INTERNATIONAL DESIGN CONFERENCE - DESIGN 2010
Dubrovnik - Croatia, May 17 - 20, 2010.

CONTEXT-FREE GRAMMAR BASED RULES FOR COMPONENT-LEVEL PRODUCT STRUCTURE MODELLING

I. Midžić, T. Stanković and D. Marjanović

Keywords: chromosome product model, function-means tree, product sub-functions, product component structure, context-free grammar

1. Introduction

Function-Means tree used for concept generation and top-down function decomposition is an object oriented representation of possible solutions attached to its suitable function or sub-function. Mean or a solution element that designers state for a possible mean of realization of the function or sub-function can be any product configuration entity; from organ to feature, including modules, sub-assemblies, parts and so on. One of the main advantages of Function-Means approach is that designers are able to identify, collect and depict alternative solutions while the overall tree structure containing information about those possible solutions and interrelations to function structure elements present in the tree. In concept generation purposes, this method may help designers not to oversee some solution to a function, but could enhance designers to utilise before defined function-mean alternative in a new product development design process or project. When this scenario occurs in cases of complex products, Function-Means method could result to wide tree structure, numerous branches and levels and consequently be difficult to maintain, search through, analyze and extract conclusions from.
The knowledge about possible solutions and functional decomposition relations between tree elements is tended to be reused or reassigned forming a new tree structure, for a new product concept generation uses. Variant product solutions in form of component structure are represented by a tree structure with the Function-Means tree object oriented characteristics. The question emerging concerns a great amount of information extracted from multiple Function-Means trees and the complexity and numerousness of detailing of those information or uttermost, expected and not very useful solutions for the designer when aiming to derive innovative solutions.
Designer's input on the matter of combining sub-trees for generating possible principle solutions (concepts) withholding the information on interrelations between nodes and its context may be substituted by an expert systems application. Expert systems [Parsaye, Chignell, 1988] work with knowledge stored in a knowledge database. Knowledge representation and structure employed by a rule-based expert system is defined by rules in IF-THEN form. Rule-based system assisting the designer additionally can offer optimizing and concept evaluating plausibility. In this paper, rule formalization is proposed for representing a simplified version of Function-Means tree structure and the impact of transposing a tree structure to rule inscription on grammar definition is analysed.
Product-Function-Component tree derived from a Function-Means tree is developed in this work and used for representing relations between function structure and component structure elements [Malmqvist, 1995] according to "the application of Theory of Domains to product modelling" - a Chromosome Model [Jensen, 1999]. Function-component relations substitute the "CorrespondsTo relationships" between the Function-Means tree and the Chromosome Model domain constituents

[Malmqvist, Schachinger, 1997]. In the Product-Function-Component tree, groups of components realizing the same function are identified at the bottom of the tree (leaves).
The main motivation for representing a Product-Function-Component tree by context-free grammar is introducing a formalized language for a future rule-based system. On a simplified example of a bicycle, we present the process of generating a component-level bicycle structure represented by grammar formalism and additional graph representation.
It is assumed that the demonstrated approach represents the initial steps for integrating rules on tree structure representation into a rule-based system under whose authority, grammar definition and rule representation is substantially utilized.

2. Motivation

Although grammars have never been the common field of interest concerning engineering design and related disciplines, inspiring work on the matter is to be found in the form of shape grammars where the derivation of rules describes the process of design evolution or creation when product's geometric or other features are considered [Hoisl, Shea, 2009]. Shape grammar formalism emerging from formal grammars first defined by linguists and dispersed in most programming languages, rely on the same foundations as any grammar-based approach. Reasoning on this matter and the motivation is to be found in wide variety of shape grammar utilisation in architecture, but also in design. Rules as the main constituents of grammar formalism are used to represent the available knowledge about the problem space and then used to generate designs [Agarwal, Cagan, 2000]. Some examples on how a certain design is generated by rule derivation demonstrate the potential of expressing variant principle solutions [Orsborn et al, 2006] and preserving the same what is in architecture and related disciplines considered style. Another adaptation of shape grammars can be found in form of product grammars [Chin, 2004] which, in a wider sense, are intended to be a mean of formalizing the knowledge that designers accumulate regarding the design of a certain product.
Variant product structures and design solutions can be presented by production rules in an intuitive and 'shallow' way as is stated by some authors [Pham, 1991]. Rule-based representation is submissive to extracting, combining and redefining new instances of rules, but since the definition of grammar is predisposed by vocabulary definition and disadvantage of the method concerning the necessity that rule writing is user-defined, rule restructuring mechanism also acquires prescribing by the user.
In this work, we explore the possibilities of representing a Product-Function-Component tree via context-free grammar rules and the dependencies of such structure to the Chromosome product Model for some future grammar-based principle solution variants generation. The realization of a product structure that is component-level based and represented by a tree at first was the main motivation for considering a Product-Function-Component tree. For the reasons of introducing a framework for grammar modelling for applications where rules could be separated from the old context and entered into a new one, derivation from the tree structure results context-free grammar rules.
An example of component-level product structure representation is to be found at [Chin, 2004] where an example of a product is subjected to product grammar methodology and intended for the automotive industry implementation.
Function-Means analysis combines knowledge on functions and means realizing them, where co-relation between functions and abstract/physical solutions are explicitly specified, the same way the Function-Component model combines abstract and concrete design features, although in a less detailed structure described in this paper for a bicycle and used to demonstrate rule writing and grammar definition modality.

3. Background

According to [Mortensen, 1999] it is possible to create four classes of design languages (process, function, organ and part languages) based on the artefact theories which allow gradual definition of a design thus functionality and the designer's reasoning can be captured. This is an analogy of the Domain Theory and Chromosome model viewpoint. Detailed relations between function-means analysis and Chromosome product Model is depicted in the work of [Malmqvist, Schachinger, 1997]. In the function-means tree, interdependent function decomposition and means assigning is performed until the functions can be associated with solutions that are constituents of the Chromosome Model; organs [Schachinger, 2004] or organs and components [Malmqvist, Schachinger, 1997]. The structure of technical systems can be considered through the concept of organs as function carriers representing the means by which the functions can be realized. Considering the technical systems with respect to its anatomy, organs represent sub-systems or sub-assemblies of the product depending on the level of detailing in the product structure. This is in many ways correspondent to the device-centric view [Chandrasekaran, Josephson, 2000] where the emphasis of the design process is on modelling the internal configuration of the artefact – its structure. In this work, organ modelling is substituted with components in order to identify groups of components realizing the same function. In this way, components are a direct solution of the means stated in the Function-Means tree and thus considered to be function carriers.

4. Product-Function-Component tree

Inspired by the way a Function-Means tree implements the interrelations with the function structure, the Product-Function-Component tree includes sub-functions decomposed from an overall function in the "verb + noun" form. Upper functions are further divided into lower functions or sub-functions depicting functional decomposition and equivalent to a Function-Means structure, components attached to each sub-function. Sub-functions are in the further text referred to as functions.

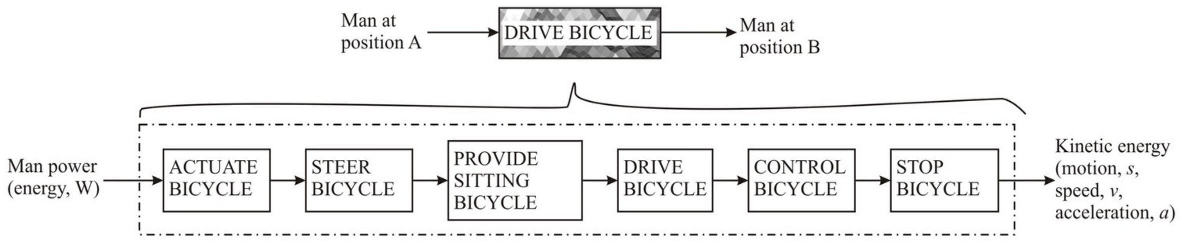

Figure 1. Overall function decomposition (bicycle)

Product-Function-Component tree is rooted with product on the top of the tree, and then a merge of functions to next tree-level is made. Since in this stage we know nothing about product sub-systems or subassemblies, i.e. we know nothing about product's component structure we assign functions to the second-tree level so they will substitute product's subsystems. Assumption made here is that a product can be broken down into subassemblies corresponding to the functions of the technical system. According to this, Product-Function-Component tree is created by merging two dependant tree structures.

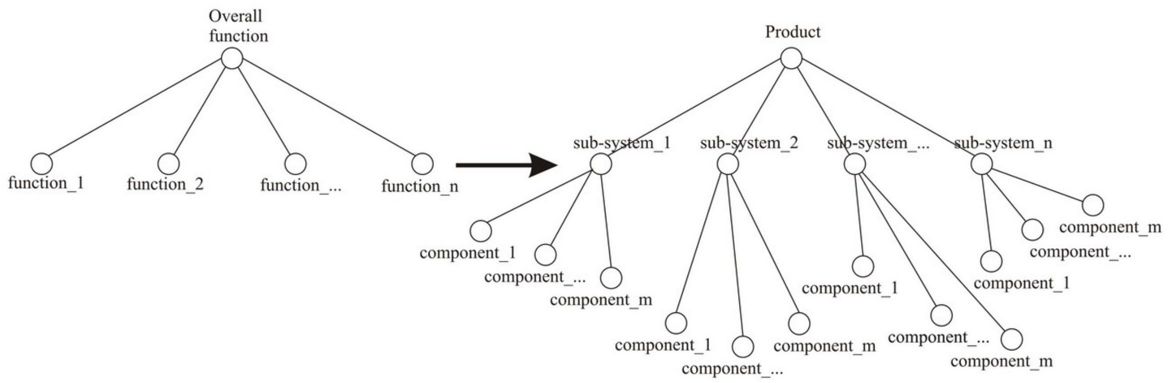

Figure 2. Merging of functions into Product-Sub-system-Component tree

Components are identified by product teardown analysis regardless to the part/assembly hierarchy.

Each component realizes one or more functions and this relation results a product component structure. Associative tree model of a technical artefact is structured as a wide three-level tree with a product on top, its functions on the next level and components that realize functions on the last level.

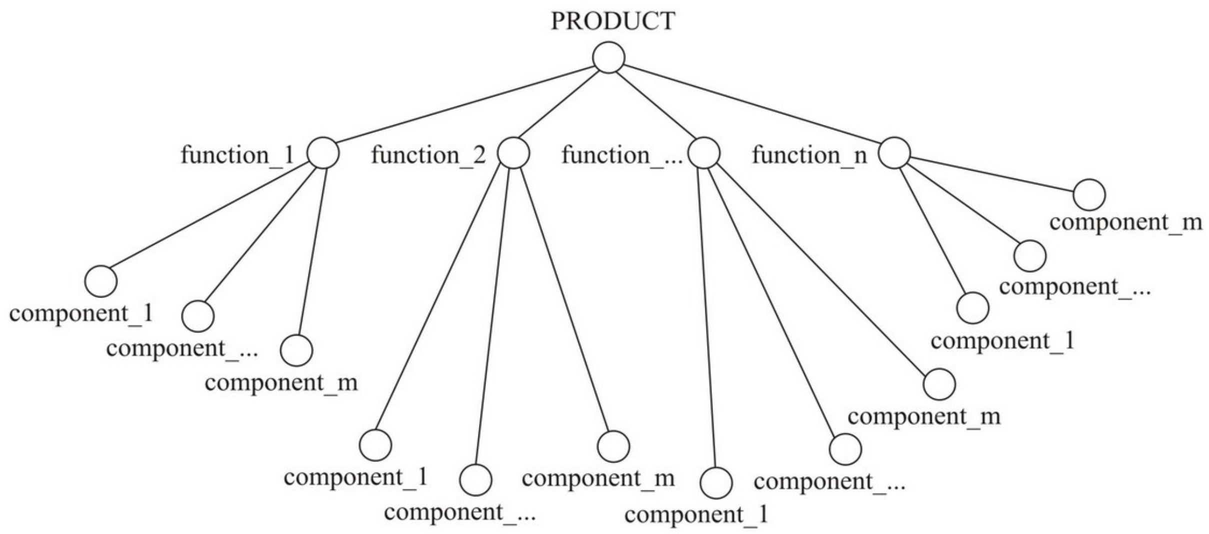

Figure 3. General model of a Product-Function-Component tree

This structure now provides the information on groups of product components realizing the same function.

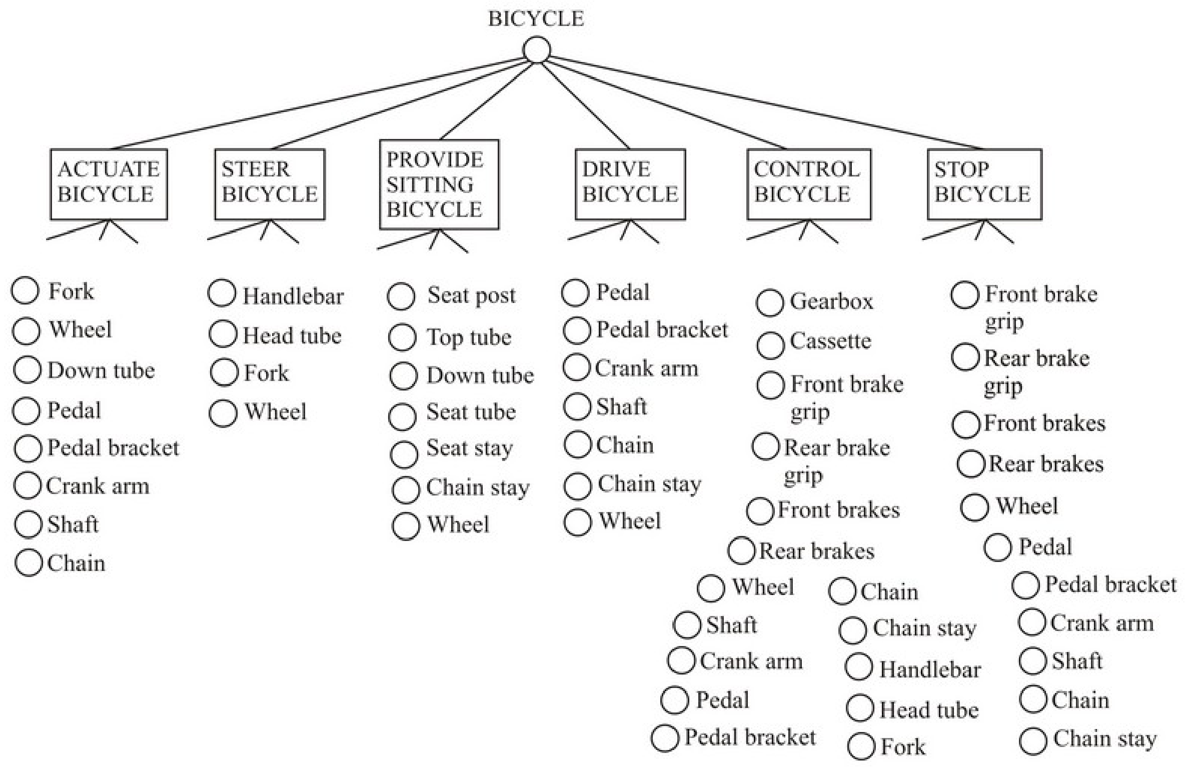

Figure 4. Bicycle Product-Function-Component tree

5. Representing Product-Function-Component tree via context-free grammar

5.1 Definition of grammar

Context-free grammar G of a formal language $L = L(G)$ is a 4-tuple $G = (\Sigma, V, S, P)$ [Jiang et al., 2000]. The vocabulary of the grammar consists of finite nonempty sets of terminals (Σ) and non-terminals (V) with $\Sigma \cap V = 0$. Terminal symbols or terminals are called the alphabet and by definition they are indivisible. Variables or non-terminals help defining the language. S is a designated variable called the start symbol and the elements of P are called rewriting rules or productions representing the recursive definition of a language. Rules consist of left-hand-side of the rule (head), the production symbol $\rightarrow$ and right-hand-side of the rule (body). In context-free languages rule P is:

$$\alpha \rightarrow \beta \tag{1}$$

where α consists of a single non-terminal or $|\alpha| = 1$. The body of the rule is:

$$\beta = (\Sigma \cup V)^{*} \tag{2}$$

Kleene closure (*) is an operation that denotes the set of every possible combinations of terminal and non-terminal symbols in the composed string of a formal language.

5.2 Terminal and non-terminal declaration

Functions are defined as non-terminals and components are defined as terminals. Rule generation is triggered by the initial element *I*. Initial element is selected from the set of non-terminals (functions).

$$I \rightarrow V \tag{3}$$

5.3 Defining rules

The strings of formal grammar are derived by applying rules from head to body ($LHS \rightarrow RHS$). This process can be illustrated by derivation of rules and a parse tree. In general, form of rules for function-component relation declaration is:

$$V \rightarrow V(\Sigma) \tag{4}$$

Derivation by rules

$$P = \{I \rightarrow V, V \rightarrow V(\Sigma)\} \tag{5}$$

with two rules in P and rule sequence (1,2,2,2) is

$$I \rightarrow V \rightarrow V(\Sigma) \rightarrow V(\Sigma)(\Sigma) \rightarrow V(\Sigma)(\Sigma)(\Sigma) \tag{6}$$

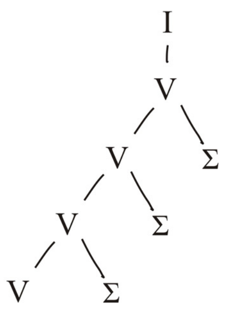

Figure 5. Parse tree of expression (7)

The final RHS of the derivation can be considered as $s(P) = \{V(\Sigma, \Sigma, \Sigma), \Sigma, \Sigma, \Sigma\}$ where function $s(P)$ is the multi-set of complete sub-trees of P [Muller-Molina, 2007].

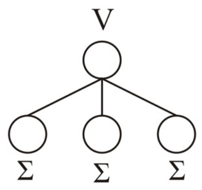

Figure 6. Representation of $s(P)$ function for expression (7)

If we choose a function from a set of V and proclaim it as the initial symbol, we could derive a multi-set for each function. In the following expressions we will illustrate grammar representation for the "actuate bicycle" function and components assigned to it.

$$\begin{aligned}
P = \{ & I \rightarrow actuate\_bicycle, \\
& actuate\_bicycle \rightarrow actuate\_bicycle\ (fork) \\
& \qquad |\, actuate\_bicycle\ (wheel) \\
& \qquad |\, actuate\_bicycle\ (down\_tube) \\
& \qquad |\, actuate\_bicycle\ (pedal) \\
& \qquad |\, actuate\_bicycle\ (pedal\_bracket) \\
& \qquad |\, actuate\_bicycle\ (crank\_arm) \\
& \qquad |\, actuate\_bicycle\ (shaft) \\
& \qquad |\, actuate\_bicycle\ (chain)\}
\end{aligned} \tag{7}$$

Rule sequence is (1,2,3,4,5,6,7,8,9) and the derivation of rules:

$$\begin{aligned}
& I \rightarrow actuate\_bicycle \rightarrow actuate\_bicycle\ (fork) \rightarrow actuate\_bicycle\ (wheel)(fork) \rightarrow \\
& \rightarrow actuate\_bicycle\ (down\_tube)(wheel)(fork) \rightarrow \\
& \rightarrow actuate\_bicycle\ (pedal)(down\_tube)(wheel)(fork) \rightarrow \\
& \rightarrow actuate\_bicycle\ (pedal\_bracket)(pedal)(down\_tube)(wheel)(fork) \rightarrow \\
& \rightarrow \ldots
\end{aligned} \tag{8}$$

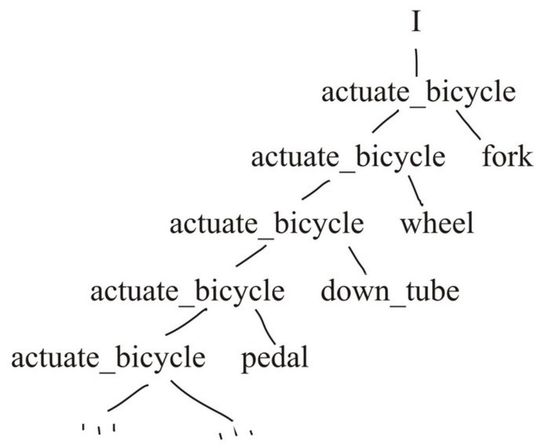

Figure 7. Parse tree of expression (9)

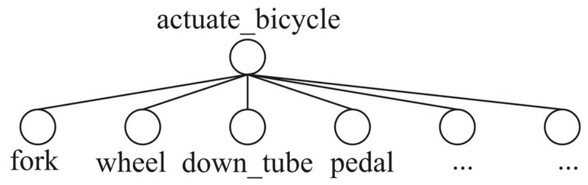

Figure 8. Representation of $s(P)$ function for expression (9)

The rest of the rules for the functions are made in the same way and the final representation of five $s(P)$ functions is equivalent to the function sub-trees in Figure 4. The final derivation of rules states that a function is realized by components assign to it.

5.4 Pursuing product grammar

Product-Function-Component tree developed for a bicycle contains all the components on the same tree-level. For example, it also contains five instances of the same component (wheel) in the same structure and other components are also multiplied. These inconsistencies are annulated by applying three product grammar methodology instructions [Chin, 2004] which include abstraction, substitution, replacement and standardization techniques performed by node replacement and avoiding overlapping of branches and nodes. Reference on how the product grammar allows for the synthesis of the different nodes and branches of the tree structure at the subsystem level to allow for innovation can be found at [Chin, 2004]. Actions while doing this are carried out on the tree structure making it a natural way of restructuring and reordering. Rule-based representation makes easier to search, store, extract, evaluate and reorder rule fragments, managing context-free and non-context free rule format, depending on the user-defined postulate.

6. Conclusions and future work

The contribution of the work consists of grammar rules determination from the simplified Function-Means tree representation in form of Product-Function-Component model developed for this paper purposes. Product-Function-Component tree structure combines knowledge about function-components relations and is correspondent to the Function-Means tree and Chromosome product Model terminology. Context-free grammar rules provide information on existing product's functions and components and the relation between them in a "Realize" or "IsSolution" relationship. Final representation is used for identifying groups of components realizing the same function, representing a principle solution of a function-means analysis and recognized as function carriers. Rule formalization is demonstrated on a simplified bicycle example.

Formal representation of knowledge often is portrayed as an open problem. Initial framework for introducing Function-Means tree representations into grammar formalism and rule definition for a rule-based system environment is been presented. The potentiality of describing, representing, storing, managing and generating possible principle solutions or variants and expanding the design alternatives supporting conceptual design in rule-based system environment is provided, but also pointed out that rule formalization and grammar definition, as well as validation and evaluation of concepts and reasoning mechanism support by the expert system are exigently depending on designer's decisions regarding rule derivation, concept evaluating criteria and perception of the design problem.

Acknowledgements

This research is part of funded project "Models and methods of knowledge management in product development" supported by the Ministry of Science and Technology of the Republic of Croatia.

References

Agarwal, M., Cagan, J., "On the use of shape grammars as expert systems for geometry-based engineering design", Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 14, pp. 431–439, USA, Cambridge University Press, USA, 2000

Chandrasekaran, B., Josephson, J.R., "Function in device representation", Engineering With Computers, 16, pp. 162–177, 2000

Chin, R., "Product Grammar: Constructing and Mapping Solution Spaces", MIT MS thesis, Massachusetts Institute of Technology, 2004

Hoisl, F., Shea, K., "Exploring the integration of spatial grammars and open-source CAD systems", International Conference on Engineering Design – ICED 2009, 2009

Jensen, T., "Functional Modelling in a Design Support System: Contribution to a Designer's Workbench", Department of Control and Engineering Design, Technical University of Denmark, Denmark, pp. 64-69, 1999.

Jiang, T., Li, M., Ravikumar, B., Regan, K.W., "Formal grammars and languages", 2000

Malmqvist, J., "A computer-based approach towards including design history information in product models and function-means trees", Proceedings of DTM – 1995, Boston, MA, USA, pp. 593–602, 1995

Malmqvist, J., Schachinger, P., "Towards an implementation of the chromosome model - focusing the design specification", International Conference on Engineering Design – ICED 1997, Tampere, 1997

Mortensen, N.H., "Design Modelling in a Designer's Workbench – Contribution to a Design Grammar", Doctoral Thesis, Department of Control and Engineering Design, Technical University of Denmark, 1999

Müller-Molina, A.J., Hirata, K., Shinohara, T., "A tree distance function based on multi-sets", 2007

Orsborn, S., Cagan, J., Pawlicki, R., Smith, R.C., "Creating cross-over vehicles: Defining and combining vehicle classes using shape grammar", Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 20, Cambridge University Press, USA, pp. 217–246., 2006

Parsaye, K., Chignell, M., "Expert systems for experts", John Wiley & Sons, Inc., 1988

Pham, D.T., "Artificial Intelligence in Design", Springer-Verlag London Limited, 1991

Schachinger, P., "Form synthesis: Interactive functions and aesthetic organs in the function-means tree", Norcode seminar, Tangby, pp. 1, 2004

Ida Midžić, mag.ing.mech.
Faculty of Mechanical Engineering and Naval Architecture
Chair of Design and Product Development
Ivana Lučića 5, Zagreb, Croatia
Telephone: 00 385 1 6168 117
Email: ida.midzic@fsb.hr
URL: http://www.cadlab.fsb.hr