Modeling the Structure and Complexity of Engineering Routine Design Problems

J. M. Jauregui-Becker, W. W. Wits, F. J. A. M. Van Houten
Faculty of Engineering Technology, University of Twente, Enschede, The Netherlands
j.m.jaureguibecker@ctw.utwente.nl

Abstract
This paper proposes a model to structure routine design problems as well as a model of its design complexity. The idea is that having a proper model of the structure of such problems enables understanding its complexity, and likewise, a proper understanding of its complexity enables the development of systematic approaches to solve them. The end goal is to develop computer systems capable of taking over routine design tasks based on generic and systematic solving approaches. It is proposed to structure routine design in three main states: problem class, problem instance, and problem solution. Design complexity is related to the degree of uncertainty in knowing how to move a design problem from one state to another. Axiomatic Design Theory is used as reference for understanding complexity in routine design.

Keywords:
Routine design, Complexity, Axiomatic Design Theory.

1 INTRODUCTION

Advances in technology and competitive markets are driving the development of products with shorter times to market. In addition to this, products are becoming more complex, with more functionality, and yet lower prices. To cope with this trend, engineers often redesign existing products to meet new requirements and optimize performances based on a steady concept. This type of design problems can be regarded as *routine*.

Although routine design occurs within a well defined domain of knowledge, such problems may be of very complex natures. Consider for example the design of injection molds. The first injection molds were designed and developed in 1868 by John Wesley Hyatt, who injected hot cellulose into a mold for producing billiard balls [1]. Much later, in 1946, James Hendry built the first screw injection molding machine, giving birth to the machines and processes we know nowadays. Since then, much knowledge on injection mold design has emerged and been formalized in books (e.g. [1] and [2]), expert systems (e.g. [3] and [4]) and the Internet. However, given the amount of components, physical phenomena and processes involved, the design of injection molds is still considered a complex task.

If such problems are modeled following the pyramid model of Gerrit Muller [5], the result is a large hierarchical multi-layered network of design functions, components and variables. As one might envision, determining strategies and methods for solving such complex problems is not a trivial task. Furthermore, if one considers that around 80% of design in industry is routine, it can be concluded that counting with a proper understanding of its complexity is a relevant topic in the field of design theory and methodology.

Having the previous as motivation, this paper presents a model to structure and determine the complexity of routine design problems. It also provides general guidelines for dealing with complexity in routine design. The future goal of the research this paper forms part of, is to effectively and systematically manage design complexity as a means for automating the generation of candidate solutions to complex routine design problems. The concepts of design complexity as stated in Axiomatic Design Theory (ADT) are used to assemble a specific model complexity for routine design.

The remainder of this paper is organized as follows:

- Section 2 present a classification of design problems and defines what routine design is.
- Section 3 describes the rationales of the synthesis process in routine design.
- Section 4 presents a new model for structuring routine design problems.
- Section 5 describes the main concepts stated in Axiomatic Design Theory.
- Section 6 presents a model of complexity in routine design by mapping ADT complexity theory onto the structuring framework presented in Section 4.
- Section 7 presents some general guidelines for dealing with design complexity

To finish, Section 8 presents a discussion and final remarks.

2 DESIGN PROBLEM CLASSIFICATION

The scope of this paper lies within the field of artifactual engineering routine design. This section explains what is meant by this at the hand of the FBS [6] model.

2.1 FBS model

FBS models a design artifact by distinguishing the following levels of object representation: Function, Behavior/State and Structure, as shown in Figure 1. The basis of the FBS model is that the transition from function to structure is performed via the synthesis of physical behaviors.

Therefore, behaviors allow characterizing the implementation of a function. As many different views of the FBS model have been developed and researched, this paper adopts the unified FBPSS model presented by Zhang et al [6]. This model is based on the analysis and generalization of the Japanese ([7], [8]), European ([9]), American ([10]) and Australian ([11]) schools of design modeling.

The FBPSS model uses the following definitions:

- Structure: Is a set of entities and relations among entities connected in a meaningful way. Entities are perceived in the form of their attributes when the system is in operation. For example, in Figure 1 the Structure is represented by an electric motor and a crank mechanism. Here, the two possible entities (structures) are the lengths of the bars L_1 and L_2.
- States: Are quantities (numerical or categorical) of the Behavioral domain (e.g. heat transfer, fluid dynamics, psychology). States change with respect to time, implying the dynamics of the system. For example, in Figure 1, the states of the structure are represented by the distance L_0 between the electric motor and the piston, the torque T of the electric motor, or the displacement of the piston s.
- Principle: Is the fundamental law that allows the development of a quantitative relation of the States variables. It governs Behavior as the relationships among a set of State variables. For the example in Figure 1, two possible principles are electromagnetism ruling the operation of the electric motor, and solid mechanics ruling the function of the crank mechanism.
- Behavior: Represents the response of the structure when it receives stimuli. Since the Structure is represented by States and Structure variables, Behaviors are quantified by the values of these variables. In the case presented in Figure 1, the two Behaviors are *Generate torque* and *Convert torque into force*.
- Function: It is about the usefulness of a system. For example, in Figure 1, one possible function of this system is to compress gas.

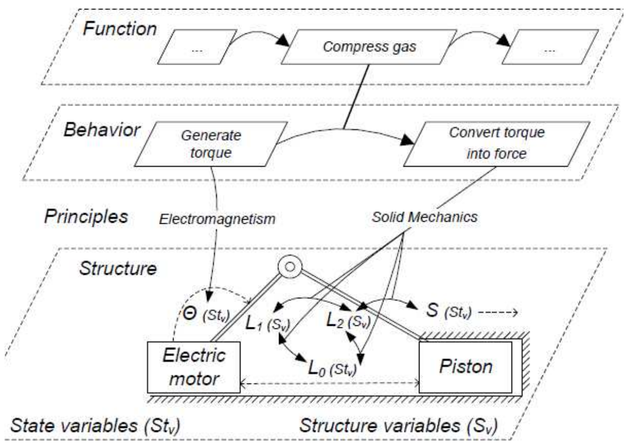


Figure 1: FBS of a crank compression mechanism.

2.2 Classification of Design Problems

If one considers a design artifact as an object with a complete FBS description, a design problem can be defined as one with an incomplete set of descriptions. As shown in Figure 2, according to the types of incomplete representations design is classified in:

- Routine design: One in which the space of functions, behaviors and structures is known, and the problem consists of instantiating structure variables.
- Innovative design: One in which the functions and behaviors are known, and the design consists of generating new structures that satisfy them.
- Creative design: One in which the functions are known, and the problem consists of determining the structures and behaviors required to satisfy them.

Nature encompasses a vast variety of behaviors (physical, chemical, human, etc). Considering physical and human behaviors, design can be classified in:

- Engineering design: Behaviors are characterized by principles stated in the laws of physics. Depending on the discipline of study, engineering design can be further classified into mechanical, electrical, chemical, geological, etc.
- Human centered design: behaviors are characterized by physiologic, psychological and emotional human reactions. Two examples are architectural design and industrial design.

Under these definitions, the scope of this paper lies within the boundaries of engineering routine design.

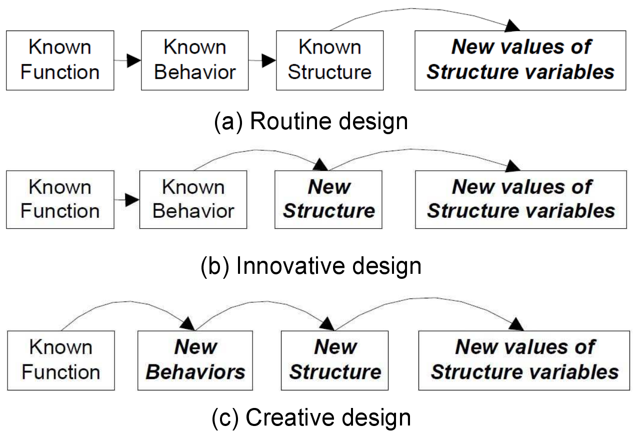


Figure 2: FBS based design problem classification

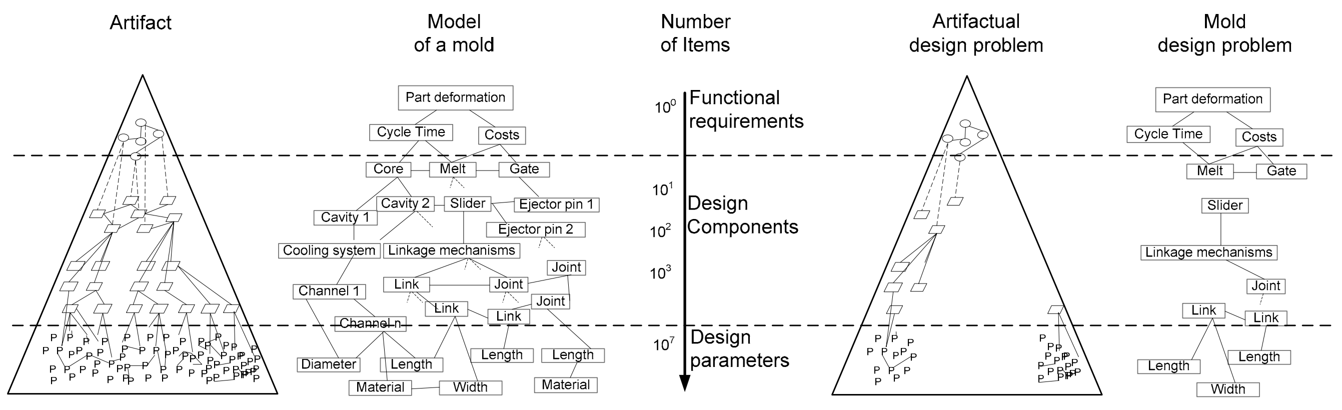
3 SYNTHESIS IN ROUTINE DESIGN

3.1 Modeling Design Artifacts

Artifacts, e.g. an injection mold, can be modeled as a hierarchical multi-layered network of interrelated components and parameters that resemble the structure of the pyramid of Gerrit Muller [5], as shown in Figure 3(a). In this model, the top layers represent functional requirements, the in-between levels represent components, and the lower levels represent design parameters of these components. Functional requirements specify the characteristics of an artifact's function, as for example the power of an electric engine. Furthermore, in this model components are composed of networks of other sub-components, and so forth. For example, sliders in injection molds are composed of mechanical linkages, which are simultaneously composed of rigid links and joints. It is characteristic to complex artifacts to have a large number of interconnected networks of components, as well as a large number of parameters, relations and constraints.

3.2 Modeling Design Problems

An artifactual design problem can be modeled as an incomplete description of an artifact, as it is shown in Figure 3(b). The descriptions known beforehand are regarded as the design requirements, and these must be satisfied by candidate solutions. Design requirements can be functional requirements, components, parameter values, or combinations thereof. Creative, innovative, and routine design problems can be represented using this



(a) The pyramid of Gerrit Muller of an design artifact

(b) The pyramid of Gerrit Muller of a design problem

Figure 3: Design artifact vs. design problem representation

model, as the differences among them reside in the type of knowledge available for generating candidate solutions. In routine design there is knowledge available about:

- the types of components that can be used to generate candidate solutions,
- how components can be connected with each others, parametric descriptions of each component, and,
- relations and constraints that relate parameters and components to functional requirements.

Furthermore, designing one type of artifact can be the subject of different types of problems, as several combinations of design requirements can be formulated.

3.3 Synthesis in Routine Design

As Figure 4 indicates, moving from an incomplete representation to a complete description is done by a synthesis process. Synthesis processes in routine design are performed by two types of tasks: (1) generating networks of components and (2) attributing values to unknown parameters. The exact strategy determining how these tasks are performed depends on the distribution of requirements throughout the different levels of detail of the problem, e.g. only on top, only at the bottom or as a mix. However, this relation is not known a priori and is different for different distributions of design requirements. As it will be shown in Section 6, complexity in routine design relates to the uncertainty in determining this relation.

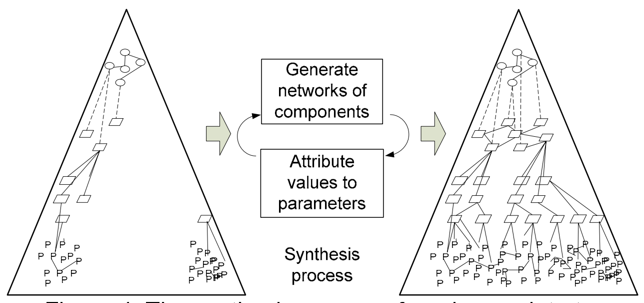

Generate networks of components
Attribute values to parameters
Synthesis process
P

Figure 4: The synthesis process: from incomplete to complete descriptions.

4 THE STRUCTURE OF ROUTINE DESIGN

In design, structure and complexity are two closely related concepts. Complexity is a property related to the degree of difficulty or uncertainty for finding a solution to a design problem [12], whereas structure is a property related to the organization of the variables and relations describing the design problem itself [13]. In order to analyze design complexity, it is necessary to understand the structure of the problem. The structure of a design problem has three important aspects to be studied:

- The consistency of the design variables: all design variables have to be related to each other's by relations that are ultimately used to determine the performances of the problem.
- The distribution of design parameters along the problem model: all parameters concentrated in one element vs. several parameters scattered through several elements; one vs. several levels of detail.
- The relation between what is known (design requirements) and what is unknown (unattributed structure variables): scattered along the problem model, concentrated in problem chunks, at the top (only functional requirements), at the bottom (only design parameters) or as mix of all these possibilities.

In order to develop a standard model of complexity in routine design, a standard way of structuring such problems is first required. Therefore, this section describes a framework that has been developed to structure design problems. The framework is inspired in the structure of analysis problems. The analysis of design complexity to be presented in Section 6 is based on this framework.

4.1 Structure in Analysis

Physicists model natural phenomena through differential and integral equations. Specific problems are solved by setting boundary conditions on the differential and integral equations, and applying solving procedures to obtain analytical expressions. The resulting expression can then be used to calculate values of variables by specifying the values of the input parameters. Consider for example the law of heat conduction shown in Equation 1. This differential equation models the phenomena of heat transfer through matter from a region of high temperature to a region of a low temperature. As this is done independent of geometry, material properties or temperature distributions, the equation is generic. To model a specific case of heat transfer the equation is rearranged by introducing boundary conditions, canceling unnecessary terms and performing mathematical manipulations. For example, heat conduction in one dimension between two flat plates results, after rearranging Equation 1, in Equation 2. The obtained equation can now be used to introduce known values and calculate the values of the required parameters, as for example in equation (3) the time required to get temperature $T = \psi$ at a point $x = \zeta$ is $t = \xi$.

$$\frac{\delta T}{\delta t} = a\frac{\delta^2 T}{\delta x^2} \tag{1}$$

$$t(x,T) = \frac{x^2}{\pi^2 \cdot a} \cdot ln\left(\frac{8}{\pi^2} \cdot \frac{T - T_W}{T_{MO} - T_W}\right) \tag{2}$$

$$t(x = \zeta, T = \psi) = \xi \tag{3}$$

If one would generalize this structure, one would notice that physicists model natural phenomena at the hand of tree phases which are solved by two types of procedures, as shown in Figure 5. On the one hand, the three phases are: the differential/integral equation, the analytical expression, and the solution. On the other hand, the procedures are: differential and integral calculus to transform the differential equation (phase 1) into an analytical expression (phase 2), and algebra or numeric methods to transform the analytical expression (phase 2) into values of unknown parameters (phase 3).

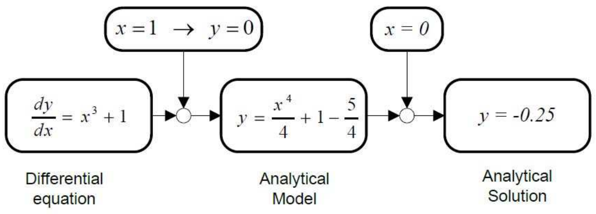

$x = 1 \rightarrow y = 0$
$x = 0$
$\frac{dy}{dx} = x^3 + 1$
$y = \frac{x^4}{4} + 1 - \frac{5}{4}$
$y = -0.25$
Differential equation
Analytical Model
Analytical Solution

Figure 5: Structure of problems in modeling natural phenomena

From a research perspective, this problem structure has allowed:

- defining the types of representations required for modeling each domain (i.e. operators),
- studying the complexity of each domain by analyzing the configuration of the used representations (i.e. equation order, equation linearity),
- developing procedures and operations to solve each problem domain (i.e. Laplace Transformations, Newton-Raphson method)
- identifying common problem structures (i.e. Poisson's equation, Laplace equation)

On the other hand, some of the major advantages from an application perspective are:

- reuse of existing problem formulation,
- utilization of standard solving methods,
- development of computer based simulation tools.

Structuring design problems as done in natural phenomena is likely to drive the automation of design problems toward more generic approaches, with advantages in both research and application. For this research, such a structure would allow the identification of features causing complexity, and do so independent from the problem semantics. Furthermore, strategies for managing design complexity can be formulated as function of problem structures.

4.2 Definitions

The structure here presented is based on the definitions presented in [14], where the different types of information contents and models used in routine design are presented. The definitions are:

- Element: is a class description of a component.
- Descriptions: characterize an element class by representing its attributes in the form of variables.
- Cardinality: Is a parameter that models the number of elements in a design solution. Its value can be unknown, known or determined by an algebraic relation.
- Embodiment: is the subset of descriptions of an element upon which instances are created to generate design solutions.
- Scenario: is the subset of environment variables, attributed to elements in the natural world and considered in measuring a design artifact's ability to accomplish its function.
- Performances: are descriptions used to express and assess the artifact's behavior.
- Analysis relations: use known theories (e.g. the laws of physics or economics) to model the interaction of the design artifact with its environment and predict its behavior. Determine the performances.
- Topology relations: define the configuration of embodiment and scenario elements by means of relations expressing belonging and connectedness.
- Objective function: weighs and adds the performances into one general indicator.

4.3 Structuring Framework

It is proposed to structure routine design problems at three different states: problem class, problem instance and problem solution. This is shown in Figure 6. Problem classes are transformed into problem instances by specifying its requirements. Requirements are specified by instantiating descriptions. Problem instances are transformed into problem solutions by algorithms that generate instances to the unknown descriptions. Therefore, one problem class can represent many problem instances, and one problem instance can have many problem solutions, as indicated in Figure 7. Under this view, solving routine design is analogues to solving problems with known differential equations.

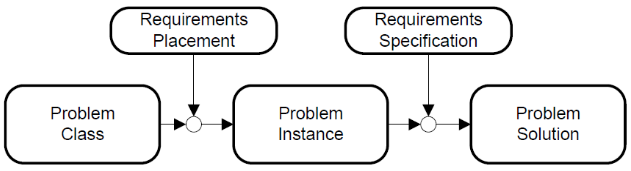


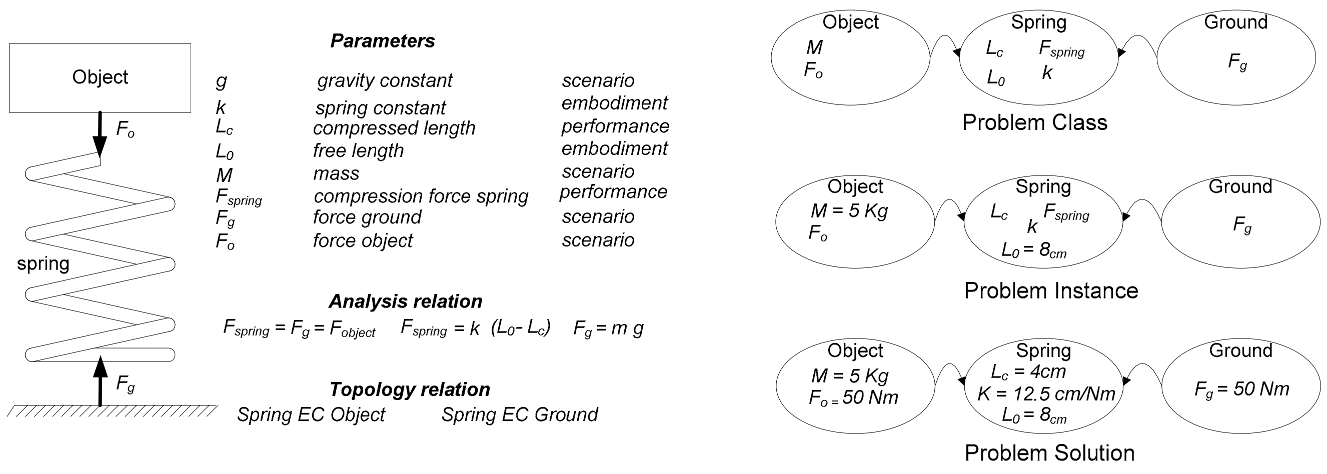
Figure 6: Framework to structure design problems

This structure can also be used to structure innovative and creative design problems. Solving innovative design problems is analogous to combining different differential equations to model various interrelated physical phenomena. Solving creative design problems is analogous to developing new differential equations. However, these two types of design problems are outside the scope of this paper.

Problem Class

A problem class is structured in:

- Elements: are considered class descriptions, and are used to represent both, embodiment and scenario elements. Elements can also be differentiated by assessing the functions and types of descriptors modeling a component.
- Relations: are considered class descriptions and are of different types, namely, topology, physical coherence, design rules, analysis relations and objective functions [14]. Their descriptions can be declared within the scope of the relation or by pointing towards descriptions of embodiment and scenario elements.



Parameters

g	*gravity constant*	*scenario*
k	*spring constant*	*embodiment*
L_c	*compressed length*	*performance*
L_0	*free length*	*embodiment*
M	*mass*	*scenario*
F_{spring}	*compression force spring*	*performance*
F_g	*force ground*	*scenario*
F_o	*force object*	*scenario*

Analysis relation

$F_{spring} = F_g = F_{object}$ $F_{spring} = k\ (L_0 - L_c)$ $F_g = m\,g$

Topology relation

Spring EC Object *Spring EC Ground*

(a) Design problem formulation of a spring (b) Structure of spring design

Figure 8: Example of phases in design structure

- Descriptions: are variables that characterize elements and relations by mathematic models. These can also be of different types: parameters, shapes, fields, topology and spatial, as described in [14].

Problem Instance

A problem instance is structured by:

- Instantiated scenario: represent scenario specifications,
- Partially instantiated embodiment elements or parameters: represent embodiment requirements, and impose constraints to the space of possible solutions.
- Instantiated performance parameters: represent the performance specifications the embodiment has to meet.

Problem Solution

The problem solution consists of fully instantiated elements, relations and parameters. For under-constrained problem-instances, many solutions may exist. This depends on how constrained the problem is. An under constrained will allow for multiple solutions, while a systems of equation type of problem will have a limited number of solutions.

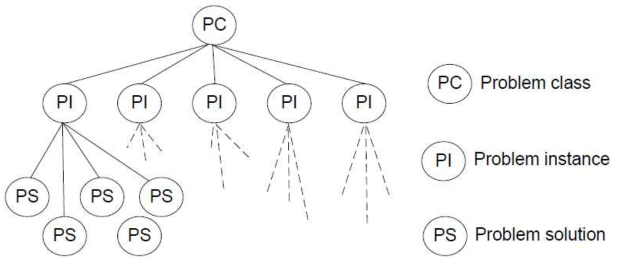


Figure 7: Problem structure dependencies

4.4 Example: Spring Design

Consider for example the spring design formulation shown in Figure 8. Figure 8(b) shows that the problem class corresponds to the declaration of the different types of parameters of the problem formulation. By setting requirements, the problem class is transformed into one (1) problem instance. However, by setting requirements to other parameters, other problem instances can be obtained. The figure shows one possible problem solution. Nevertheless, for this problem instance, many possible problem solutions are possible.

5 COMPLEXITY IN AXIOMATIC DESIGN THEORY

As stated in [15], the complexity of systems has been studied from two different perspectives: the physical domain and the functional domain. On the one hand, complexity in the physical domain is seen as an inherent characteristic of physical things, including algorithms and products. According to this view, systems with many parts are more complex than those with less. Examples of such studies are computational complexity [16] and complex emergent systems [17]. On the other hand, complexity in the functional domain is seen as a relative concept that evaluates how well we can satisfy “what we want to achieve" with “what actually is achievable". From this perspective, axiomatic design theory [15], incompleteness of information [18] and multi-disciplinary complexity [19] are some of the different models to understand the complexity of systems.

This paper is based on the notions of design complexity stated in Axiomatic Design Theory (ADT) [15]. The basic idea of this model is that without difficulty in understanding (or making, operating, etc.), a system is not complex. In this sense, complexity is the property of a system that makes it difficult to understand with the available knowledge about its constituents parts. Tomiyama further elaborates this view, by stating that complexity can be studied from the view point of knowledge structure [19], identifying two types of complexity: complexity by design and intrinsic complexity of multi-disciplinarily. The former is attributed to the structure of the design problem, while the latter deals with behavioral characteristics.

This paper adopts the ADT definition of complexity, and focuses on complexity by design. The three main reasons why this model has been chosen as framework in this research are: (1) complexity is regarded as a relative property, (2) complexity is the consequence of engineering activities, and (3) it is assumed that complexity can be managed. The ADT model of complexity is used to identify different types of complexities in routine design. This section presents a summary of ADT’s model of design complexity.

5.1 Axiomatic Design Theory

ADT is based on the hypothesis that there are fundamental principles that govern good designs [20]. Its two founding axioms are: (1) maintain the independence of the

Functional Requirements (FRs) and (2) minimize the information of the Design Parameters (DPs).

FRs are the set of requirements that characterize the needs of the artifact in the functional domain, while DPs are the variables that characterize the design in the physical domain. The relation between the FRs and the DPs is represented in equation form as:

$$\{FR\} = [A]\{DP\} \tag{4}$$

where A is the Design Matrix (DM) of the problem.

Depending on the DM, a design can be coupled, decoupled or uncoupled. Consider for example a problem with two FRs and two DPs. When the design is coupled, the FRs cannot be satisfied independently because of the interdependence with both DPs, as shown in Equation 5. In a decoupled design, shown in Equation 6, the DPs have to be solved in a particular order so that FRs are achieved. In uncoupled designs (Equation 7), the FRs are independent from each others, and no particular order is required for solving the DPS.

$$Coupled: \begin{Bmatrix} FR1 \\ FR2 \end{Bmatrix} = \begin{bmatrix} x & x \\ x & x \end{bmatrix} \begin{Bmatrix} DP1 \\ DP2 \end{Bmatrix} \tag{5}$$

$$Decoupled: \begin{Bmatrix} FR1 \\ FR2 \end{Bmatrix} = \begin{bmatrix} x & 0 \\ x & x \end{bmatrix} \begin{Bmatrix} DP1 \\ DP2 \end{Bmatrix} \tag{6}$$

$$Uncoupled \begin{Bmatrix} FR1 \\ FR2 \end{Bmatrix} = \begin{bmatrix} x & 0 \\ 0 & x \end{bmatrix} \begin{Bmatrix} DP1 \\ DP2 \end{Bmatrix} \tag{7}$$

5.2 Complexity model

In ADT, complexity is defined as “the measure of uncertainty in achieving the functional requirements of a system within their specified design range”. When the range of a system changes as a function of time, it is regarded as a system with time-dependent complexity. When the range does not change as a function of time, it has a time-independent complexity. “Time” is used in a general sense, signifying the progression of “events".

Time-independent complexity is classified into time-independent real complexity and time-independent imaginary complexity. The former is a consequence of the system range not being inside the design range. The latter occurs when there are many FRs and the design is a decoupled design. It is called imaginary because this corresponds to a situation in which the different orders in solving the design matrix have different attributed levels of difficulty. A system with imaginary complexity can satisfy the FRs at all times if we vary DPs in the right order.

Time-dependent complexity is the uncertainty caused by the increase or decrease of the number and types of DPs during the design process itself. ADT classifies time-dependent complexity into combinatorial and periodic complexity. Design problems with combinatorial complexity experience a continued growth of their DPs in time. For example, constructing a sentence by the combination of words has combinatorial complexity. As the number of words (the DPs in this case) increases, keeping semantic and syntactic consistency among them becomes more difficult. On the other hand, periodic complexity is the case in which the increase of parameters is restarted after a period of time (or succession of actions). An example described in [15] is air traffic control. Air traffic in large airports follows a wave pattern that depends on the time of the day. When the traffic is at its peak, air controllers deal with very complex situations. However, at low traffic times their task becomes significantly simpler.

ADT suggests three main strategies for managing design complexity: (1) minimize the number of FRs, (2) eliminate time-independent real complexity and time-independent imaginary complexity, and (3) transform a system with time-dependent combinatorial complexity into a system with time dependent periodic complexity. This paper adopts this model of design complexity, and explores its characteristics for routine design problems.

5.3 Translating ADT terminology

ADT terminology is translated into the design structuring framework (see Section 4.2) as follows:

- Functional Requirements (FRs): correspond to the functions, performances and scenario descriptions as described in Section 4.
- Design Parameters (DPs): correspond to the embodiment descriptions in the routine design formulation, and model elements and parameters.
- Design Matrix (DM): is formed by the analysis, topology and physical coherence constraints.

6 MODEL OF COMPLEXITY IN ROUTINE DESIGN

The model of complexity obtained in this research is the result of mapping the ADT complexity model presented in Section 5 onto the structuring framework presented in Section 4. The result is a set of complexity types in problem classes and another for problem instances. Complexity of problem classes deals with incorrect problem formulations, while complexity in problem instances deals with deriving strategies for solving it.

6.1 Complexity of Problem Classes

Time-independent complexity captures the complexity of a system in which the time dimension does not limit its ability to achieve its functional requirements. In other words, the range of the system does not change over time. In routine design, the process of moving from problem classes to problem instances is not a synthesis process by itself. This is rather a human process that involves specifying which are the parameters and elements characterizing the input of the problem. Therefore, complexity here is related to how well the problem has been formulated, and regards two types of uncertainties. One is the uncertainty of having all of the required information in the problem formulation. The second is the lack of differentiation between problem chunks and its interrelations. Figure 9 illustrates this idea by showing an inconsistent and unstructured problem in Figure 9(a) and a consistent and structure one in Figure 9(b).

6.2 Complexity of Problem Instances

Time-independent Real Complexity

Real complexity appears in multi-objective problems, where one parameter has to satisfy contradicting objectives. This is caused when several disciplines determine an artifact’s behavior. For example, the more lanes a highway has, the more traffic it can accommodate. At the same time, as the number of lanes increases, the number of accidents also increases. Designing highways with the objectives of traffic maximization and accidents minimization has a contradiction, and therefore is a problem with time-independent real complexity.

Time-independent Imaginary Complexity

Imaginary complexity originates from the fact that design requirements are set at different combinations of parameters and elements. As consequence, it is not known a priori in which order the problem will be solved. Furthermore, when the DM is decoupled, a particular order is required for solving the problem. Imaginary complexity depends on the relations between known and unknown cardinalities as well as on the relation between instantiated and non instantiated elements and parameters. The former is regarded in this work as knowledge distribution, while the latter is regarded as requirements distribution.

From a knowledge distribution viewpoint, the more cardinalities that are known in a problem instance, the lower its uncertainty is regarding the number of elements to instantiate. Moreover, as the cardinalities of elements are often interrelated among each other, modifying one automatically leads to the modification of others. In this sense, knowledge distribution refers to the distribution of known cardinalities among the elements of the problem.

Requirements distribution refers to the distribution of instantiated and non instantiated elements and parameters. For elements, complexity regards the uncertainty of having to apply bottom-up or top-down approaches, while for parameters it regards the uncertainty of knowing in which order to solve the constrained system.

Time-dependent Combinatorial Complexity

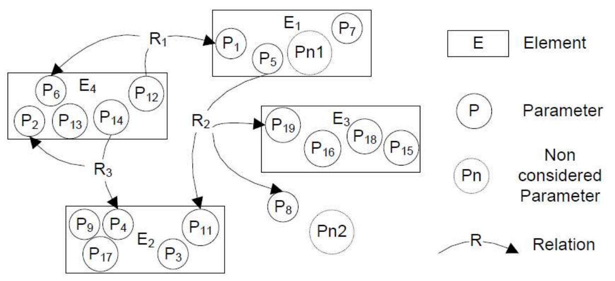
For problem instances, combinatorial complexity occurs when the design problem consists in generating complex topologies or shapes in which embodiment elements are instantiated several times within one design solution (for example, the number of gears required in a gear box). This results in a DM with time-dependent varying size and terms. When no knowledge is available about the number of instances required to satisfy the FRs, the problem presents time-dependent combinatorial complexity. One way of recognizing this type of complexity is by assessing the cardinalities of the elements in the problem formulation. When the ranges of cardinalities are not known, or cannot be written as function of other parameters, the design problem has combinatorial complexity.

7 COMPLEXITY MANAGEMENT

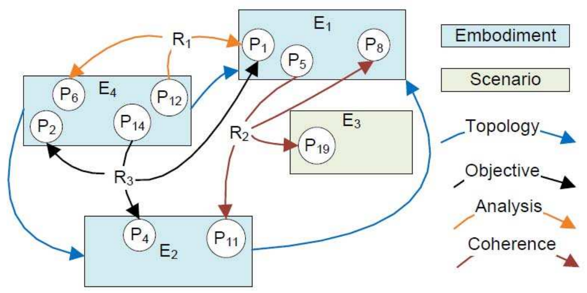
7.1 Problem classes

Ignorance of FRs and DPs is related to the failure to properly understand them in a design task. This is caused by a faulty or incomplete description of the functions, performances and scenarios in the problem formulation. As result, one would be addressing a wrong problem. Complexity emerging from incomplete problem formulations is time-independent imaginary complexity. In ADT, imaginary complexity is defined as the uncertainty that arises because of the designer's lack of knowledge and understanding of a specific design itself. In order to solve this, the FRs of the problem have to be identified and related to the problem's DPs. Then, the problem can be reformulated in terms of the emerging relations between FRs and DPs. By doing so, the imaginary component of the complexity can be managed. Two methods can be used to manage such complexities:

- FBS based formulation: Described in [21], this method aids the exploration of the information required to formulate a given design problem. The goal of the method is to obtain a consistent mapping between a problem function, its behaviors, and the parameters and relation in its formulation.
- ADT based decomposition: Described in [22], this method deals with the decomposition of the problem into smaller problem chunks. The goal is to distribute the information of the problem among different abstraction levels.



(a) Inconsistent and unstructured



(b) Consistent and structured

Figure 9: Complexity in problem classes.

7.2 Problem instances

Time-independent real complexity can be managed by using constraint satisfaction and optimization techniques. In the field of Multidisciplinary Design Optimization [23] several techniques have been developed to cope with such problems, as for example Collaborative Optimization (CO), Bi-Level Integrated System Synthesis (BLISS) and Analytic Target Cascading (ATC).

Time-independent imaginary complexity can be managed by determining strategies that establish in which order to solve the design problem. Notable techniques are based on Design Structuring Matrix manipulations [24].

Time-dependent combinatorial complexity can be managed by making parametric models of the topology of the system. The field of Computational Design Synthesis (CDS) has developed grammar based approaches that can be used to cope with this type of complexity ([25]).

7.3 Integration challenge

Having this overview, one can say that the real challenge lies in the development of a framework to integrate these techniques. Such a framework requires representations that can be used for modeling design knowledge (elements, parameters, and relations) in a generic fashion while supporting the utilization of aforementioned techniques simultaneously. Furthermore, such representations should allow the development of methods for determining solving strategies as a function of the organization of the building blocks used to represent a design problem and permit the independent modeling of abstraction levels.

8 DISCUSSION

In general terms, solving routine design can be performed in two different fashions: developing specific methods for specific problems, or developing generic methods for problem families. It is the second approach that can enable the future development of software that automates this process, and where the focus of this paper lies. Such an approach is confronted with three main challenges. Firstly, design problems have to be translated into terms that allow their study independent of its semantic contents, thus to define a common language. Secondly, basic problem characteristics have to be identified which consists of finding the common ground of these problems. And thirdly, general problem solving approaches have to be developed. The first challenge was researched by the authors and presented in [14]. The second challenge is what this paper focuses on: investigating different types of complexity in design. Its result is a generic description of the basic components of design complexity that can be found in routine design. Further research is required to develop generic methods for managing these complexities, which is the third challenge. It is expected that these approaches will lead to robust standard methods that are capable of automating the generation of candidate solutions to routine design problems.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the support of the Innovation-Oriented Research Programme 'Integral Product Creation and Realization (IOP IPCR)' of the Netherlands Ministry of Economic Affairs, Agriculture and Innovation.

REFERENCES

[1] Douglas, M.B., 1999, Plastic Injection Molding: Manufacturing Startup And Management.

[2] Menges, M., 1986, How to make injection molds, Hanser.

[3] Mok, C.K., Chin, K.S. and Ho, J.K.L., 2001, An Interactive Knowledge-Based CAD System For Mould Design in Injection Moulding Processes. The International Journal of Advanced Manufacturing Technology, 17(1), pp27-38.

[4] Chan, W.M., Yan, L., Xiang, W. and Cheok, B.T., 2003, A 3D CAD knowledge-based assisted injection mould design system. The International Journal of Advanced Manufacturing Technology, 22(5), pp387-395.

[5] Muller, G.J., 2007, Design objectives and design understandability.

[6] Zhang, W.J., Lin, Y. and Sinha, N., 2005, On the Function-Behavior-Structure Model for Design. Canadian Design Engineering Network conference.

[7] Umeda, Y. and Tomiyama, T., 2003, FBS modeling: Modeling scheme of Function for Conceptual Design. In Working Papers of the 9th Int. Workshop on Qualitative Reasoning About Physical Systems. Amsterdam. pp271-278

[8] Umeda, Y. and Tomiyama, T., 1997, Functional Reasoning in Design. AI in Design, 12(2), pp41-48.

[9] Pahl, G., Beitz, W., Feldhusen, J. and Grote, K.H., 2007, Engineering Design: A Systematic Approach, Springer.

[10] Chandrasekaran, B., Goel, A. and Iwasaki, Y., 1993, Functional Representation as Design Rationale. IEEE Compute, 26, pp48-56.

[11] Gero, J.S. and Kannengiesser, U., 2004, The situated function-behaviour-structure framework. Design Studies, 25, pp373-391.

[12] Suh, N.P., 1999, A Theory of Complexity, Periodicity and the Design Axioms. Research in Engineering Design, 11(2), pp116-132.

[13] Simon, H., 1973, The Structure of Ill Structured Problems. Artificial Intelligence, 4(3), pp181-201.

[14] Jauregui-Becker J.M, Tragter H and van Houten, F.J.A.M., 2009, Structure and models of artifactual routine design problems for computational synthesis. CIRP Journal of Manufacturing Science and Technology, 1(3):120-125.

[15] Suh, N.P., 2005, Complexity: Theory And Applications, Oxford University Press.

[16] Arora, S. and Barak, B., 2009, Computational Complexity: A Modern Approach, Cambridge.

[17] Murthy, V.K. and Krishnamurthy, E.V., 2009, Multiset of Agents in a Network for Simulation of Complex Systems, Springer Verlag, New York.

[18] Ueda, K., Synthesis and emergence - research overview, 2001, Artificial Intelligence in Engineering, 15, pp321-327.

[19] Tomiyama, T., 2006, Dealing with Complexity in Design: A Knowledge Point of View. Design Methods for Practice, pp137-146.

[20] Suh, N.P., 1998, Axiomatic Design Theory for Systems. Research in Engineering Design,10, pp 189-209.

[21] Jauregui-Becker, J.M., Tragter, H., Kokkeler, F.G.M. and Houten, F.J.A.M.v., 2007, On the Parametric Definition of Design Problems for Computational Synthesis. In International Conference on Engineering Design.

[22] Jauregui-Becker, J.M., Tragter, H. and Houten, F.J.A.M.v., 2009, From how much to how many: a method to develop representations for computational synthesis. In International Conference on Engineering Design 2009. Stanford, California, USA.

[23] Papalambros, P.Y. and Wilde, D.J., Principles of Optimal Design- Modeling and Computation. (Cambridge University press, 2000).

[24] Browning, T.R., 2002, Applying the design structure matrix to system decomposition and integration problems: a review and new directions. Engineering Management, IEEE Transactions on, 48(3), pp292-306.

[25] Kurtoglu, T., Swantner, A. and Campbell, M.I., 2008, Automating the Conceptual Design Process: From Black-box to Component Selection. Design Computing and Cognition '08, pp553-572.