INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED'09
24 - 27 AUGUST 2009, STANFORD UNIVERSITY, STANFORD, CA, USA

FROM *HOW MUCH* TO *HOW MANY:* A METHOD TO DEVELOP REPRESENTATIONS FOR COMPUTATIONAL SYNTHESIS

J. M. Jauregui-Becker[1], H. Tragter[2], F. J. A. M. van Houten[3]
(1,2,3) University of Twente

ABSTRACT
This paper presents *from how much to how many* as a method to parameterize artifactual routine design problems for computational synthesis. The goal is to develop representations with low levels of complexity to ease the initialization of a computational synthesis process. To achieve this, complexity management guidelines from axiomatic design theory are used. The case study of Cooling for Injection Molding (CIM) is used to demonstrate the application of the method. The resulting representations are used to develop a Computational Synthesis System of CIM design. Generated design solutions indicate the method is successful for developing representations for CS, and in this way, initializing such processes.

Keywords: Design representations, design complexity, axiomatic design.

1 INTRODUCTION
Research in Computational Synthesis (CS) studies algorithmic procedures to automate the generation of designs. The idea here is that by combining "low-level" building blocks "high level" functionalities can be achieved. CS methods vary from straight forward implementation of artificial-intelligence (e.g. [1]), constraint solving (e.g. [2]) and optimization techniques (e.g. [3]) down to much more specialized approaches, as for example engineering shape grammars, function-based synthesis methods and kinematic synthesis, described in [4].
Figure 1 shows a flowchart with the general processes a CS System (CSS) should reassemble in order to automate the generation of design solutions [5]. Firstly, the design problem is formulated, which in engineering is done by declaring variables and constraints, and by constructing objective functions. This information is assembled into representations (building blocks) that support the algorithms to generate candidate solutions. Candidate solutions satisfy all constraints of the problem independent of how well the goal is achieved. An evaluation step analyses the results by calculating its performance and decides whether to accept, adjust or reject a candidate solution. Guidance drives the generation process in a given direction to improve the generated solutions.
Once a design problem is formulated, the first challenge encountered when developing a CSS is the definition of the building blocks, or representations, upon which the candidate designs can be generated. This is motivated by the fact that representations depend on the characteristics of the problem formulation as well as on the desired level of detail of the solutions. As indicated in [5], the following guidelines should be considered when developing representations for CS:

- An increase in the number of representations generally leads to an exponential expansion of the solution space.
- Representations have to express the relevant characteristics of a design while allowing room for modeling details. By doing so, the generation process can focus on designing concepts while the guidance process can concentrate in optimizing candidate solutions.
- By building-in design constraints in the representations, generating infeasible solutions is easily avoided. However, bringing excessive number of constraint results in too limited solution spaces.

In addition to these remarks, the complexity of a design problem also relates to its representations [6]. This follows from Suh's Axiomatic Design Theory (ADT) [7], where design complexity is defined as the measure of uncertainty in achieving the Functional Requirements (FR) of a system within its specified range of Design Parameters (DPs). For example, designing a building with 3D CAD

wireframe features is more complex than designing it with 3D CAD parametric features. This is because wireframe features do not allow modeling solids, and have a higher measure of uncertainty.
A great diversity of representation schemes can be found in CS design literature. While some representations are specific for one domain (e.g. super quadrics for shape design [8]), other representation schemes, like grammars and parametric models, have proven to be of a more generic character. For example, design grammars have been successfully used in computational synthesis methods for shapes design (e.g. [9, 10]), product design configurations (e.g. [11]) and conceptual design (e.g. [12, 13]). However, few methods describe the development of representations at the hand of the characteristics of a given design problem, as it is argued in [5]. In this context, the purpose of this paper is to present *from how much to how many* as a method to aid the development of representations suited for CS. The method does so by emphasizing on the management of design complexity, which follows from the argumentation presented by Campbell et al [14]: "the type of basic building blocks used for design generation inherently affects the complexity and variation in the candidate designs which is in turn reflected in the effectiveness and time of the search process". The authors believe that such types of methods in CS are analogous to discretization methods in Finite Element Method (FEM) as both serve to translate human representations into models suited for computational processes. Moreover, they can boost up CS technology by supporting preprocessing operations just as discretization methods have done for FEM [5]. The focus in this paper is set on artifactual routine design problems, which proceed within a well-defined space of functions, behaviors and structures [8].

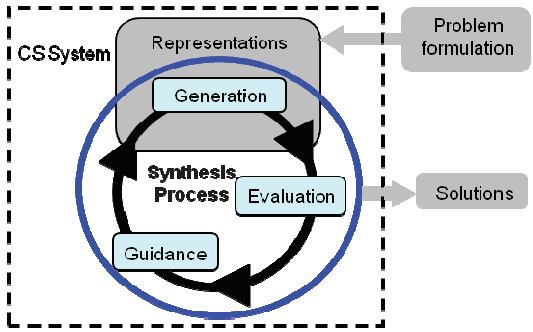


Figure 1: Computational Synthesis model [5].

This paper is structured in five parts. Section 2 presents the foundations firstly by introducing complexity management in ADT, and secondly by summarizing a framework for structuring artifactual routine design. Section 3 introduces the design of Cooling systems for Injection Molding (CIM) as the case study used in this paper. Section 4 presents *from how much to how many.* Firstly, the sources of complexity in artifactual routine design are explained at the hand of ADT and the structuring framework presented in Section 2. Secondly, the strategies used for dealing with complexity are introduced: (1) identify FRs and DPs, (2) reformulate FRs in terms of DPs, and (3) separate into smaller problem chunks that result from decoupled and uncoupled Design Structuring Matrix (DSM). Section 4 finishes by presenting the resulting representations for CIM design as well as a brief description of the CS method for its automation. Section 5 concludes that *from how much to how many* provides a systematic approach to determine, for a given design problem, the building blocks upon which design solutions can be generated. By doing so, the initialization of a synthesis process is supported, which according to Cagan et al [5] has not received enough attention in literature.

2 FOUNDATIONS

From how much to how many is founded on the notions of design complexity management stated in Axiomatic Design Theory (ADT), which are shortly summarized in Section 2.1. Section 2.2 summarizes the framework presented in [15] for structuring and modeling of artifactual routine design. This framework is used in this paper to formulate a design problem in a systematic way, as well as to identify its sources of complexity.

2.1 Complexity in Axiomatic Design Theory

According to Suh [7], complexity in design has been studied from two different perspectives: the physical domain and the functional domain. In the physical domain –which includes most engineers, physicists and mathematicians– complexity is seen as an inherent characteristic of physical things, including algorithms, products, processes, and manufacturing systems. This kind of thinking leads to the idea that systems with many parts are inherently more complex than those with less. In the functional domain, complexity is seen as a relative concept that evaluates how well we can satisfy "what we want to achieve" with "what is achievable". This view is used in this paper to explain the idea of moving design problems from *how much* to *how many*.

ADT is based on the hypothesis that there are fundamental principles that govern good designs [16]. Its two founding axioms are:

1. Maintain the independence of the Functional Requirements (FRs)
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:

$$FR = [A]DP \tag{1}$$

where A is the Design Structuring Matrix (DSM) of the problem. Depending on the DSM, 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 2. In a decoupled design, shown in Equation 3, the DPs have to be solved in a particular order so that FRs are achieved. In uncoupled designs (Equation 4), the FRs are independent from each others, and no particular order is required for solving the DPS.

$$Coupled: \quad \begin{Bmatrix} FR1 \\ FR2 \end{Bmatrix} = \begin{bmatrix} x & x \\ x & x \end{bmatrix} \begin{Bmatrix} DP1 \\ DP2 \end{Bmatrix} \tag{2}$$

$$Decoupled: \begin{Bmatrix} FR1 \\ FR2 \end{Bmatrix} = \begin{bmatrix} x & 0 \\ x & x \end{bmatrix} \begin{Bmatrix} DP1 \\ DP2 \end{Bmatrix} \tag{3}$$

$$Uncoupled \quad \begin{Bmatrix} FR1 \\ FR2 \end{Bmatrix} = \begin{bmatrix} x & 0 \\ 0 & x \end{bmatrix} \begin{Bmatrix} DP1 \\ DP2 \end{Bmatrix} \tag{4}$$

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 function of time, it is regarded as a system with time-dependent complexity. When the range does not change as function of time, it has a time-independent complexity. "Time" is used in a general sense, signifying 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 also classified in two types: time-dependent combinatorial complexity and time-dependent periodic complexity. The former occurs when the number of DPs explodes as function of time or if the system range continues moving the design range in time. The later occurs when uncertainties affect the system during a period. In this case, the system resets after each period causing the removal of arisen complexity. Time-independent imaginary complexity and time-dependent periodic complexity can occur only when we must satisfy many FRs at the same time, whereas time-independent real complexity and time-dependent combinatorial complexity can exist regardless of the number of FRs that must be satisfied at the same time.

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,
3. Transform a system with time-dependent combinatorial complexity into a system with time dependent periodic complexity

2.2 Design problem structure and model

According to [15], routine design problems can be structured and formulated in the following terms:

- embodiment elements and scenario elements that describe the initial state of the design,
- objective function, performance indicators and analysis relations to assess the goal of the design,
- topology relations and physical coherence constraints to indicate the set of logical states that have to hold for the design artifact to exist,
- confinement constraints that limit the values embodiment, scenario and performance descriptions are allowed to reach.

The following definitions apply:

- Element: is a class description of a design artifact component.
- Descriptions: characterize an element class by representing its attributes in the form of variables.
- 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 those descriptions used to express and assess the artifacts behavior, and are calculated using analysis relations.
- Analysis relations: use known theories, for instance the laws of physics or economics, to model the interaction of the design artifact with its environment and to predict its behavior.
- Topology relations: define the configuration of embodiment and scenario elements by means of relations expressing convergence, connectedness and continuity.
- Objective function: weighs and adds the performances of the design into one general indicator.

3 CASE STUDY

3.1 Design of CIM

Injection molding is an important manufacturing technique for producing plastic parts. This technique consists of injecting a hot polymer into the impression of a mold. Here, the plastic is cooled down to a solid state by a series of cooling channels drilled into the mold. The cooling stage is of great importance in the injection molding process, as it affects the productivity and the quality of the final part. The design process of CIM is divided into three successive processes [17]:

1. Preliminary design: consists of determining the possible locations of cooling channels in the vicinities of hot spots. At this phase, the channels do not form a circuit.
2. Layout design: consists in connecting previously positioned channels into circuits to assure a coolant is able to flow through it. Inlet and outlet channels are included to exchange the coolant with external cooling devices.
3. Detailed design: Optimizes cooling channels position to minimize warpage and thermal residual stress by applying small changes to the channels positions.

This paper focuses on preliminary and layout design, as it is in these phases where the cooling concepts are generated.

3.2 Computational Synthesis of CIM

Little research in CIM has focused on automating its design process. Although many papers explain optimization techniques of cooling channels placement, only the investigation carried by Li [18, 19, and 20] targets the generation of the candidate solutions. His method consists of decomposing the part geometry into several predefined shapes. Subsequently, three techniques are collaboratively used to generate candidate solutions, namely, case-based design, graph-based search and heuristic search. Case-based design maps the shape features on predefined solution, obtaining a preliminary design, which is capture in a graph model. A graph based transversal algorithm is employed to search for candidate cooling circuits. Finally, a heuristic search develops the candidate solutions into layout designs that contemplate tentative manufacturing plans.

3.3 Design problem formulation

The formulation of CIM has been done according to the structuring framework previously explained in Section 2.2. Figure 2 graphically represents the problem by showing the elements by nodes and

relations by arcs. Labels are used to specify the models of the elements and relations. For explanatory reasons, not all descriptions and relations have been included.
As the figure shows, *Channel* is regarded as embodiment element, while *Mold Part* and *Plastic Part* are regarded as scenario elements. The goal of the design, expressed by the objective function, is to minimize the time for cooling the plastic part and to minimize the temperature differences in the plastic part. Cooling time and temperature distribution are therefore regarded as performances.
Physical coherence, topologic and analysis relations are shown as arcs connecting the elements, with labels specifying the relations. Analysis is represented by the equation of Laplace, and allows the calculation of the performances (temperature distributions and cooling time). Detailed information on the analysis methods can be found in [17]. Topology relations specify that elements *Channel* are connected to each other inside the mold and are not allowed to share its space with the element *Plastic Part*. Physical coherence constraints are used to define: (a) the minimum distance between *Channel* and *Plastic Part* (PCC-1), (b) the minimum distance between *Channel* (PCC-2), (c) the diameter values of the *Channel* (PCC-3) and (d) the allowed distance between *Channels* and the *Mold Parts* (PCC-4). In [17], knowledge from expert designers has been used to formulate these three quantities as a function of the plastic part thickness. Furthermore, other physical coherence constraints, such as non-drillable surfaces and inlet/outlet surfaces, are also described. As the model is used for explanatory reasons, the latter have been kept out of the formulation.

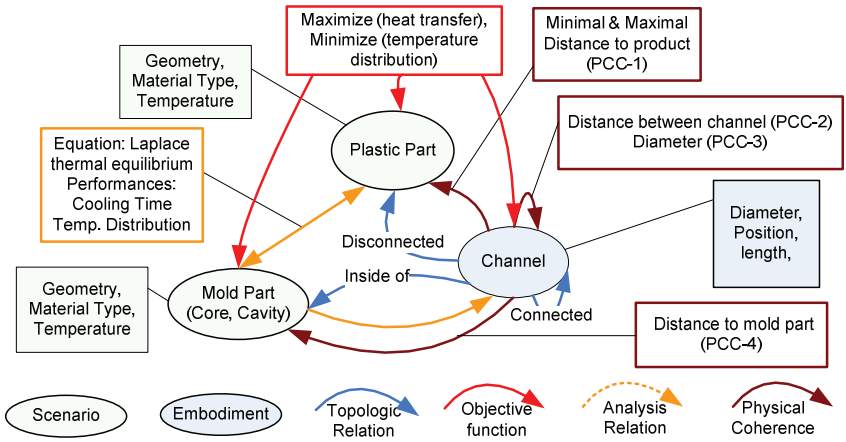


Figure 2: Semantic Networks for cooling layout design.

4 *FROM HOW MUCH TO HOW MANY*

In this paper, *from how much to how many* is presented as a method to develop representations of design problems for CS by applying complexity management strategies. In terms of ADT, this is defining the DPs (and its ranges) of the problem as well as its relations with the FRs, such that design complexity is minimized. Section 4.1 presents the sources of complexity in artifactual routine design in the terms of the problem formulation in Section 2.2. The strategies for complexity management upon which the method is based are introduced here as well. *From how much to how many* consists of the three steps shown in Figure 3, which are both applied in the functional domain (Section 4.2) and in the physical domain (Section 4.3). First, an identification phase determines the space of FRs and classifies the types of DPs involved in the problem. Secondly, a reformulation assembles the obtained FRs and DPs into a new decomposed problem model. Finally, the decomposed problem is separated into several problem chunks, whose instantiation order is determined by the resulting DSM.

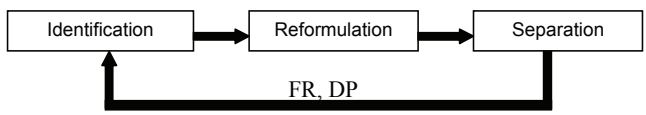


Figure 3: Steps in from how much to how many

4.1 The strategies in complexity management

Time-independent imaginary complexity

In routine design, it is common to cope with problem formulations with no explicit FR. In some cases, this leads to the apparent idea that the same DPs are related to different FRs, and as such, that the problem has a coupled DSM (Equation 2). Therefore, a more intensive search process to find feasible solutions is required. In order to avoid such states, the FRs of the problem have to be identified and related to the problem's DPs, which is done in the identification step of the method. Then, the problem can be reformulated in terms of the emerging relations between FRs and DPs, as done in the second step of the method. By doing so, the imaginary component of the complexity can be managed.

Time-independent real complexity

This appears when the ranges of the FRs are not in agreement with the DPs. By separating the design problem into a sequence of problem chunks, the range of the design can formulated as a function of the DPs, and by doing so, focus the search processes of each FR. This is achieved in the separation step of the method for both the functional and the physical domain.

Time-dependent combinatorial complexity

This occurs when the design problem consists in generating, for example, complex topologies or complex shapes where 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 DSM 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. In the method, this complexity is tackled by introducing periodicity in the DPs at the separation step of the physical domain. By doing so, the system becomes more deterministic and therefore requires less information to find a solution.

4.2 Functional Domain

In the functional domain, *from how much to how many* consists of identifying functional elements, identifying its instantiation order and encapsulating the resulting structures into new problem formulations. Here, each functional element is considered as one high level DP related to one (or a group of) FR(s).

Identification

Identification in the functional domain allows obtaining the functional map of the artifact being designed. This is achieved by:

1. Listing the functions of each of the elements involved in the formulation.
2. Elements undergoing more than one non-additive function (independent functions) are split into new element definitions: one element for each function.
3. Formalizing how one element's function acts upon other elements.

Consider the case of injection molding cooling systems. To decompose the element Channels we start by listing its functions:

- Function 1: to cool down the melt. Channels are placed close to the part geometry and arranged such that heat is transferred in a homogenous manner from the melt to the coolant flowing through it.
- Function 2: to transport the coolant. Coolant is transported between channels absorbing heat (function 1) to constitute cooling circuits.
- Function 3: to exchange coolant with the environment. Channels are used to connect the cooling circuits with the external surface of the mold.

As these functions are not additive but complementary, the element channel is decomposed into three different elements: (a) *Absorber channels* to fulfill function 1; (b) *Exchanger channels* to fulfill function 2; (c) *Connector channels* to fulfill function 3. Figure 4 presents the resulting channels and their functions assembled in a model. As shown, the new element *Absorber channel* applies its function to the scenario element *Plastic Part*, *Connector channels* apply their function to the element *Absorber channel*, and the *Exchanger channel* applies its function to both the *Connector channel* and the *Absorber channel*.

Reformulation

The reformulation step consists in making a new problem formulation based on the functional elements that resulted from applying the previous step. To do so, the new topology relations among functional elements are formalized. In addition, the topology relations of the mother elements are inherited by the new elements. The direction of the emerging topologic relations among functional elements equals the direction in which functions are applied among them. This follows from the principle that the direction in which functions are applied expresses how one element instantiation is limited by the previous instantiation of others. The resulting map of elements and topologic relations can be seen as the set of syntactic rules indicating the constraints imposed in the elements instantiation order.

In Figure 5 the reformulated CIM is presented. The figure shows both the functional elements and the topology relations that have emerged. Here, each channel element type can be connected to another channel element of the same type (relations 3, 5 and 9). *Connector channel* elements can be connected to *Absorber channels* (relation 12). However, the relation does not work in the opposite direction. This is because *Connector channels* function is constrained by the previous existence of an arrangement of *Absorber channels*, as indicated in Figure 5. A similar situation occurs for the relation between *Exchanger channels* and the partial circuits formed by the arrangement of *Absorber* and *Connector channels* (relations 6 and 8): the existence of the former is constrained by the previous existence of the latter.

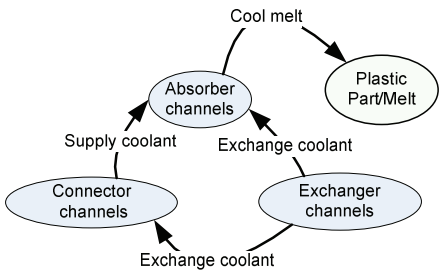


Figure 4: Functional elements cooling layout design

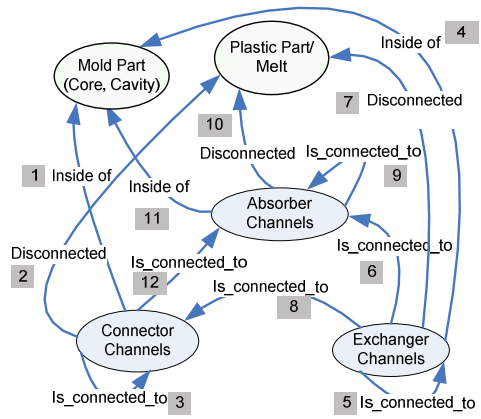


Figure 5: Decomposed cooling layout design

Separation

Now that all emerging relations have been formalized, it is assessed whether the design problem can be separated into smaller chunks. According to ADT, this is possible if the DSM is uncoupled or decoupled. Therefore, the separation step consists in assembling the DSM of the reformulated problem and deriving an appropriate instantiation order that manages the time-independent imaginary complexity of the system. In [21] a method is reported to define the instantiation order when the decomposition results in many elements that have multiple relations.

For the case of CIM, a DSM is assembled by considering the relations between the FRs and the DPs shown in Figure 5:

$$\begin{matrix} FRs \\ \begin{Bmatrix} Cool\ Melt \\ Transport\ Coolant \\ Exchange\ Coolant \end{Bmatrix} \end{matrix} = \begin{matrix} DSM \\ \begin{bmatrix} 9 & - & - \\ 12 & 5 & - \\ 6 & 8 & 3 \end{bmatrix} \end{matrix} \begin{matrix} DPs \\ \begin{Bmatrix} Absorber\ Channel \\ Connector\ Channel \\ Exchanger\ Channel \end{Bmatrix} \end{matrix} \quad (5)$$

As shown, the DSM is decoupled and square, and can be separated into three independent problem formulations. The first is shown in Figure 6(a). Here, *Absorber channels* are first instantiated by taking into considerations the relations shown in the figure. In a similar manner, *Connector channel* are

designed considering the *Absorber channel* as scenario elements, shown in Figure 6(b). The formulation in Figure 6(c) is assembled for the design of *Exchanger channels*.
If the number of FRs does not match the number of DPs, the resulting DSM is not square. In such a case, a DSM of the relation among the DPs is better suited for determining the appropriate instantiation orders.

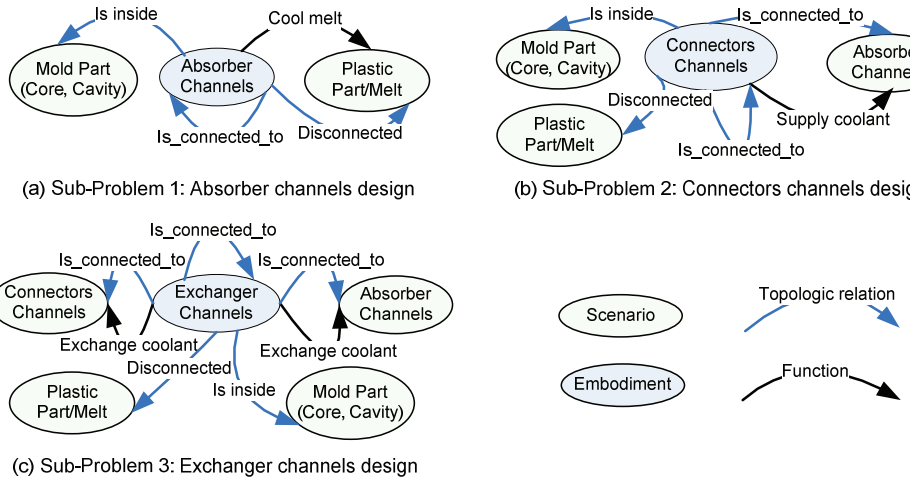


Figure 6: Resulting decomposed problem formulations.

4.3 Physical Domain

The application of *from how much to how haw many* in the physical domain is based on the design descriptions classification presented in [15]. According to this research, descriptions in artifactual routine design can be categorized into five attributive dimensions:

- Parameter: models the properties applicable to a whole element. These can be of different nature, as for example numeric, symbolic, logic, predicate and combinations among them.
- Space: describes the position of the elements and depends on the chosen coordinate system (Cartesian, Cylindrical of Spherical) and the dimensions of interest (1D, 2D, or 3D).
- Field: uses parameters and geometric vectors to describe properties that hold for specific regions of the elements. Fields are specified together with an incident zone, which is the spatial place where the field influences an element. An incident zone can be a volume, an area, a line or a point.
- Shape: describes the form of an element or groups of elements. Commonly used models are based on geometry and shape graphs.
- Topology: uses topology relations to describe the disposition of elements in design. Cardinality is used to measure the number of times an element is instantiated in the topology.

According to [15], each dimension is represented by different types of mathematical models and requires different algorithms to generate solutions. *From how much to how many* uses this classification to decompose elements in different primitives, where each primitive encapsulates attributes that correspond to one dimension. By doing so, building blocks with focused search ranges are found. Furthermore, the method manages imaginary complexity by firstly integrating physical coherence constraints into the resulting primitive elements, and secondly by introducing periodicity in the system (as defined in Section 2.1). Subsequent sections explain how to apply the method in the physical domain.

Identification

In this step, descriptions are categorized around the five attributive dimensions. In the case study of CIM design, the identification of primitive elements is done by analyzing the type of description used to model them. As the elements *Absorber channels*, *Connector channels* and *Exchanger channels* are modeled with the same descriptions, the identification has been generalized in an element *Channel*,

which represents either one of the three before mentioned types. *Channels* have attributes in two domains:

- Space: models the position of a channel inside the mold. The position is modeled by three coordinates in the Cartesian coordinate system.
- Shape: models the geometry of a channel by the descriptions diameter (D) and length (L).

The scenario element *Mold Parts* can be modeled in different dimensions: Shape, Fields and Parameter. As *Mold Parts* are not subject of design and its shape and parametric representation are fixed, it is chosen to use the representational dimension Fields.

Reformulation

Reformulation consists of encapsulating each of the identified description dimensions into a new primitive element and formalizing the physical coherence constraint among them. In this way, a new problem formulation is obtained.

For the case of CIM shown in Figure 7, the following primitive elements are derived:

- Points: encapsulate the Space dimension of the representations. A *Point* contains a description about its position *(x,y,z)*. *Points* are related to the scenario elements by the physical coherence constraints PCC-1 and PCC-4. A set of *Points* indicates the path followed by the channel.
- Segment: encapsulate the Shape dimension. A *Segment* contains the descriptions diameter (D) and length (L). The physical coherence relation PCC-2 determines the minimum allowed distance between two points to avoid break of the mold, while the relation PCC-3 determines the diameter of the channel as function of the distance between a *Point* and *Mold Part* or *Plastic Part*.

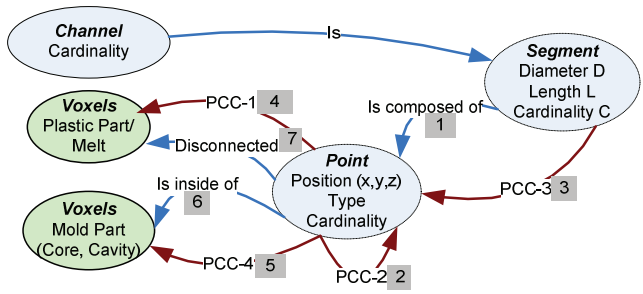


Figure 7: Primitives in functional element "Absorber Channel"

Given that the three functional elements are decomposed into the same primitives, the following notation is used to differentiate among them:

- *Absorber channels* have primitives *Blue Points and Blue segments.*
- *Connector channels* have primitives *Green Points and Green segments.*
- *Exchanger channels* have primitives *Brown Points and Brown segments.*

These six primitive elements define the information contents of the DPs of the problem.

On the other hand, the element *Mold Parts* is also decomposed into primitives. Here, the dimension of concern is Fields and, as such, voxel elements are chosen. Voxels are cubic units and can be used to describe a mold's shape and position.

Separation

This step consists of separating the reformulated problem into smaller chunks by assessing the DSM of the resulting problem. As the interest here lays in the physical domain, the DSM has to describe the interrelations among the primitive elements, or DPs, of the problem. For the case of CIM design, the resulting DSM becomes:

$$\begin{Bmatrix} Points \\ Segments \\ MoldPart \\ PlasticPart \end{Bmatrix} = \begin{bmatrix} 2 & 0 & 5,6 & 4,7 \\ 3,1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} Points \\ Segments \\ MoldPart \\ PlasticPart \end{Bmatrix} \quad (6)$$

According to this, *Points* have to be instantiated firstly and *Segments* secondly.

Now that the problem has been separated, the combinatorial aspect of the complexity is tackled. In the case study of CIM, combinatorial complexity arises from the uncertainty of how many Channels are required and how they should be connected. This is managed by predefining a 3D grid of points without "color" within the voxel mesh. By doing so, periodicity is brought in the system as suggested in ADT. By setting the distance between *Points* according to relation PCC-2, imaginary complexity is also removed from the system, as relation 2 can be removed from the DSM shown in equation 6. In order to integrate the physical coherence constraints PCC-1 and PCC-4 into DPs, one extra type of *Points* is declared, namely, *Black Points. Black points* define the positions where channels cannot be placed to avoid mold breakage. In Table 1, the logic relations that determine the colors of *Points* are presented.

Table 1. Logic relations determining the color of Points

Point	*ID*	*Logic relation with scenario elements*
Blue	C1	Surrounded by [(core voxels) OR (cavity voxels)] AND [product voxels]
Green	C2	Surrounded by [(core voxels) OR (cavity voxels)]
Brown	C3	Surrounded by [(core voxels) OR (cavity voxels)] AND [exchanger voxels]
Black	C4	Surrounded by [non drillable voxels]

4.4. Results

Figure 8 shows the resulting problem formulation of CIM after applying *from how much to how many*. As shown, a hierarchical model consisting of three levels of abstraction (*Points* design, *Segment* design and *Channel* design) is obtained. Each element in the model is considered a DP, while the topology relations among them represent the syntactic rules determining their structure. Physical coherence constraints represent semantic rules assuring no physical inconsistencies occur. The complexity of the system has been reduced by specifying the ranges of each DP: *Points* as function of the scenario elements *Plastic Part* and *Mold Parts*, *Segments* as function of *Points,* and *Channels* as function of *Segments*. Furthermore, the order of instantiation was determined for each abstraction level at the hand of its DSM. In combination with a CS method (e.g. A-design [22]), the representations in Figure 8 can be used for automating the design of CIM.

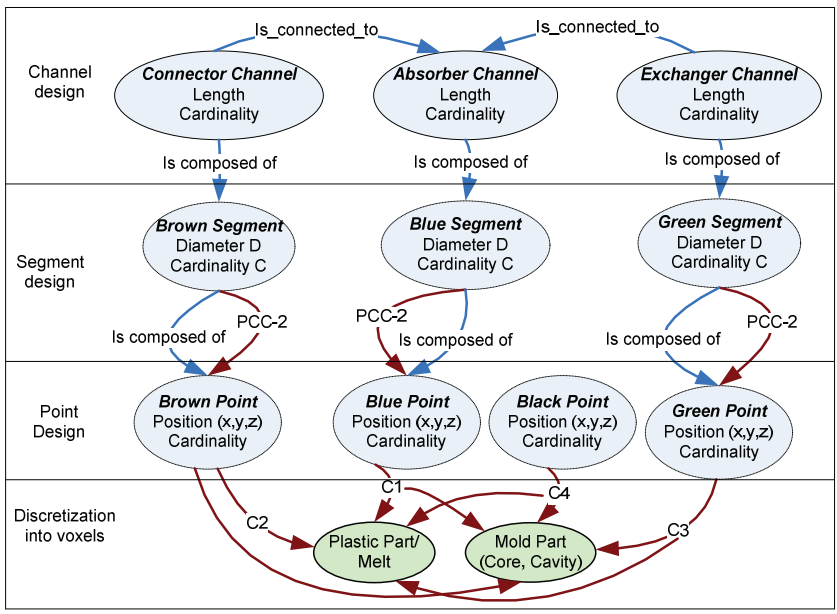
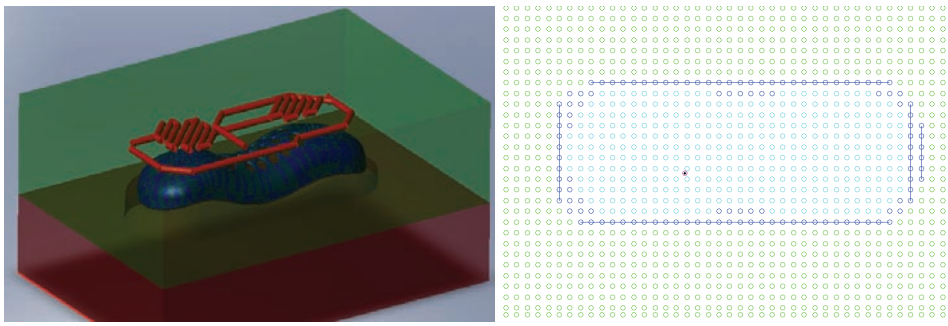


Figure 8: Decomposed cooling design problem

In this paper, these representations have been implemented into software with the aim of assessing their feasibility in supporting the computational synthesis of CIM. A generation and test algorithm is used for the generation of solutions, while a qualitative analysis is used for evaluating the obtained solutions. This is fully described in [23], where an overview of the implementation is presented. At the hand the instantiation orders resulting from applying the method, the CS of CIM consists of:

1. Making a voxel model of the solid parts *Mold Part* and *Plastic Parts*.
2. Generating a mesh of *Points* on top of the voxel model and attributing color to the points according to the relations shown in Table 1.
3. Generating *Absorber segments* by assembling pairs of blue points.
4. Connecting *Absorber channels* with *Connector channels.* The latter generated by assembling pairs of green points.
5. Connecting *Absorber channels* and *Connector channels* with *Exchanger channels*. Again, the latter generated by assembling pairs of green points.

Figure 9(a) shows one automatically generated CIM solution of a telephone mold, while Figure 9(b) shows the grid of *Points* and *Absorber channels* in one section of the mold.



(a) CIM candidate solution *(b) Example of generated Absorber Channels*

Figure 9: Images of CSS for CIM design

5 CONCLUSIONS

From how much to how many is presented as a method to aid the development of representations for computational synthesis by applying principles of complexity management stated in Axiomatic Design Theory (ADT). The goal is to support the development of Computational Synthesis Systems (CSS) by providing a systematic approach to determine, for a given design problem, the building blocks upon which design solutions can be generated. Results from applying the method to the design of Cooling systems for Injection Molding (CIM) indicates it is successful in supporting the initialization of a synthesis process, which according to Cagan et al [5] has not received enough attention in literature. Further experimentation is required to prove the effectiveness of the method.

ACKNOWLEDGEMENT

The authors gratefully acknowledge the support of the Dutch Innovation Oriented Research Program 'Integrated Product Creation and Realization (IOP-IPCR)' of the Dutch Ministry of Economic Affairs.

LITERATURE

[1] Krishnamoorthy, C. S., *Artificial Intelligence and Expert Systems for Engineers*, 1996 (CRC Press Inc).

[2] Schotborgh, W.O., Kokkeler, F.G.M., Tragter, H., Bomhoff, M.J. amd Houten, F.J.A.M. van. A Generic Synthesis Algorithm for Well-Defined Designs. In: *CIRP Design Conference on Design Synthesis,* 2008.

[3] Cagan J., Grossmann I.E. and Hooker J. A Conceptual Framework for Combining Artificial Intelligence and Optimization in Engineering Design. *Research in Engineering Design*, 1997, 9, pp. 20-34.

[4] E. K. Antonsson and J. Cagan. *Formal Engineering Design Synthesis*, 2001 (Cambridge

University Press).
[5] Cagan J., Campbell M.I., Finger S. and Tomiyama T. Framework for Computational Design Synthesis: Model and Applications. *Journal of Computing and Information Science in Engineering*, 2005, ASME.
[6] Ameri, F., Summers, J., Mocko, G. and Porter, M. Engineering design complexity: an investigation of methods and measures. *Research in Engineering Design*, 2008, 19(2), 161-179.
[7] Suh N.P. Axiomatic Design Theory for Systems. *Research in Engineering Design*, 1998, 10, pp 189–209.
[8] Jaklic, A., Leonardis, A. and Solina, F. Segmentation and Recovery of Superquadrics Series, *Computational Imaging and Vision*, 2001, 20, pp 12-39.
[9] McCormack, J. P., Cagan, J., and Vogel, C. M. Speaking the Buick Language: Capturing, Understanding and Exploring Brand Identity with Shape Grammars, *Des. Stud*, 2004, 25(1),1-29.
[10] Orsborn S, Cagan J, Pawlicki R and Smith R. Creating cross-over vehicles: defining and combining vehicle classes using shape grammars. *Artificial Intelligence in Engineering Design and Manufacturing,* 2006, 20(3), 217–246
[11] Campbell M.I., Cagan J. and Kotovsky K. Agent-based synthesis of electromechanical design configurations. *Journal of Mechanical Design*, 2000, vol. 122/61.
[12] T. Kurtoglu, A. Swantner, and M. I. Campbell. Automating the Conceptual Design Process: From Black-box to Component Selection. In *Design Computing and Cognition '08*, 2008, pp. 553-572.
[13] Startling A.C. and Shea K. A Parallel Grammar for Simulation-Driven Mechanical Design Synthesis. In: *International Design Engineering Technical Conferences 2005*, Long Beach, 2005, 2(A), 427-426.
[14] M. I. Campbell and R. Rai. A Generalization of Computational Synthesis Methods in Engineering Design. In: *AAAI Spring Symposium Series*, Palo Alto, CA, 2003.
[15] Jauregui-Becker, J.M., Tragter, H. and van Houten, F.J.A.M. Structure and models of artifactual routine design problems for computational synthesis. *CIRP Journal of Manufacturing Science and Technology*, 2009, 1(3), 120-125.
[16] Suh, N.P. Complexity in Engineering. *Annals of CIRP*, 2005, 54(2), 581-598.
[17] Menges, Mohren. *How to make injection molds*, 1986 (Hanser Publishers).
[18] Li C.L., Li C.G. and Mok, A.C.K. Automated layout design of plastic mold cooling system. *Computer Aided Design*, 37, 645-662.
[19] Li CL and Li CG. Manufacturable and functional layout design of cooling system for plastic injection mould - an automatic approach. In: *Proceedings of the international conference on manufacturing automation*, 2004, pp. 47-54.
[20] Li, C.L. and Li, C. G. Plastic injection mould cooling system design by the configuration space method. *Computer-Aided Design*, 2008, 40(3), 334-349
[21] Jauregui-Becker, J.M., Wits, W.W. and van Houten, F.J.A.M. Reducing design complexity of multidisciplinary domain integrated products: a case study. In: *Proceedings of the 41st CIRP Conference on Manufacturing Systems*, 2008, Vol 1, pp. 149-154
[22] Campbell M.I., Cagan J. and Kotovsky K. The A-Design approach to managing automated design synthesis. *Research in Engineering Design*, 2003 14: 12-24.
[23] J. M. Jauregui-Becker, H. Tragter and F. J. A. M van Houten. Toward a bottom-up approach to automate the design of cooling systems for injection molding. *Computer Aided Design and Applications*, 2009, 6(4), 447-459.

Contact: Juan M. Jauregui-Becker
University of Twente
Faculty of Engineering Technology
Laboratory of Design, Production and Management
P.O. Box 217, 7500 AE Enschede
The Netherlands
T: +31 53 489 2266
F: +31 53 489 3631