TOPOLOGICAL DESIGN OF A HIGH ALTITUDE PLATFORM (HAP) USING A SYSTEM DESIGN LANGUAGE

Dipl.-Ing. Manan Haq
Design - Construction
TAO Technologies GmbH
Nobelstraße 15, D-70569 Stuttgart
haq @ tao-group.de

Prof. Dr.-Ing. habil. Bernd Kröplin
Institute for Statics and Dynamics of Aerospace Structures
University of Stuttgart, Germany
Pfaffenwaldring 27, 70569 Stuttgart, Germany
kroeplin @ isd.uni-stuttgart.de

ABSTRACT

In this paper the application of graph grammar based design languages is presented. Design languages consist of a set of vocabulary and a set of rules which have a specific order within a production system. The design graph generated by the design language is a formal and a domain independent representation of the design object. This allows to apply modifications by rules to the design graph. Using the example of a High Altitude Platform system (HAP system) it is shown how the design language approach enables the automation of design processes by facilitating the generation of a large variety of design variants very effectively.

1. INTRODUCTION

Design of complex systems such as the topological design of aerospace systems is a particular challenge for design engineers. Several multidisciplinary sub-systems of different domains are building a network of nodes. The nodes are interacting with each other by exchanging information. In case of modifications in one node these information can propagate through all other nodes of the network and vice versa. For analysis and simulation purposes digital models of each domain are generated separately. They are not fully linked with each other, which causes a lack of model consistency. Therefore parametrical and topological changes can only partially propagate through adjacent models of each domain. Thus modelling, management and model consistency problems exist within the design process of these systems.

However, graph based design languages are well suited for building master-models of such complex systems where different domains can exist at the same time and place. Such a design language consists of a set of basic components, also called vocabulary, and a set of rules, also called design patterns. These two sets enable the designer to define a master- model in form of a design graph. It is a formal - and at this level still domain independent - representation of a design object. This formal domain independency allows the designer to make modifications by design rules to the design graph or master model. So that the applied changes are propagated consistently through all geometrical, physical and other models contained therein. Out of this master-model sub-models of each domain can be derived. Thus model consistency over various design domains is therefore intrinsically maintained.

The systematic is shown by an example of a High Altitude Platform (HAP, FIG 1). This is a Lighter-Than-Air System for telecommunication and telemetry purposes [19]. A design language for such a HAP system is developed within the software tool named “Design Compiler 43” where sub-domains such as geometrical configuration-, CAD-, rise analysis-, electronics-, drive-, aerodynamics- and structural-model are embedded therein. By this design language various parametrical and topological configurations of HAP systems are analysed.

2. RELATED WORKS

In early stages of computational design synthesis, rule-based information processing concepts of various kinds have been developed to improve the formalisation of the design process.

The Lindenmayer-Systems (also called L-Systems) were developed in biology to describe the growth of plants and trees [20]. L-Systems are based on *string grammars*, where abstract symbols are represented by characters from an alphabet and encode the topological plant structures through their rules expressed as string substitutions.

Shape grammars were described by Gips and Stiny [12]. A set of geometrical shape rules forms the design grammar that is used to generate different truss designs. Mitchell applied the concept of shape grammars in the field of architecture to support the design process [7]. In these approaches, the rules are expressed directly in geometrical form. Further, Shea and Cagan [31] developed a shape grammar for truss structure generation and optimisation. With this technique, geodesic-like domes, building roof structures and transmission towers were generated and the design topologies were optimised in terms of weight and strength. Furthermore, parallel

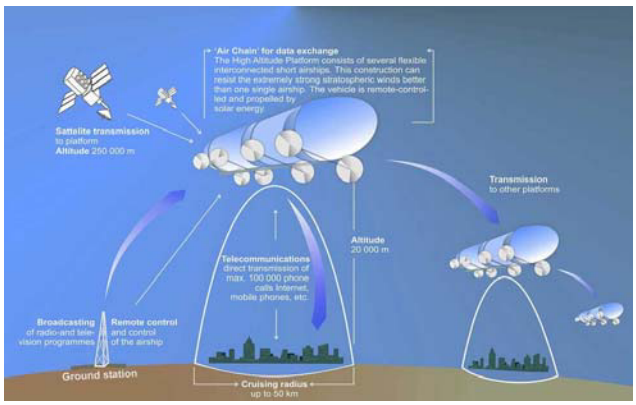

FIG 1. Concept of HAP System

grammars, based on a function-behaviour-structure, are applied by Starling and Shea [33] for the parametric synthesis of mechanical systems. Finally, Agrawal and Cagan [1] developed a shape grammar for coffee makers. In this grammar the so-called shape rules were coupled with certain product functions. Also, a detailed cost analysis of the manufacturing and assembly of novel coffeemakers was obtained. In order to overcome the limitation of consistency between function and form,

Finger and Rinderele [11] used the formalism of *graph grammars [13]* for form and function configuration of mechanical systems. Campbell et al. combined the method of function structures and graph grammars for the conceptual design of products [32]. Saitou et al. generated the so-called topology graphs for decomposition-based assembly synthesis of the structural characteristics of products [10], [21] and [22], such as for the optimal feasibility of manufacturing and assembly.

Alber and Rudolph [2], [3], [4] introduced the approach of a generic graph grammar where a so-called design graph is generated as an intermediate domain-independent representation. Subsequently, this graph representation is translated into the various object domains and multiple domain-dependent models of the design object can be generated. On the basis of this approach a design language was developed to generate truss structures such as transmission towers. In the same research group, Irani and Rudolph [17] developed a graph grammar for conceptual design of space stations, while Schaefer and Rudolph [27] applied the formalism of graph-based design grammars to support the automated design process of satellites. Finally, Kormeier et al. [18] and Haq and Rudolph [14], [15], [16] defined a graph-based design language for aircraft surface conceptual design and developed a design language for generic space-frame structure design.

3. APPROACH OF GRAMMAR BASED DESIGN LANGUAGE

3.1. Languages for engineering design

Natural languages are means of verbal and written communication between humans. They are able to transport most if not all relevant information in a specific situation of a problem to other humans. The linguistic structure of natural languages is formed out of a so-called vocabulary and a set of rules that describe the syntactically correct form of a sentence (i.e., the correct order of the vocabulary in that sentence). It is important to note that the vocabulary serves just as set of symbolic placeholders. In order to give such a structured assembly a meaning (or in other words to interpret a sentence), each symbol is assigned with a certain meaning. This aspect is called semantics. In the domain of pragmatics the effects of such a sentence on humans in a specific situation, under certain boundary conditions and other context-related constraints is considered [16].

In engineering, formal languages [9] are most typically known as string-based *programming languages* (e.g., Java, Pascal, C, C++ etc.). In analogy to these string-based formal languages a graph-based design language can be defined to describe technical objects. For this purpose the components of a design object are considered as the vocabulary (i.e., the graph nodes) as the basic 'building blocks' of the design language. They can subsequently be composed together by graph-based design rules. Within this approach, a grammar of a graph-based design language for technical objects is developed, which allows creating a set of possible rule combinations – a so-called *production system* – which represents a generative description of a technical object. It proves later to be advantageous to handle complex objects on this basis within a broad range of structural properties and forms [15], [16].

3.2. Graph grammar based design language

With the approach of graph grammar based design language the engineer is able to define a formal graph based but domain-independent representation of a design object. Within the rule-based design process a design graph of an engineering artefact is generated by executing a specific sequence of rules that represents a building plan of a design object. This means that the first step in developing a design language is to define the vocabulary and the rules of the grammar of the design language (FIG 2). The vocabulary elements may be seen as abstract containers that represent the basic geometrical and functional components of the desired object domain. The rules of the design language are the design patterns consisting of the building instructions for the design object. In a production system, a specific set of design rules is composed together, which represents a building plan of an object (FIG 2). These tasks must be done once in the beginning of a design process. After this effort, the vocabulary and the rules can be reused over and over again. By executing a set of design rules, a new design graph can be generated or an existing design graph can be manipulated further. The reexecution of a set of design rules with different boundary conditions (typically contained in the axiom) leads to the generation of different models under (re-) use of the know-how encoded in the design patterns. In analogy to the aforementioned string-based formal languages the graph-based language translation can be described as a two-step translation process: In a *first translation step*, a design graph is created by executing the production system of the desired engineering artefact. Such a design graph contains the complete information about the desired engineering design object. It describes the design object on a high abstraction level. Therefore it is one common representation for all domains, such as for the geometrical, for the physical and/or for the functional model domain. This enables to maintain the design model consistency between these domains through constraint processing techniques. In the *second translation step*, the achieved design graph is interpreted via interfaces for analysis or visualisation purposes. This may be arranged individually for each area of application. For post processing purposes, analysis tools can be connected to the design compiler '43' via specific interfaces [15], [16]. In the following sections, the syntax of the design rules and the vocabulary definitions are described in more detail.

3.2.1. Vocabulary definition

Within the definition of graph grammar based design language, the nodes in a design graph and in a design rule

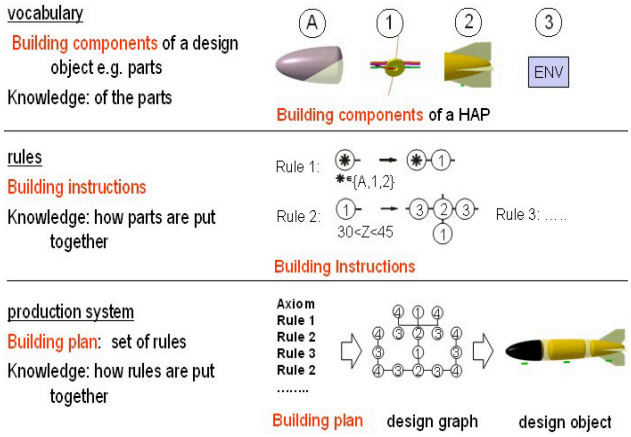

FIG 2. Elements of a design language

are called vocabulary of a design language. The vocabulary are abstract placeholders for real geometrical objects, procedures or ideas, which can be composed to form a complete technical system during a design process. FIG 2 shows some vocabulary of a HAP system. Some of them are of geometrical and of functional nature. They contain complete information to define the desired devise or element of a HAP These are the description of geometry in the form of a script for CAD modeller CATIA V5 [6]. Physical information is embedded in the form of equations for a symbolic computer algebra program (Mathematica 5 [23]). The interfaces of the vocabulary, the so-called ports, are defined to exchange data between other nodes in design graph. Furthermore, parameters and variables are defined to embed the measures of the design object in the vocabulary for parameterisation purposes. Such a definition of each vocabulary is deposited in an XML file that enables easy and structured extension and documentation capabilities.

3.2.2. Rule definition

The syntax for design rules as it is implemented in the design compiler '43' is represented by a geometrical arrangement of graph nodes and edges in a plane. It is divided into the four quadrants Q_1–Q_4 as illustrated in FIG 3. Such a four-quadrant scheme is a derivation from the original X-scheme, which is quite common in the field of graph grammars [13]. The left two quadrants Q_1 and Q_2 represent the *if*-part and are also called the *conditional part* of the graph rule. The right two quadrants Q_3 and Q_4 represent the *then*-part and may also be called the *generative part* of the graph rule. In case of rule execution, a search is performed to check if there is any (sub-)graph matching between the graph in the conditional part and the actual existing design graph. If a matching is found, the generative part of the rule is executed and the modifications represented in the graph rule are mapped onto the actual design graph. As represented in the rule in FIG 3, the following modifications can be applied to the design graph: All nodes in Q_1 and all edges leading to these nodes will be removed from the design graph. Nodes in Q_3 will be added to design graph. Edges represented in the 'then-part' (Q_3 and Q_4) are assigned to the design graph. Attributes of newly pasted or already existing nodes in the then-part in Q_3 and Q_4 of the rule can be set or modified [16].

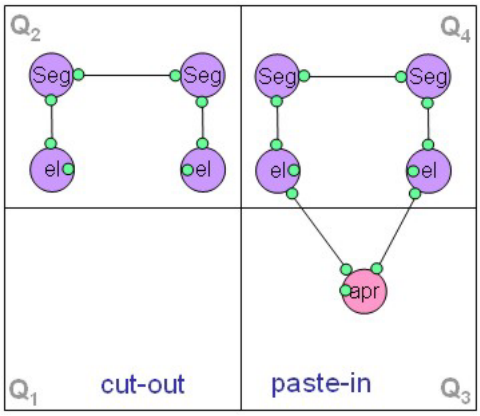

FIG 3. Four quadrants of a rule [15]

3.2.3. Production system

A production system specifies the syntactically correct arrangement of the rules of the design language in a program. It presents a building plan of a design object, which insures a feasible semantics of the describing object. If the rules are not executed in a specific right order, it may be that an object can be generated but it fails the aspired behaviour of the desired object as a whole. Similar to the natural languages where the meaningful massage of the sentences are derived from the syntactical right order of the words, in the design languages the behaviour of the design object as a whole (i.e., the semantics) depends on both the correctness of the vocabulary and there syntactically correct arrangement in a design rules, which have also a syntactical correct order in a production system.

Every production system consists of an axiom and a set of rules (FIG 2). The axiom presents a starting graph of a design object and specifies a necessary condition of an execution able program. Which means that in case of a missing axiom a production system can not be executed. The set of rules followed by the axiom presents a building plan of the object. A production system can consist of many sets of rules. The sequence of execution of rules within a set has a specific order. Otherwise it may be that the set of rules can not be completely executed so that the generation of the design object with its aspired behaviour could be fail. The sequence of execution of the sets of rules of different domains could have an arbitrary order. This facilitates the design engineer to generate a sub graph of different design domains independently.

4. DEVELOPING THE DESIGN LANGUAGE FOR HAP-SYSTEMS

In this section the development process of the graph grammar based design language of a HAP System is described. At first the preliminary step of feasible decomposition and aggregation of the complete system is presented. By this the vocabulary and rules of the design language are defined, which is an essential condition for developing a design language for any engineering object. At least, the rules are composed to a production system, which presents a building plan of the HAP system. By executing the production system a design graph is generated. It is an abstract representation of the HAP

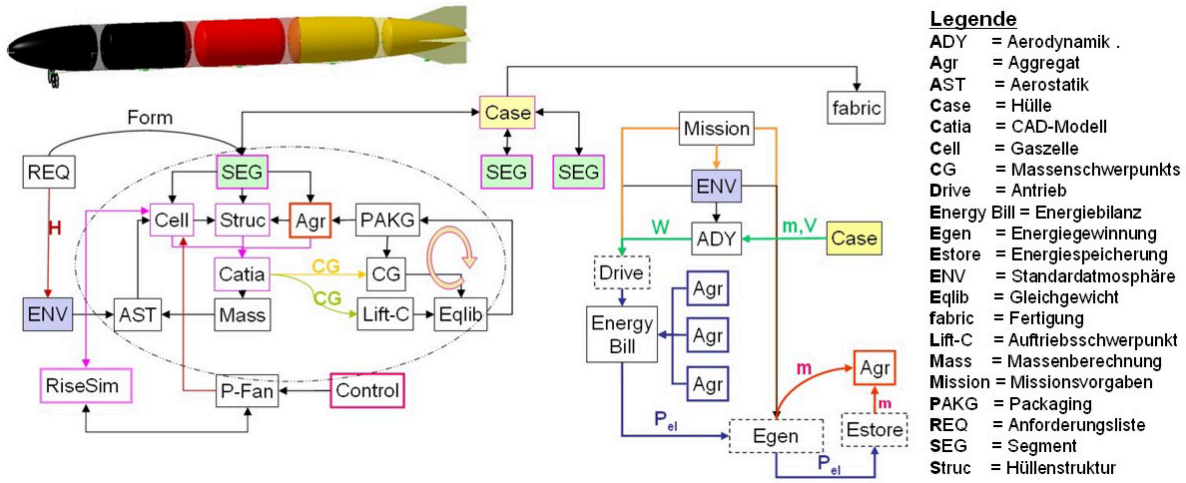

FIG 4. Overview and system decomposition of HAP system

system including all necessary domains and facilitating model consistency simultaneously over all domains. Finally, the aspects of granularity within vocabulary and rule definition are discussed. It is shown that a right balance of granularity between vocabulary and rules of a design language facilitates a high rate of freedom in designing engineering artefacts.

4.1. System decomposition and aggregation

First step of developing any design language is the decomposition process of the desired design object. In case of the HAP system the complete system has to be decomposed in its basic building blocks. The decomposition has not only to be made for the structural body of the HAP but also for all other domains of the system. This concerns to the functional-, electrical- and energy model etc. (FIG 4). The decomposition process for vocabulary definition and the aggregation process for rules definition are not done one after another, but they proceed hand in hand and are complementing each other in iterative steps. The right level of granularity in the decomposition gives the direction of quality of the design language. On one hand the aim is to decompose the complete system in general building blocks which can be commonly used for the whole system. On other hand the aim is to define a minimum number of general rules or building instructions, which facilitates a syntactically correct arrangement of the vocabulary to their neighbours within a graph.

FIG 4 illustrates an overview of the HAP system. It consists of a diagram where the sub systems of the HAP with there neighbourhood relations are presented schematically. Such a representational form is an initial step for the decomposition process of the desired object. It is neither an absolutely top-down nor an absolutely bottom up presentation, but a compound of both. The diagram has also not a clear tree structure but a tree which leafs are adhered in there ends.

The objects or the nodes in the diagram can be broken down into there extractive elements until a feasible commonality is distinguishable between them. As presented in FIG 4 the hull [*CASE-node*], consists of many segments, [*SEG-nodes*]. Each segment consists of a cell, structure and system components: [*Cell-node*], *[Struc-node*] and [*Agr-node*]. There is also a [*Req-node*] which

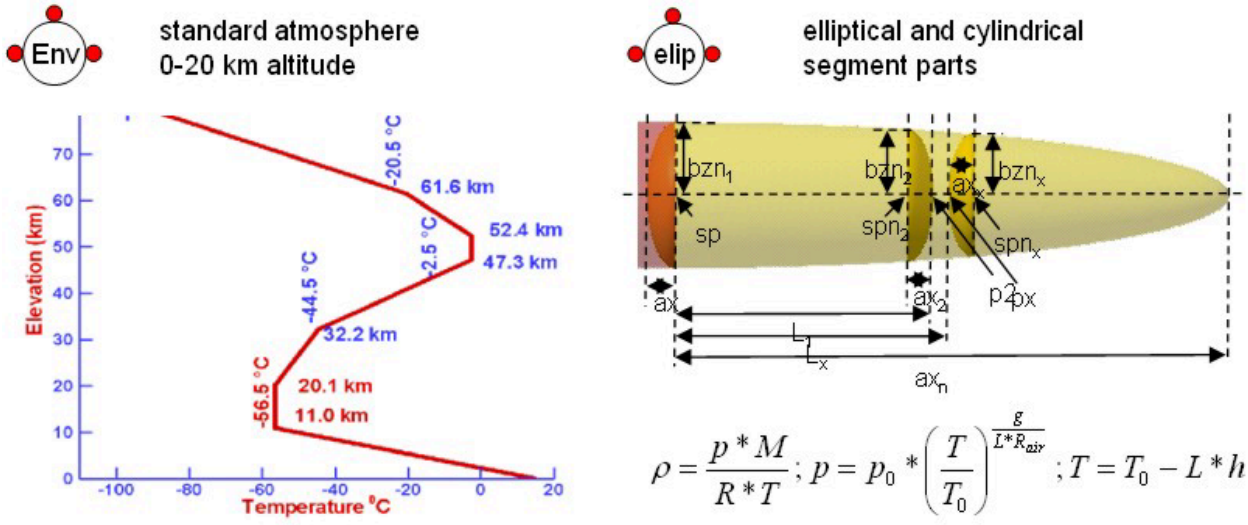

FIG 5. Functional and geometrical vocabulary of the design language

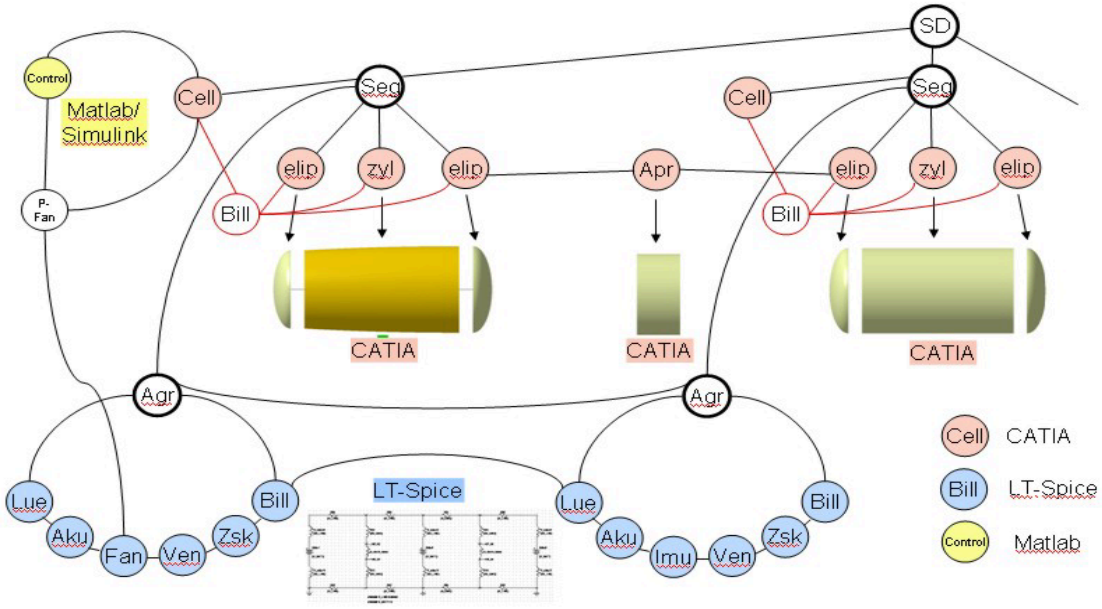

FIG 6. System aggregation by the HAP design language

contains the requirement parameters for the desired HAP. A [*ENV-node*] consist of equations describing the atmosphere from the ground to 20 km altitude. The [*AST-node*] is responsible for the aerostatic calculations of each segment of the HAP. The control system is implemented in the [*Control-node*], which e.g. regulates the pressure, [*P-Fan-node*], in each segment during the rise period, [*RiseSim-node*], of the HAP from the ground to the target altitude. The edges between the nodes represent the relations between them. Thus a commonality is distinguishable between many objects contained in the diagram. These objects can be picked out and defined as common objects or the vocabulary of the design language. While defining the rules, these vocabulary can be used to derive object or graph nodes of the design graph and there relations in form of edges. These general relations between the objects in the diagram are assertive for the definition of the rules. E.g. every segment consists of a cell and every cell is connected with a control node.

We present the hull geometry of the HAP as an example to make the decomposition and aggregation process more clear. As presented in FIG 6 the geometry of the hull consist of five segments. Therefore a vocabulary is defined as a general placeholder, which enables during the rule definition the derivation of specific objects of segments. Each segment consists of an elliptical and cylindrical segment part (FIG 5, 6). So we define one vocabulary for a cylindrical and one vocabulary for an elliptical segment part. It is distinguished in the graph in FIG 6 that common rules can be defined. One rule adds two elliptical segment part nodes to every segment node. Another rule adds a cylindrical segment part node between two elliptical segment part nodes. A further general rule which can be defined is to connects two segments with an apron by adding an apron node between two segment nodes (FIG 6 rule). We can proceed by this way extracting common vocabulary and rules out of such a diagram of each domain for expanding and improving the design language.

5. SYSTEM ANALYSIS EXAMPLES WTH THE DESIGN LANGUAGE

In the chapter before we have presented the development process of the design language. It makes clear that a feasible decomposition and aggregation are an importance act for vocabulary and rule definition a design object.

Such a unique abstract representation of the HAP system provides various analysis possibilities for the design engineers which are not performable in such a less effort

In this chapter, the enhancement of the developed design language of the HAP system is demonstrated by means of some analysis examples.

5.1. Parametrical and topological analysis

In the first example a parametrical and topological analysis is performed to the HAP system. As mentioned before, a production system of a design language consists of an axiom and a set of rules. Modifications can be done on the one hand by changing the parameters in the axiom or in any other rule in the set. On the other hand the topology of the HAP system can be modified by executing rules, which can add or delete corresponding nodes from the design graph.

In FIG 7 (right) a production system with an axiom and a set of rules is presented. In the same fig left we can see the results of parametrical modifications done by changing the axiom of the production system. In the first line we have a HAP system with five segments and in the corresponding axiom there are also five segment nodes. In the second line the HAP system consist of three segments and a smaller diameter. The jump from first HAP to third HAP is only the modification of parameters. The third, fourth and fifth HAP are from the same class, the difference is up to the number of the segments.

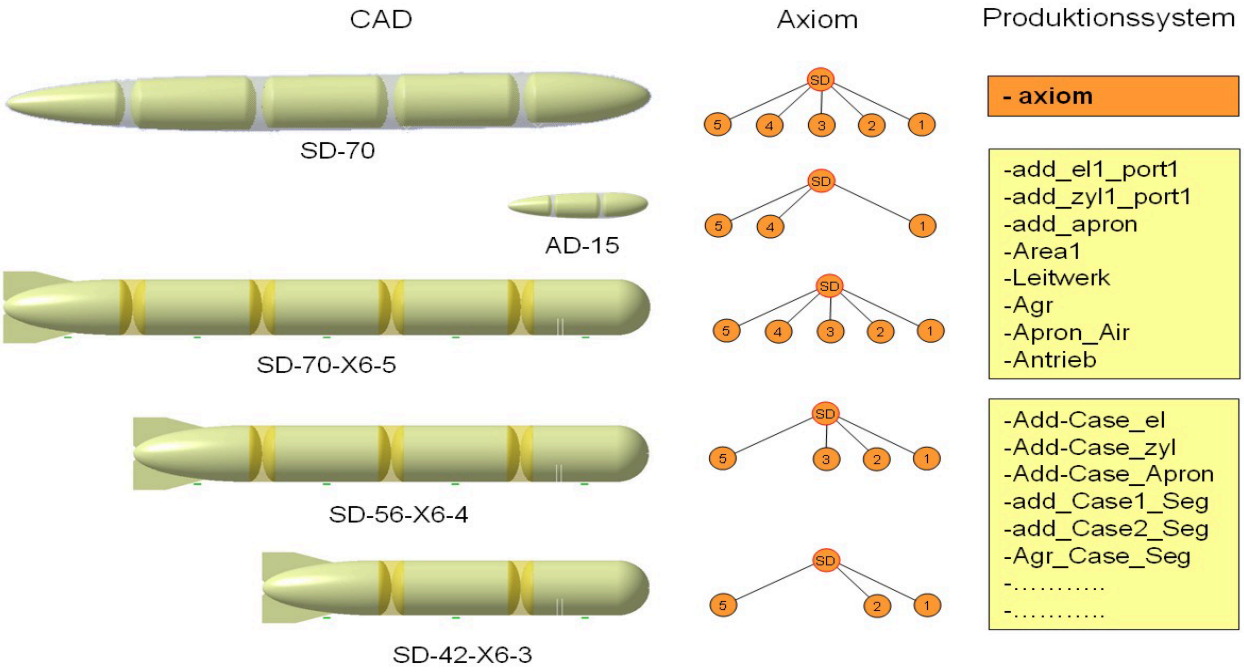

FIG 7. Parametrical and topological analysis of the structure

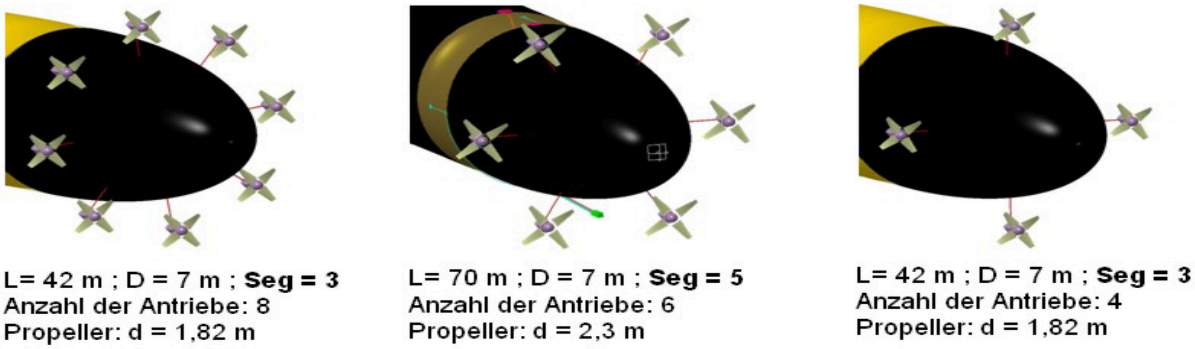

FIG 8. Parametrical and topological analysis of the drive system

In FIG 8 we can see topological modifications concerning the drive system. The part of the grammar responsible for the dimensioning of the drive system consists of a set of electrical motors with there technical parameters, a set of rotor blades and electrical cables. On the one hand the drive system dictates the dimensions of the power supply system. On the other hand the size of the HAP system dictates the dimensions of the drive system. The mass of the drive and power supply system must not exceed the given limits by the lift of the HAP system. In such a case the size of the HAP system has to be enlarged, which means a larger drive power and therefore a larger motor with bigger rotor blades and more energy cells. So the dimensioning problem begins again from the start.

5.2. Optimization of wiring harness

System integration is a complex problem within the design of HAP systems. For each mission there are various possibilities for the configurational integration of the board systems. For each configuration a suitable dimensioning of the wire harness is necessary. Even an optimum for the wiring harness can be reached. Such a problem can not be described analytically. Therefore it can also not be solved by commercial optimization tools. It is a multi objective optimization problem. One objective is to choose a thick diameter of the power cable to reduce the attrition power and therefore less battery power is needed. The other objective is to reduce the mass of the cable by choosing a less diameter of the cable.

For solving the problem mentioned above parameterized rules and vocabulary are defined. Therefore the design engineer only needs to generate a list of system components with there location in each segment. FIG 9 presents such an example, where system components are placed on a truss of a segment (FIG 9, left). The set of rules responsible for the wiring are executed, which automatically generates a parameterized wiring harness which can be optimized with genetic algorithms. The rules have an access to a database with a set of commercial cable. For analysis purposes an electronic tool “LT-spice” is used (FIG 9, right). It generates automatically a circuit plan of the system configuration and the wire harness, which is calculated for the result feedback to the design graph for optimization loops. Hence, by this way of multi objective optimization a pareto boundary is generated. By

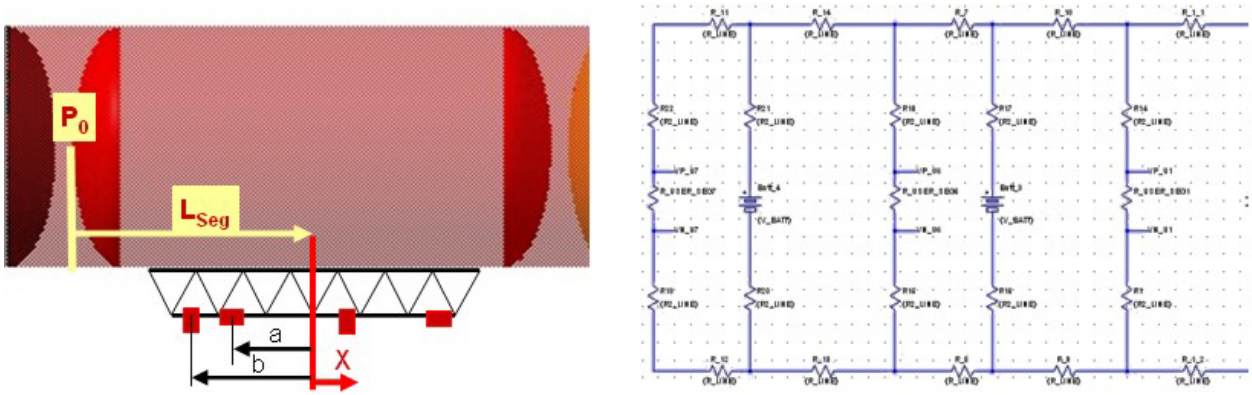

FIG 9. System integration, analysis and optimization of wiring harness

consulting further weighting factors and restrictions the design engineer can now evaluated the boundary curve and pick an optimum solution which fulfils the mission requirements. One of the weighting factors could be to obtain a wiring harness which is optimized to mass and attrition power and simultaneously meeting the goal of cost function. In case of an alternative system integration the same grammar is executed which generates an optimized circuit plan of the desired HAP configuration. Thus the design engineer is relieved of modelling the circuit planes manually and is therefore able to concentrate on the creative aspects of engineering design. Furthermore the design languages facilitate generating a large variety of design alternatives which are not conceivable in such short time periods.

5.3. Rise analysis

To determine the behaviour of the HAP during the rise phase from the ground to the target altitude, a detailed analysis is required. It consists of the expanding behaviour of the helium gas and its swapping in the cell during rising. Also its angle of approach is essential. With this the drag (air resistance) can be assessed. And finally its behaviour of straitening up can be analysed.

FIG 10 shows a rise and a straightening up analysis of a HAP system between 0m and 20000m altitude. The HAP is filled with a certain amount of helium so that the hull is plump after reaching the target altitude of 20000m. Considering the thermodynamic equations of the atmosphere, the gas has on the ground level only a 10 % of the expansion of volume as in 20 km altitude. Therefore the helium cell is not plump on the ground and the gas can swash back and forth in the segment. As illustrated in FIG 10 each segment aspire to reach a momentum free stabile attitude. For this the helium gas swash's to the front and the each segment props up and has an approach angle of about 65° degrees. During the rise phase the helium gas expands and it leads to a straightening up of the HAP.

This was done for a HAP system of 70 m length. The curve is illustrated in FIG 11. Before we launch the 70 m HAP it was essential to do preliminary tests with a 15 m HAP demonstrator system to examine the striating up behaviour in low altitudes for optical observations. This can efficiently done in heights of about 2 or 3 km. So the question is that, how a HAP system of 15 m length should be filled with helium gas so that a similar rising behaviour

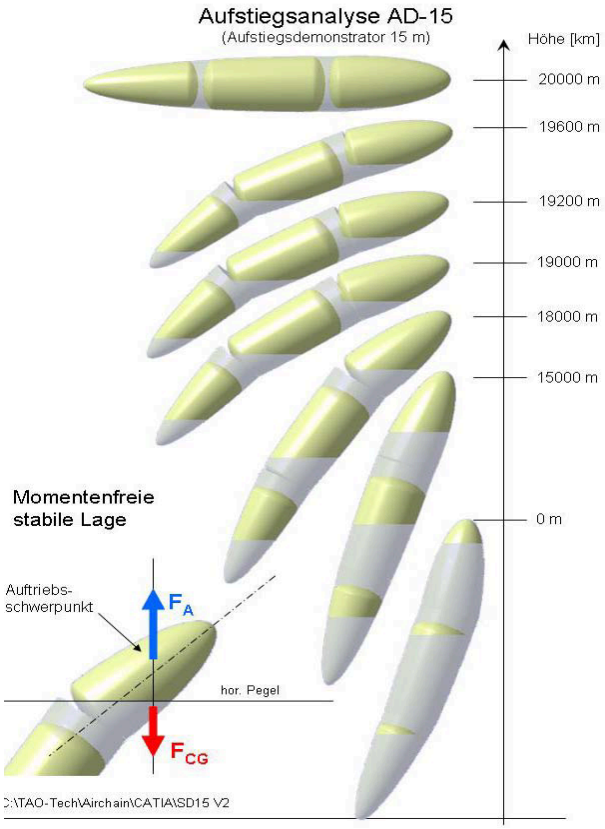

FIG 10. Rise analysis from 0m to 20000m

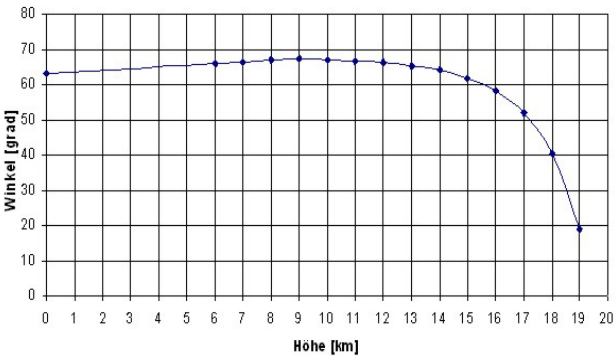

FIG 11. Straitening up behaviour of the 70m HAP

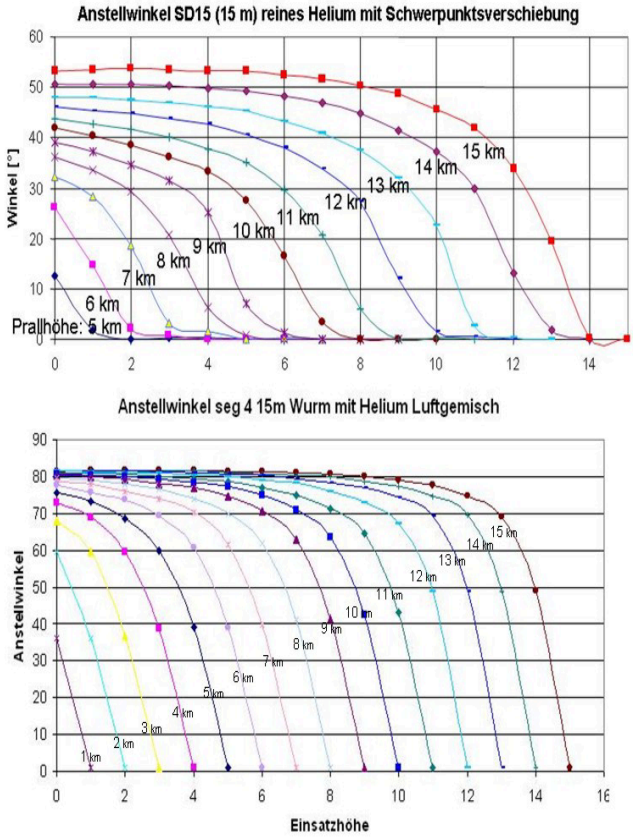

FIG 12. Angle of approach of the 15m demonstrator

could be acquired. The analysis was done for two cases. Both are presented in FIG 12.

In the first case (FIG 12 upper diagram) the gas cell of each segment was filled with pure helium for the target altitude. Each curve presents a course during the rise phase. The HAP was filled with pure Helium and weighted for the target altitude, so that the segments were plump at the target altitude.

In the second case (FIG 12 lower diagram) the gas cells were loaded with less minimum weight and filled with helium, so that the HAP could reach the target altitude. Additionally air was mixed to the helium for plumping the segments at the target altitude. The curve for each altitude is presented in FIG 12.

It can be easily indicated that only the second case is well suited for a prior test for optical observation of the rise behaviour of the HAP system. We can see that the rise behaviour of the 15 m HAP is quite similar to the 70 m HAP in only 2 or 3 km altitude.

It has to be mentioned that the diagrams in FIG 12 could not be generated analytically. The points on the curves of each diagram are a result of numerical calculations of different states during the rise of the HAP system. Performing the numerical calculation manually, it would require a time period of about 3 hours for each point. There are about 300 points in both diagrams, so that manual calculations for both diagrams are not conceivable in feasible time periods.

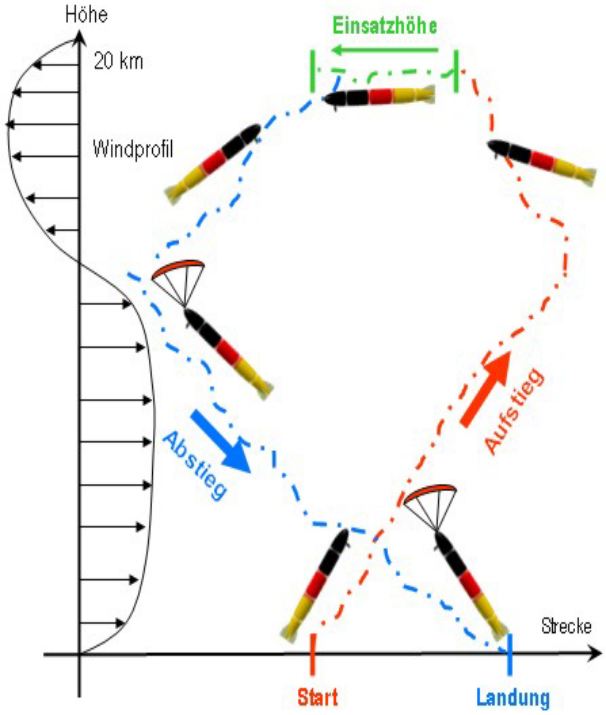

FIG 13. Rise and descent path of HAP System

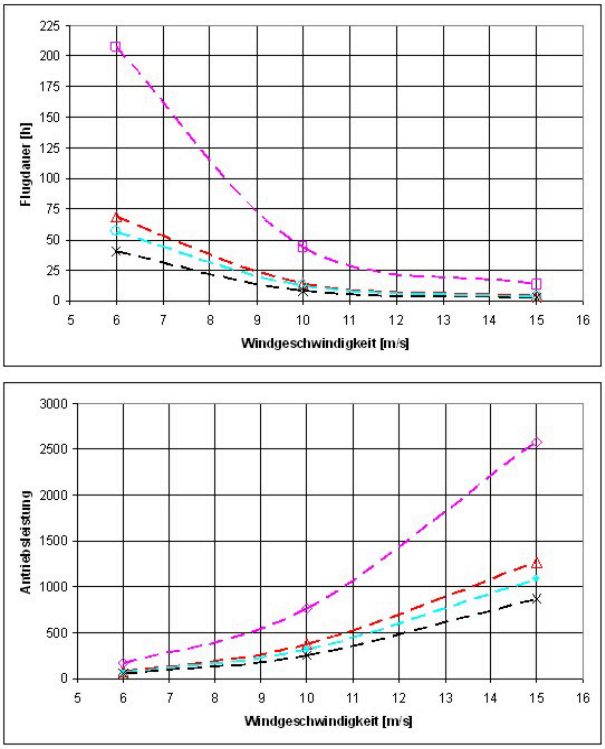

FIG 14. Flight duration and drive power analysis

5.4. Mission planning

The same design language can be used for mission planning and analysis. One of the mission analysis goals is to assess the rise and descent path of the HAP by a given air speed map (FIG 13). By this way the driftage by the wind can be acquired. So that the necessary manoeuvres to handle the HAP within the rise and descent path could

be assessed. Such an analysis is essential for the dimensioning of the drive and power supply system. E.g. if it is asserted that the calculated landing position before discharging the descent manoeuver is too far from the starting point, than it has to be estimated how much drive power is necessary to reach a starting point for the descent path which leads to a reachable landing point. This is very important information, which dictates the duration of the mission and the particular time of starting the descent maneuver.

An other analysis is illustrated in FIG 14. It shows a diagram with the mission duration over the air speed for 6m/s, 10m/s and 15m/s in 20 km altitude. Each curve presents a different HAP configuration. The HAP with 42m length has an estimated flight duration of about 40 h by an air speed of 6m/s. With the same length and an increased airspeed of 10m/s the flight duration is reduced to about 10 h and by an airspeed of 15m/s the duration is only about 5 h. Hence the over all behaviour of the four curves is of cubical nature.

The next diagram contributes for a better understanding of the first diagram. It presents the drive power over the air speed for each HAP configuration. With increasing air speed the air drag of the HAP increases simultaneously which leads to a higher driver energy consumption. Thus the flight duration decreases in a cubical manner by increasing the speed of the HAP.

Considering the results in the diagram it is observable that a feasible duration of the HAP mission can be realized between an air speed between 5m/s and 10 m/s. In case of higher air speed the portable energy in form of batteries is not enough to facilitate a feasible duration of the HAP mission. The diagrams in FIG 14 provide a deep insight into the problematic of the design of power supply and drive system and their relations to the neighbour domains. It is considerable to develop design alternatives of energy and drive system which enables flight durations above the value of battery based power supply systems.

6. CONCLUSION AND FURTHER WORK

In this paper a graph grammar based design language has been developed for the topological design of High Altitude Platform systems (HAP systems). By this the design engineer is able to generate design variants of different parametrical and topological nature. The approach of graph grammars facilitates to perform several kind of analysis in a very short time period. The implementations of these analysis were not done separately for each domains but incorporating all domains consisting in the HAP system. In spite the fact that the design of HAP system is in its very nature non deterministic and also not analytical, the approach of design language even enables the analysis of such systems, so that a deep insight into the problematic of both, the overall design and the design of each domain is gained. Furthermore the approach of graph grammars assists the design engineer to generate design alternatives in a conceivable time period.

The use of graph grammar based design languages to formalize the design of HAP system offers many opportunities for future research and industrial application. Through implementing more details in the graph grammar definition and incorporating further domains in the design language, further analysis can be perform which assist the engineer designing HAP systems in a more precisely way.

Acknowledgements

The authors would like to express their sincere thanks to the colleagues at TAO-Technologies and at Institute for Statics and Dynamics of Aerospace Structures for their contributions that have provided knowledge and understanding of HAP systems.

References

[1] Agrawal M. and Cagan J., 1997, “Shape Grammars and Their Languages – A Methology for Product Design and Product Representation” Proceedings of ASME Design Engineering Technical Conferences, DETC97-DTM-3867, Sacramento, California, USA
[2] Alber R., Rudolph S. and Kröplin, B., 2002, “On Formal Languages in Design Generation and Evolution” WCCMV, Fifth World Congress on Computational Mechanics, Vienna, Austria
[3] Alber R. and Rudolph S., 2002, “On a Grammar-Based Design Language that Supports Automated Design generation and Creativity”, Proceedings of IFIP WG5.2 Workshop on Knowledge Intensive CAD (KIC-5), Malta, Malta
[4] Alber R. and Rudolph S., 2003, “43” - A Generic Approach for Engineering Design Grammar”, Proceedings of American Association for Artificial Intelligence 2003, Spring Symposium Technical Report
[5] Bölling, M., 2005, “Multidisziplinärer Vorentwurf von Luftschiffen (in German), Diploma Thesis, Faculty of Aerospace Engineering, University of Stuttgat, Germany
[6] CATIA V5 R12, Dassault Systems, www.dassault-systemes.com
[7] Antonsson E. K. and Cagan J., 2001, Formal Engineering Design Synthesis, Cambridge University Press, USA
[8] Cetin O. L. and Saitou K., 2004, “Decomposition-Based Assembly Synthesis for Structural Modularity” ASME Journal of Mechanical Design, 126, pp. 234-243.
[9] Chomsky N. 1957, “Syntactic Structures”, Mouton, The Hague
[10] Cetin O. L. and Saitou K., 2004, “Decomposition-Based Assembly Synthesis for Maximum Structural Strength and Modularity” ASME Journal of Mechanical Design, 126, pp. 244-253.
[11] Finger S. and Rinderele J. R., 1989, “A transformational Approach to Mechanical Design using a bond Graph Grammar”, In Proceedings of First ASME Design Theory and Methodology Conference, ASME, New York
[12] Gips, J. and Siny, G. (1980) ‘Production systems and grammars: a uniform characterization’, Environment and Planning B:Planning and Design, Vol. 7, pp.399–408
[13] Göttler H.,1988, “Graphgrammatiken in der Softwaretechnik”, Springer Press, Berlin
[14] Haq M. and Rudolph S., 2004, “”EWS-Car”: A Design Language for Conceptual Car Design”, VDI-Berichte 1846, “Numerical Analysis and Simulation in Vehicle Engineering”, pp. 213-237.

[15] Haq M. and Rudolph S., 2005, “A Design Language For Generic Space Frame Structure Design”, IFIP TC5 Working Conference on CAI, pp. 201-215, ULM, Germany.
[16] Haq M. and Rudolph S., 2006, “Design Acceleration of Automotive Structures by Structure Design Language”, Proceedings of 2006 Conference on “Virtual Product Development (VPD) in Automotive Engineering”, Munich, Germany.
[17] Irani M. R. and Rudolph S., 2003, “Design Grammars For Conceptual Design of Space Stations”, Proceedings of IAC International Astronautical Congress, 2003, IAC-03-T.P.02, Bremen, Germany
[18] Kormeier T., Alber R. and Rudolph S., 2003, „Topological and Parametrical Design of Aircraft Surfaces using Design Grammars“, Proceedings of DGLR Symposium German Airospace Congress, Munich, Germany
[19] Kröplin B., Kungl P., 2005, “Verbundforschungsvorhaben Airchain, Autonome hochfliegende Plattformen”, Abschlussbericht für das Arbeitspaket ISD III, Integration und elektrische Vernetzung Bordsystem, www.isd.uni-stuttgart.de/airchain/
[20] Lindenmayer A. and Prusinkiewicz P., 1996, “The Algorithmic Beauty of Plants”, Springer
[21] Lyu N., Saitou K. “Decomposition-Based Assembly Synthesis of A 3D Body-In-White Model for Structural Stiffness”, Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI. USA
[22] Lyu N., Saitou K. “Decomposition-Based Assembly Synthesis of Space Frame Structures using Joint Library”, Proceedings of ASME Design Engineering Technical Conferences, DETC2000-14533, Baltimore, Maryland, USA
[23] Mathematica 5.1, Wolfram Research, www.wolfram.com
[24] Nishigaki H., Nishiwaki S., Amago T., Kikuchi N., 2000, “First Order Analysis for Automotive Body Structure Design”, Proceedings of ASME Design Engineering Technical Conferences, DETC2000-14533, Baltimore, Maryland, USA
[25] Pahl G. and Beitz W., 1996, “Engineering Design, a systematic Approach” second edition, Springer, Berlin
[26] Prats M., Jowers I., Garner S. and Earl C., 2004, “Improving Product Design Via a Shape Grammar Tool”, Proceedings of International Design Conference - Design 2004, Dubrovnik, Croatia
[27] Schaefer J. and Rudolph S., 2005, “Satellite Design by Design Grammars”, Aerospace Science and Technology,2005, 9, pp 81-91
[28] Schmidt L. C. and Cagan J., 1996, „Grammars for Machine Design“ Proceedings of Artificial Intelligence in Design 1996, Stanford
[29] Schmidt L. C., Shetty H. and Chase S. C., 2002, “A Graph Grammar Approach for Structure Synthesis of Mechanisms”, ASME Journal of Mechanical Design, 122, pp. 371-376.
[30] Schumacher Axel, 2004, “Optimierung mechanischer Strukturen, Grundlagen und industrielle Anwendungen”, Springer-Verlag, Heidelberg, Berlin, Germany
[31] Shea K. and Cagan J., 1998, “Generating Structural Essays from Languages of Discrete Structures”, J. S. Gero and F. Sudweeks (eds), Artificial Intelligence in Design 1998, pp. 365-385, Kluwer Academic Publishers, Netherlands
[32] Sridharan P. and Campbell M. I., 2004, “A Grammar for Function Structures”, Proceedings of ASME Design Engineering Technical Conferences, DETC2004-57130, Salt Lake City, USA
[33] Starling, A. and Shea, K. ,2005 ‘A parallel grammar for simulation-driven mechanical design synthesis’, *ASME 2005 Internationsl Design Engineering and Technical Conference*, Long Beach, California, USA, published by ASME, 2005, (DETC2005-85414)