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

THE MAIN MYSTIFICATIONS INGRAINED IN ENGINEERING DESIGN

Vladimir Sedenkov
Belarusian State University

ABSTRACT

Amazing phenomenon of engineering design is that it manipulates with non-existent and yet to be called into being entities in the same way as with real objects (products and processes). Thus a product design is structured like available product; design process, which will be completed only after design obtaining, is simulated similarly to real processes; and design pseudo-problem is decomposed into subproblems in the same manner as a soluble problem. In this paper, we have dwelt only on the major, in our view, myths, namely design problem realization, product and design process models. But those are tailed with unavoidable derived mythology. The family of myths substantially affects the progress in design theory and methodology. The goal of this paper is to compare two ways – to follow the myths and to abjure those. After presentation the next myth, its origin and effects, we propose a possible anti-myth and expose related prospects.

Keywords: Design problem, meta design process, platform kit

1 INTRODUCTION

It should be noted that mythology phenomenon in engineering design has a growth trend: the synthesis is repeatedly treated as inverted analysis [1], thereupon the theory of technical systems (analysis) gradually turns into design theory (synthesis) [2]. To examine the mythology sources, let us engage an ancillary concept of two interactive and in possession of common knowledge but disjoint worlds – *real* and *virtual*. Each world is represented by its *intrinsic objects* and communication facilities with the concurrent world or *nonintrinsic objects*.

Real world is the percept, subject of inquiry and transformation into more comfortable, safe, diverse and beneficial world. Basic processor in this world is a human being (*H*, informal processor). Intrinsic objects of the real world are people, products, processes, services, relations between all of those and representations of these entities. Topical activity for intrinsic objects are analysis, modeling, optimization and employment.

Nonintrinsic objects of the real world are *tasks* (needs) and *designs* (realizable abstractions of the future). The tasks are outgoing (forwarded), while designs are ingoing (received) nonintrinsic objects. Topical operation for designs is transformation into intrinsic objects of the real world, i.e. physical synthesis (expansion of the world).

While the real world is considered (in the context of designing) as designs implementator, the virtual world serves for designs incubator, where the part of a master processor performs computer (*C*, formal processor).

AXIOM 1: Just as a product (process, service) is obtained through implementation of a product design, so a product design should be obtained through implementation of product's design of design: *design of design* (ML_n) $\rightarrow$ *design* $\rightarrow$ *artifact*.

An extension of the above string to the left (design of design of design, etc.) is replaced with the sequence of design-of-design maturity levels (ML_i, $i=0,1,2,\ldots,n$). This axiom nominates intrinsic objects of the virtual world – *design-of-designs*. Topical operations for these objects are synthesis of design-of-design terminal state (ML_n) and its transformation into design. Design-of-design representations have been intrinsic objects of the virtual world, as well.

Nonintrinsic objects have here inverse roles relatively to those they have in real world: the tasks serves for ingoing or received objects, while designs are outgoing or forwarded to real world objects. Topical for tasks is transformation into intrinsic objects (representations) of the virtual world. Basic principles of the two worlds coexistence are as follows.

1. Intrinsic *objects* of one world cannot become intrinsic objects of another world. This also means that representations (data structures) of intrinsic objects in real world are irrelevant for the virtual world, and vice versa (that is abstractions for the available present – particularly hierarchies – are unacceptable as abstractions of the absent future).
2. *The categories* of the real world – structuring, modeling, decomposing, etc. – are irrelevant for intrinsic objects of virtual world, and vice versa.

Thus, the design engineering mythology may be considered as the consequence of disregard for the stated principles (transfer of methods, objects, representations and categories from the *H*-centered world of discrete analysis into processor-independent – formal or informal – world of continuous synthesis) or, what is the same, of the lack of the virtual world concept itself.

In any case, the world intersection entails a progress in aberrant tendency, which is observable for a long time, but attempts to counteract it are still ineffective. Thus universities cherish hope of a teachable, learnable and holistic design process (*DPR*), as well as of design language, which will change the present semi-intuitive one. Design practice needs for the proper computer support of *DPR*, guidelines on how to counteract the growth of *DPR* complexity, and on design methods and knowledge structuring to apply those efficiently.

At the same time, design theory is weakened by confusion of research themes. Creative thinking and innovation remain to be external to design science core [3]. Besides, the pure *H*-orientation of all design science subsections does not leave niches for impactful (on partner basis) interaction with computer. This situation just conduces to mythology growth.

In the first part of the paper, we briefly present the fundamentals of Continuous Process Theory – a specially made basic formalism, which runs through the entire paper content. Every sequent section presents the next myth, its genesis and consequences, whereupon a possible anti-myth is discussed. The general discussion is presented in Section 6. In conclusion, we consider one of the prospects in post-mythological engineering design: the end-to-end platform approach to design affairs management – from a unified design platform to the platform of design science.

2 CONTINUOUS PROCESS THEORY – THE FORMAL TOOL AT HAND

The starting point of the theory is representation of any process *PR* by its scheme:

$$PR=(D,\ P), \tag{1}$$

where *D* is a procedure to produce the process result, and *P* is a processor intended to perform *D*. A set of processes (processes schemes) is added with three relations, coupling any two members of the set:

- *providing relation* (or *p-relation*): an output of PR_2 serves for the input of PR_1 ($PR_2 \xrightarrow{p} PR_1$);
- *relation of determination* (*d-relation*): the output of PR_2 is a new state of $D \in PR_1$ or $P \in PR_1$ ($PR_2 \xrightarrow{d} PR_1$);
- empty or *λ-relation* (PR_1 and PR_2 are mutually independent).

Non-empty relations are used to make continuous structures out of processes – the *n-order* structures (here $n=\overline{1,3}$). For instance, conditional PR_1 requires determination of its *D* and *P* via performance of respective processes: *"search for D"* (*SD*) and *"search for P"* (*SP*):

$$SD \xrightarrow{d} PR_1 \xleftarrow{d} SP \tag{2}$$

PR_1 is the *core* of the structure while *SD* and *SP* form the structure's *tail*. Notation (2) is the example of the *first order structure* (*n=1*). The structures with *n=1* serves for the elements of the *second order structures*, shaped by the alternative relation. One of the motives for structure formation may be as follows.

Associate with each process scheme a level of its *uncertainty* (*UL*) as *UL* of scheme's components.

- A process, which has *UL=0*, is called *physical*: its *D* and *P* are real.
- *UL=1* corresponds to a *logical* process: its *D* and *P* have descriptions sufficient for their physical implementation.
- A *virtual* process has *UL=2*: its *D* and *P* exist only as mental images.
- *UL=3* is assigned to a *conditional* process (PR^C): its result has been declared but *D* and *P* are presented by their symbols only.

Constructive proof of logical runability for PR^C consists in stepwise reduction of its *UL*. A step of reduction is referred to as *determination* of conditional, virtual or logical process. While two-stroke determination of PR^C, the objective of the *virtual* (downward) determination is the reduction $UL=3 \rightarrow UL=2$; during the second or the stroke of *logical* (upward) determination, the reduction $UL=2 \rightarrow UL=1$ will take place. The outcome of this two-stroke determination cycle of PR^C is a *super-tree* (*S*-tree) – an arc-bichromatic graph, where each *S*-node is an ordinary tree (Figure 1).

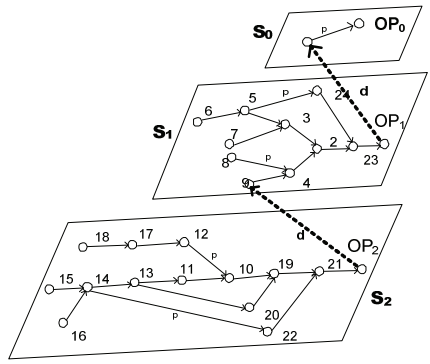

Figure 1. The fragment of the third order structure out of processes

3 THE MYTH NO. 1: PRODUCT MODELING

There is no stable concept from science how a product should be modeled – neither in general, nor for a specific application context [4]. Moreover, the most widely spread concept – hierarchical product decomposition: systems, organs, parts – does not clear the air in the case of whether this model is initiated by design process or the model itself gives birth to *DPR* stages (conceptual, embodiment, detailed design) [5].

But, omitting theory windings, let us see the matter in a different way. System analysis, outside of any relation to design process, uses hierarchical representation of available product for its ends anyway. It turns out that the same representation is borrowed by *DPR*. Hence, the observed naked truth is that the model used for analysis of a (*existent*) product serves for the model of synthesis of (*inexistent*) design. How valid is such borrowing?

3.1 The properties and effects of product structure hierarchical representation

Hierarchical decomposition of an available product is oriented, in the first place, to its analysis. The subject of such analysis is a human being (*H*). One more property of the hierarchy "systems, organs, parts": this is the data structure with high level of discrecity (or, in terms of [6], with high level of granularity chosen for abstracting parts and wholes).

At the same time, the main activity in product development is synthesis. This highly iterative procedure needs another assistance of computer (*C*) in contrast to analysis. Besides, this procedure should be explicit in view of intensive participation of *C*. However, the use of model inherent in object analysis to perform design synthesis eliminates an active part for *C* (*H*-orientation of the model "systems, organs, parts") and leaves synthesis implicit at any level of hierarchy.

The foregoing shows the need of separation of object and design models: *H*-centered object models are intended for object analysis or its physical implementation, while the intended purpose of *P*-independent design models is design synthesis. The set of models both intended for object analysis and of particular design states (spatial models, for instance) is replenished by methods developers and IT suppliers. A synthesis-oriented model, i.e. the model of design dynamics, still remain to be constructed.

3.2 Anti-myth: design dynamics modeling

Synthesis-oriented design representation, waiting to put in existence, has to ensure:

- systematic usage of *C* (hence, to be a *P*-independent data structure);

- design evolution mapping (a specific design progress concept is necessary);
- explicit synthesis (to be continuous design representation).

Just as systems, organs and parts have been the *object's categories* with relationship that shapes a model of available object, so design model construction needs *design's categories* with their relationship (the use of categories of yet inexistent object is inconsistent).

Recall the sequence of design-of-design maturity levels (Section 1): $ML_1, ML_2, \ldots, ML_n$. This sequence is called *diachronic* (*dh*) or "historical" structure (*St*) of design-of-design and serves for it quasi-dynamic representation. Componentization of each ML_i is referred to as *synchronous* (*sh*) *structure* or semantics (*Sm*) of respective ML_i. If the semantics of each ML_i is determined, we have *approximation model* of the design-of-design (and design as well): *AM=(St, Sm)*. Otherwise, there will be a quasi-approximation model – *qAM=(St, Sm*)*, where *Sm** is abstract semantics.

Thus, we take for the design's categories, its maturity levels ML_i or states. Incident ML_i and ML_{i+1}, $i=1,2,\ldots, n-1$, are linked with *embedding relation* ($\prec$). In compliance with the conventional general design progress concept (design evolution), we link by this relation the following four design states (*design goals*): *Prototype* (PRT), *Market version* (ITM), *Manufacturing version* (COM) and *Artifact* (ART).

$$\mathrm{PRT} \prec \mathrm{ITM} \prec \mathrm{COM} \prec \mathrm{ART} \tag{3}$$

To smooth the discrecity of design representation in terms of its states, split the development of each state into four stages. Each stage results in *design subgoal*: qSYS (a minimal set of product units that validates the concept of goal attainment), *system* or SYS (the extension of *q*SYS with control functions), *quasi-design* or *q*DES (space layout of the SYS constituents) and *design* or DES (the *q*DES components are assigned with shape, materials, grades of finish and all necessary joints):

$$\mathrm{qSYS} \prec \mathrm{SYS} \prec \mathrm{qDES} \prec \mathrm{DES} \tag{4}$$

Combine all design states into its diachronic representation – the data structure named *quasi-hierarchy* (*q*-hierarchy) – by closing the nesting hierarchy, i.e. making the latter actual across horizontal as well. The resulted *dh*-structure is shown in Figure 2.

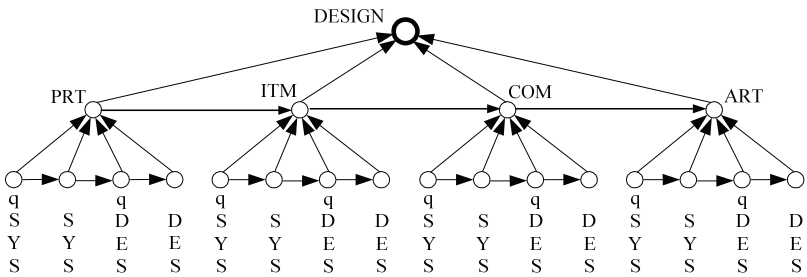

Figure 2. Primary dh-structure for a product design

Now clarify obtaining each of design states depicted in the *q*-hierarchy by a node. To this end, refine the generic concept of design development (evolution) till particular *design progress concept* or *DPC*. We take for the part of *DPC* the *evolution of individual*, that is adaptation of the current design state to a new state of *operation environment* of the sought-for object. The taken concept claims concurrent designing of a product and its operation environment (*OE*).

3.2.1 Product operation environment representation

OE is a family of processes $\{PR_q\}$, which a product deals with throughout its lifecycle. There are only three kinds of relations between a product and a process from the family: a product can play the role of the *input*, *disturbance* or *processor* with respect to any process from $\{PR_q\}$ (Figure 3*a*). Hence, we can break $\{PR_q\}$ up to three constituent sets: $\{PR_{q1}\}$, $\{PR_{q2}\}$ and $\{PR_{q3}\}$ (Figure 3*b*).

Processes from $\{PR_{q1}\}$ place on their input a number of *requirements* (*Rq*); this set is shared by the product life cycle stages (six in our case). Processes from $\{PR_{q2}\}$ impose *constraints* (*Cs*) on their disturbance. Processes from $\{PR_{q3}\}$ specify for their processor the *operation conditions* (*Cn*). In turn,

each constituent set is divided into motivated (we drop the details) number of subsets – the hierarchy in Figure 3*b*. As product-oriented *OE* should also be under design, it needs (Axiom 1) *P*-independent diachronic representation (hierarchies are *H*-oriented structures). Having taken the members of each set of processes (Figure 3*b*) for the vector of 3D space, we shape *dh*-structure of *OE* design. The latter is referred to as *&-cube* (Figure 3*c*).

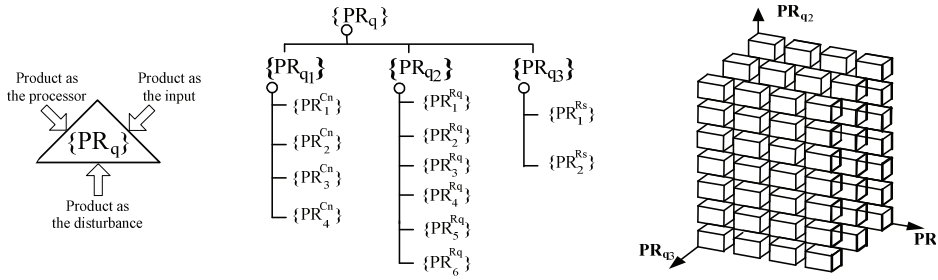

Figure 3. OE design dh-structure construction

Embedding relation links incident cells of &-cube along the *track* of its scanning. This track leads from the starting cell of *&*-cube – ($X_0Y_0Z_0$) – to its end cell – ($X_{max}Y_{max}Z_{max}$). Here are examples of tracks (brackets signify elements iterations along the track up to their depletion):

- *(X(Z)(Y(Z)))* – layer-by-layer &-cube scanning along axis *Y*;
- *(X(Y)(Z))* – all *Z*-vectors, except the last one, are omitted in vertical layers (reduced layers);
- *(X)(Y)(Z)* – reduced *&*-cube (omitted all *Y*-vectors, except the last one);
- *(X)(Z)(Y(Z))* – only the last vertical layer is taken wholly;
- *(X)(Z)((Y)(Z))* – the layer from the previous track is reduced.

Thus, the choice of *&*-cube scanning track defines also both relevant power of $\{\mathbf{PR_q}\}$ and variants of its structuring. These variants are also valid for the sets {Rq} and {Cs}, generated by $\{\mathbf{PR_q}\}$ members (*OE* processes), and consistent with concurrently derived design properties and characteristics. Incidentally, the obtained structuring for properties and characteristics is more insightful than in [7].

3.2.2 Design platform

Thus, there is no object design without design of its *OE*. To make these two the results of the same *DPR*, let us unify representation of *dh*-structure of both designs, leaving requirements of *P*-independence and continuity for the new representation in force. The unified *dh*-structure is obtained by substitution of *&*-cube for terminals in the above *q*-hierarchy (Figure 2). After that, design *dh*-structure is valid for both product and *OE* design, taking the status of $St \in qAM$. Whereupon, it takes only to refine $Sm^* \in qAM$ to come from *qAM* to approximate model of a product design (AM^P) and approximate model of *OE* design (AM^{OE}). The case of quasi-approximation design model is referred to as *design platform*: $qAM=(St, Sm^*)$. In general case, Sm^* is represented by the scheme of conditional (i.e. to be determined, see Section 2) process.

3.2.3 From design platform to product design

Thus, the transition from design platform $qAM=(St, Sm^*)$ to approximation model for a product design $AM^P=(St, Sm^P)$ consists in refinement of abstract semantics $Sm^* \in qAM$. Diachronic representation of $St \in qAM$ (the sequence of design states specified by the track of *&*-cube scanning) may be considered only as quasi-dynamics of a design. Design dynamics representation is obtained under stipulation that representation of $Sm \in AM^P$ is continuous. How to get it?

Any dynamics is associated with a process. Hence, the case in point is representation of each *dh*-structure element of $St \in qAM$ in terms of processes. This is possible if the content of any such element, from the very first one, should be the determination of the process scheme (Section 2) – namely, the scheme of *operation process* (*OP*) of the sought-for product. In this case, each *sh*-component will be specified by the structure of processes. This opportunity is presented by *continuous process theory*.

Thus, with the assumption that designer perceives the future product in terms of its operation processes, we shall try to display his reasoning by:

1. shaping the sequence of design-of-design states (*dh*-structuring);
2. choosing the number of states that provides designer with a comfort design increment ("the quantum of evolution") for explicit synthesis;
3. dynamic representation of these states (by a structure of processes for each ML_i).

Successful virtual designing of a product comes to an end with producing the design-of-design – dynamic (continuous) semantics *Sm* for the end element ($X_{max}Y_{max}Z_{max}$) in the last &-cube of $St \in qAM$ (Figure 4*a*). The obtained sequence of ML_i in whole has been AM^P – *approximate model* of the product design-of-design. Continuous semantics of each &-cube's cell is represented by *P*-independent data structure – *S*-tree of processes (Figure 1). Each super-node (S_i-node, $i=0,1,2,...$) of the *S*-tree is represented by an ordinary tree rooted with OP_i – an operation process of respective product constituent.

The terminal state of design-of-design, received in cell ($X_{max}Y_{max}Z_{max}$) of the last &-cube of AM^P, will be transformed into realizable design.

3.2.4 From design platform to OE design

$Sm^* \in qAM$ refinement while AM^{OE} construction consists in

- delimitation of *OE* borders in compliance with the object under design (selection of relevant phases in its life cycle and definition of &-cube scanning track);
- product ML_i-driven identification of the set of *OE* processes in each &-cube cell along the scanning track (cells attribution in &-cube from Figure 3*c*);
- cells attribution in &-cube from Figure 4*b* by {Cn}∪{Rq}∪{Rs} families, generated by processes from the corresponding cell in &-cube on Figure 3*c*.

The family members may conflict; in this case, an attempt should be made to pack them into the compromise Pareto set. The completed *OE* design-of-design has been the content of the terminal cell ($X_{max}Y_{max}Z_{max}$) along the &-cube scanning track.

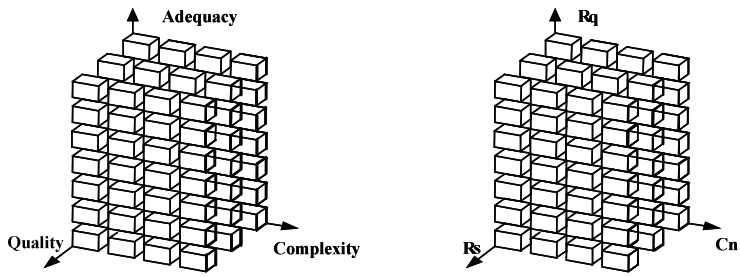

Figure 4. Basic members (&-cubes) of design dh-structure for a product – a), *and OE* – b)

4 THE MYTH NO. 2: FROM DPR MODEL TO DPR

Each designing event is unique in the sense that appropriate design process in its procedure manifestation has not been a replica of some previous *DPR*. Hence, a design process constructed off-line is a particular case with a restricted use. This situation is reflected by

AXIOM 2: Executable (procedural) design process does not exist a priori:
its shaping terminates concurrently with design obtaining.

There are four main questions that arise: What does *DPR* emerge from? What is the process of *DPR* construction? What is the subject and content of this process?

We call the process, which develops *DPR*, a *meta-DPR (mDPR)*: it shapes *DPR*, supports its realization and, therefore, indirectly draws the sought-for design. What this *mDPR* should be?

4.1 *mDPR* determination

4.1.1 Remarks on the notion on DPR model

Design process model is an outline of flow of work during design activity. The base for such a flow shaping may be composed of various aspects of product representation (constrains, life cycle phases, structure, features, etc.) [5, 8], personified experience and heuristics of a designer [9], technologies borrowed from other disciplines or even total design activity from the identification of the market/user need to the selling of the successful product [10].

The main peculiarities of *DPR* representation by a model are its close association with a product and disregard of design evolving characteristics as such. Among the rest character properties of *DPR* models, their incompleteness (by definition) and *H*-orientation. Add those with a brief comments.

Dependence on a product. Relations of the well known systematic *DPR* model [5] with the model of a product (systems, organs, parts) have predetermined the sequence of activities in the scope of *DPR* (conceptual, embodiment, detail design), which assumes availability of a real or virtual prototype for the product under design. If a prototype does not exist, subsystem separation (conceptual design) becomes a doubtful and unnatural procedure for early design stages: subsystems nomination and component binding (often not univocal) to one or another subsystem are rather typical for closing design stages.

Human orientation. DPR models are *H*-oriented. They can be interpreted up to executable *DPR* by a human designer only. *DPR* models entail disintegrated *DPR* in the case of relatively complex designs.

DPR model from a perspective of Continuous Process Theory. Within *CPT*, the part of process representation is performed by the two-component scheme of a process (Section 2), which has three levels of uncertainty corresponding to conditional, virtual and logical process. Thus, the *DPR* model in its ordinary sense takes up in *CPT* the place of object (procedure *D*) in the design process scheme – *DPR=(D, P)*.

4.1.2 mDPR determination: the common way

The object value of conditional *mDPR* (*D*=product representation transformation) predetermines for the part of the process input a product representation. In [5], for instance, a product is represented by the levels of its structural decomposition (systems, organs, parts). Then virtual $D \in mDPR$ is determined as a stepwise reduction of the level of abstraction in product representation (this accords with deriving of conceptual, embodiment and detail design respectively). In the absence of a prototype, the outlined scheme of design obtaining does not work.

In [11], a product is represented in another way – by a set (vector) of functional requirements. Virtual determination of *D* is seen here as the transformation of a given set of functional requirements into a defined set of design parameters. But this case does not much differs from the previous one in the main thing: in the absence of a prototype, the set of functional requirements becomes so vague that the way of design deriving becomes pointless if a product under design is not trivial.

As for determination of *mDPR* upon subject, it gives in both cases $P=H^C$ [12].

4.1.3 Another way of mDPR determination

While *DPR model* is the process conventional presentation, *DPR design* representation is validated by

AXIOM 3: Just as a physical product is the outcome of *product* design processing, so a product design has been an outcome of *design process design* implementation (processing).

Virtually, *DPR design* is a description of intended for physical implementation (or analysis) product design synthesis and/or progress concept (Design Formation Concept or *DFC* and Design Progress Concept or *DPC*, respectively). For instance, *DPR* design based on *DPC* conditionally qualified as "evolution of population" is presented in [13] and has been a very helpful instrument for a product design structural optimization.

The important properties of design representations have been their precision, continuity and explicitly stated design formation/progress concept. These properties are inherited and expanded by the technological process of the concept realization – *mDPR*: it becomes explicit, regular, *P*-independent and stimulant of intensive use of formal processor (*C*). In turn, such *mDPR* produces holistic, continuous, teachable and learnable product design process.

It should also be noted that the choice of *DPC*, built into the *DPR* design, predetermines the way of misalignment of product complexity growth and complexity growth of produced *DPR*. This possibility is unavailable in the scope of model approach.

4.1.4 Summary

1. None of the current design science sections takes account of the formal processor (*C*), which has nominally been a designer's partner but practically retains the status of the on-call tool. Partly it is because the synthesis is often tractable as inverted analysis and consequently inherits its technique. (Incidentally, such mechanistic synthesis hardly has any prospects.) But while the analysis takes the tool status of *C* as a norm, the iterative by nature synthesis is, on the contrary, very problematic without computer activism.
2. The focus of design theory and methodology should be on the constantly available *meta-DPR* instead of a priori unavailable *DPR*. The planned design system has also be primarily aimed at *mDPR* support, and design automation is associated exactly with *mDPR*, not *DPR*.
3. *mDPR*, giving rise to complex, non-holistic and ill-observable *DPR*, can not be considered as adequate. As these properties are stipulated by the properties of one or another *DPR* model associated with *mDPR* object part, the first-priority task has been the search for a proper (instead *DPR* model) cooperator for *mDPR*.

4.2 Anti-myth: *DPR* design instead *DPR* models

Axiom 3 declares the alternative input of *mDPR* (*design process design*) and *mDPR* function (*DPR design implementation*). Let us construct the required *DPR* design, in which design formation concept simultaneously is the design progress concept.

As any other design, *DPR* design matures through evolving. Let the chosen design progress concept for *DPR* (DPC^{DPR}) is the same as for the product one – "evolution of individual", that is successive adaptation of the current design state to a new state of design environment. But design environment for *DPR* design differs from the environment for a product. For the former, we take the power of intended operational space of *DPR* design:

- 3Di (domain-, task- and processor-independence),
- 2Di (domain- and processor-independence),
- 1Di (*P*-independence).

Thus, *DPR* design should have three maturity levels (*ML*): ML_{3D}^{DPR}, ML_{2D}^{DPR} and ML_{1D}^{DPR} respectively.

To construct ML_{3D}^{DPR}, we use $qAM=(St, Sm^*)$ – the 3Di design platform. The refinement of $Sm^* \in qAM$, when moving to AM^{DPR}, results in iteration of the twain of processes (Figure 4): product design state synthesis (SPR^P) and *OE* design state synthesis (SPR^{OE}). This twain is assigned as *Sm* to each item (*&*-cube cell) of *DPR* design *dh*-structure. (Thus, the response to the product complexity growth will be the growth of number of iterations for the same pair of processes.)

DPR design with ML_{2D}^{DPR} is produced by imparting to ML_{3D}^{DPR} the ability to adjust its structure to a design task (a sought-for product and design goal). Lastly, *DPR* design with ML_{1D}^{DPR} is obtained by adding ML_{2D}^{DPR} with domain specific knowledge (applied software, theories, gained experience, design methods, etc.), structured in compliance with the structure of ML_{3D}^{DPR} and provided with a meta-*DKMS* (inter-domain *Design Knowledge Management System*).

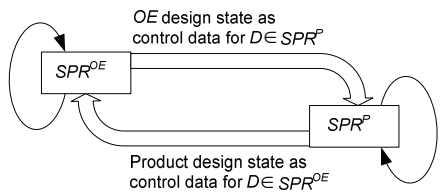

Figure 5. AM^{DPR} semantics

5 THE MYTH NO. 3: Design problem

5.1 Generic problem realization

When a conditional PR^C is declared, it needs to be determined upon D and P (Section 2). In that case, the processes SD (search for D) and SP (search for P) should be executed for PR^C:

$$SD \xrightarrow{d} PR^C \xleftarrow{d} SP \tag{5}$$

This triple of processes represents the elementary process structure based on **d**-relation.
On the other hand, any problem statement is "what is needed" (the required output of process PR^C) and "what is given" (I^P – the input of $P \in PR^C$). Thus, we identify the linear notation of structure (5), added with $P \in PR^C$ input, as a *problem scheme* (*PrB*):

$$PrB=<<SD,\ SP> < PR^C>> \tag{6}$$
$$I^P$$

PR^C determination transforms it into PR^L (*logical* process, i.e. ready for physical implementation). Following [14], we distinguish a "*problem solution*" (determined $D,P \in PR^L$ – the results of <*SD, SP*> performance) and an "*answer to the problem*" (the output of PR^L). We call *problem realization* the activity consisting in problem solution deriving and answer computing. The problems that have both solution and answer are *soluble*.
But there are many more problems (posed directly upon a required result), which have an answer but have no solution. Those are referred to as insoluble of the *second kind*. (The problems that have neither solution nor answer have insolubility of the *first kind*.) For all that, the answer availability to such a problem indicates that instead of original (insoluble) some another problem is actually realized, the answer to which coincides with the answer to the initial problem (or the answer to the latter is a part of an answer to another problem). Another problem is said to be *conjugate problem* (*CP*) with respect to original one. Knowledge engineering gives a classic example of such a pair of problems: insoluble original – diagnosis problem, and soluble conjugate – knowledge inference problem. Answer deriving to a problem through realization of its *CP* is called *quasi-* or *q-realization* of the problem.

5.2 Design problem quasi-realization

Design problem (*DP*) has the reputation of ill-defined [15] or even wicked [16]. Write down for *DP* its scheme:

$$DP=<<SD,\ SP>< DPR>> \tag{7}$$
$$I^P = \text{needs and requirements}$$

Since the need and requirements cannot be transformed into a goal design (the former and the latter are different conceptual worlds [17]), *DP* has no solution. At the same time, an answer to *DP* (a design) is possible. Hence, *DP* is a *phantom* or *q*-realizable *problem*.
The replacement of ambiguous characteristic of "ill-defined" for constructive "insoluble of the second kind" means searching for *DP* its CP^{DP}. It begins with searching for a process conjugate to $DPR \in DP$ from (7). The current design paradigm propose (on default) for this role (inherently, the role of *mDPR*) the process of a stepwise concretization of abstract description of a prototype relevant for a product under design. The problem scheme restored upon such *mDPR* looks as follows:

$$CP^{DP}=<<SD,\ SP>< mDPR>> \tag{8}$$
$$I^P = \text{system image of a prototype}$$

5.3 Anti-myth: adequate conjugate problem and its tackling

Axiom 3 (Section 4.1.2) defines both the process conjugate to *DPR* and this process input. Restore upon this process the scheme of the adequate conjugate problem for *DP* (CP_A^{DP}):

$$CP_A^{DP} =<<SD,\ SP>< mDPR>> \tag{9}$$
$$I^P = DPR \text{ design}$$

We have constructed *DPR* design in Section 4.2, so proceed now to CP_A^{DP} solution, that is $mDPR \in CP_A^{DP}$ determination.

Let us first obtain 2Di solution to CP_A^{DP}, i.e. when *DPR* design, entering the problem input, has ML_{2D}^{DPR}. In this case, we obtain not *mDPR*, but a so called quasi-*DPR* (*qDPR*). Though *qDPR* performance cannot produce domain-specific *DPR* (2Di *DPR* design implementation should result in 2Di *DPR*, which is inoperative), its destination is to serve for *mDPR* platform.

Indeed, $D \in qDPR$ is the procedure realizing *&*-cubes (Figure 4*a*) traverse with triggering in each cell the twain of processes (Figure 5). At the same time, the facilities of *qDPR* implementation (a special purpose OS) have gotten the name of domain-independent *Design Machine* [18], considered as the platform for domain-specific design system – the facilities that support *mDPR* implementation.

The next step produces 1Di solution to CP_A^{DP}: determination of *mDPR=(D, P)* upon object *D* results in $D \in qDPR$ added with abilities to interact with design knowledge management system from ML_{1D}^{DPR}.

6 DISCUSSION

On the one hand, the material of this paper does not hold a rank of design theory, so there is no point to compare it with the available design theories. Besides, we also deem that "the development of design theory is essentially still in a pre-theory stage" [19]. So the paper content should be perceived as the site stripping for initiation a proper design theory (that is its platform building). The further theory development is, in our view, a long run process assuming participation in it the entire design research community.

On the other hand, that what is usually declared by authors as design theory may be reduced per se to the common triple: a proposed conjugate problem, Design definition naturally stipulated by this problem, and more or less schematic description of the process for this definition implementation. In other words, the theories under offer actually may also be treated as the attempts to propose a design theory platform. At that rate, a number of items emerge for comparison, though we shall confine ourselves to a lightning version of such comparison whereas a detailed one would require a separate paper.

Let us initially make more specific the notion of a proper design theory. To this end, we formulate the main requirements the theory should meet.

1. *Self-sufficiency*. It is ensured by availability of sufficient scientific foundation specially built for this theory. In contrast, an external base of scientific principles, commonly engaged by design theories, concurrently involve concomitant problems specific for a native discipline but could hardly be effectively employed for design goals.
2. *Adequacy*. This property depends on choosing of a conjugate problem that replaces insoluble design one. CP^{DP} realization should entail at least the explicit realization of the structure synthesis problem, tackling the problem of design process complexity, and possibility to design a new product that has never been designed before.
3. *Consistency*. It is provided by the condition when the way of thinking and the way of doing, promoted by the theory, are sides of the same coin but not alternative coins.
4. *P-independency*. This precondition eliminates many barriers for computer to cooperate with a designer on the partnership basis.
5. *Purposiveness*. This means that from the very beginning all theory tools must be aimed at practice problem solving. The necessity of theory adaptation to practice needs emerged post factum inevitably dilutes the original theory but does not attain the adaptation goals.

In the light of these requirements, it may be said that on the way to a proper design theory we have removed the inveterate myths that encumbered the approaches to such theory initiation and proposed the embodiment of this way. The latter is characterized by the following.

- The inherent base of scientific principles of the theory – *CPT* – is intensively employed by all its divisions and parts. (The available design theories often appeal to external facilities – set theory, logic, etc. But strong tools commonly play poor parts. For instance, the use of logic in original C-K theory [3] is highly argued but low intensive.)
- The part of CP^{DP} is performed by the problem of design process design realization that has resulted in the choice of the most natural, in our view, design progress concept (evolution of

individual), development of a unified design representation (design platform), and splitting of design process into *DPR* proper and *meta-DPR*. Evolutionary synthesis or EVOS – exactly so was designated the proposed design of design theory platform. (For reference, CP^{DP} in the theory based on model [5] is formulated as "stepwise reducing the level of abstraction in a prototype description", while C-K theory [3] poses the problem of "recursive generation of propositions with the status of concepts till those can be transformed into propositions with the status of knowledge".)

- The selected design progress concept and continuous design representation enable to make design structure synthesis explicit and to develop new products. (Design representation and product structure synthesis in the available design theories have not been clearly articulated. Besides, in German systematic [5] and axiomatic design theory [11], the presence of a prototype for the product under design is essential.)
- Concern of practice (first of all, a process that yields a design and a system that supports this process performance) is the main focus of EVOS from the very beginning. (C-K theory needs for adaptation to demands of practice though unable to supply a want in full.)
- *DPR* splitting for *DPR* proper and *meta-DPR* shifts the focus from the former to the latter. This means that the complexity problem now associates with not *DPR* but *meta-DPR*. At the same time, *meta-DPR* complexity is not as pressing as previously for *DPR* – *meta-DPR* structure does not depend critically on the structure of a product under design. (The available design theories continue inefficient dealing with *DPR*, which is actually the embodiment of two different processes – the goal one and technological.)
- *P*-independence of *meta-DPR* enables to change the status of computer in designing from the instrument on demand to designer's partner. (All available design theories are *H*-centered.)
- The proposed way of thinking and way of doing consist in a problem-free manner. (Design theories at hand keep in focus either the first, like C-K theory [3], or the second, like VDI Guideline [20].)
- Others from among proposed platforms enable to discuss the new technology of design computerization (design system platform, Section 5.3) and clear up a matter with design automation. (The well-known design theories do not start these questions.)

7 CONCLUSION: THE PLATFORM APPROACH IN DESIGN 'INDUSTRY'

We have discussed the main, in our view, myths of engineering design – design problem, design process, and product representation. Renunciation of these myths shifts the focus from the design problem to its conjugate problem, from a design process to meta-design process, from representation of a product to representation of a design (product design, operation environment design or any process design) – particularly dynamic representation. The change of focus also opens up new prospects – the platform approach to design industry, for instance.

For instance, the *design platform* outlines yet another constructive definition of designing: transformation of *qAM* into *AM* through the refinement of abstract semantics $Sm^* \in qAM$. (We have used the unified design platform to produce designs of a product, its operation environment and design process.)

The major role of 3D independent *meta-DPR platform* (*qDPR*) consists in transfer the key platform's properties (complete, holistic and continuous structure) to domain-specific meta-*DPR* (and upward to design system).

The use of *design system platform* (design machine, *DM*) leads to the new efficient and effective technology of design computerization (automation): the compilation a wind range of domain-, product-, user-, and media-oriented design systems via replication of domain-independent *DM* and further extension of each specimen with domain-specific SW and design knowledge management system (design knowledgemation).

In turn, the bulk of facilities used for the platform technology description may shape, in our view, *design language platform*, which should allow to change from the current substantially intuitive notion base of designing to the more strict one and thereby to serve for design coordination over distance and across professions.

Thus, design industry has the opportunity to start to gather its regular platform toolkit, the item of which should be a Design Theory platform.

REFERENCES

[1] Weber, C. CPM/PDD – an extended theoretical approach to modeling products and product development processes. In *Proceedings of the 2nd German-Israeli Symposium on Advances in Methods and Systems for Development of Products and Processes*, Stuttgart, July 2005, pp.159-179.

[2] Andreasen, M.M. and McAloone, T.C. Applications of the theory of technical systems – experiences from the "Copenhagen school". In *Proceedings of the AEDS 2008 Workshop*, Pilsen, November 2008, pp.1-18.

[3] Hatchuel, A. and Weil, B. A new approach of innovative design: an introduction to C-K theory. In *International Conference on Engineering Design, ICED'03*, Stockholm, August 2003.

[4] Weber, C. Theory of technical systems (TTS) – its role for design theory and methodology and challenges in the future. In *Proceedings of the AEDS Workshop*, Pilsen, November 2008, pp.107-121.

[5] Pahl, G. and Beitz, W. *Engineering Design*, 1983 (Springer, Berlin-Heidelberg).

[6] Ariyo, O.O ., Eckert, C.M. and Clarkson, P.J. Hierarchical decomposition for complex product representation. In *International Design Conference, DESIGN'08*, Dubrovnik, May 2008, pp.737-744.

[7] Eder, W.E. and Hosnedl, S. *Design Engineering*, 2008 (CRC Press, Boca Raton).

[8] Ullman, D.G. *The Mechanical Design Process*, 1992 (McGraw-Hill, New York).

[9] Kusiak, A. (ed.), *Concurrent Engineering: Automation, Tools and Techniques*, 1993 (Wiley, New York).

[10] Pugh, S. *Total Design: Integrated Methods for Successful Product Engineering*, 1991 (Addison-Wesley).

[11] Suh, N.P. *The Principles of Design*, 1990 (Oxford University Press).

[12] Sedenkov, V. An attempt to answer perennial design questions. In *Proceedings of the 6th Seminar EDIProD 2008,* Gdynia-Zielona Gora, September 2008, pp. 21-30.

[13] Vajna, S., Clement, S., Jordan, A. and Bercsey, T. The Autogenetic Design Theory: An Evolutionary View of the Design Process. *J. Eng. Design*, 2005, 16(4), 423–440.

[14] Polya, G. *Mathematical Discovery*, *Volume* 2, 1965 (John Wiley&Sons, New York – London).

[15] Simon, H.A. *The Science of the Artificial*, 1970 (MIT Press, Cambridge).

[16] Rittel, H. and Webber, M. Dilemmas in a general theory of planning. *Policy Science*, 1979, 4, pp.155-169.

[17] Meijers, A.W.M. The relational ontology of technical artifacts. In *The Empirical Turn in the Philosophy of Technology*, Oxford, 2000 (Elsevier Science).

[18] Sedenkov, V. and Guziuk, Z. Design machine: theory and implementation. In *Proceedings of the International Symposium TMCE*, Lausanne, April 2004, (pp.1119-1120, executive summary) CD.

[19] Dixon, J. R. On research methodology towards a scientific theory of engineering design. *Artificial Intelligence for Eng. Design, Analysis and Mafnufacturing*, 1988, 1(3), 145-157.

[20] VDI Guideline 2221: *Systematic Approach to the Design of Technical Systems and Products*, 1987 (VDI, Düsseldorf).

Contact: Vladimir Sedenkov
Belarusian State University
SW Engineering Department
4, Nezavisimosty Ave.
220030, Minsk
Belarus
Tel: Int. +375 17 2723344
Fax: Int. +375 17 2265548
E-mail: sedenkov@hotmail.com

Vladimir is Senior Lecturer of Computer Science in the Software Engineering Department at Belarusian State University. He teaches and researches in engineering design and computer-aided design. He is interested in many aspects of design and computing, in particular how computer aids can assist design in the organization and management of the information used in design.