processes. Like other scientific disciplines, design theory also needs to be completed with the formulation of laws by means of an adequate and accurate language. The laws would set the limits for the theoretical level of design and define the science of design. The examples

* Corresponding author.

0736-5845/99/$ - see front matter
PII: S0736-5845(99)00029-0

of law are the two axioms proposed by Suh [4]. Gomez-Senent et al. [5] have also listed nine principles of design. With to the language, possible forms include graphics, flow charts, natural languages and mathematics. Theoretically, mathematics is the most accurate and powerful language for scientific purposes. Because design studies have been focused on design methodologies, design philosophy, and knowledge-based design, natural and graphic languages are the major representation tools in the research [6]. The purpose of this part of the paper is to propose a mathematical framework to represent the entities involved in design processes. In the next section, the basic framework of the mathematical representation of design problems is proposed to govern the following discussions. Then design specifications, product descriptions, and product performance are investigated in separate sections. Conclusions are given in the final section.

2. Framework of mathematical representations

Design is an intelligent activity that begins with design requirements and ends with a product description as is shown in Fig. 1. In a typical design process, design requirements are represented by design specifications. Based on the specifications, candidate design descriptions are generated. The product description must be evaluated against the prescribed design requirements to determine if the designed product satisfies the requirements. The process iteratively generates conceptual, configuration, and detailed designs. The design requirements can be motives or demands for a completely new product, the complaints on the performance of existing products, or the failure due to malfunctions of existing products. Direct evaluation against design requirements is usually not possible because they are usually given by vague customers, requirements. Thus, the first step in engineering design is design requirement formulation, which translates design requirements into design specifications. The design process then provides a mapping from design specifications to design descriptions. Therefore, a comprehensive design theory should include at least three basic parts:

1. A general framework to describe and formulate design problem.
2. A language to define the two ends of a design process.
3. A theory to address the processes of formulation and design.

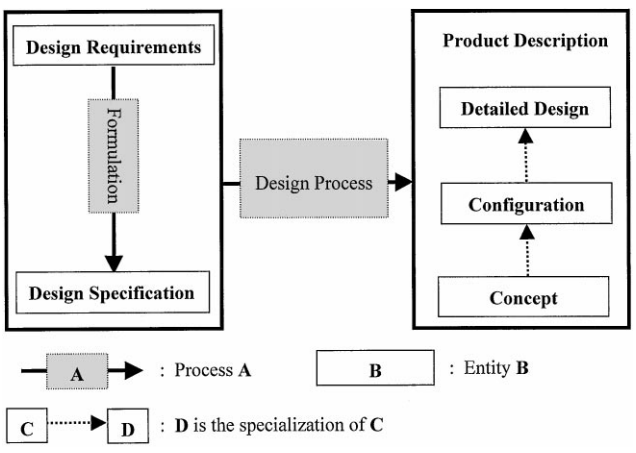

Fig. 1. Design activity.

In this paper, our focus is on the second part: the development of a language for design. This lays a foundation for a scientific formulation of designs and design processes. The other two questions are addressed in the Part I of this paper [7].

The development of a language for design should start by finding a strict definition and a robust representation of the terms involved in design processes without going to the details of concrete design tasks [8]. It can be easily seen from Fig. 1 that the terms include design requirements, design specifications, and product descriptions. Moreover, since design is a dynamic process from abstract, simple and conceptual to specific, complex and detailed representations, design descriptions evolve and change through a design process. The representation schemes should be able to support the changing descriptions along the continuously evolving design processes. Research has been conducted on formulating the entities involved in design process, such as Yoshikawa [9], Salustri and Venter [10], Cheng and Zeng [11], and Maimon and Braha [12].

Most of the existing work focused on product data models to support the development of product information systems and CAD systems. According to our present studies, the problem is that the dynamic nature of design process has not been captured. Consequently, the design process cannot be reasonably modeled. The present work is fundamentally different from existing design object modeling [11,12]. The mathematical framework proposed in this paper attempts to embody the entities in design processes at different levels of abstraction and complexity. Therefore, it can be used in the entire design process.

Design specifications manifest themselves as a set of desired product properties which represent the geometrical, physical, economic, and other design-related properties of the product. Denoting design specifications and product properties as R^d and E, respectively, we have

$$R^d = \{r_j^d : r_j^d = \lambda(e^i, [e^i]),\ e^i \in E,\ i = 1, 2, \ldots, j = 1, 2, \ldots, n_r\} \quad (1)$$

where $[e^i]$ is a constraint on product property e^i. The symbol n_r is the number of listed design requirements.

According to the definition, a design specification can be viewed as a predicate $\lambda(e^i, [e^i])$.

Product descriptions, denoted by S, are the representation of design solutions. Design solutions are usually described by concepts, configurations, or product drawings, depending on stages of a design process. They are usually defined by a set of product properties

$$S = \{e^i : i = 1, 2, \ldots, n_s, e^i \in E\}, \tag{2}$$

where n_s is the number of properties with which a product can be completely defined. For example, a rectangle can be sufficiently defined by its shape, length, and width where n_s is three. We can also define it by four vertex points where n_s is 4. Any more information in both cases would be redundant.

A mechanical design process is generally divided into conceptual, configuration, and detailed design phases. The main objective of conceptual design is to develop concepts to meet design specifications. Configuration design refines design concepts to concrete product architectures and components. Key design parameters for critical design features are also determined at this stage. Detailed design determines all detailed parameters including dimensions, tolerances and other design parameters of all components where a product is described by engineering drawings or geometric models. Three types of product descriptions are involved as shown in Fig. 1, which are concept, configuration, and detailed design. In fact, there are no explicit boundaries between design stages. Design iterations occur throughout entire design processes. Every earlier design process will generate some new design requirements or will refine the original design requirements to redefine the design problem. A designer may begin a design task at any stage with design requirements and product descriptions. Hence, a design representation scheme must be able to support the entire dynamic design process in an integral, unified, and continuous form, as is shown in Fig. 2.

In the following sections, these entities under the framework will be discussed in more detail, beginning from design specifications. The elements involved in defining design specifications and product descriptions are then specified.

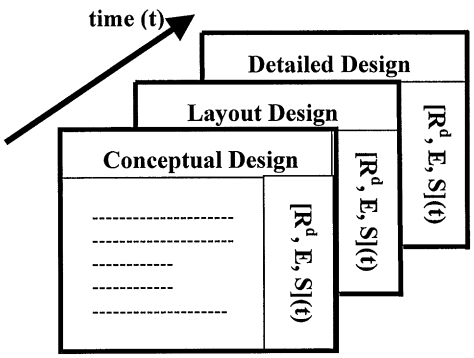

Fig. 2. A framework of design representation.

3. Design specifications

3.1. Design requirements

From the product life cycle point of view, any product design must take into account a number of requirements regarding functionality, safety, manufacturability, assembly, testing, shipping, distribution, operation, services, re-manufacturing, recycling and disposal [15]. These requirements can be investigated by viewing a product as an object in its working environment, as is shown in Fig. 3. A product responds to an action from its working environment in a way depending on the laws of the environment. The action–response mode of a product is called the performance of the product. A product may have many kinds of performances. For example, the change of cost of a product with the change of raw materials is a market performance of the product; the stress distribution of a product under environmental forces is a mechanical performance of the product. Obviously, the aforementioned requirements are constraints imposed on either the structure or the performance of a product (see Fig. 4). The examples of structural requirements are constraints on dimensions, shapes, configurations, and materials whereas the examples of performance requirements are safety, functionality, manufacturability and so forth.

3.2. Design specifications

Denoting design specifications for structural and performance requirements as R^s and R^p, respectively, we can further define design specifications R^d as

$$R^d = R^s \cup R^p. \tag{3}$$

According to Eq. (1), two sets of product properties are necessary to facilitate the definition of design requirements, which are the sets for product description and product performance. They are called the structural property set E^s and the behavioral property set E^b that are responsible for representing product description and

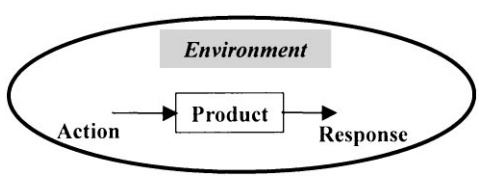

Fig. 3. Product environment.

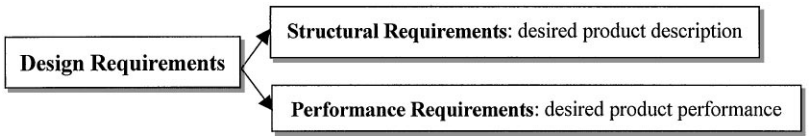

Fig. 4. Classification of design requirements.

Table 1
Properties in mechanical design

Name	Type	Value	Examples
Direction	Structural	$\{x, y, z\}$	⟨direction, x⟩
Coordinate	Structural	$R \times R \times R$	⟨coordinate, ⟨21.0, 10.0, 30.0⟩⟩
Length	Structural	R	⟨length, 50.0⟩
Shape	Structural	String	⟨shape, circle⟩
Density	Structural	R	⟨density, 15.0⟩
Material_name	Structural	String	⟨material_type, steel⟩
Viscosity	Structural	R	⟨viscosity, 0.6⟩
Force_name	Behavioral	String	⟨force_name, torque⟩
Force_magnitude	Behavioral	R	⟨force_magnitude, 500.0⟩
Deflection	Behavioral	R	⟨deflection, 5.0⟩
Velocity	Behavioral	R	⟨velocity, 50.0⟩

product performance, respectively. They constitute a partition of the product property set E as follows:

$$E = E^{s} \cup E^{b}. \tag{4}$$

3.3. Properties

A product property is an observable, measurable or otherwise known characteristic related to the product [10]. It is defined by its name e_n and its value e_v. If the sets of property name and property value are denoted as D and R, respectively, then property set, E, can be represented as

$$E \subseteq D \times R;\ e^i = \langle e^i_n, e^i_v \rangle,\ e^i \in E,\ e^i_n \in D,\ e^i_v \in R. \tag{5}$$

D and R can be taken as the domain and the range of a property, respectively.

All the properties in a design field constitute a property pool. Table 1 gives some examples of structural and behavioral properties widely used in mechanical design. The structural and behavioral properties of a product constitute a state of the product.

By observing Eq. (1), we can say that a design specification is actually a set of properties by itself. A specifications value can be either a single value or multi (including infinite) values in a range R. A specification can then be seen as the generalization of a constrained property.

In fact, the property is not only the foundation of representing design requirements but also the base of product descriptions, which is to be shown in the succeeding sections. It is much broader than point set in topology, which lays the foundation of solid modeling.

Obviously, more complete definitions of design specifications, product descriptions and product performances should be further formalized according to the framework given in Fig. 2. The following sections will provide detailed discussions.

4. Product description

In this section, our focus will be placed on the mathematical representation of product descriptions according to the framework given in Fig. 2. As product descriptions include all the results generated in the dynamic design process, the representation scheme must imply different levels of product descriptions. On the other hand, in a real world design, the number of potential products is infinite. It is essential to develop a finite means to handle the infiniteness existing in the problem. As a result, the developed representation scheme should be able to handle the infiniteness, the complexity, and the abstraction related to product descriptions. A natural way to handle infiniteness in scientific research is to define a set of basic elements and combination rules. By combining the basic elements, any object can be constructed. Correspondingly, the discussion in this section will include three parts: primitive products, complex products, and levels of complexity and abstraction. The first two parts deal with the infiniteness problem. The third part shows how this approach embodies the levels of complexity and abstraction naturally.

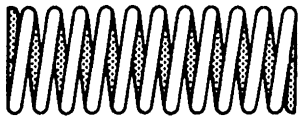

Fig. 5. A compression coil spring SP1 [18].

4.1. Primitive products

Since any product can be decomposed into sub-assembly components and their relations, it is reasonable to assume the availability of a set of primitive components and connectors [16]. They are the basic product elements whose performance can be obtained without turning to other product elements. In mechanical design, we have the primitive components such as gears, shafts, bearings, and primitive connectors such as attachment (physical contacts, fixations and stops, joints, fasteners, couplings, welds and the like), positioning (relative distance or angle between components, alignment including coaxial, collinear, parallel, perpendicular and flush alignments), motion (cam-controlled objects, trajectory of joints and end-effects, etc) and containment (e.g., components contained within the same housing) [17]. From the representation point of view, they do not have any difference. For the sake of simplicity, both primitive components and primitive connectors are named after primitive product denoted by S_a^i. Primitive product is defined by a set of structural properties E^s as

$$\begin{aligned} S_a^i = \{ & e_s^j \mid \forall j, 1 \leqslant j \leqslant n, e_s^j \in E^s, \forall (e_s^j)_n \in D, \\ & \exists (e_s^j)_v \in R, e_s^j = \langle (e_s^j)_n, (e_s^j)_v \rangle \}, \\ S_a = \{ & S_a^i \mid i = 1, 2, 3, \ldots \}. \end{aligned} \tag{6}$$

By taking the straight-sided helical compression spring shown in Fig. 5 as an example, a formal description and an informal explanation of the spring are given in Table 2.

Fig. 6 gives a class of primitive springs in mechanical design. They are represented as

$$\begin{aligned} \text{primitive\_springs} = \{ & \text{compression\_coil\_spring, extension\_coil\_spring, torsion\_bar, torsion\_coil\_spring,} \\ & \text{flat\_spring, volute\_spring, flat\_spiral\_spring, belleville\_spring, leaf\_spring} \} \end{aligned} \tag{7}$$

4.2. Complex products

Any product can be described as a set of components related to each other through component connectors:

$$S = S^c \cup S^v \tag{8}$$

where S^c and S^v are the sets of all the components included in a product and the component connectors of these components, respectively.

$$\begin{aligned} S^c &= \{ S_i \mid i = 1, 2, \ldots, n_c \}, \\ S^v &= \{ T_i \mid i = 1, 2, \ldots, n_t \} \end{aligned} \tag{9}$$

where n_c and n_t are the numbers of components and component connectors of the product, respectively. Therefore, a product can be further described as

$$\begin{aligned} S = \{ & X_i \mid \forall i, 1 \leqslant i \leqslant n_c, X_i = S_i; \\ & \forall i, n_c < i \leqslant n_c + n_t, X_i = T_i \}. \end{aligned} \tag{10}$$

The product description given in Eq. (10) is based on the definition of 'components' which are products by themselves. This renders the definition recursive and brings in a hierarchical structure of the product description as is shown in Fig. 7. By defining $S(k, i_k, j_{k-1})$ as the node at the i_kth position in the kth layer with a parent node at the $j_{(k-1)}$th position in the $(k-1)$th layer, the product can be described recursively as

$$\begin{aligned} & S(0, 0, 0) = \{ S(1, i_1, 0) \mid i_1 = 1, 2, \ldots, n_1 \}, \\ & \forall i_k 1 \leqslant i_k \leqslant n_k S(k, i_k, j_{k-1}) \\ & \qquad = \{ S(k+1, i_{k+1}, i_k) \mid i_{k+1} \\ & \qquad = m_0 + 1, m_0 + 2, \ldots, m_n \} \\ & \forall i_k, S(n, i_k, \_) \in S_a, \\ & n_0 = 1, n_k = \sum_{i_k=1}^{n_{k-1}} n(k+1, i_k), \\ & \qquad m_0 = \sum_{i=1}^{i_k - 1} n(k+1, i), m_n = m_0 + n(k+1, i_k), \end{aligned} \tag{11}$$

where $S(0, 0, 0)$ is the root node, $n(k+1, i_k)$ is the number of child nodes of the i_kth node in the kth layer of the tree and n_k is the total number of nodes in the kth layer of the tree. A typical block of the tree is shown in Fig. 8.

Indeed, Eq. (11) has provided a formal approach to representing any product based on the primitive product set given in Eq. (6). The objective of representing infinite number of products by a finite means has thus been realized.

By substituting Eq. (11) into Eq. (10), we get

$$\begin{aligned} & S[0] = \{ S(0, 0, 0) \} \\ & \forall k 1 \leqslant k \leqslant n S[k] = \{ S(k, i_k, \_) \mid i_k = 1, 2, \ldots, n_k \} \\ & \qquad \forall i_n S(n, i_n, \_) \in S_a. \end{aligned} \tag{12}$$

Table 2
Description of spring

Informal description		
Property name	Property value	Formal description
Outside diameter ($d0$)	$V_1 = 25.0$ mm	$\langle d0, 25.0\rangle$, $V_1 \in \{7.0, 8.0, \ldots, 19.0, 20.0, 22.0, 25.0, 28.0, \ldots, 60.0, 65.0\}$
Wire diameter (d)	$V_2 = 2.0$ mm	$\langle d, 2.0\rangle$, $V_2 \in \{1.0, 1.25, 1.4, 1.6, 1.8, 2.0, 2.25, 2.5, 2.8, 3.2, \ldots, 10.0\}$
Inside diameter ($d1$)	$V_3 = 21.0$ mm	$\langle d1, 21.0\rangle$, $V_3 \in \{x \mid x = D0 - 2d\}$
Free length (L_0)	$V_4 = 234.0$ mm	$\langle L_0, 234.0\rangle$, $V_4 \in R$ (the set of real numbers)
Active coils (N_A)	$V_5 = 11$	$\langle N_A, 11\rangle$, $V_5 \in Z^+$ (the set of integer numbers)
Inactive coils (N_I)	$V_6 = 2$	$\langle N_I, 2\rangle$, $V_6 \in Z^+$ (the set of integer numbers)
Pitch (p)	$V_7 = 20.0$ mm	$\langle p, 20.0\rangle$, $V_7 \in R$ (the set of real numbers)
Material (m)	$V_8 =$ carbon steel	$\langle m, \text{carbon\_steel}\rangle$, $V_8 \in \{\text{carbon steel, alloy, stainless steel, } \ldots\}$
SP1 = $\{\langle d0, 25.0\rangle, \langle d, 2.0\rangle, \langle d1, 21.0\rangle, \langle L_0, 234.0\rangle, \langle N_A, 11\rangle, \langle N_I, 11\rangle, \langle p, 20.0\rangle\}$		

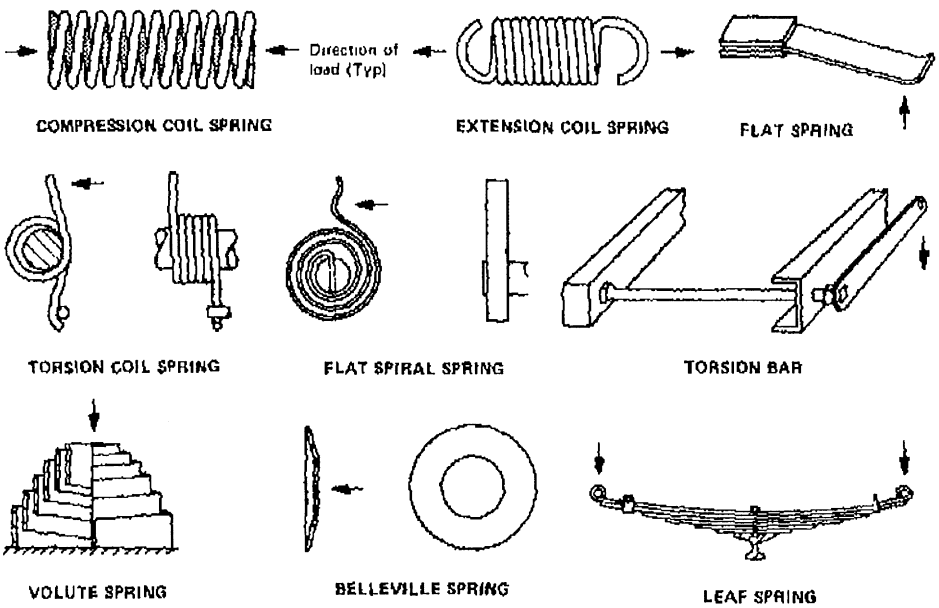

Fig. 6. Types of mechanical springs [16].

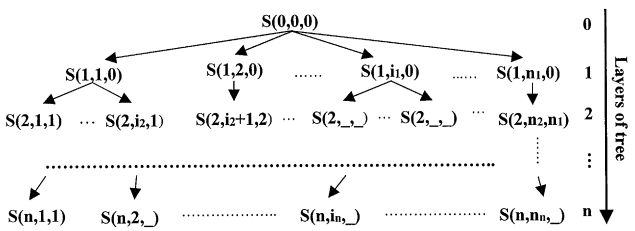

Fig. 7. Tree structure of product description.

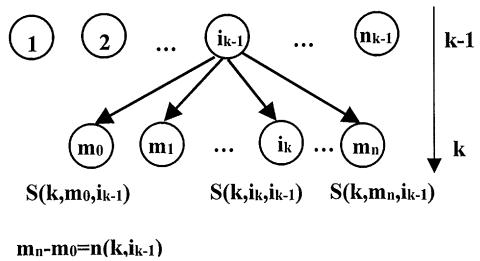

Fig. 8. Basic block of product description tree.

which is actually Eq. (10) refined by considering different layers in the hierarchical structure of product description. In fact, Eqs. (6), (11) and (12) constitute a mathematical representation of product descriptions. It will be seen in the later discussion that this representation implies levels of complexity and abstraction of product description.

A rocker arm assembly example shown in Fig. 9 can be used to explain the above notions. It consists of the following components: RM_1 = valve; RM_2 = spring; RM_3 = rocker arm; RM_4 = tappet; RM_5 = follower assembly; RM_6 = cam assembly.

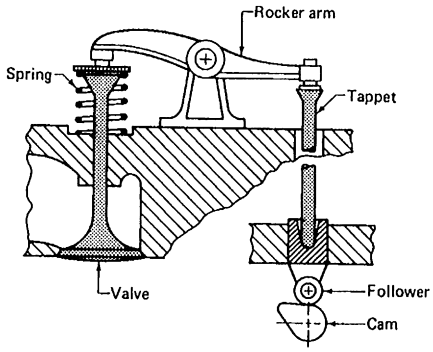

Fig. 9. A rocker arm assembly [18].

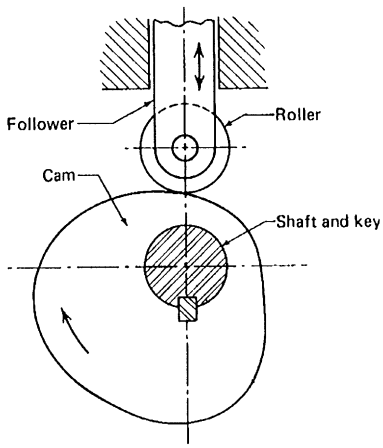

Fig. 10. Cam assembly [18].

The rocker arm can then be represented according to Eq. (12) as

$$\mathrm{RM} = \{\mathrm{RM}_i \mid i = 1, 2, \ldots, 6\}, \tag{13}$$

where every component can be further defined by other components until the component becomes a primitive product.

In the description given in Eq. (13), RM_2 is not necessarily to be refined since spring is one of the primitive products. In contrast, however, RM_6 must be further described since it is not primitive by itself. It is shown in Fig. 10, which is composed of: $\mathrm{CAM}_1 = \mathrm{Cam\ body}$, $\mathrm{CAM}_2 = \mathrm{Cam\ shaft}$, $\mathrm{CAM}_3 = \mathrm{Cam\ key}$.

Accordingly, the cam can be formally described as

$$\mathrm{CAM} = \{\mathrm{CAM}_i \mid i = 1, 2, 3\}. \tag{14}$$

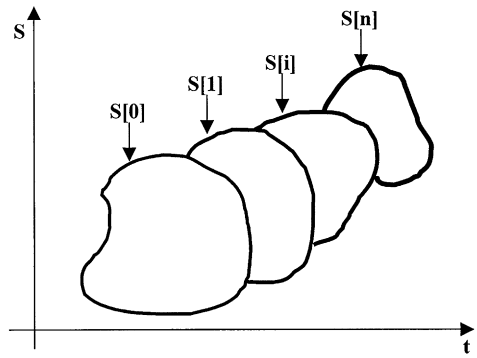

Fig. 12. Product descriptions in evolving design process.

In this way, we have a tree structure of the rocker arm as is shown in Figs. 11 and 12. The cam body, shaft, key and so on can be defined with a set of properties, respectively.

4.3. Levels of complexity and abstraction

As is implied in Fig. 2, product descriptions should evolve with design process. So it is essential to verify that if Eq. (12) satisfies the condition. Indeed, in Eq. (12), each $S[k]$ gives a representation of product with respect to the components of different levels of complexity. As the value of index k increases, product description $S[k]$ will become more detailed. This means that index k represents the degree to which the definition of product description is detailed. Therefore, index k is called the complexity level of product description. $S[k]$ is the kth-order product description. In the tree structure of product description in Fig. 7, if a component cannot be further decomposed, the number of its successor will become one. This component is thus a primitive product. If all the leaf nodes in Fig. 7 become primitive products, then the product is said to be well defined. Otherwise, the product description is partially defined, which is indeed an abstraction of a well-defined product description. Correspondingly, we have the notion of abstraction levels for product descriptions. The levels of complexity and abstraction for product descriptions are shown in Table 3. Obviously, lower order product description is the abstraction and the type of a higher order ones. This fact matches with a basic logical principle perfectly: the

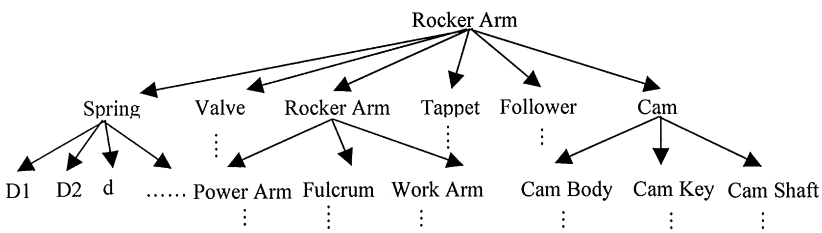

Fig. 11. Tree structure of rocker arm.

Table 3
Levels of complexity and abstraction of product description

Product definition		Complexity level	Abstraction level
$S[0]$	M	0	n
$S[1]$	M	1	$n-1$
$S[n]$	M	n	0

more a concept's intention is, the narrower the concept's extension has.

A design process usually begins with a partially defined product description. In extreme cases, it begins from $S[0]$. The final design is obtained by expanding the leaf nodes into components with greater level of complexity. The process is shown in Fig. 11. This representation scheme provides a top-down approach of supporting the dynamic design process from generic and simple to concrete and complex product descriptions, as represented by Fig. 2.

Based on the notion of complexity and abstraction given above, the abstraction of primitive products becomes primitive product type. It is the product description after the product properties defining the product assume values in broader ranges. Denoting the set of primitive product type as S_{at}, we have

$$\begin{aligned} S_{\mathrm{at}}^{i} &= \{S_{\mathrm{a}}^{k} \mid k = 1,2,\dots\} \\ &= \{e_{\mathrm{s}}^{j} \mid \forall j, 1 \leqslant j \leqslant n, \forall (e_{\mathrm{s}}^{j})_{\mathrm{n}} \in D, \\ &\qquad \forall (e_{\mathrm{s}}^{j})_{\mathrm{v}} \in R, e_{\mathrm{s}}^{j} = \langle (e_{\mathrm{s}}^{j})_{\mathrm{n}}, (e_{\mathrm{s}}^{j})_{\mathrm{v}} \rangle, e_{\mathrm{s}}^{j} \in E^{s}\}, \\ S_{\mathrm{at}} &= \{S_{\mathrm{at}}^{i} \mid i = 1, 2, \dots, m\}. \end{aligned} \tag{15}$$

The type "compression coil spring" can then be defined as

$$\text{Compression\_coil\_spring} = \{\langle d0, V_1\rangle, \langle d, V_2\rangle, \langle d1, V_3\rangle, \langle L_0, V_4\rangle, \langle N_A, V_5\rangle, \langle N_I, V_6\rangle, \langle p, V_7\rangle, \langle m, V_8\rangle\}. \tag{16}$$

The difference between primitive product and primitive product type lies in that the property value of product is defined as a narrower range compared with that of product type. The extreme case is that each property just takes exactly one value in the description of product whereas each property may assume a value varied in a prescribed range.

Eqs. (6), (11), and (12) together constitute a complete definition of a product description. The primitive product in Eq. (6) defines the lowest level of abstraction and the highest level of complexity of a product description. In fact, the choice and the definition of primitive products vary with design domains and designers' expertise and preferences. Needless to say, different design domains have different primitive products. Meanwhile, experienced designers will have more complex primitive products in mind compared with naive ones.

It should be noted here that the tree structure of product descriptions in this paper is different from traditional representations in product modeling. The first and the most essential difference lies in that the base of the present representation is the property set which might be point set (geometrical information) or a concept set (feature information, physical information and any other linguistic information). But the point set in topology is the base of CSG solid modeling. Another profound difference is that the traditional product modeling can only support the representation of product information after the product has been designed whereas the present approach represents a product along the whole dynamic design process from design concept to final detailed geometry and dimensions.

5. Product performance

To complete the definition of design requirements given in Eq. (3), this section discusses product performance. The discussion is going to refine the action–response pattern of a product given in Fig. 3 by taking into account the product descriptions from the last section. Therefore, the discussion will involve two aspects: complexity and abstraction. From the complexity point of view, the performance of a product is realized by its constituent components. From the abstraction point of view, performances may experience a process from abstract to concrete with the progression of design.

5.1. Performance

As is shown in Fig. 3, product performances are responses of a product to imposed external actions according to the laws in product's working environment. Moreover, according to Eq. (12) a product may consist of a set of components. Thus, the action–response performance pattern is realized by mutual interactions among the constituent components which form a performance network of components shown in Fig. 13. In a product assembly, any component of the assembly may be subject to the actions from components that connect to it. As a result, the state of the component, which has been

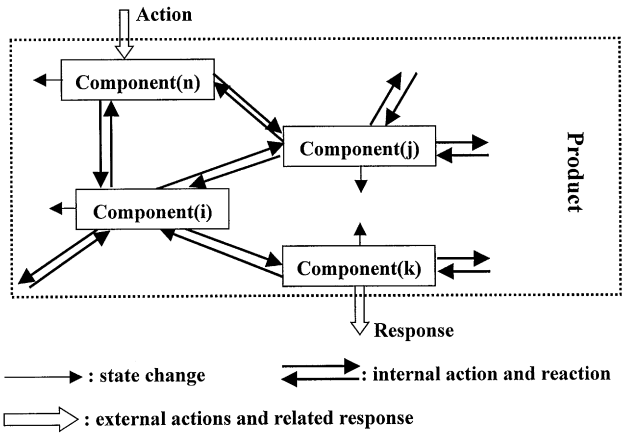

Fig. 13. Performance network of the components in a product.

defined as structural and behavioral properties of a product, may be changed to resist the actions, and the component may also act on and/or react to the connected components because of its state change. In this way, the global performance of a product is accomplished by the performance of its individual components.

An example is given in Fig. 14, which represents the performance of the cam assembly in Fig. 10. It will be formally described later in this section.

Since the structure of influence network shown in Fig. 13 depends upon how a product is decomposed into sub-assemblies, a basic block of the performance network is necessarily to be disclosed to facilitate the mathematical representation of the network. "component(i)" is a typical constituent in the influence network in Fig. 13. It is given in Fig. 15 by separating the group of actions on and response from the component(i). The expression "action(k)" is a representative of actions on component(i) with the expression "reaction(k)" as the corresponding response. The expression "action(j)" is a representative of component(i)'s actions on its connected components with the expression "reaction(j)" as the corresponding response from the connected components which in turn acts on component(i) from its connected components due to its actions on those connected components. The expression "state change(i)" is a representative of the state change of component(i) due to all the actions on and reactions to it.

In Fig. 15, action(k) and reaction(j) together constitute the actions imposed on component(i), whereas reaction(k), action(j), and state change(i) become the responses from component(i). The figure indeed shows the performance pattern of components included in a product. It is through its components' interactions that a product exhibits global performance. Because both actions and reactions are same in nature, for the sake of simplicity, from now on in this paper, they are uniformly called actions. Hence the performance scheme in Fig. 15 can be further represented in Fig. 16.

Therefore, the action and response in Fig. 3 become action(j), action(k) and state change(i), respectively. They are physical entities with structural carriers, which can be formally represented as

$$a \in A \subset SYP(E^{b})$$
$$r \in R \subset SYP(E^{b}) \tag{17}$$

where a, A are the actions and action set; r, R the response and response set; $P(X)$ the power set of set X; and S: product description.

For the cam assembly example in Fig. 10, with reference to the cam, the rotation of shaft and key is an action on the cam while the rotation of the cam itself is a response of the cam to the action. If the rotational velocity of the shaft is ω_s, then the rotational velocity of the cam should also be ω_s. According to Eq. (17), the action and the response related to the cam can be described as

Action on the cam:

$$\text{rotation\_of\_shaft\_and\_key} = \{\text{shaft\_\&\_key\_description, cam\_description, contact\_point,} \langle \text{rotation\_rate}, \omega_s \rangle, \langle \text{rotation\_direction, clock\_wise} \rangle\}.$$

Response of the cam:

$$\text{rotation\_of\_cam} = \{\text{cam\_description}, \langle \text{rotation\_rate}, \omega_s \rangle, \langle \text{rotation\_direction, clock\_wise} \rangle\}. \tag{18}$$

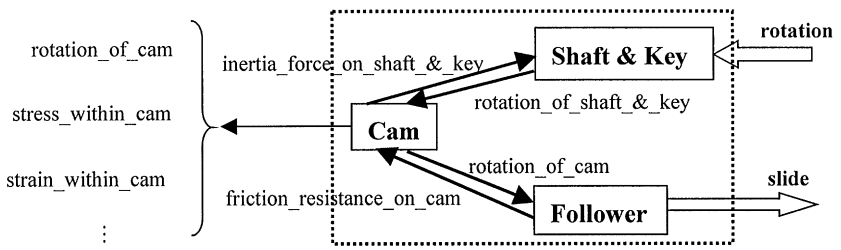

Fig. 14. Performance network of cam assembly.

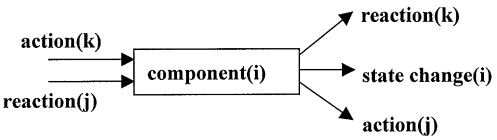

Fig. 15. Typical constituent of performance network of components.

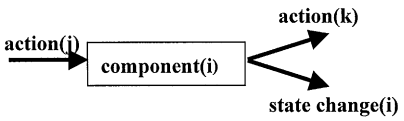

Fig. 16. Component performance.

Moreover, if we consider the follower, then the rotation of cam is an action on the follower and the slide of follower becomes a response.

Action on the follower:

$$\text{rotation\_of\_cam} = \{\text{cam\_description, follower\_description, contact\_point, } \langle \text{rotation\_rate}, \omega_s \rangle, \langle \text{rotation\_direction, clock\_wise} \rangle\}$$

Response of the follower: (19)

$$\text{slide\_of\_follower} = \{\text{follower\_description}, \langle \text{slide\_direction, vertical} \rangle, \langle \text{slide\_velocity}, v \rangle\}$$

Eq. (17) means that action and response can be described as a set of product properties. The product description provides structural information related to actions and responses and the behavioral properties provide physical information related to actions and responses. Then a performance, denoted by p, can be defined as a tuple of action a and response r,

$$p = \langle a, r \rangle. \tag{20}$$

A performance of the follower in Fig. 10 is an example, which is represented as

$$p = \langle \text{rotation\_of\_cam, slide\_of\_follower} \rangle \tag{21}$$

It indicates that the follower will slide vertically with velocity v following the rotation of cam with rotational rate ω_s.

Naturally, product performances, as products responses to external actions, can be mathematically defined as a relation from action set A to response set R. Denoting product performances as P, we have

$$P \subset A \times R, \tag{22}$$

where P is the set of element p.

5.2. Performance in different levels of complexity and abstraction

Since the definitions of action and response rely on product descriptions in the terms defined in Eq. (17), the performance in Eq. (22) should also be defined based on the product descriptions. That is to say, we should study performances from three aspects: primitivity, complexity and abstraction, as was done to product descriptions.

Firstly, when the product descriptions in Eq. (17) are primitive products, we have a set of primitive product performances P_a:

$$A_a \subset S_a \cup P(E^b); \quad R_a \subset S_a Y P(E^b); \quad P_a \subset A_a \times R_a. \tag{23}$$

Secondly, just as any product can be decomposed into subassemblies and components until all components become primitive products, the influence network in Fig. 13 can also be decomposed into component performances until all performances become primitive performances. Similar to Eq. (12), we have the following representation of performances:

$$P[0] = \{P(0, 0, 0)\}$$

$$\forall k\ 1 \leqslant k \leqslant nP[k] = \{P(k, i_k, \_) \mid i = 1, 2, \dots\},$$

$$\forall i_n P(n, i_n, \_) \in P_a. \tag{24}$$

Thirdly, according to Eq. (12) and Table 3, product descriptions also imply different levels of abstraction. In the same way, the levels of abstraction for product performances are embodied in $P[k]$, as is given in Table 4. The levels correspond to different stages of design processes. This is illustrated in Fig. 17. It can be seen as the equivalent of Fig. 2 in the case of product performances. It can be seen from Fig. 17 and Table 4 that Eq. (24) embodies the evolving process of product performances as design progresses. Eqs. (23) and (24) have given a complete representation of any product performance in any stage of design.

6. Conclusions

Like other scientific disciplines, a design theory needs to be completed with the formulation of laws by means of

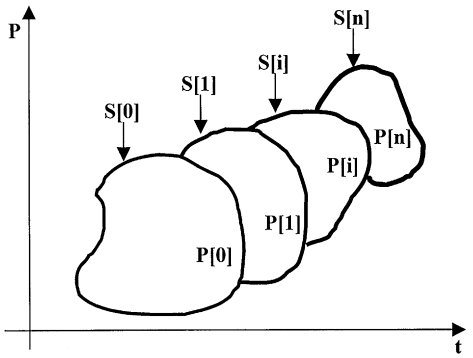

Fig. 17. Product performances in evolving design process.

Table 4
Product performance in different levels of abstraction and complexity

Product description	Action	Response		Product performance
$S[0]$	$A[0]$	$R[0]$	M	$P[0]$
$S[1]$	$A[1]$	$R[1]$	M	$P[1]$
$S[2]$	$A[2]$	$R[2]$	M	$P[2]$
$S[n]$	$A[n]$	$R[n]$	M	$P[n]$

Notes: 1. Index n is the level of complexity. 2. $A[n] \subset S[n] \cup P(E^b)$; $R[n] \subset S[n] \cup P(E^b)$.

an adequate and accurate language. The laws address the fundamental aspects of design and the design process while the language aims to provide representation tools for expressing notions involved in the laws. This part of the paper discussed the language aspect of design science, starting from the basic and general descriptions of design process. Design specifications, product descriptions, and product performances are discussed using set theory, based on the argument that design begins with design requirements and ends with product description. It is also argued that the language should be able to support dynamic evolving design processes from abstract and simple to concrete and complex design descriptions.

The fundamental aspect of these discussions is the definition of product property set which includes basic structural and behavioral characteristics. Design requirements, including structural and performance ones, are formulated into design specifications defined as constraints on product properties. Product descriptions deal with the structural part involved in the design requirements whereas product performances describe the behavioral aspects of design requirement definition. The definition of product descriptions is based on the set of primitive products as well as the levels of complexity and abstraction. This product description provides a single and uniform representation scheme to support entire dynamic design processes.

Product performance is described as the response of a product to external actions in its working environment. The performance of a product is realized through the performance of its constituent components. Those components interact with each other. Together, they form a performance network that transmits the actions on the product to the responses to its environment. The basic block is disclosed to define the fundamental performance pattern of a component. Mathematically, product performance is defined as a relation from an action set to a response set. Finally, along the levels of complexity and abstraction, performances are formulated in different levels.

Acknowledgements

The research reported in this paper is supported by the Natural Science and Engineering Council of Canada (NSERC) through Research Grant OGP0105754. The authors also thank Ms. L. Li and Mr. N. Schemenauer for their suggestions on the improvement of the paper.

References

[1] Kirschman CF. et al. Classifying functions for mechanical design. Proceedings of The 1996 ASME Design Engineering Technical Conference and Computers in Engineering Conference, August 18–22, 1996, Irvine, CAL 96-DETC/DTM-1504.
[2] Gu P. Recent development in design theory and methodology research. Proceedings of International Conference on Manufacturing Sciences, June, 1998. Wuhan, China, p. 21–26.
[3] Sodhi R, Turner JU. Towards modeling of assemblies for product design. Comput Aided Des 1994;26(2):85–97.
[4] Suh NP. The principles of design. Oxford: Oxford University Press, 1990.
[5] Gomez-Senent E et al. The design dimensions: a design theory proposal. Proceedings of ICED'97, 1997. p. 467–89.
[6] Hsu W, Woon IMY. Current research in the conceptual design of mechanical products. Comput Aided Des 1998;30(5): 377–89.
[7] Zeng Y, Gu P. A science-based approach to product design theory. Part I: formulation and formalization of design process, J. Product Process Dev, 1999, Robotics and Computers Integrated Manufacturing, 1999; 15(4).
[8] Messac A, Chen W. The engineering design discipline: is it confounding lexicon hindering its evolution? Proceedings of DETC'98, September 13–16, 1998, Atlanta, GA, DETC98/DTM-5658.
[9] Yoshikawa H. General design theory and a CAD system, in Man-machine Communications in CAD/CAM, Tokyo, Oct. 2–4, 1980, Proceedings of IFIP WG5.2, Amsterdam: North-holland, 1981. p. 35–8.
[10] Salustri FA, Venter RD. An axiomatic theory of engineering design information. Eng Comput 1992;8(4):197–211.
[11] Cheng GD, Zeng Y. Strategies for automatic finite element modeling. Computers & Structures 1992;44(4):905–9.
[12] Maimon O, Braha D. On the complexity of the design synthesis problem. IEEE Transactions on Systems, Man and Cybernetics 1996;26(1):141–50.

[13] Eastman CM, Fereshetian N. Information models for use in product design: a comparison. Comput Aided Des 1994;26(7): 551–72.
[14] Gorti SR, Gupta A, Kim GJ, Sriran RD, Wong A. An object-oriented representation for product and design processes. Comput Aided Des 1998;30(7):489–501.
[15] Gu P, Hashemian M, Sosale S. An integrated modular design methodology for life-cycle engineering. Ann CIRP 1997;46(1): 71–4.
[16] Gui J, Mantyla M. Functional understanding of assembly modeling. Comput Aided Des 1994;26(6):435–51.
[17] Gu P, Sosale S. Product modularization for life cycle engineering. Robotics and Computer Integrated Manufacturing. 1999;15(4): 331–9.
[18] Hindhede U, et al. Machine design fundamentals: a practical approach. New York: Wiley, 1983.