PERGAMON

Robotics and Computer Integrated Manufacturing 15 (1999) 331–339

Robotics and Computer Integrated Manufacturing

www.elsevier.com/locate/rcim

A science-based approach to product design theory Part I: formulation and formalization of design process

Y. Zeng, P. Gu*

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

Abstract

This paper, first of the two parts series, addresses a science-based approach to the study of design processes. Design process is an evolving process that begins with design requirements and ends with product descriptions. A general design governing equation is proposed to characterize the design process. We also proposed a design process model that embodies synthesis and evaluation processes, design problem redefinition process, and design decomposition process. The basis of the design process model is primitive design which is formalized as a six-step process. Any design problem can be decomposed into primitive designs. Utilizing the design representation scheme we established using set theory, the design process is mathematically formulated and formalized. The detailed explanations are provided to illustrate the design governing equation, the design process model and the mathematical formulation. © 1999 Elsevier Science Ltd. All rights reserved.

Keywords: Science-based design studies; Design governing equation; Design process; Set theory

1. Introduction

Research into design and design processes has a long history. However, not until 1960s did the concepts and methods of systems science begin to influence design studies substantially. Significant progress has been made in several disciplines including engineering, philosophy, psychology, computer science and so on, in many aspects such as phenomena, nature, cognitive models and computation of design [1]. Some observations have been made to characterize design, which include that design is a creative act full of style, that the design problem is holistic and ill-structured, and that some of design problems can be dealt with by information processing theories while others cannot. The research is multi-faceted, dealing with design education, design methodologies, and design automation. The fundamental underlying issue is the understanding of design. The research seeks answers for some very fundamental yet obvious questions like "what is design?", "what are the natures of design?", "how is a design task accomplished?" and so forth. These basic questions lead to research into design science which will fundamentally drive design from art towards science.

Like any of other engineering sciences, design science should include two fundamental parts: laws and languages. The laws establish basic assumptions and principles for design processes whereas the languages provide a basic means to represent the laws. The work accomplished by Suh [2], Tomiyama and Yoshikawa [3], and Marston et al. [4] are examples of the former and those done by Salustri and Venter [5], Umeda et al. [6], Cheng and Zen [7], Eastman and Fereshetian [8] and Gui and Mantyla [9] are the examples of the latter. Maimon and Braha [10] tried to put two parts together from the computational point of view. Zeng et al. [11] attempted to set up a formal framework to integrate design process and design objects in one uniform system. Gorti et al. [12] also proposed an object-oriented representation for product and design processes. However, the proposed languages often do not sufficiently support design processes. Intensive science-based design studies are just recent efforts. The research is by no means concluding and profound. The entire exploration is still in pre-theory stage [13]. One of the major reasons is the lack of a good combination of precise mathematical representation languages and laws governing design processes. Some of the existing researches deal with empirical explorations of design (including design methodologies). Other efforts have been devoted to the development of formal representation of design information. The representation schemes often do not sufficiently support design processes. Our aim is two fold: (1) to establish

* Corresponding author.

a basic mathematical representation scheme to define the objects involved in the entire design process; (2) investigate design process with the mathematical representation of design objects. The focus of this paper, first of a two part series, is placed on the study of design process. The mathematical representation scheme is discussed in the Part II of the series by Zeng and Gu [14].

This paper is organized as follows. In the following section, design problem will be formally defined. It proposes a general design governing equation which embodies basic natures of design. Then a mathematical representation scheme, supporting and elaborating the representation of design requirements and products, is provided. In the fourth section, a mathematical model for design process is proposed, which incorporates the widely accepted design process models. We also discuss the basic assumptions necessary for design processes. The final section gives the concluding remarks.

2. Formulation of product design problem

Design is an intelligent activity that begins with design requirements and ends with a product description, as is shown in Fig. 1. In a design process, design requirements are represented by design specifications. Based on the specifications, candidate product descriptions are generated. The product descriptions are evaluated against 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, complaints about the performance of existing products, or the failure due to malfunctions of existing products. The first step in product design is design requirement formulation, which translates design requirements into design specifications [15]. The design process then provides a mapping from design specifications to product descriptions. Therefore, a comprehensive design theory should include at least three basic elements: (1) A general framework to describe and formulate design problems; (2) A language to define the two ends of a design process; and (3) A theory to address the formulation and the design processes. In this paper, our focus is on the first and the third parts, in particular, on the definition of design problem and the mathematical models of design process.

Design requirements are classified into structural and performance requirements, which are constraints imposed on the structure and the performance of a product, respectively. Examples of structural requirements are constraints on the basic dimensions, shapes, configurations, and materials whereas examples of performance requirements are safety, functionality, and manufacturability.

Design specifications are the formulation of design requirements, which manifest themselves as a set of de-

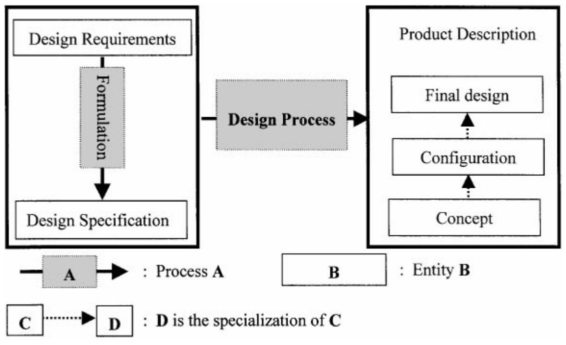

Fig. 1. Design activity.

sired product descriptions or product performances. Denoting product descriptions and product performances as $\boldsymbol{S}$ and $\boldsymbol{P}$, respectively, $\boldsymbol{X}$ as their union gives

$$\boldsymbol{X} = \boldsymbol{S} \cup \boldsymbol{P} \tag{1}$$

we have design specifications $\boldsymbol{R}^{\mathrm{d}}$

$$\boldsymbol{R}^{\mathrm{d}} = \lambda(\boldsymbol{X}, [\boldsymbol{X}]) \tag{2}$$

where $[\boldsymbol{X}]$ is the constraint, quantitatively and/or qualitatively, on entity $\boldsymbol{X}$. According to the definition, design specification can be viewed as a predicate $\lambda(\boldsymbol{X}, [\boldsymbol{X}])$. In the present paper, we will use design specifications and design requirement interchangeability. Both of them refer to the definition given by Eq. (2).

Product descriptions, denoted by $\boldsymbol{S}$, are the representation of design solutions. Design solutions are usually described by concepts, configurations, or product drawings, depending on stages of design process.

Product performance is the response of a product to external actions exerted on it according to the laws in product's working environment.

A product design process can be 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 [13]. Correspondingly, three types of product descriptions are involved as shown in Fig. 1. They are conceptual, configuration, and detailed design. In fact, there are no explicit boundaries between design stages. Design iterations occur throughout entire design process. Every earlier design process will generate 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.

Apart from different representation forms of final product in different design stages, each of those design

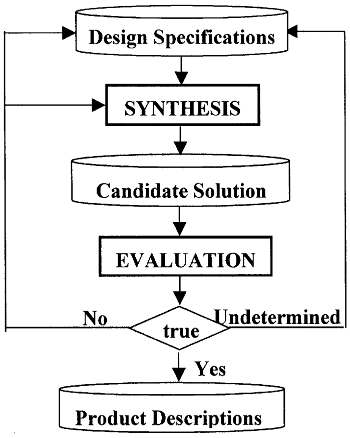

Fig. 2. Basic design process.

subtasks shares some basic features of problem solving. Each subtask is generally finished in two phases: synthesis and evaluation, as is shown in Fig. 2 [16]. At the beginning of design process, a set of design specifications are given, which are defined with respect to descriptions of the product to be designed. To satisfy the specifications, some possible design proposals are then generated for further justification against the specifications. If a design proposal satisfies the specifications or the product description is not detailed enough to be evaluated, then it might be accepted as a design solution or it should be refined further. Otherwise the proposal must be modified or new design proposals should be recommended or the whole design problem should be reformulated. The process is shown in Fig. 2.

In the above process, the synthesis process is responsible for proposing a set of candidate product descriptions based on design specifications. The process of mapping, denoted by $\boldsymbol{K}^{\mathrm{s}}$, from design specifications to product descriptions is defined by

$$\boldsymbol{K}^{\mathrm{s}}: \boldsymbol{R}^{\mathrm{d}} \rightarrow \boldsymbol{S} \quad \text{or} \quad \boldsymbol{S} = \boldsymbol{K}^{\mathrm{s}}(\boldsymbol{R}^{\mathrm{d}}). \tag{3}$$

The mapping can be abstract knowledge, past design cases, and any other information supporting the process. The available synthesis knowledge $\boldsymbol{K}^{\mathrm{s}}$, in most cases, is plausible. Product descriptions cannot be determined for certain. In other words, for one design specification, there may exist many product descriptions to satisfy it, as is shown in Fig. 3. This is a divergent process. The more product descriptions are generated, the more chances the final design is optimal, and of course, the more resources the design processes will consume.

Meanwhile, the evaluation process is used to evaluate candidate products against design specifications. Theoretically speaking, once a product description is given, its properties, including those related to product structure and performance, can be derived from related property knowledge. Denoting the knowledge as $\boldsymbol{K}^{\mathrm{p}}$, we have

$$\boldsymbol{K}^{\mathrm{p}}: \boldsymbol{S} \rightarrow \boldsymbol{X} \quad \text{or} \quad \boldsymbol{X} = \boldsymbol{K}^{\mathrm{p}}(\boldsymbol{S}) \tag{4}$$

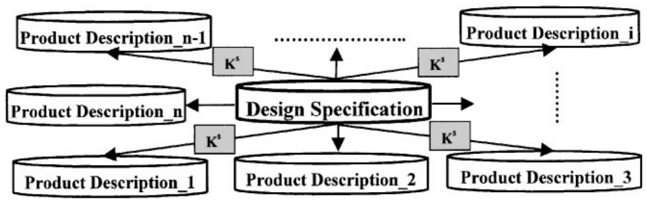

Fig. 3. Mapping from design specification to product descriptions.

Again, $\boldsymbol{X}$ is the union of product descriptions $\boldsymbol{S}$ and product performances $\boldsymbol{P}$. The equation means that as long as a product is defined, related structural and performance information of the product can be obtained through property knowledge. The knowledge can be disclosed scientific principles, well-designed experiments, designers' expertise, and other means supporting the process. This is a convergent process that selects the best final product that satisfies the design requirements. It is a deterministic process. From the logical point of view, there always exist some way to determine if a product description satisfies the prescribed design specifications. The process assigns a Boolean value to design specification predicate $\lambda(\boldsymbol{X}, [\boldsymbol{X}])$ defined in Eq. (2) [17].

$$\lambda = \begin{cases} 1 & \text{when } \boldsymbol{R}^{\mathrm{d}} \text{ is satisfied,} \\ 0 & \text{when } \boldsymbol{R}^{\mathrm{d}} \text{ is unsatisfied,} \\ -1 & \text{when } \boldsymbol{R}^{\mathrm{d}} \text{ cannot be decided for satisfaction.} \end{cases} \tag{5}$$

Boolean values 0, 1 and -1 represent false, true and undetermined, respectively. The objective of a design process is to generate a product description S so that

$$\boldsymbol{R}^{\mathrm{d}} = (\lambda(\boldsymbol{X}, [\boldsymbol{X}]) = 1). \tag{6}$$

This equation transforms design problem solving into a process of looking for solutions that make all the design specification predicates λ to assume a truth value 1. The process is illustrated in Fig. 4.

Since $[\boldsymbol{X}]$ is predefined in some way for a design problem, the truth value of predicate $\lambda(\boldsymbol{X}, [\boldsymbol{X}])$ depends only on $\boldsymbol{X}$. Let

$$\lambda^*(\boldsymbol{X}) = \lambda(\boldsymbol{X}, [\boldsymbol{X}]) \text{ when } \lambda(\boldsymbol{X}, [\boldsymbol{X}]) = 1. \tag{7}$$

Eq. (6) can then be simplified as

$$\boldsymbol{R}^{\mathrm{d}} = \lambda^*(\boldsymbol{X}) = \lambda^*(\boldsymbol{K}^{\mathrm{p}}(\boldsymbol{S})). \tag{8}$$

The two processes can be seen as two operators: synthesis and evaluation operators, acting on the solution space of design problems. The synthesis operator tries to stretch the space whereas the evaluation operator attempts to shrink the space. The interaction of both operators gives rise to the final design solutions as shown in Fig. 5.

In temporal order, the above design process can be described in Fig. 6. The number under the arrows indicates the order of process. The symbol above arrows

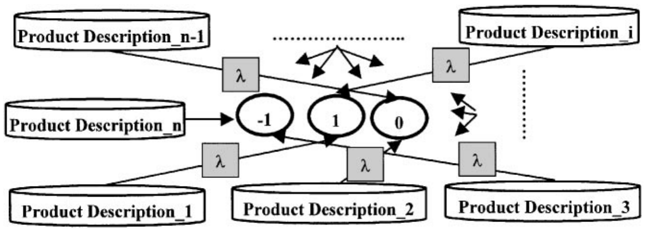

Fig. 4. Mapping from product descriptions to design specifications.

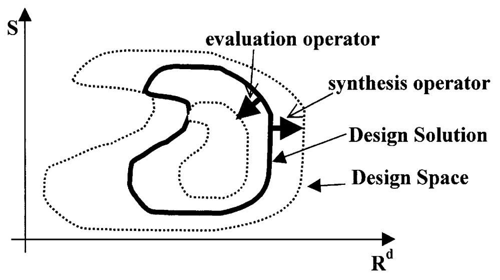

Fig. 5. Design space under synthesis and evaluation.

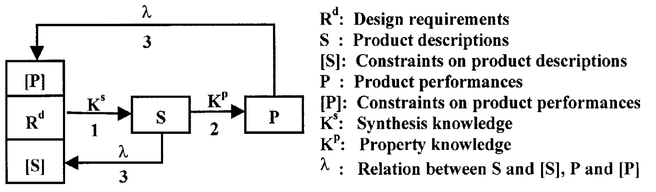

Fig. 6. Temporal order of design process.

represents the function of the process. This is indeed another version of Fig. 2. It can be seen from the figure that the process from $\boldsymbol{S}$ to $\boldsymbol{P}$ completely relies on $\boldsymbol{S}$ which comes from $\boldsymbol{R}^{\mathrm{d}}$ according to $\boldsymbol{K}^{\mathrm{s}}$. This means two things: first, process $\boldsymbol{K}^{\mathrm{p}}$ follows process $\boldsymbol{K}^{\mathrm{s}}$; second, final solution $\boldsymbol{S}$ requires that both processes be logically true at the same time. More formally, it can be represented as

$$\boldsymbol{R}^{\mathrm{d}} \to \boldsymbol{S} \text{ is true if and only if } \lambda(\boldsymbol{X}, [\boldsymbol{X}]) \text{ is true}$$
$$\text{where } \boldsymbol{X} = \boldsymbol{K}^{\mathrm{p}}(\boldsymbol{S}). \qquad (9)$$

In this process, the interpretation of the truth value of design requirements and the determination of the value of product descriptions are mutually dependent. This fact makes design problem solving different from other kinds of problem solving such as diagnostics, prediction, classification and so on ([18,19]). This is also where the difficulties of design come from.

By substituting Eq. (8) into Eq. (3), we get the following equation:

$$\boldsymbol{S} = \boldsymbol{K}^{\mathrm{s}}(\lambda^*(\boldsymbol{K}^{\mathrm{p}}(\boldsymbol{S}))) \quad \text{or} \quad \boldsymbol{S} = \boldsymbol{K}^{\mathrm{s}} \cdot \lambda^* \cdot \boldsymbol{K}^{\mathrm{p}}(\boldsymbol{S}). \qquad (10)$$

This equation is called the design governing equation. This equation is a recursive equation and is the mathematical representation of Eq. (9). Tentative design solutions are first generated using synthesis knowledge. The tentative solutions of design are then used to look for new solutions which are fixed points of a compound function constructed from synthesis and property knowledge. This is an iterative process until the final solutions satisfy all the requirements. This equation underlies design processes. It corresponds to the differential equations in classic engineering sciences. It governs design activities. The purpose of design process models is to find solution approaches for this equation.

Based on the above observations, design can be further defined as a transformation from design requirements $\boldsymbol{R}^{\mathrm{d}}$ to product description $\boldsymbol{S}$ that satisfies the requirements according to property knowledge $\boldsymbol{K}^{\mathrm{p}}$ determined by the product description $\boldsymbol{S}$.

3. Product design representation

The purpose of this section is to review our basic framework representing the entities appeared in the design equation given in Eq. (10). Based on the framework, design process models can be established and described in a formal way. To support the dynamic design evolving process, a mathematical framework of design representation is given in Fig. 7. It requires that every object involved in design processes includes a time factor. These objects include design requirements, product descriptions, and product performances.

Since design requirements are defined with respect to product descriptions and product performances, only product descriptions and performances are addressed.

In representing a product, the first thing is to define a set of primitive products whose performance can be defined independently. They are basic components and connectors. Examples in product design are gears, shafts, bearings, springs, fasteners, and welds. They are denoted by $\boldsymbol{S}_{\mathrm{a}}^{i}$

$$\boldsymbol{S}_{\mathrm{a}} = \{\boldsymbol{S}_{\mathrm{a}}^{i} \mid \boldsymbol{i} = \mathbf{1}, \mathbf{2}, \ldots, \boldsymbol{m}\} \qquad (11)$$

where $\boldsymbol{S}_{\mathrm{a}}$ is the set of primitive products. A product is decomposed into components related by connectors down to the level in which the components are predefined primitive products. This generates a tree structure of product description as is shown in Fig. 8. The product can then be defined as

$$\boldsymbol{S}[\mathbf{0}] = \{\boldsymbol{S}(\mathbf{0}, \mathbf{0}, \mathbf{0})\}$$
$$\forall \boldsymbol{k}, \mathbf{1} \leqslant \boldsymbol{k} \leqslant \boldsymbol{n}, \boldsymbol{S}[\boldsymbol{k}] = \{\boldsymbol{S}(\boldsymbol{k}, \boldsymbol{i}_k, \_) \mid \boldsymbol{i}_k = \mathbf{1}, \mathbf{2}, \ldots, \boldsymbol{n}_k\}$$
$$\forall \boldsymbol{i}_n, \mathbf{1} \leqslant \boldsymbol{i}_n \leqslant \boldsymbol{n}_n, \boldsymbol{S}(\boldsymbol{n}, \boldsymbol{i}_n, \_) \in \boldsymbol{S}_{\mathrm{a}} \qquad (12)$$

where $\boldsymbol{k}$ is the layer of tree and $\boldsymbol{n}_k$ is the number of nodes in the $\boldsymbol{k}$th layer of the tree. $\boldsymbol{S}[\boldsymbol{k}]$ is the product description with respect to the nodes in the $\boldsymbol{k}$th layer of the tree. $\boldsymbol{S}(\boldsymbol{k}, \boldsymbol{i}_k, \_)$ represents the $\boldsymbol{i}_k$th node in the $\boldsymbol{k}$th layer of the tree. The third index is designed to represent the

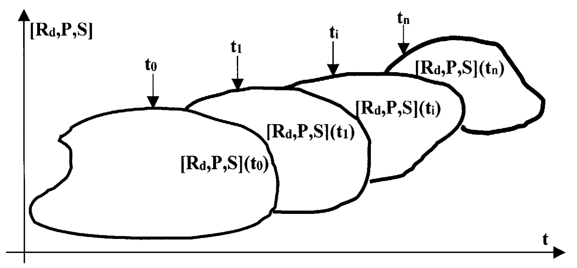

Fig. 7. A framework of design representation.

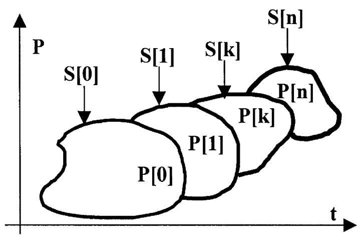

Fig. 9. Product descriptions and performances in evolving design process.

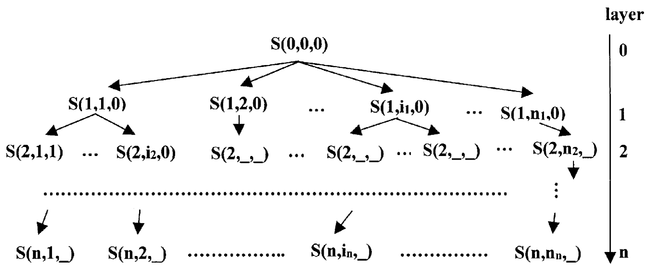

Fig. 8. Tree structure of product description.

relative position each node in the tree with reference to its parent node. It is of no relevance for the present discussion, so we use a dash " – " to represent it for simplicity.

As can be seen from Eqs. (1) and (4), product performances depend on product descriptions. Corresponding to product descriptions, three aspects of product performance should be taken into account: primitivity, complexity and abstraction.

Firstly, corresponding to primitive products, we have a set of primitive product performances $\boldsymbol{P}_{\mathrm{a}}$

$$\boldsymbol{P}_{\mathrm{a}} = \{\boldsymbol{P}_{\mathrm{a}}^{i} \mid \boldsymbol{i} = 1, 2, \ldots, \boldsymbol{n}_p\} \tag{13}$$

where $\boldsymbol{P}_{\mathrm{a}}^{i}$ is a primitive product performance and $\boldsymbol{n}_p$ is the number of possible primitive performances.

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

$$\begin{aligned} &\boldsymbol{P}[0] = \{\boldsymbol{P}(0, 0, 0)\} \\ &\forall \boldsymbol{k}, 1 \leqslant \boldsymbol{k} \leqslant \boldsymbol{n}, \boldsymbol{P}[\boldsymbol{k}] = \{\boldsymbol{P}(\boldsymbol{k}, \boldsymbol{i}_k, \_) \mid \boldsymbol{i}_k = 1, 2, \ldots\}\boldsymbol{P}_{\mathrm{a}} \\ &\qquad = \{\boldsymbol{P}_{\mathrm{a}}^{i} | \boldsymbol{i} = 1, 2, \ldots, \boldsymbol{n}_p\} \\ &\qquad \forall \boldsymbol{i}_n, \boldsymbol{P}(\boldsymbol{n}, \boldsymbol{i}_n) \in \boldsymbol{P}_{\mathrm{a}}. \end{aligned} \tag{14}$$

The representation schemes in Eqs. (11) and (14) imply the evolution of product descriptions and performances of design process. It is shown in Fig. 9. As the value of index $\boldsymbol{k}$ increases, product descriptions $\boldsymbol{S}[\boldsymbol{k}]$ and product performances $\boldsymbol{P}[\boldsymbol{k}]$ become more detailed and concrete until the components become primitive products. Hence, a design process usually begins with a definition of abstract product description and performance which has a small value of $\boldsymbol{k}$. The value of $\boldsymbol{k}$ increases as design process evolves. This representation scheme provides a top–down approach to supporting the dynamic design process from generic and simple to concrete and complex product descriptions, as is required by Fig. 7.

4. Formal design process model

One of the objectives in science-based design studies is to establish design process models that act like an algorithmic solution to the general design governing equation Eq. (10). It should provide a step-by-step guidance to naïve designers or foundations for the development of CAD systems. The models should be the refinements of design processes shown in Fig. 2 by elaborating the synthesis and evaluation processes. There have existed many formal results in modeling evaluation process. Suh [2] proposed two axioms to determine if a designed product is good. Some others (Lee and Thornton [20]; Law and Antonsson [21]; and Simpson et al. [22] studied the approaches to evaluating the quality of a product. Comparatively, more efforts are needed for the science-based research for the integral design process. In this section, we attempt to study design process based on the representation scheme and the general design governing equation introduced in previous sections. We will not cover specific details which are to be portrayed in our later work.

An algorithm, according to traditional understanding, is a finite, unambiguous description of a procedure to solve a class of problems. Fundamentally, it consists of primitive recursive functions and unification operations. Any complex problem can be solved by the unification of primitive recursive functions ([23]). Similarly, in studying formal design process models, we proposed a three-step problem solving model. The first step is to define primitive design. The second is to decompose original design problems into primitive ones to generate primitive design solutions. The third is to combine the generated primitive products.

4.1. Primitive design

As was assumed in the last section, in any domain of product design there exists a set of primitive products whose descriptions and performances are well defined. Moreover, according to Eq. (4), we can define a set of property knowledge for the primitive products as

$$\boldsymbol{K}_{\mathrm{a}}^{p} \subset \boldsymbol{S}_{\mathrm{a}} \times \boldsymbol{P}_{\mathrm{a}}. \tag{15}$$

Again, $\boldsymbol{S}_{\mathrm{a}}$ is primitive product description set and $\boldsymbol{P}_{\mathrm{a}}$ is primitive performance set. In fact, $\boldsymbol{K}_{\mathrm{a}}^{p}$ can be knowledge, experiments, or existing expertise. One product may have many kinds of performances as is shown in Fig. 10.

Based on the above notion, it is reasonable to assume the existence of a set of primitive design requirements $\boldsymbol{R}_{\mathrm{a}}^{d}$, which are constraints on the descriptions and performances of primitive products.

$$\boldsymbol{R}_{\mathrm{a}}^{d} = \lambda(\boldsymbol{X}, [\boldsymbol{X}_{\mathrm{a}}]),$$
$$\boldsymbol{X}_{\mathrm{a}} = \boldsymbol{S}_{\mathrm{a}} \cup \boldsymbol{P}_{\mathrm{a}}. \tag{16}$$

Primitive product descriptions, primitive product performances, and primitive design requirements constitute a design solution space. Any design problem solving is accomplished by searching through this space. The final solution should satisfy the general design governing equation defined in Eq. (10). By denoting corresponding primitive synthesis knowledge and primitive property knowledge as $\boldsymbol{K}_{\mathrm{a}}^{s}$ and $\boldsymbol{K}_{\mathrm{a}}^{p}$, respectively, we obtain the primitive design governing equation as

$$\boldsymbol{S}_{\mathrm{a}} = \boldsymbol{K}_{\mathrm{a}}^{s} \cdot \lambda^{*} \cdot \boldsymbol{K}_{\mathrm{a}}^{p}(\boldsymbol{S}_{\mathrm{a}}). \tag{17}$$

This equation is different from Eq. (10), although they share the same form. The definition of solution space is clearer here because the primitive design governing equation can be solved based on predefined descriptions and performances of primitive product as well as primitive design requirements. The following algorithm is proposed to solve Eq. (17) according to Figs. 2 and 6.

Step 1: determine a set of candidates of primitive product $\boldsymbol{S}_{\mathrm{a}}$ according to given design requirements $\boldsymbol{R}_{\mathrm{a}}^{d}$ and pre-defined synthesis knowledge $\boldsymbol{K}_{\mathrm{a}}^{s}$:

$$\boldsymbol{S}_{\mathrm{a}} = \boldsymbol{K}_{\mathrm{a}}^{s}(\boldsymbol{R}_{\mathrm{a}}^{d}). \tag{18}$$

Step 2: determine the performances of candidate products $\boldsymbol{P}_{\mathrm{a}}$ related to design specifications by a set of product property knowledge $\boldsymbol{K}_{\mathrm{a}}^{p}$:

$$\boldsymbol{P}_{\mathrm{a}} = \boldsymbol{K}_{\mathrm{a}}^{p}(\boldsymbol{S}_{\mathrm{a}}). \tag{19}$$

Step 3: evaluate the candidate products against design specification

$$\boldsymbol{R}_{\mathrm{a}}^{d} = \lambda^{*}(\boldsymbol{X}_{\mathrm{a}}); \boldsymbol{X}_{\mathrm{a}} = \boldsymbol{S}_{\mathrm{a}} Y \boldsymbol{P}_{\mathrm{a}}. \tag{20}$$

Step 4: If design requirements are satisfied, end the process. Otherwise go to step 1.

This four-step process is named after primitive design process. There are several points worth to be noted with regard to the above process. Firstly, property knowledge $\boldsymbol{K}_{\mathrm{a}}^{p}$ depends on the design specifications to be satisfied and the product descriptions to be evaluated. For a design problem, property knowledge is a subset of the set of laws in product's working environment. Mathematically, it can be represented as a correlation class of the set of environmental laws with respect to product descriptions and design specifications.

$$\boldsymbol{K}_{\mathrm{a}}^{p} = [\boldsymbol{S}_{\mathrm{a}} \cup [\boldsymbol{P}_{\mathrm{a}}], \boldsymbol{L}]_{\approx}, \tag{21}$$

where $\boldsymbol{L}$ is the set of laws in product's working environment. $[\boldsymbol{S}_{\mathrm{a}} \cup [\boldsymbol{P}_{\mathrm{a}}], \boldsymbol{L}]_{\approx}$ is a correlation class of set $\boldsymbol{S}_{\mathrm{a}} \cup [\boldsymbol{P}_{\mathrm{a}}]$ with respect to set $\boldsymbol{L}$. Correlation class is defined based on correlation relation '$\approx$' between two sets $\boldsymbol{Y}$ and $\boldsymbol{Z}$

$$\forall \boldsymbol{y} \in \boldsymbol{Y} \forall \boldsymbol{z} \in \boldsymbol{Z} (\boldsymbol{y} \approx \boldsymbol{z} \rightarrow \boldsymbol{z} \approx \boldsymbol{y}). \tag{22}$$

The correlation class of set $\boldsymbol{Z}$ with respect to set $\boldsymbol{Y}$, denoted as $[\boldsymbol{Z}, \boldsymbol{Y}]_{\approx}$, is a subset of $\boldsymbol{Y}$ consisting of all those elements in $\boldsymbol{Y}$ that has correlation relation with each element in $\boldsymbol{Z}$.

$$[\boldsymbol{Z}, \boldsymbol{Y}]_{\approx} = \{\boldsymbol{y} \in \boldsymbol{Y} | \forall \boldsymbol{z} \in \boldsymbol{Z}, \boldsymbol{y} \approx \boldsymbol{z}\}. \tag{23}$$

If $\boldsymbol{K}_{\mathrm{a}}^{p}$ turns out to be an empty set, it means that the performance knowledge of a product should be acquired. This is why modeling, analysis, experiments and prototyping are often required in accomplishing a design task.

Secondly, if we define three spaces to represent design specifications, product descriptions and product performances, then the mappings among them can be used to describe the above solution processes. It is shown in Fig. 11, which can be seen as the extension of Fig. 4. The process consists of six steps:

(1) a set of tentative primitive product descriptions $\boldsymbol{S}_{\mathrm{a}}$ is generated according to given primitive design specifications $\boldsymbol{R}_{\mathrm{a}}^{d}$;
(2) the tentative primitive products will define a set of new design specifications $(\boldsymbol{R}_{\mathrm{a}}^{d})'$, which extends original design specifications into augmented design specifications $(\boldsymbol{R}_{\mathrm{a}}^{d})^{+}$;

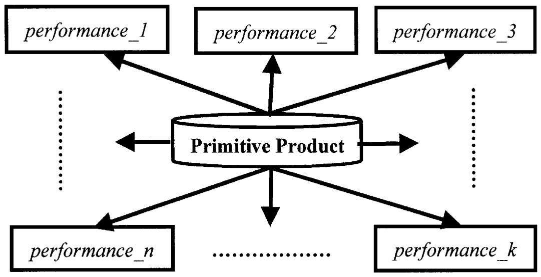

Fig. 10. Product description-performance relationship.

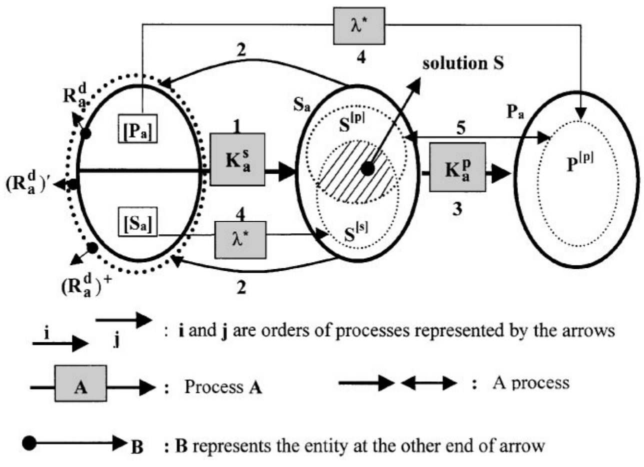

Fig. 11. Mappings between R_a^d, S_a and P_a.

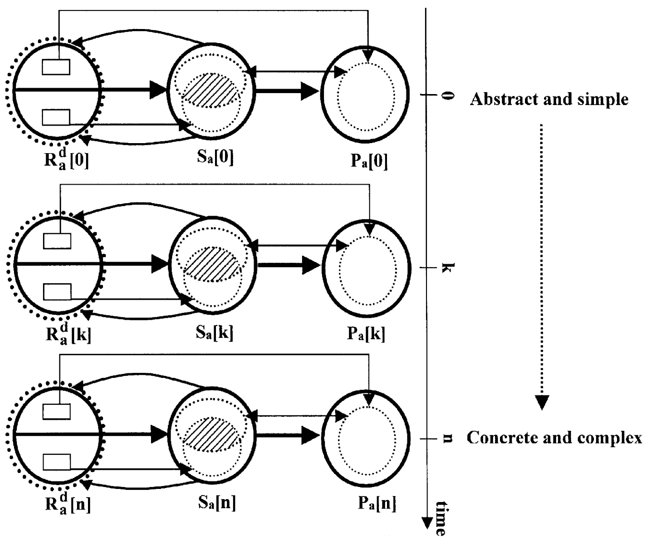

Fig. 12. Different levels of abstraction of primitive design.

(3) according to the generated tentative product specifications S_a and augmented design specifications $(R_a^d)^+$, the correlated performances P_a of the tentative product are derived;
(4) the tentative product specifications S_a and the correlated performances P_a are evaluated against the augmented design specifications $(R_a^d)^+$. Legal subsets of product descriptions $S^{[s]}$ and product performances $P^{[p]}$ are then obtained;
(5) a subset of product descriptions $S^{[p]}$ corresponding to product performances $P^{[p]}$ is derived;
(6) the intersection of product specifications S_a, $S^{[s]}$, and $S^{[p]}$ constitute the final design solution.

Symbolically, the above process can be defined as:

1. $\forall R_a^d, \exists K_a^s\,(K_a^s\colon R_a^d \to S_a)$,
2. $\forall S_a \exists (R_a^d)'(R_a^d)^+ = R_a^d \cup (R_a^d)'$
3. $\forall S_a[P_a] \exists K_a^p((K_a^p = [S_a \cup [P_a], L]_{\approx}) \wedge (K_a^p\colon S_a \to P_a))$,
4. $\exists S^{[s]} \in S_a \exists P^{[p]} \in P_a(\lambda^*(S^{[s]})) \wedge (\lambda^*(P^{[p]}))$ if $S^{[s]} = \Phi \vee P^{[p]} = \Phi$, go to 1,
5. $\forall P^{[p]} \exists S^{[p]} \in S_a(K_a^p\colon S^{[p]} \to P^{[p]})$ if $S^{[p]} = \Phi$, go to 1,
6. $S = S_a I S^{[s]} I S^{[p]}$ if $S = \Phi$, go to 1.

It should be noted that the choice of primitive products and, in turn, the primitive product performances and design requirements is in fact artificial and relies on designers' expertise and knowledge as well as the state of the art of technology. For example, experienced designers have more primitive products in mind so that their designs are usually more alive and flexible, compared with naïve designers. They also have more complex primitive products, which make their generations of design faster in many cases.

The proposed mathematical representation of primitive design fits well with the existing design process models. They correspond to different levels of complexity and abstraction of product description in the definition of synthesis knowledge in Eq. (3). When the design description is well defined, it becomes a case-based design. When the design description is just partially defined, it is a knowledge-based design. The primitive design in different levels of complexity and abstraction is shown in Fig. 12.

4.2. Decomposition of design specifications and combination of products

In real-life design problems, it is not always easy to find a proper set of design knowledge to map all the given design specifications onto product descriptions and then accomplish the design task according to the procedures given above. The original design specifications R^d can be divided into two parts with respect to the primitive design specifications R_a^d. The first is the intersection of the set R^d and the set R_a^d. The second is the subset of R^d that does not have any intersection with the primitive design specifications R_a^d. For the latter case, a kind of requirement structure is required to transform the specifications into primitive ones. The structure can be deduced by substituting Eqs. (12) and (14) into Eq. (2). Hence we have

$$R^d\,[k] = \lambda\,(X(k), [X(k)]),$$
$$X(k) = S\,[k] \cup P\,[k]. \tag{24}$$

The equation means that the evolution of design specifications with design process can also be represented with reference to the levels of complexity and abstraction of product descriptions. The above equation also provides a way to decompose the original design specifications R^d into a set of primitive design specifications R_a^d

$$R^d = \{r_i^{ad} \mid r_i^{ad} \in R_a^d, i = 1, 2 \ldots\}. \tag{25}$$

In transforming design specifications into product descriptions, the specifications can be gradually decomposed into simpler specifications which can be ultimately decomposed into primitive ones. This process allows product description to be generated incrementally. However, as is portrayed in Fig. 10, a product may have some performances besides those used in defining design specifications. These performances may conflict with the already generated solution. This makes the problem subject to continuous redefining. This is the so called ill-structure of design problems. Therefore, multitude steps of decomposition may be required. A recursive, dynamic, and incremental decomposition is given below:

$$\boldsymbol{R}_1^d = \tau(\boldsymbol{r}_1^{\mathrm{ad}}, \boldsymbol{R}^d) \quad \boldsymbol{R}_i^d = \tau(\boldsymbol{r}_i^{\mathrm{ad}}, \boldsymbol{R}_{i-1}^d) \quad \boldsymbol{r}_i^{\mathrm{ad}} \in \boldsymbol{R}_{\mathrm{a}}^d. \tag{26}$$

where τ is decomposition operator of design specifications, which decompose a design specification $\boldsymbol{R}_{i-1}^d$ into a primitive design specification $\boldsymbol{r}_i^{\mathrm{ad}}$ and another design specification $\boldsymbol{R}_i^d$. It is defined as

$$\tau(\boldsymbol{r}_i^{\mathrm{ad}}, \boldsymbol{R}_{i-1}^d) = (\boldsymbol{R}_{i-1}^d / \{\boldsymbol{r}_i^{\mathrm{ad}}\}) \cup (\{\boldsymbol{r}_i^{\mathrm{ad}}\} \updownarrow (\boldsymbol{R}_{i-1}^d / \{\boldsymbol{r}_i^{\mathrm{ad}}\})),$$
$$\boldsymbol{r}_i^{\mathrm{ad}} \in \boldsymbol{R}_{\mathrm{a}}^d, \tag{27}$$

where '/' represents the difference of sets and β represents the conflicts between sets. This equation shows that the newly generated design specification $\boldsymbol{R}_i^d$ consists of two parts. One is the design specification given by subtracting the primitive design specification $\boldsymbol{r}_i^{\mathrm{ad}}$ from the design specification $\boldsymbol{R}_{i-1}^d$. Another is the conflict between the primitive design specification $\boldsymbol{r}_i^{\mathrm{ad}}$ and the subtraction of $\boldsymbol{R}_{i-1}^d$ by $\boldsymbol{r}_i^{\mathrm{ad}}$. Indeed, the above definition provides a design specification structure for a design problem, which takes into account the dynamic nature of design. Therefore, in design problem solving process, the solution process will redefine the problem, which makes design an ill-structured problem [24].

Naturally, following design requirement decomposition is the product combination, which incrementally combines the generated primitive product into a completed partial product:

$$\boldsymbol{S}_{\mathrm{p}}^0 = \Phi,$$
$$\boldsymbol{S}_{\mathrm{p}}^i = \xi(\boldsymbol{S}_{\mathrm{a}}^i, \boldsymbol{S}_{\mathrm{p}}^{i-1}),$$
$$\boldsymbol{S}_{\mathrm{a}}^i = \boldsymbol{K}_{\mathrm{a}}^s(\boldsymbol{r}_i^{\mathrm{ad}}) = \boldsymbol{K}_{\mathrm{a}}^s \cdot \lambda^* \cdot \boldsymbol{K}_{\mathrm{a}}^p(\boldsymbol{S}_{\mathrm{a}}^i), \tag{28}$$

where ξ is the product combination operator. $\boldsymbol{S}_{\mathrm{p}}^i$ is a partially defined product. When a generated primitive product is combined into the partially defined product, some of their performances may conflict with each other. The conflict becomes a new requirement. The conflict is refined as the performance or structural conflicts between newly generated primitive product and existing partial product.

$$\{\boldsymbol{r}_i^{\mathrm{ad}}\} \updownarrow (\boldsymbol{R}_{i-1}^d / \{\boldsymbol{r}_i^{\mathrm{ad}}\}) = \boldsymbol{X}_{\mathrm{a}}^i \updownarrow \boldsymbol{X}_{\mathrm{p}}^i,$$
$$\boldsymbol{X}_{\mathrm{a}}^i = \boldsymbol{K}_{\mathrm{a}}^p(\boldsymbol{S}_{\mathrm{a}}^i)\, \boldsymbol{X}_{\mathrm{p}}^i = \boldsymbol{K}_{\mathrm{a}}^p(\boldsymbol{S}_{\mathrm{p}}^i),$$
$$\boldsymbol{X}_{\mathrm{a}}^i = \boldsymbol{S}_{\mathrm{a}}^i \cup P_{\mathrm{a}}^i \boldsymbol{X}_{\mathrm{p}}^i = \boldsymbol{S}_{\mathrm{p}}^i \cup \boldsymbol{P}_{\mathrm{p}}^i. \tag{29}$$

Again $\updownarrow$ represents the conflicts between sets. Compromises or redesigns are required when conflicts occur in a design process.

By combining primitive design process and design decomposition process, we can get a formal model of the entire design process as follows:

```
Design ($\boldsymbol{R}^d, \boldsymbol{S}$){
  $\boldsymbol{R}_0^d = \boldsymbol{R}^d$;
  $\boldsymbol{S}_{\mathrm{p}}^0 = \Phi$;
  $\boldsymbol{i} = \mathbf{0}$;
  repeat {
    $\boldsymbol{i} = \boldsymbol{i} + \mathbf{1}$;
    $\boldsymbol{r}_i^{\mathrm{ad}} \in \boldsymbol{R}_{i-1}^d \cap \boldsymbol{R}_{\mathrm{a}}^d$; //decomposition of design
                                    specifications
    $\boldsymbol{S}_{\mathrm{a}}^i = \boldsymbol{K}_{\mathrm{a}}^s(\boldsymbol{r}_i^{\mathrm{ad}}) = \boldsymbol{K}_{\mathrm{a}}^s \cdot \lambda^* \cdot \boldsymbol{K}_{\mathrm{a}}^p(\boldsymbol{S}_{\mathrm{a}}^i)$; //primitive design
                                                process
    $\boldsymbol{S}_{\mathrm{p}}^i = \xi(\boldsymbol{S}_{\mathrm{a}}^i, \boldsymbol{S}_{\mathrm{p}}^{i-1})$; //combination of primitive
                                    product into partial products
    $\boldsymbol{X}_{\mathrm{a}}^i = \boldsymbol{K}_{\mathrm{a}}^p(\boldsymbol{S}_{\mathrm{a}}^i)$;   //properties of primitive and
                                    partial products
    $\boldsymbol{X}_{\mathrm{p}}^i = \boldsymbol{K}_{\mathrm{a}}^p(\boldsymbol{S}_{\mathrm{p}}^i)$;
    $\boldsymbol{R}' = \boldsymbol{X}_{\mathrm{a}}^i \updownarrow \boldsymbol{X}_{\mathrm{p}}^i$; //conflicts between 'partial and
                                    primitive products
    $\boldsymbol{R}_i^d = (\boldsymbol{R}_{i-1}^d / \{\boldsymbol{r}_i^{\mathrm{ad}}\}) \cup \boldsymbol{R}'$; //new design specifications
  until $\boldsymbol{R}_i^d = \Phi$; //all design specifications are satisfied
  $\boldsymbol{S} = \boldsymbol{S}_{\mathrm{p}}^i$; }
```

This is a formal realization of the design process required by Fig. 1. The process progresses by refining the abstract and simple ideas of product to concrete and complex descriptions of product until the final solution is achieved. The equations used here are defined in Eqs. (17) and (25) through Eq. (29).

In the above process, if primitive design specifications $\boldsymbol{R}_{\mathrm{a}}^d$ that match original design specifications $\boldsymbol{R}^d$ are comprehensive enough, the process can end within very few loops. The extreme case is parametric design in which only local modifications are needed for design.

5. Conclusions

Like other scientific disciplines, a design theory needs to be completed with the formulation of laws by means of a robust 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 law aspect of design science, starting from basic and general descriptions of design process. Based on the argument that design begins with design requirements and ends with the product description, a general design governing equation is established, which embodies the inner recursive relationship between design specifications and product descriptions using set theory. It is argued that design can be seen as a transformation from design requirements to product description that satisfies the requirements according to property knowledge related to the product description.

Utilizing the product design representation scheme we proposed using set theory, which includes time factor, design process is also discussed at primitive and general levels. Corresponding to primitive products, a primitive design process model is developed. The model consists of six steps which include synthesis, problem redefinition, and evaluation. To solve a general design problem, the design process is gradually decomposed into primitive ones with a design specification decomposition operator. The final solutions are generated by incrementally integrating the primitive designs into one product description with product combination operator. Entire design process is formulated and formalized with a set theory-based mathematical language.

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 Mr. N. Schemenauer for his suggestions on the improvement of the paper.

References

[1] Cross N. Developments in design methodology. Chichester: Wiley, 1984.
[2] Suh NP. The Principles of Design. Oxford: Oxford University Press, 1990.
[3] Tomiyama T, Yoshikawa H. Extended general design theory, Design Theory for CAD. Amesterdam: Elsevier Science Publishers B.V., 1987. p. 95–130.
[4] Marston M, Bras B, Mistree F. The applicability of the axiomatic and decision-based design equations in variant design. Proceedings of DETC'97, DETC97/DTM3884, 1997.
[5] Salustri FA, Venter RD. An axiomatic theory of engineering design information. Eng. Compu. 1992;8(4):197–211.
[6] Umeda Y, Takeda H, Tomiyama T, Yoshikawa H. Function behavior and structure. In: JS Gero (Ed.) Applications of artificial Intelligence in Engineering V, Vol. 1 Berlin: Springer Verlag. pp. 177–194.
[7] Cheng GD, Zeng Y. Strategies for automatic finite element modeling. Computers and Structures, 1992, Vol. 44, No. 4, pp. 905–909.
[8] Eastman CM, Fereshetian N. Information models for use in product design: a comparison. Computer-Aided Design 1994;26(7):551–72.
[9] Gui J, Mantyla M. Functional understanding of assembly modeling. Computer-Aided Design, 1994;26(6):435–51.
[10] Maimon O, Braha D. On the complexity of the design synthesis problem. IEEE Transactions on Systems, Man and Cybernetics, 1996;26(1):141–50.
[11] Zeng Y, Jing J, Liu H, Xie X. Formal framework for architectural design. Final report for the Natural Science Foundation of China, 1996.
[12] Gorti SR, Gupta A, Kim GJ, Sriram RD, Wong A. An object-oriented representation for product and design processes. Computer-Aided Design 1998;30(7):489–501.
[13] Gu P. Recent development in design theory and methodology research, Proceedings of International Conference on Manufacturing Sciences, June, 1998, Wuhan, China, pp. 21–6.
[14] Zeng Y, Gu P. A science-based approach to product design theory. Part II: formulation of design requirements and products. Robotics and Computer Integrated Manufacturing. 1999;15(4):314–52.
[15] Otto KN. Forming product design specifications. Proceedings of DETC'96, 96-DETC/DTM-1517, 1996.
[16] Jones JC. A method of systematic design. In: N. Cross (Ed.) Developments in Design Methodology. Wiley, Chichester, 1984.
[17] Treur J. A logical analysis of design tasks for expert systems. International Journal of Expert Systems, 1989;2:233–53.
[18] Zeng Y, Cheng GD. On the logic of design. Des Stud 1991;12(3):137–41.
[19] Roozenburg NFM. On the pattern of reasoning in innovative design. Des Stud 1993;14(1):4–18.
[20] Lee DJ, Thornton AC. The identification and use of key characteristics in the product development process. Proceedings of DETC'96, 96-DETC/DTM-1506, 1996.
[21] Law WS, Antonsson EK. Multi-dimensional mapping of design imprecision. Proceedings of DETC'96, 96-DETC/DTM-1524, 1996.
[22] Simpson TW, Rosen D, Allen JK, Mistree F. Metrics for assessing design freedom and information certainty in the early stages of design. Proceedings of DETC'96, 96-DETC/DTM-1521, 1996.
[23] Davis MD, Weyuker EJ. Computability, Complexity and Languages. New York: Academic Press, 1983.
[24] Simon HA. The structure of ill structured problems. Arti Intell 1973;4(2):181–201.