Strojniški vestnik - Journal of Mechanical Engineering 54(2008)6, 398-412
UDC 658.51

Paper received: 28.2.2008
Paper accepted: 15.5.2008

Product Family Modelling in Conceptual Design Based on Parallel Configuration Grammars

Eugeniu-Radu Deciu[1,2,*] - Egon Ostrosi[1] - Michel Ferney[1] - Marian Gheorghe[2]
[1]Université de Technologie de Belfort-Montbéliard, France
[2]University "Politehnica" of Bucharest, Romania

In this paper, a product family modelling in conceptual design approach based on configuration grammars is proposed and developed. Product modelling is essential during the conceptual design both in the functional and the structural design spaces. However, there is no adapted formal representation to support the modelling of configurable products. Grammars can be considered as formal powerful tools to represent the relationship inside the configurable products. Therefore, we propose a configuration grammar design approach which is based on two types of grammars: a functional grammar for configuration and a structural grammar for configuration. The two configuration grammars work simultaneously on different abstraction levels of the product family. Therefore, the evolution of the product model, from the stage of functional structure to the stage of physical structure, can be represented in an adapted way. The configuration grammars design approach constitutes an effective tool for the representation and the modelling of configurable product family. The proposed design approach is validated by the application to a design case of industrial products - the design of an external gear pump family. The results of the application validate the pertinence of our approach.

Keywords: product family design, configuration design, product configuration, modelling, CAD

1 INTRODUCTION

Design of configurable product family or *design for configuration* has emerged as an efficient tool to deal with the new challenges of a constantly dynamic and volatile market [1]. Design for configuration is the process which generates a set of *product configurations* based on a *configuration model* and is characterized by a configuration task [1] to [5]. The *configuration task* then consists in finding the configuration of a product by defining the relations between its components in order to satisfy a set of specifications and a set of constraints imposed on the product [1] to [5].

An essential characteristic of the conceptual design of a configurable product family is the *product modelling* [1], [5] and [6]. The effective modelling of a configurable product family must be capable to represent the complex relationship between the components of a product on the one hand, and the members of the family, on the other hand [7] and [8]. This modelling must also be capable to represent the product structures both in the functional space and in the structural space of design. Furthermore, the modelling must deal with the problem of *generation* and *derivation* of the different products, and thus carry out the variety of the *new* and *innovative* products [9].

However, in conceptual design, there is no adapted formal representation to support the modelling of the configurable products [10]. Grammars can be considered as formal powerful tools to represent the strong structural relationship inside the configurable products [7], [8], [10] to [14]. Grammar-based design systems have the potential to automate the design process and allow a better exploration of design alternatives [15] to [17].

This paper proposes and develops a configuration grammar design approach to support the computer-aided-design for product family modelling. Two interrelated subjects are considered in this research:

1. What are the properties of configurable products? *(Properties extraction of configurable products)*
2. How can we develop such generative configuration grammars capable to handle the configurable products structures in both, functional and structural, design spaces?

*Corr. Author's Address: Université de Technologie de Belfort-Montbéliard, Laboratoire M3M 90 010 Belfort Cedex, France, d_eugen@hotmail.com

(Development of Configuration Grammars Design Approach for Product Family Modelling)

This paper is structured as follows.

In the first section, the problem of design for configuration of product family is presented and the use of design grammars during the conceptual design is introduced.

In the second section is developed the proposed configuration grammar-based design approach for product family modelling. The theoretical bases of the proposed approach are set-out. First, the properties of the structures of configurable products are extracted. Then, using the extracted properties of the configurable structures, the functional and the structural configuration grammars are defined and developed. The inference of configuration grammars and the algorithm of generation are also indicated.

In the third section is presented the case study of an external gear-pump family modelling and representation. The design case study illustrates and validates the proposed design approach based on configuration grammars in the conceptual design.

The conclusions and the perspectives of this research study are finally presented.

2 CONFIGURATION GRAMMAR-BASED DESIGN APPROACH FOR PRODUCT FAMILY MODELLING

In this paper, we propose a grammar-based design approach for product family modelling in conceptual design, which is based on two configuration grammars (Fig. 1):

- *a functional grammar for configuration* (**FGC**), *and*
- *a structural grammar for configuration* (**SGC**).

These two configuration grammars work in parallel in two design spaces, the functional space and the structural space, in order to represent, model and configure a product family.

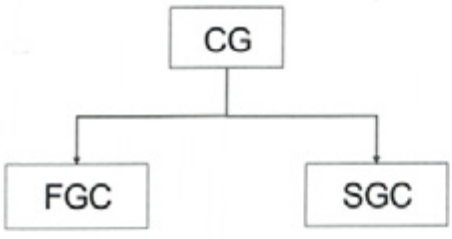


Fig. 1. *The configuration grammar-based design approach*

The physical structure must accomplish the functional structure of the product family, and this means that the set of product functions has to be accomplished, in the structural space, by a set of physical solutions. So, the construction of the physical structure of the product family is based on its functional structure. In this paper we propose the development of both structures in parallel, by the means of the configuration grammars approach.

The functional structures are of great importance in the development of the configurable product family. The functional structure of a product is used to represent the product functions. Then the first configuration grammar, **FGC**, is used to construct and to represent the functional structure of the product family in the functional design space.

The second configuration grammar, **SGC**, is used to construct and to represent the physical structure of the product family in the physical design space. The proposed **SGC** grammar has two forms of representation, respectively:

- *a structural attributed graph grammar for configuration* (**SAGGC**), and
- *a structural grammar for configuration based on features* (**SGCF**).

In the next sections are developed the proposed configuration grammars.

2.1 Properties of the Structures of Configurable Products

To define the configuration grammars, we have extracted a set of properties of the structures of configurable products [12].

From the engineering design point of view, the *feature-component-module-product* relationships are adequate structural means for a general product representation. Since such means are ***recursive***, any *proper granularity level of representation* must be introduced to asses design for configuration possibility.

We note with *structure(i)*, the level *i* of a precedence relationship. For instance a *component* is a *structure(2)* level. Then, the properties are defined to represent and generate the structures of a configurable product family and are defined as follows:

- **Property 1. The significant structures.** The significant structures(i) handled by the grammars are: the *primitive structures* or the *terminals*, the *intermediate structures* or the *non-terminals* and the *final structure*.
- **Property 2. The configuration features.** Each structure(i) is provided with a particular set of attaching elements, called the set of *configuration features* [11] and [12];
- **Property 3. The configuration connections.** Each pair of structures, structure(i)-structure(j), is connected through the configuration features. The association of two or more, features of configuration, belonging to different structures, generates new elements called *configuration connections* [11] and [12].
- **Property 4. Generation of new structures.** From the interconnection between the structures, primitive or intermediate, evolves a higher level structure to these ones [10]. The association of the configuration features produces the joint features on the one hand, and the tie features on the other hand [9] and [10]. The consequences of this property are [12]:
  - *Addition* – refers to the possibility of a structure(i) to be added to the existent product structure (of the product family);
  - *Suppression* – refers to the possibility of a structure(i) to be removed from an existent product structure (in a product family);
  - *Recurrence* – refers to the possibility of adding (and repeating) several times the same structure(i) in order to generate and configure the product configuration;
  - *Replacement or swapping* – refers to the possibility of a structure(i) of the product structure to be replaced by another structure(i);
  - *Change of attributes* – refers to the possibility of a structure(i) to change some of its attributes.
- **Property 5. Geometric and topologic constraints.** The interconnection between the structures can occur if and only if the structures satisfy some conditions defined on the geometric and topological domains (i.e. the geometric and topologic constraints). For instance, *spatial orientation* – indicates the relative spatial orientation of each structure in the product structure.

2.2 Functional Grammar for Configuration

Different authors have proposed several functional grammar approaches to address the functional design space [7], [18] and [19]. In a similar way, we propose a functional grammar for configuration to generate the product functional structure.

Given the set $\mathbf{F}=\{f_1, f_2, f_3, \ldots, f_i, \ldots f_m\}$, with $i \in [1,m]$, the set of m functions of the family of product FP, these functions define the functional structure or the functional graph for each product in the family FP. Then, to generate (construct) the functional graph, characteristic for each product, we define the functional graph grammar for configuration (**FGGC**).

The functional graph grammar for configuration is defined as the rewriting graph system expressed like a quadruple (1):

$$FGGC = (V_N, V_L, P, S) \qquad (1),$$

where:

V_N = is the alphabet of labeled nodes that represent the set of elementary functions of the product family;

V_L = is the alphabet of nodes labels that designate the functions names;

P = is the set of graph productions (the rules of graph productions);

$S = \{O\}$ is the start symbol of the graph.

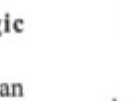
The alphabets of the nodes and labels are defined as follows:

- *the alphabet of labeled nodes:*

The alphabet of the labeled nodes is defined as the union of the terminal alphabet and the non-terminal alphabet of nodes.

$V_N = V_N^N \cup V_N^T$, where:

V_N^T 

V_N^N 

- *the alphabet of labels:*

The alphabet of the labels is defined as the union of the terminal alphabet and non-terminal alphabet of labels.

$V_L = V_L^N \cup V_L^T$ where:

$V_L^N = \{f\}$ is the non-terminal alphabet of labels;

$V_L^T = \{function\_name_i\}, i \in [1,m]$ is the terminal alphabet

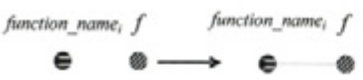
of labels, m is the total number of functions of the product family and the variable *function_name*$_i$ is the specific name of each function.

The set of productions *P* is formed of the following rules:

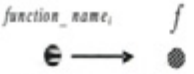
P$_1$) Generation of the first (non-terminal) node of the functional graph:

P$_2$) Transformation of a non-terminal functional node into a terminal functional node of the functional graph:

P$_3$) Addition of a non-terminal functional node to a final functional node of the functional graph:



P$_4$) Transformation of a terminal functional node into a non-terminal functional node of the functional graph:



P$_5$) Suppression of a non-terminal functional node from the functional graph:

Each rule of production of the functional graph grammar for configuration (**FGGC**) is defined like a triplet (G_L, G_R, T), composed of the two graphs G_L and G_R, called the left side and the right side, and T is called insertion function.

A production rule is applied to the graph G whenever we wish to change the structure of G. More precisely, the production is carried out by locating an isomorphism of G_L in G, by eliminating G_L from G, by replacing G_L with G_R and by inserting G_R in G, using the function T.

2.3 Structural Configuration Grammar

In this approach, we propose the representation and the modelling of a product family in the structural design space, by the help of the structural grammar for configuration (**SGC**).

The structural grammar for configuration has two forms of representation:

A) the first representation is based on a structural attributed graph grammar for configuration,

B) the second representation is based on a structural grammar of configuration based on features.

These two forms of the structural grammars work together and are complementary in their purpose to model and configure a product family.

2.3.1 Structural Configuration Attributed Graph Grammar

The structural attributed graph grammar for configuration (**SAGGC**) is defined as a seven-tuple:

$$SAGGC = \{N, E, A, B, S, P, O\} \qquad (2),$$

where:

- $N = \{N^T, N^N\}$ is the alphabet of terminal and non-terminal nodes;
- $E = \{E^T, E^N\}$ is the alphabet of terminal and non-terminal edges;
- A = is the set of nodes attributes;
- B = is the set of edges attributes;
- S = is the start symbol;
- P = is the set of production and transformation rules for the nodes of the attributed graph grammar.

In the above definition, the sets N^T and respectively N^N have the following definitions:

- $N^T = \{V^T_{structure}, V^T_{joint-tie\ features}\}$ represent the terminal alphabet of nodes and consists of a terminal alphabet of significant configuration structures and a terminal alphabet of joint and tie features of configuration.

The two alphabets are defined as follows:

- $V^T_{structure} = \{structure(i)\}$ is a finite, non-empty set of significant and primitive configuration structures of a product called the *terminal alphabet of configuration structures*. For example, the terminal structures are the parts of a configurable mechanical product.
- $V^T_{joint-tie\ features} = feature_j$ is a finite, non-empty set of configuration features called *the terminal alphabet of joint features and tie features*.

For example, the generic term *feature(j)* is defined by the following set:

feature$_j$ = {ext_thread, cone, planar_circ_face,

key_groove, parallel_groove, ext_spline, ext_cyl_face, ext_plane_face, gear, ext_cyl_area, ext_prismatic_area}.

The set N^N has the following definition:

- $N^N = \{V^N_{structure}, V^N_{joint-tie\ connections}, S\}$ represent the non-terminal alphabet of nodes and consists of a non-terminal alphabet of significant structures, a non-terminal alphabet of joint and tie connections, and the start symbol S.
- $V^N_{structure}$ = is a finite, non-empty set of non-primitive significant configuration structures, called *the non-terminal alphabet of configuration structures*. The non-terminal structures are the modules/products/families of mechanical configurable product family.
- $V^N_{joint-tie\ connections} = connection_k$ represent a finite, non-empty set of connections of configuration between the configuration features called the *non-terminal alphabet of joint and tie connections*.

For example, the generic term *connection(k)* is defined as follows:

$connection_k$ = {plane_connection, circ_plane_connection, cylindrical_connection, cylindrical_stepped_connection, slot_connection, threaded_connection}

- $E = \{E^T, E^N\}$ is the terminal and non-terminal alphabet of edges.
- E^T = is the terminal alphabet of edges. A terminal in E^T, represent a relation between a terminal of $V^T_{structure}$ and a terminal of $V^T_{joint-tie\ features}$, where $E^T \subseteq V^T_{structure} \times V^T_{joint-tie\ features}$.
- E^N = is the non-terminal alphabet of edges. A terminal in E^N, represent a relation between a non-terminal of $V^N_{structure}$ and a non-terminal of $V^N_{joint-tie\ connections}$, where $E^N \subseteq V^N_{structure} \times V^T_{joint-tie\ connections}$.
- S = is the start symbol of the configuration grammar.
- P = is the set of graph productions of the attributed graph grammar. The set of production and transformation rules satisfy the property of generation of new structures of the configuration grammar.

The inference of the properties of *addition*, *suppression* and *recurrence* makes possible to generate new structures, and consequently, the corresponding rules define the set of production rules of the configuration attributed graph grammar. The properties of *replacement* and *change of attributes* make possible to transform a structure of a given level (primitive, intermediate or final) into another structure, and consequently, the corresponding rules define the set of transformation rules of the configuration attributed graph grammar.

Then, in our **SAGGC** approach, we distinguish three types of production rules:

- the *production of addition* ($P^{addition}$),
- the *production of suppression* ($P^{suppression}$),
- the *production of recurrence* ($P^{recurrence}$).

The graph representation of these productions is indicated in Figure 2.

Also, in our approach **SAGGC**, we distinguish two types of transformation rules:

- the *transformation of replacement*,
- the *transformation of change of attributes*.

The transformation of replacement $T_i^{replacement}$ has the following graph representation (Fig. 3).

Formally, the transformation $T_i^{replacm}$ has the expression:

$$\{X,x\} \xrightarrow{T_i^{replacm}} \{Y,y\} \qquad (3).$$

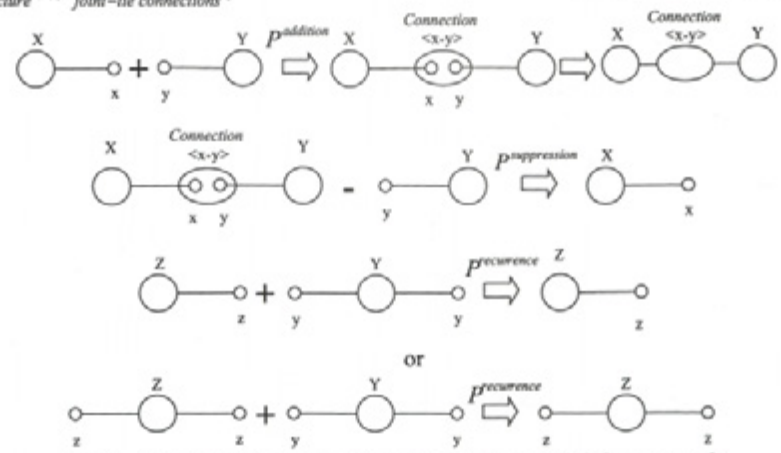


Fig. 2. *Set of productions rules used in the SAGGC approach*

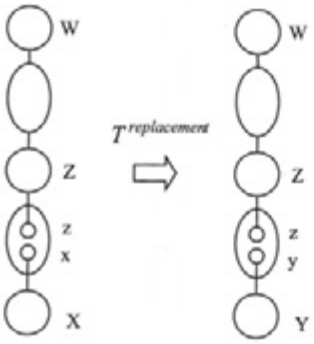


Fig. 3. *Example of the transformation replacement of structures*

The transformation of change of attributes $T_i^{chg\ attb}$ has the following graph representation (Fig. 4).

Formally, the transformation $T^{chg\ attb}$ has the following expression:

$$\{X(A_1), x\} \xrightarrow{T^{chg\ attb}} \{X(A_2), x\} \quad (4),$$

- $O = \{+, -\}$ is a set of operators applied on the structures. These operators satisfy the *property 4*.

2.3.2 Structural Configuration Grammar Based on Features

A *configuration language* describes the generation of configuration structures, joint elements and tie elements [10]. So, a configuration grammar based on features provides the formal and generic description for this configuration language. Then, the *structural grammar for configuration based on features*, **SGCF,** is defined as the 8-tuple:

$$SGCF = \left\{V^T_{structures}, V^T_{joint-tie\ features}, V^N_{structures}, V^N_{joint-tie\ connections}, S, \nabla, \Lambda, P\right\} \quad (5),$$

where:

$V^T_{structures} = \{structure(i)\}$ is a finite, non-empty set of primitive significant structures called the *terminal alphabet of configuration structures*. Where *structure*(i), is a terminal configuration structure.

$V^N_{structures} = \{STRUCTURE(i), \cdots, S\}$ is a finite, non-empty set of non-primitive significant structures called the *non-terminal alphabet of configuration structures*. Where *STRUCTURE*(i), is a non-terminal configuration structure and S is a special non-terminal symbol called the *start symbol*.

$V^T_{joint-tie\ features} = \{0, feature_j, \cdots, \nabla\}$ is a finite, non-empty set of configuration features called the *terminal alphabet of joint and tie features*. The term *feature*$_j$ has the same meaning like in the **SAGGC** definition.

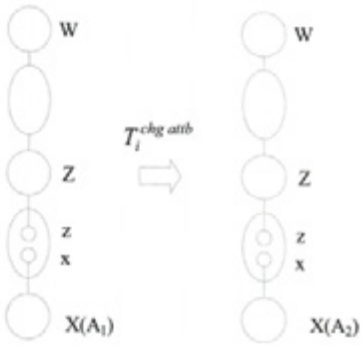


Fig. 4. *Example of the transformation change of attributes of the structures*

$V^N_{joint-tie\ connections} = \{O, \ldots, connection_k, \cdots \nabla, \Lambda\}$ is a finite, non-empty set of configuration connections between the configuration features called the *non-terminal alphabet* of *joint* and *tie connections*. The term *connection*$_k$ has the same meaning like in the **SAGGC** definition.

S, ∇, Λ are respectively the starting symbol of configuration structures, the starting symbol of joint connections and respectively of tie connections.

$P: \left\{\begin{bmatrix}\alpha\\ \Gamma_\alpha\\ \Delta_\alpha\end{bmatrix} \rightarrow \begin{bmatrix}\beta\\ \Gamma_\beta\\ \Delta_\beta\end{bmatrix}\right\}$ is a finite, non-empty set of productions rules.

The following conditions must be held concerning the terminal vocabulary and the non-terminal vocabulary, of configuration structures and respectively of joint features-tie features:

$V^T_{structure} \cap V^N_{structure} = \varnothing$ and,

$V^T_{joint-tie\ features} \cap V^N_{joint-tie\ connections} = \varnothing$

$(V^T_{structure} \cup V^N_{structure}) \cap (V^T_{joint-tie\ features} \cup V^N_{joint-tie\ connections}) = \varnothing$

To define the generation of new structures in a non-ambiguous way, we have to state a set of conditions on the production rules. The interconnection between structures can occur if and only if the structures satisfy some conditions defined on the geometric and topological domains. These conditions are represented by the geometric constraints and topological constraints. For instance, *spatial orientation* – indicates the relative spatial orientation of each structure in the product structure.

The geometric and topological constraints are imposed at each level of production rules. This

means that the grammar productions rules must meet obligatory conditions or constraints before being applied.

Then, a *conditional production rule C* is defined as follows:

$$\begin{bmatrix} \alpha \\ \Gamma_\alpha \\ \Delta_\alpha \end{bmatrix} \xrightarrow{C} \begin{bmatrix} \beta \\ \Gamma_\beta \\ \Delta_\beta \end{bmatrix} \quad (6),$$

where $C = \begin{bmatrix} C_{\alpha\to\beta} \\ C_{\Gamma_\alpha\to\Gamma_\beta} \\ C_{\Delta_\alpha\to\Delta_\beta} \end{bmatrix}$ are semantic conditions associated to each level of a productions rule.

2.4 Inference of Configuration Grammars

2.4.1 Inference of Configuration Grammars on the Multiple Abstraction Levels of the Product Structures

The configuration grammar design approach is based on two parallel grammars that are developed to address the product family modelling in two design spaces during the conceptual design: the functional space and the structural space.

Moreover, the two proposed configuration grammars work simultaneously on four different abstraction levels of the significant structures:
- the *part level*,
- the *module level*,
- the *product level*,
- the *family level.*

Then, the functional grammar is used in the functional space of design to generate the functional structure of a product, structure which is translated, in the structural space, in a physical structure solutions corresponding to the previously four levels.

In this paper we present only the inference of configuration grammars on the part level and the product level of configurable product structures.

Next we indicate the application steps of our configuration grammars approach on the part level generation.

2.4.2 Inference of Configuration Grammars on the Part Level

On *the part level,* our structural configuration graph grammars have a similar representation, in a certain limit, with the representation form of the graph grammars suggested by Fu et al. [20] for part structure generation. Comparing to that paper, the main difference is the purpose of our approach, which is to configure a product family. Firstly, our contribution is to define the concept of significant regions as structures with a functional signification in the configuration of the part. Secondly, the introduction of the concepts of *configuration features* and *configuration connections,* on the one hand, and of *conditional grammars,* on the other hand, constitutes important contributions that make possible to handle and connect the structures, in order to generate a variety of part configurations.

Steps of inference of configuration grammars on the part level

1) Generation of the functional structures of the product.
2) Identification of the set of elementary functions to be accomplished by the part.
3) Identification of the possible configuration features which can accomplish and materialize the elementary functions required.
4) Join of the features and identification of the regions with a functional signification in the component structure.
5) Join of the functional significant regions, generation and validation of the various configurations of the part.

According to the variation of the elementary functions, the structural space is concretized in several alternatives of part regions. These regions are combined in order to generate several part configurations. The various configurations are validated according to the main functions that are defined initially.

These steps are summarized in the following algorithm of generation of the configurations structures of a part (Fig. 5). The same algorithm is applied to all abstraction levels of a product family in order to generate a variety of configurations, where each level will has its corresponding significant structure.

Concerning the application of the algorithm of generation, an interesting research direction could be how to improve the design information extraction from the data base. In this direction, different methods and approaches could be considered such as the probability-based approach proposed by Kim et al. [21] and the conceptual design algorithm proposed by Žavbi and Duhovnik [22].

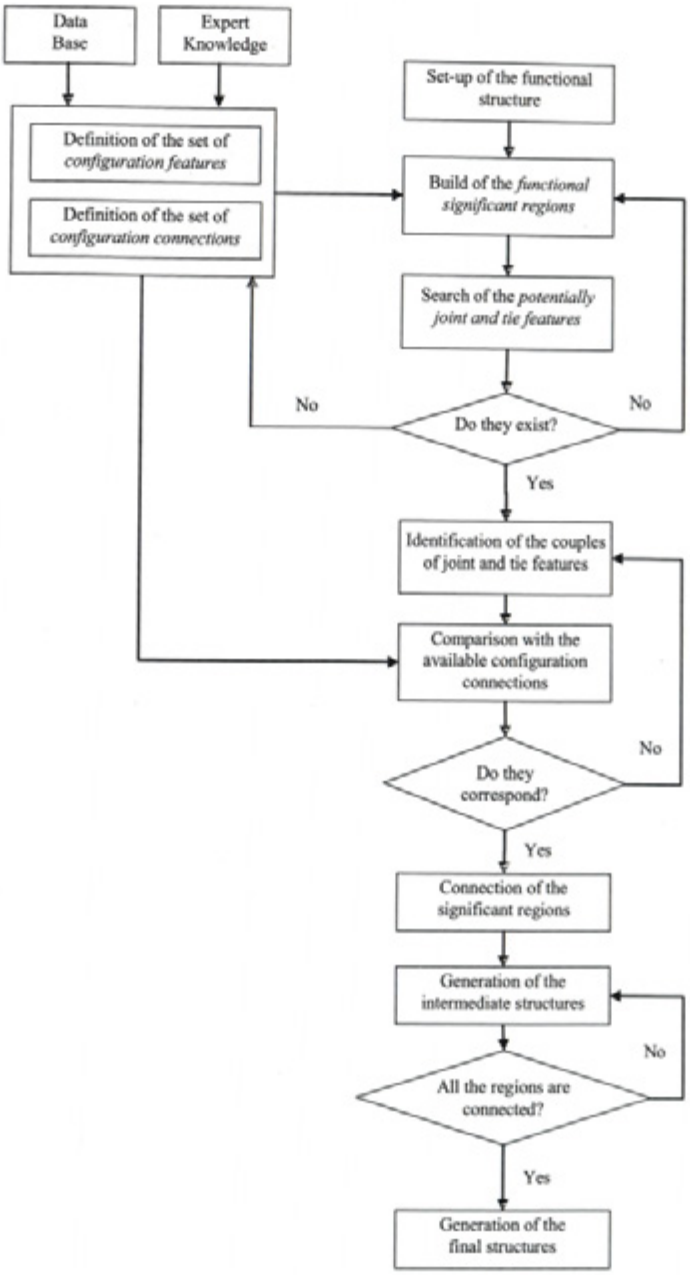


Fig. 5. *Algorithm of generation of the product structures using the configuration grammars*

3 INDUSTRIAL APPLICATION

3.1 External Gear Pump Structure

In this section, we present the application of our configuration grammar approach to the design case study of an external gear pump family.

The external gear pumps are widely used in hydraulic systems, due to their simplicity, reliability, and very high power ratings. External gear pumps are fixed displacement and are used in different applications, such as: hydrostatic mobiles units, equipments for transport, machine tools and other applications with a large number of variants and configurations.

The structure of the gear pump is composed of the following main components [22] (Fig. 6): *flange (1); body (2); thrust seal (3); bearings (4);*

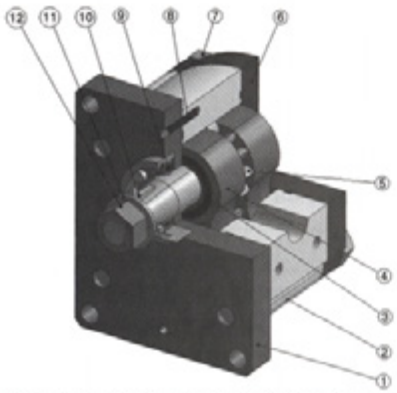


Fig. 6. *Generic structure of the external gear pump*

shafts (5, 5'); rear cover (6); screws (7); dowel pin (8); lip seal (9); woodruff-key (10); washer (11); nut (12).

The purpose of this application is to represent, to model and to generate the structures of an external gear pumps family by the help of configuration grammars inferred on different level of abstraction.

In this case study, we are indicating the inference of configuration grammars on the part level and respectively, on the product level.

3.2 Inference of Configuration Grammars on the Part Level

First, we present the inference of configuration grammars approach on the part level.

The steps of inference of the configuration grammars approach correspond to the steps presented in the paragraph 2.4.2.

3.2.1 Functional Significant Regions

On the *part level*, one or more configuration features are joined together, in order to generate a region with a functional meaning. We call this region a *functional significant region*.

Let us take as example the generation of the driving shaft of the external gear pump. The structure <*SHAFT*> is composed of three main functional significant regions:

- *Region I* – the left area of the shaft;
- *Region II* – the central area of the shaft; and
- *Region III* – the right area of the shaft.

Each one of these three regions is made up of a collection of one or more configuration features which are "assembled" together in order to accomplish a common function, i.e. the corresponding function of the region (Fig. 7).

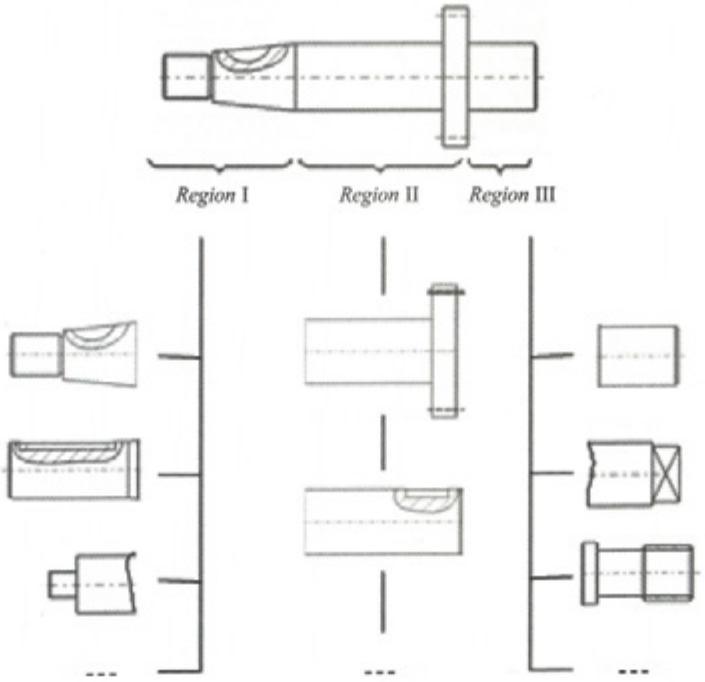


Fig. 7. *The functional significant regions of the structure <SHAFT>*

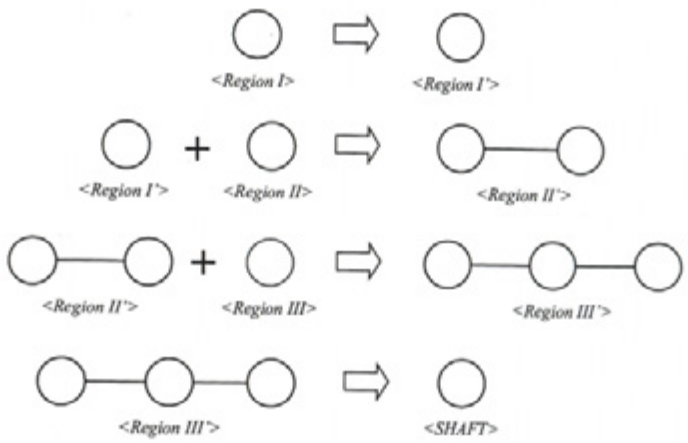


Fig. 8. *Graph productions to generate the functional significant regions of the <SHAFT>*

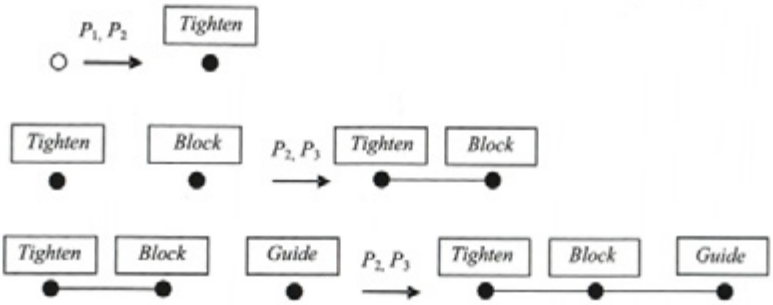


Fig. 9. *Productions to generate the functional structure of region I*

The set *P* of graph productions describes the generation of the structure <*SHAFT*>, starting from its functional significant regions, and is given above (Fig. 8)

In the following section, we present the generation of each of the three significant regions of the <*SHAFT*>.

Generation of region I of the <SHAFT> structure:

As we specified, the feature association is made with the purpose to accomplish one/many functions of the product.

In this case, the function which must be accomplished by *region I* (the left area of the part) is "to transmit the torque of the movement", with elementary sub-functions: "to tighten", "to block" and "to guide".

The three functions are materialized respectively by three features: <*External_thread*>, <*Key_Groove*> and <*Conical_Area*>.

Other geometric alternatives of the configuration features are available, in order to generate various configurations of the same region (Fig. 7).

Thereafter, we present the productions inferred to generate the functional structures of the regions, and respectively the physical structures of the regions.

In the functional space, the functional structure of *region I* is generated by the inference of the productions of **FGC** grammar (Fig. 9).

In the structural space, the production rules used to generate *region I* of the <*SHAFT*> structure are as follows (Fig. 10).

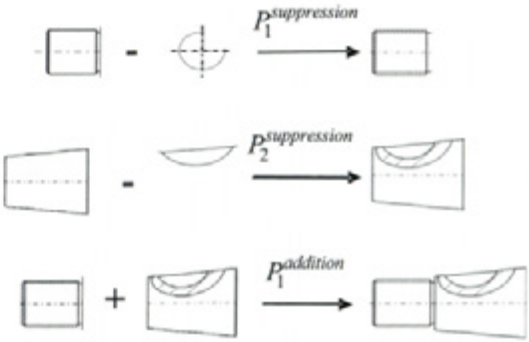


Fig. 10. *Productions to generate the region I of the <SHAFT>*

Below, are indicated the formal and the graph representations of the first production.

The formal representation of the production which generates *region I* is (7):

$$P_1^{add}: \left| \begin{array}{c} \left[\begin{array}{c} \langle CONICAL\_THREAD\_AREA \rangle \rightarrow \langle CYL\_THREAD\_AREA \rangle \langle CONICAL\_AREA \rangle \\ \left[\langle PlaneConnection \rangle \rightarrow \langle ExtPlaneFace_1 \rangle \langle ExtPlaneFace_1 \rangle \right] \end{array} \right] \\ \left[\begin{array}{c} \langle ExtThread \rangle \rightarrow \langle ExtThread \rangle \langle 0 \rangle \\ \langle ExtCyl Face_2 \rangle \rightarrow \langle 0 \rangle \langle ExtCyl Face_2 \rangle \\ \langle KeyGroove \rangle \rightarrow \langle 0 \rangle \langle KeyGroove \rangle \\ \langle ConicalLatFace \rangle \rightarrow \langle 0 \rangle \langle ConicalLatFace \rangle \end{array} \right] \end{array} \right| \quad (7).$$

The graph representation of the production presented above is (Fig. 11).

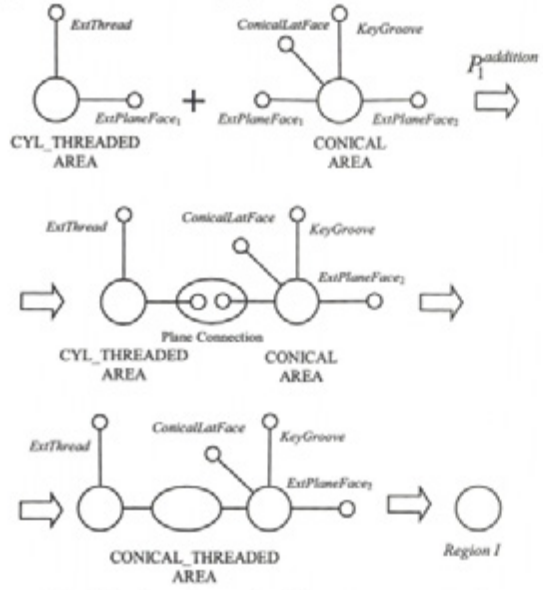


Fig. 11. *Graph productions to generate the region I*

Generation of region II of the <SHAFT> structure:

In the case of the central area of the *<SHAFT>*, the function to be accomplished is "to turn around its own axis" and "to transmit the rotational movement to the driven shaft". We distinguish two cases of possible solutions: the first one, when the two functions are materialized together into the same single solution, and the second one, when the functions are materialized by two distinct solutions.

The set of features which materializes the two functions is composed of three features: *<Hub>* for the first case; and respectively *<Cylindrical_Area>*, *<Parallel_Groove>* for the second case.

In the functional space, the production used to generate the functional structure of *region II* is (Fig. 12).

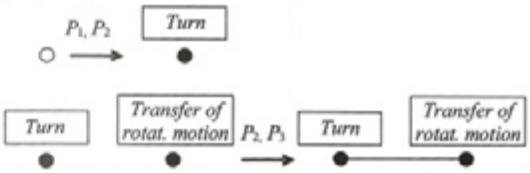


Fig. 12. *Productions to generate the functional structure of the region II*

The formal representation of the production which generates *region II* is:

$$P_1^{add}: \left| \begin{array}{c} \left[\begin{array}{c} \langle CONICAL\_THREAD\_AREA \rangle \rightarrow \langle CYL\_THREAD\_AREA \rangle \langle CONICAL\_AREA \rangle \\ \left[\langle PlaneConnection \rangle \rightarrow \langle ExtPlaneFace_1 \rangle \langle ExtPlaneFace_1 \rangle \right] \end{array} \right] \\ \left[\begin{array}{c} \langle ExtThread \rangle \rightarrow \langle ExtThread \rangle \langle 0 \rangle \\ \langle ExtPlaneFace_2 \rangle \rightarrow \langle 0 \rangle \langle ExtPlaneFace_2 \rangle \\ \langle KeyGroove \rangle \rightarrow \langle 0 \rangle \langle KeyGroove \rangle \\ \langle ConicalLatFace \rangle \rightarrow \langle 0 \rangle \langle ConicalLatFace \rangle \end{array} \right] \end{array} \right| \quad (8).$$

In the structural space, the production which generates the *region II* of the *<SHAFT>* structure shown in Figure 13.

The graph representation of the production presented is (Fig. 14).

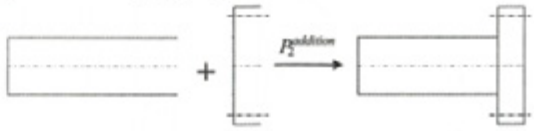


Fig. 13. *Production to generate the region II of the <SHAFT>*

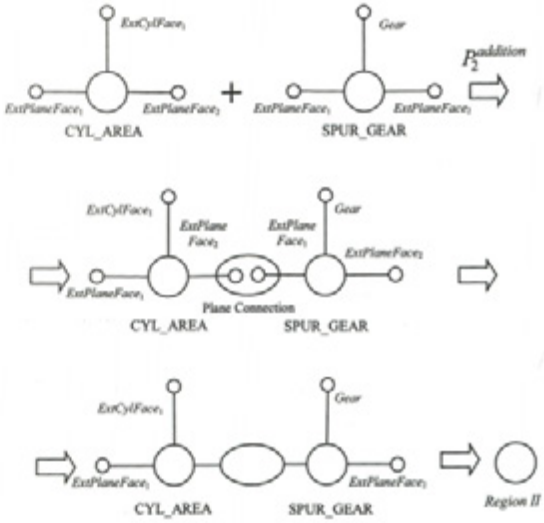


Fig. 14. *Graph productions to generate the region II*

Generation of region III of the <SHAFT> structure

In the case of *region III* of the *<SHAFT>* structure, the functions to be accomplished are: "to turn around its own axis" and if required "to transmit the torque of the movement". For this example, we considered only the first function.

So in the functional space, the functional structure of *region III* consists only in one function (Fig. 15).

In the structural space, the corresponding feature which materializes this function is the *<Cylindrical_Area >* (Fig. 16).

Fig. 15. *Production to generate the functional structure of the region III*

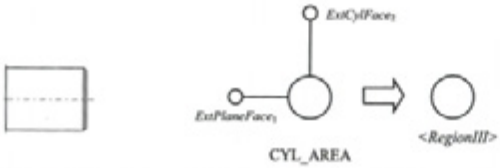


Fig. 16. *Graph production to generate the region III of the <SHAFT>*

Generation of the final structure <SHAFT>:

This stage consists in connecting the structures, which were previously generated for the three main regions, in one single structure, i.e. the final structure of the *<SHAFT>*.

To generate the configuration of the *<SHAFT>* part structure implies the arrangement of the structures of the significant regions *I*, *II*, *III*, their connection according to the set of configuration productions (Fig. 8) and the validation of the final generated structure according to the initial functional structure.

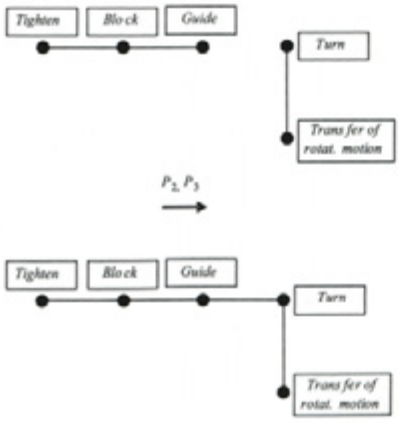


Fig. 17. *Production to generate the functional structure of the region II'*

In functional space, the rule of production to generate the functional structure of *region II'* has the given form in Figure 17.

In the structural space, the production used to generate the structure *<SHAFT>* is given in Figure 18.

The production rule to generate the final functional structure of the *region III'*, i.e. *<SHAFT>*, is given in Figure 19.

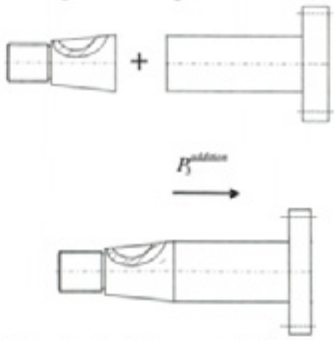


Fig. 18. *Production to generate the region II' of the <SHAFT>*

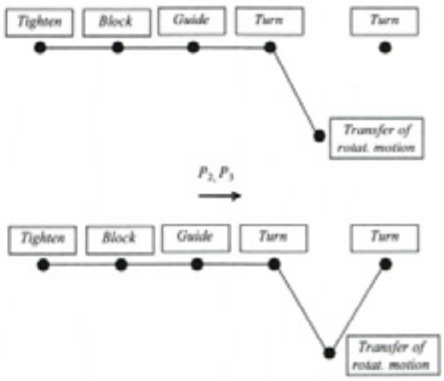


Fig. 19. *Production to generate the functional structure of the region III'*

In the structural space, the production used to generate the physical structure of the *<SHAFT>* is given in Figure 20.

The result of this production inference, Figure 20, represents the final structure of the *<SHAFT>*.

Once the final structure is generated, we have inferred the transformation of replacement for each of the three functional significant regions

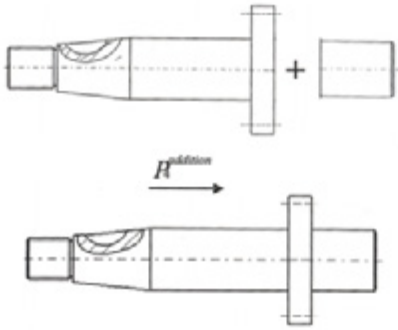


Fig. 20. *Production to generate the final structure of the <SHAFT>*

constituting the structure *<SHAFT>*, and we have generated several configurations of the structure shaft. Some of the configurations that were generated are indicated in the Figure 21.

The same mechanism of configuration is used to generate the other parts of the external gear pump. The alternative configurations of the parts are then used in the generation of the structures of an external gear pump family.

3.2.2 Inference of Configuration Grammars on the Product Level

The configuration grammars are inferred on the *product level* in order to generate different instantiations of the external gear pumps.

So, based on the significant structures, on the one hand, and on the transformation rules, on the other hand, we can produce new and novel product structures.

The inference of the replacement rule on the *product level* produces the replacement of a part in a given product structure. By using this transformation rule in the case of an external pump structure, the replacement of several parts, we can generate a variety of configurations of gear pumps, i.e. a family of pumps.

In Figure 22 are presented some configurations of gear pump structures generated from the inference of the *transformation of replacement*.

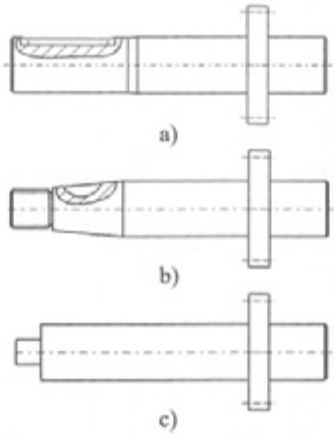


Fig. 21. *Various configurations generated for the <SHAFT> structure*

Fig. 22. *Various configurations of gear pumps generated after the inference of the transformation of replacement*

4 CONCLUSIONS

In this paper we have proposed a product family modelling approach in conceptual design based on configuration grammars.

The proposed approach is based on two configuration grammars: a *functional grammar for configuration* (**FCG**) and a *structural grammar for configuration* (**SCG**). These two grammars work in parallel and are used to generate simultaneously the functional structure and respectively the physical structure of a configurable product family.

The **FGC** grammar is used to construct and to represent the functional structure of the product family in the functional design space.

The **SGC** grammar is used to construct and to represent the physical structure of the product family in the physical design space. The **SCG** grammar has two forms of representation: an *attributed graph grammar for configuration* and a *grammar for configuration based on features*.

The contributions of our design approach are:

1. The configuration grammars are defined on the properties of the structures of configurable products.
2. The proposed grammars are based on feature-component-module-product relationships, considered as adequate structural means for a general product representation.
3. The functional domain and structural domain are considered simultaneously in order to generate the functional structure and the physical structure of a configurable product family.
4. The configuration grammars, defined on the functional design space and on the structural design space, are complementary and work in parallel in the two design spaces.
5. An algorithm of generation of the product structures using the configuration grammars was proposed.
6. The configuration grammars work simultaneously on multiple levels of abstraction of the significant structures.

We have applied the proposed configuration grammars-based design approach to the design case study of an external gear pump family. In this design case, the configuration grammars were inferred on the part and the product level, in order to generate the external gear pump family. The results obtained from the inference of configuration grammars validate and demonstrate the interest of our approach.

Our current research work is concentrating on the possibilities of implementation of the configuration grammar design approach in CAD environment software.

The proposed method meets functional requirements in a qualified way, but not necessarily quantified (e.g. stress limit constraints). The introduction of the behaviour space in the definition of design grammars could be a new open direction of research. Another direction of research is how the proposed method can facilitate the application of analysis based examinations.

5 REFERENCES

[1] Tiihonen, J., et al. Modelling configurable product families. *Proceedings of the 12th International Conference on Engineering Design (ICED'99)*, vol. 2, Munich, p.1139-1142, 1999.

[2] Snavely, G.L., Papalambros, P.Y. Abstraction as a configuration design methodology. *Advances in Design Automation*, (New York: ASME) DE - (65)-1, p. 297-305, 1993.

[3] Brown, D.C. Defining configuring. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Cambridge University Press, vol. 12, p. 301-305, 1998.

[4] Sabin, D., Weigel, R. Product Configuration Frameworks – A survey. *IEEE Intelligent Systems*, vol. 13(4), p. 32-85, 1998.

[5] Soininen, T., Tiihonen, J., Männistö, T., Sulonen, R. Towards a general ontology of configuration. *AIEDAM*, vol. 12(4), p. 357–372, 1998.

[6] Männistö, T., Soininen, T., Sulonen, R. Modeling configurable products and software product families. *Presented at the IJCAI'01 Workshop on Configuration*, Seattle, USA, 2001.

[7] Siddique, Z., Rosen, D. Product platform design: a graph grammar approach. *Proceedings of DETC'99*, ASME Design Engineering Technical Conferences, 1999.

[8] Du, X., Jiao, J., Jiao, J. Tseng, M. Graph grammar based product family modeling. *Concurrent Engineering: Research and Applications*, vol. 10(2), p. 113-128, 2002.

[9] Ostrosi, E., Ferney, M. Feature modeling grammar representation approach, *AIEDAM*,

vol. 19(4), p. 245-259, 2005.

[10] Ostrosi, E., Ferney, M., Deciu, E.R., Garro, O. Feature-based reasoning for designing structural configuration in advanced CAD systems. *5th International Conference on Integrated Design and Manufacturing in Mechanical Engineering (IDMME 2004)*, 5-7 April, Bath, UK, 2004.

[11] Deciu, E.R., Ostrosi, E., Ferney, M., Gheorghe, M. Configuration grammar-based design approach for product family modeling in advanced CAD systems. *Proceedings of the 16th International Conference on Engineering Design (ICED'07)*, ISBN 1-904670-02-4, Paris, France, 2007.

[12] Deciu, E.R., Ostrosi, E., Ferney, M., Gheorghe, M. A configuration grammar design approach for product family modeling in conceptual design. *Proceedings of the 7th International Symposium on Tools and Methods of Competitive Engineering (TMCE'08)*, Izmir, Turkey, 2008.

[13] Schmidt, L.C., Cagan, J. Grammars for machine design. *Gero J.S., Sudweeks F. (ed.), Artificial Intelligence in Design'96*, Kluwer Academic Publishers, p. 325-344, 1996.

[14] Schmidt, L.C., Cagan, J. Optimal configuration design: an integrated approach using grammars. *ASME Journal of Mechanical Design*, vol. 120(1), p. 2-9, 1998.

[15] Mullins, S., Rinderle, J.R. Grammatical approaches to design. Part 1: An introduction and commentary, *Research in Engineering Design*, vol. 2(3), p. 121-135, 1991.

[16] Rinderle, J.R. Grammatical approaches to engineering design. Part II: Melding configuration and parametric design using attribute grammars, *Research in Engineering Design*, vol.2(3), p. 137-146, 1991.

[17] Chase, S. A model for user interaction in grammar-based design systems. *Automation and Construction*, vol. 11(2), p. 161-172, 2002.

[18] Starling, A.C., Shea, K. A grammatical approach to computational generation of mechanical clock designs. *Proceedings of International Conference in Engineering Design, ICED'03*, Stockholm, Sweden, 2003.

[19] Schmidt, L.C., Shi, H., Kerkar, S. A constraint satisfaction problem approach linking function and grammar-based design generation and assembly. *ASME Journal of Mechanical Design*, vol. 127(2), p. 196-205, 2005.

[20] Fu, Z., De Pennington, A., Saia, A. A graph grammar approach to feature representation and transformation. *International Journal of Computer Integrated Manufacturing*, vol. 6(1&2), p. 137-151, 1993.

[21] Kim, S., Ahmed S., Wallace, K. Improving information extraction using a probability-based approach. *Journal of Mechanical Engineering – Strojniski vestnik*, vol. 53(7-8), p. 429-441, 2007.

[22] Žavbi R., Duhovnik, J. Conceptual design chains with basic schematics based on an algorithm of conceptual design. *Journal of Engineering Design*, vol. 12(2), p. 131-145, 2001.

[23] External gear-pumps catalogs: Hesper, Rexroth Bosch, Enerflux Industrie.