Computer Modelling and New Technologies, 2005, Vol.9, No.2, 49-55
Transport and Telecommunication Institute, Lomonosov 1, LV-1019, Riga, Latvia

TOWARDS APPLYING THE METHOD OF AGGREGATIVE FUNCTIONAL DESIGN IN MULTI-USER ENVIRONMENT

E.S. NAPALKOV, V.V. ZARS

CAD Center,
Department of Engineering Science and Transport, Riga Technical University,
Str. Ezermalas 6, Riga LV-1014, Latvia; e-mail: napalkov@latnet.lv

The paper is dedicated to the development of a method for conceptual design based on virtual exchange of users knowledge to be represented in form of the so-called aggregative design components. In order to support aggregative components a frame-based structure of the abstract product is used.

Keywords: *Aggregative Design Component, Abstract Product Structure and Fuzzy Logical Retrieval*

1. Introduction

Advanced technique of synthesizing conceptual design solutions is mainly based on application of methodology of functional design [1]. General statement is to link design functions with the product behaviour, and then this behaviour with the product structural (physical) embodiment to achieve the overall design specification. However, there are some problems of functional design that have not yet explored sufficiently. One of these is design in dynamic environments, in particular, under conditions of Internet/Intranet environment by virtual exchanging knowledge being taken from private experiences of involved partners. The main barrier is incompatibility of information formats used in different Product Support Systems. Moreover, knowledge about emergence of valid information for the current design case is not obvious when data exchange occurs. Therefore the transferred information needs to be managed.

This trend touches upon many questions concerning methods, infrastructure and organization for product data sharing, for distributing work, for detecting and resolving conflicts. The critical issues in sharing and managing design information are:

- patching heterogeneous databases together to create some middleware infrastructure;
- developing a single product data model to create homogeneous design environment.

The first approach is used to exchange product data in a federated manner, while as each of involved partners solves certain subtask in general task of collaborative design [2]. In case of using a single tool for CAD-files generation and visualization (e.g. the VRML format) the involved partners enable to interact with each other through accessing solutions designed. A major disadvantage is that functional structure and a product configuration has to be conformed previously. The second approach is based on global databases concept, standard data packages, or agreed ontology of the design domain needed in significant implementation and customisation effort [3]. Therefore the development of more flexible collaboration models for conceptual design is necessary.

Our analysis revealed that one could support both individual and collaborative design process relying on the federated conception within so-called the abstract product structure (APS). The proposed scheme of user knowledge exchange is represented in the next section.

2. Representation of abstract product structure

In the proposed scheme, APS is considered as a multilevel frame-based network constructed as a result of functional decomposition of some design domain. This enables the users to represent explicitly and to share own knowledge with each other over instances of describing the selected class frames. The feature is a support of alternative views on functional and structural properties of the same design artefacts (design components). So, a common interest of all involved partners is to intensify these knowledge exchanges to reuse them for deducing inferences and improving a quality of products being designed for own needs.

The development of APS is based on distributing all design concepts into five functional layers corresponding to definition of: machine part design features (the 1st layer *C*); machine parts (the 2nd layer *D*);

subassemblies (the 3rd layer *S*), and assemblies (the 4 layer *A*). Each of these layers comprises finite number of class functions used for modelling functionality of associated design concepts. The highest 5 layer defines the overall function of the design domain investigated.

In particular, analysing different types of transferred energy in the mechanic domain *M (*force, torque, pressure, temperature) we have categorized functions of all assemblies into four class functions such as *transference*, *fixing*, *adjusting* and *driving.* Analysing different types of mechanical relations between interacting parts we have categorized functions of all subassemblies into ten class functions such as *transference/move*, *transference/rotate*, *fixing/position*, *fixing/seal*, *adjusting/limit*, *adjusting/control*, *driving/rotate*, *driving/move*, *catching* and *mating*. For functional classification of machine parts, seven classes have been established such as *rotating*, *moving*, *limiting*, *sealing*, *positioning*, *controlling* and *fastening*. At last, for functional classification of machine part features, their influence upon definition of a machine part working function, basing schemes and receptivity to external power loads has been taken into account. As a result, we have restricted by nine class functions of design features such as *operating/move*, *operating/rotate*, *operating/limit*, *reinforcing*, *basing* and others.

A primary model of APS is constructed by establishing rigid hierarchical relations between class functions of adjacent layers. This implies the definition of so-called *abstract class frames* as some two-layer (*A-S*), (*S-D*), and (*D-C*) hierarchical structures, each of which includes the vertex class function and its subclass functions of lower layer [4]. Duplicated from the experimental WEB-page a partial list of such abstract class frames is shown on Fig. 1. It is determined that a number of subclass functions in any abstract class frame can be restricted by four ones to interpret unambiguously the most essential functional and structural properties of existent design components.

The interpretation process consists in using slots such as *Concept Name*, *Image Name*, *Instance Function Name, Instance Function Entity* and others to represent the design component sequentially in the following forms:

- the *conceptual class frame* related to the selected abstract class frame;
- the instance frame related to the selected conceptual class frame.

Add Edit Delete Function Classes: Filter

No	Function Class	Layer	SubClass 1	L1	SubClass 2	L2	SubClass 3	L3	SubClass 4	L4	Select
1	Adjust/Control	S	Controlling	D	Moving	D	Positioning	D	Rotating	D	○
2	Adjust/Limit	S	Limiting	D	Moving	D	Positioning	D	Rotating	D	○
3	Adjusting	A	Adjust/Control	S	Adjust/Limit	S	Catching	S	Mating	S	○
4	Basing	C									○
5	Catching	S	Limiting	D	Fastening	D	Moving	D	Rotating	D	○
6	Controlling	D	Basing	C	Operate/Control	C	Reinforcing	C			○
7	Driving	A	Driving/Move	S	Driving/Rotate	S	Catching	S	Mating	S	◉
8	Driving/Move	S	Moving	D	Limiting	D	Controlling	D	Positioning	D	○
9	Driving/Rotate	S	Rotating	D	Limiting	D	Controlling	D	Positioning	D	○
10	Fastening	D	Basing	C	Operate/Fasten	C	Reinforcing	C			○
11	Fix/Position	S	Positioning	D	Fastening	D	Limiting	D	Moving	D	○
12	Fix/Sealing	S	Sealing	D	Limiting	D	Fastening	D	Moving	D	○

1 2 3

Figure 1. Representation of abstract class frames

At first, it is required to interpret general behavioural aspects by describing the design component in terms of design concepts to be shared between the vertex and subordinate class functions of the selected abstract class frame, for example:

CFR [<Driving, *Hoisting mechanism*>, <Driving/Move, *Driven drum of winch*>,
<Driving/Rotate, *Electric motor*>, <Catching, *Reduction gear*>, Mating, *Brake clip*>],
where *CFR* is a name of conceptual class frame.

Secondly, it is necessary to provide design concepts for instance functions and graphical images in accordance with functional and structural decomposition of the design component, for example:

IFR [<Hoisting mechanism, *Image*, *Provide a hoisting the palletised loads*>,
< Driven drum of winch, *Image, Provide a feed motion of winch cable*>,
<Electric motor, *Image*, *Create a torque on main drive shaft*>,

<Reduction gear, *Image*, *Transform a torque on driven shaft*>,
<Brake clip, *Image*, *Reduce a rotary inertia of drive shaft*>],
where *IFR* is a name of instance frame.

Depending on a purpose of design each graphical image can display a real physical structure, sketch or schematic circuit with application parametric equations, computing algorithms and other mathematical models intended for behaviour simulation of the design component. Therefore, template facets with attributes of function entities (related to instance functions) and design concepts are used to complete the interpretation process. In this case, the user assigns a value space for the most important functional and design attributes. As a result, the interpreted design component is transformed into two-layer function/means structure consisting of the vertex design component and appropriate subcomponents of lower layer (or design organs). These function/means structures can be called by *aggregative design components* (ADCs). Based on user knowledge capture they are stored in separated library.

In synthesizing design solutions, ADCs have to play a role of solution elements. In fact, it required modifying the functional design approach (FD-technology) to develop the method of so-called *aggregative functional design* (AFD-technology).

3. Mathematical model of AFD process

In order to outline main features of the proposed method it is advisable to describe general mathematical model of functional design process. As is known, it is aimed to find those design components whose output functions can achieve the desired output functions. Given a library of design components associated with simple behaviours this task is solved starting with decomposing of the purpose function inquired in the user functional requirements (query).

Let C denote general set of design components and $F(q)$ denote a set of functions inquired in query q; $C = \{c_i\}$, $i \in I$. Then, mathematical model of synthesized function/means tree (in general case, we deal with the graph with contours) can be represented as follows:

$$G = < H, F(q), \psi, R >, \quad (1)$$

where design components $H \subseteq$ C are nodes of the graph; $F(q)$ are its edges; ψ is an incident or that identifies subsets of nodes $H(f)$ incidental with an edge f ($\forall f \in F(q)$); R is subjective mapping on subsets $H(f)$ so as for each component $c_i \in H(f)$, a set of relevant components c_i^g can be found; at that $g \in N^+$, and $N^+ = \{1,2,\ldots\}$.

The graph G is oriented one. It allows one to determine input-output relations on a set H of design components. The certain disadvantage is a danger of initiating information explosions that can lead to generating entire forest of function/means graphs.

In the considered AFD process, the graph G is structured from ADCs enabling to achieve the design solution in smaller number of iterations. Let S_i be a subset of ADCs connected with a use of the i-th design component; $S_i = \{s_{ij}\}$, $j \in M_i$. It follows that we can represent the synthesized graph in extended form like:

$$G^i = < H^i, F^i(q), \psi^j, R^i >, \quad (2)$$

where $H^i = \bigcup_i \bigcup_j S_i$; $F^i(q)$ is an extended set of edges such that $H(f) = \bigcup_i H^i(f^i)$, $\forall f^i \in F^i(q)$;

ψ^i is an extension of ψ on sets H^i and $F^i(q)$, R^i denotes uncertain subjective mapping.

Thus, in AFD process, the main emphasis is placed of on performing transformation as follows:

$$< H^i, F^i(q), \psi^i, R^i > \rightarrow < H, F(q), \psi, R >, \quad (3)$$

in which two design libraries should be used, such as:
- ADC library for generating function/means structures;
- solution library for optimising function/means structures.

The first one requires the application of auxiliary procedures to avoid an appearance of design solutions with redundant structures. The second one enables at once to select the solution prototype with an irredundant structure for the present situation. AFD-process in more detail is described in the next section.

4. Synthesis of design solutions

In contrast to functional design the AFD process is based on combining of "top-down" and "bottom-up" steps for deducing inferences. These steps are the following:
- retrieving ADCs for the purpose function inquired ("top-down" step);
- retrieving vertex components for the design organ selected ("bottom-up" step);
- coupling relevant vertex components ("top-down" step);
- correcting a structure of design solution being synthesized ("bottom-up" step).

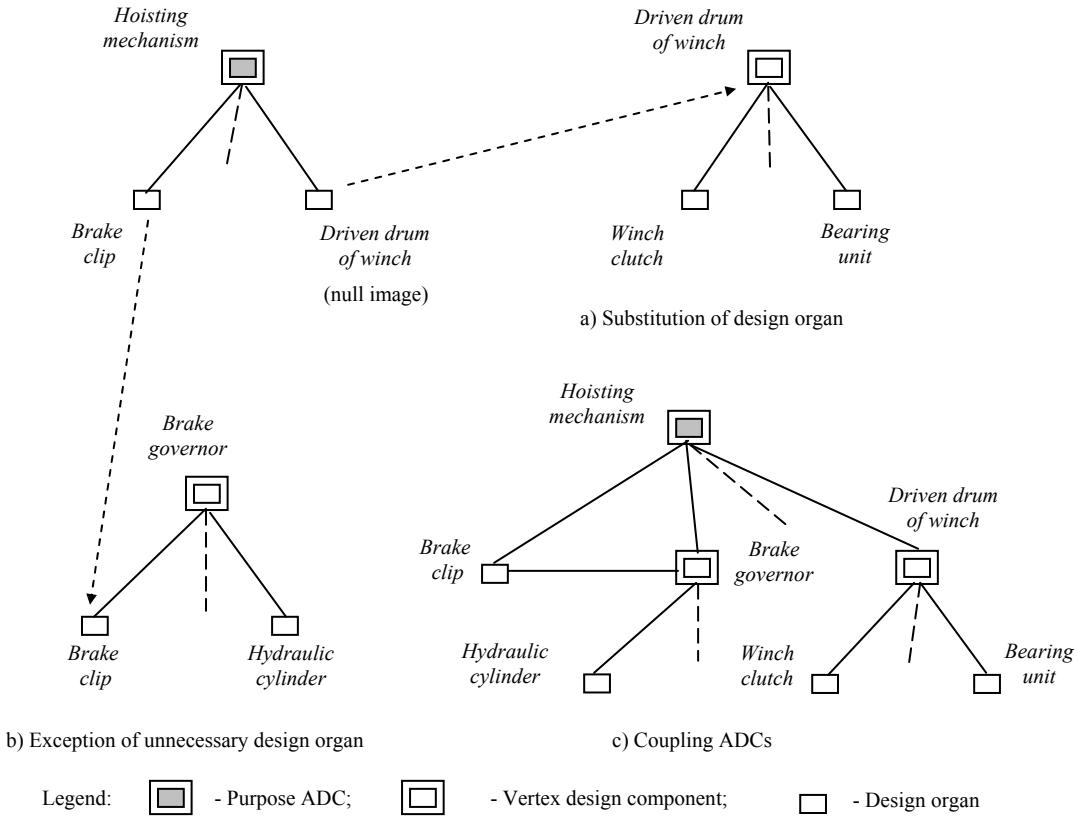


Figure 2. Deducing inferences from ADCs

At the first step the most preference is given to those ADCs whose organ structures contain sub-functions inquired by the user. For retrieval, a library of ADCs is used. Here, one should have in mind that all inquired sub-functions are to belong to the (k–1)-th description layer with respect to the design problem being solved on the k-th description layer. This means that the designed assembly will include only subassemblies and other assemblies, the designed subassembly will include only parts and other subassemblies, and the designed part will include only design features and other parts. If exists some ADC that satisfies all functional requirements, then an attempt is made to retrieve its real prototype in a case library by comparing with the design constraints given. Otherwise, a message is generated about missing or contradictory design parameters to create new prototype.

In order to satisfy uncovered functional requirements the second step concerned with extension of current ADC is executed. For that it is necessary to retrieve additional ADCs whose organ structures are set intersected with current ADC at least by one design organ. The selected organ can have either null or complete graphical image. Null image means that the design organ is defined on (k–1)-th description layer as only the design concept, while it's the behavioural properties should be found on k-th description layer. Thus the design organ is becoming the vertex component of additional ADC. Restricted to reductive description of behaviour (in form of conceptual class frames) such example is shown on Fig. 2a. Other example shown on Fig. 2b is related to finding vertex component for the design organ with

complete graphical image. In both examples, the main goal is to select the most preferable ADCs that comprise maximal number of uncovered sub-functions. For retrieval of possible prototypes, the case library can be also used.

The third step is aimed to verify the compatibility of ADCs within a structure being synthesized. According to underlying principles of functional design the connections between design components are specified as input-output relations between their functions. Two components can interact with each other on condition that their functions have the same entity. Therefore it is necessary to analyse entity attributes before two components can be connected together.

The described AFD process solves this task by analogy. The feature is a use of such procedure for only vertex design components. In order to narrow a seek area of design solutions the functional requirements must include entity attributes for each function inquired.

The forth step allows one to modify relations between ADCs coupled. It involves the procedures of substitution and exception of unnecessary design organs. In particular, the substitution procedure is performed with respect to design organs that have null description (Fig. 2a). The exception procedure enables to remove one of two repeatable design organs used for coupling vertex components. As shown on Fig. 2b it leads to resulting in contours into the graph synthesized. Other application of the exception procedure may be a remove of design organs that do not contain sub-functions indicated in requirements. The resulting graph for a modified structure consisting of three ADCs is shown on Fig. 2c.

The described AFD process is cyclic. It is finished when one succeeded in covering all sub-functions inquired. Otherwise, it is necessary to reject the first-choice hypothesis concerning the purpose design fact (in our example it is called by *Hoisting mechanism*) to continue the AFD process by finding other ADC for the purpose function inquired.

5. Retrieval of relevant ADCs

In reality, the described AFD process is one of possible scenarios that the user can select in synthesizing solutions by means of the Reasoning Engine within a frame-based APS. Other scenario (a backward approach) is to synthesize design solutions starting with analysis of given sub-functions to work then a way to the purpose function. Therefore, the development of effective search procedures for retrieving relevant information is required. Since we deal with multiple class frames in APS, it is better to use the fuzzy logic for removing partial uncertainty occurred in selecting relevant ADCs [5].

Suppose that the user query q specifies the relationships $F(q) \subseteq P \times F$, where P is a set of entity attributes p and F is a set of functions f to be satisfied. Let $((p,c),\omega)^{\gamma} \in [0,1]$ be a weight of the pair $(p,c)^{\gamma}$ in which the entity attribute p is matched with the design organ c belonging to γ-th function, $c \in C$. Then the total weight $(c,\omega)^{\gamma}$ of the design organ c about the query q can be represented in form of the Euclidean distance $d\,(c,q)^{\gamma}$ by identifying the specified attributes p of the design organ c.

Allow a set of such design organs $C^{\gamma} = \{(c,\omega)^{\gamma} / c\}$, and the membership function μ: $C^{\gamma} \times C^{\gamma} \to [0,1]$ to evaluate the similarity degree $\mu\,(c,\omega)^{\gamma} = ((c,\omega), l)^{\gamma}$ of each design organ with respect to the user query q, where l is a value of linguistic variable, $C^{\gamma} \subseteq C$.

For that, we divided a set C^{γ} into clusters according to three terms of linguistic variable such as *little-suited*, *almost acceptable* and *acceptable* design organs about the query. These terms and values of linguistic variable l evaluate the similarity degree of design organ c by evaluating its membership degree with each of above-mentioned clusters. For clustering, as shown on Fig.3a, we used two trapezoidal and one triangular membership functions with three critical points (0,2, 0,5, 0,8) within the common interval [0,1] of measuring the Euclidean distance.

Further, let $(((c,\omega), l),^{\gamma} s)^{\eta}$ denote membership degree of design organ c in organ structure of the s-th ADC that belongs to η - th function, $s \in S$. Then the total membership degree $((s,\omega), l)^{\eta}$ of the s-th ADC can be computed by using the superposition operation as following:

$$((s,\omega),l)^{\eta} = \sup_{\gamma} \{(((c,\omega),l)^{\gamma}, s)^{\eta}\} \quad , \tag{4}$$

where superposition operation includes the procedures of interpolation and defuzzification of input membership functions.

For our goal, it is sufficiently to use only one output term represented by a truncated triangular membership function on Fig. 3b. Therefore the restricted number of basic fuzzy rules (like IF – THEN) is required to set all combinations for weighting design organs. One of such general combinations is the following:

IF ((*c*, *ω*), *Appl.*)^1 ∧ ((*c*, *ω*), *Appl.*)^2 ∧ ((*c*, *ω*), *Alm.*appl.)^3 ∧
∧ ((*c*, *ω*), *Little-suit.*)^4 *THEN* ((*s*, _), _),

where upper indexes correspond to listing design organs in organ structure of ADC evaluated.

In the process of defuzzification, a computed value of output membership function is automatically transformed in Euclidean distance, for example:

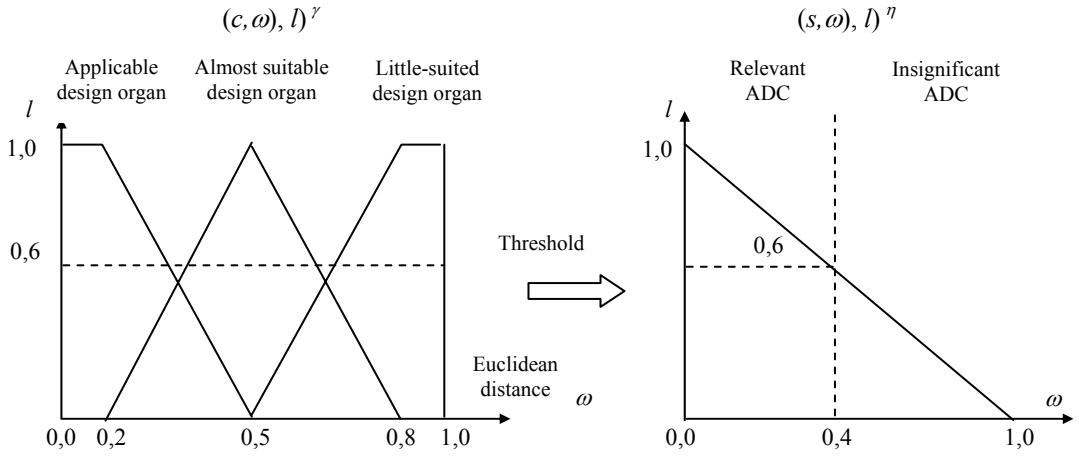
IF ((*c*, 0.18), *Appl.*)^1 ∧ ((*c*, 0.12), *Appl.*)^2 ∧ ((*c*, 0.45), *Alm.*appl.)^3 ∧
∧ ((*c*, 0.80), *Little-suit.*)^4 *THEN* ((*s*, 0.489), *Less accept.*)

that corresponds to selecting ADC with insignificant organ structure (subject to all membership degrees are computed with a threshold of applicability equal to 0.6).

Thus, in order to select relevant ADCs it is necessary to aim at:

$$((s,\omega),\ l)^{\eta} \le (s,0{,}4)^{\eta}, \qquad (5)$$

where $\omega = 0{,}4$ is an extreme value of Euclidean distance defuzzificated.
Besides, one should proceed from the assumption that vertex components of ADCs certainly satisfy the input functional requirements. It follows that the final task can be a finding optimal design solution (*S**, *W**) with minimal weight-average Euclidean distance *W* about the query *F(q)* on total set of solution variants $\{(S,W) / S\}$, $S^* \subseteq S$.



a) Input membership functions

b) Output membership function

Figure 3. Membership functions used in fuzzy logical rules

In order to avoid the complete enumeration of possibilities it is required to generate additional matrixes that specify relations on pairs of elements (*F*(*q*), *S*) and (*S*,*C*). In this case, optimal design solution is achieved in depending on total set of design organs used. Formally, this task can be referred to a finding optimal coverage for three-fraction graph specified on these matrixes.

6. Conclusions

In this paper we have proposed an approach for individual and collaborative conceptual design of mechanical products in multi-user environment. Allowing the notions of abstract product frame-based structure and the aggregative design component we have elaborated the way for sharing the user functional and structural knowledge over the system of product data virtual management. The use of these notions has led to the development of AFD-technology based on combining and modifying the methods of functional design and case-based design in deducing inferences.

It is indicated that the main advantages are:

- essential reducing iterations in the process of structuring design solutions;
- more easy adoption of design solutions in the process of their prototyping.

We believe that the proposed method of conceptual design offers good opportunities especially for SME to build virtual cooperation and thereby maintain competitiveness within the global market.

References

[1] Pahl G., Beitz W. (1996) *Engineering Design – A Systematic Approach*. 2nd edition. London: Springer Verlag.

[2] Huang G.O., Mak K.L. (2001) Issues in the development and implementation of Web applications for product design and manufacture, *Journal of Computer Integrated Manufacturing,* **14**(1), 125-135.

[3] Wang L., Shen W., Xie H., Neelamkavil J., Pardasari A. (2002) Collaborative conceptual design: a state-of-the-art survey, *Journal CAD,* **34**(13), 981-996.

[4] Napalkov E., Zars V. (2003) Model of Graphical Interpretation of Fuzzy Design Patterns, *Proceedings of Riga Technical University,* **5**, 89-96.

[5] Napalkov E. (2000) Knowledge Capture and Reuse. In: *Proceedings of Nordic-Baltic-Russian (NBR'2000) Summer School "Applied Computational Intelligence".* St-Peterburg, 192-207.

Received on the 21st of June 2005