AEDS 2008 WORKSHOP
31 October – 1 November 2008, Pilsen - Czech Republic

DESIGN PROBLEM *q*-REALIZATION AND ITS OUTCOMES

Vladimir Sedenkov

Keywords: meta-design process, design industry platforms, design science structure

1. Introduction

Design problem (*DP*) has a stable reputation of ill-structured [Simon 1970] or even wicked [Rittel 1973]. At the same time, absent adequate realization of *DP* retains unclear the design process (*DPR*) itself. The multiple conjectural models of *DPR* merely promote its disintegration. In turn, the lack of *DPR*-oriented knowledge entails inarticulateness of design automation, inefficient and non-technological utilization of design methods and knowledge, vague ways to counteract *DPR* complexity growth, chaotic picture of design related research. All of this causes the feeling of "the end of design methodologies" [Nauman 2004].

On the other hand, absent *DP* realization leaves doubtful the outlines and power of Design Science, disabling it to field many questions induced by design theory and practice. Thus, the appeal to engineering Design Science consolidation [Birkhofer 2006], as well as the claim "designing is a discipline in its own right" [Andreasen 2001], have an ambiguous effect: the both of items cause no doubts to be raised but evidently swap the events, leaving the cart before the horse (there is neither the mater to consolidate nor separate discipline yet).

More concisely, the lack of adequate *DP* realization has turned into dam that, in our view, hinders both Design Science progress and design practice enhancement. In this paper, we propose a way of *DP* quasi-realization and consider its outcomes.

2. The glimpse into continuous process theory – a formal tool at hand

Process scheme: a process (*PR*) representation by the pair (*D*, *P*), where *D* is a procedure to produce the process result, and *P* is a processor intended to perform *D*: *PR=(D, P)*. 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*):

$$SD \xrightarrow{d} PR_1 \xleftarrow{d} SP \qquad (1)$$

PR_1 is the *core* of the structure while *SD* and *SP* form the structure's *tail*. Notation (1) is the example of the *first order structure* ($n=1$). The structures with $n=1$ serves for the elements of the *second order structures*. Let us construct one of those.

In short, any problem statement is "what is needed" (delivered by a virtual PR_1) and "what is given" (provided by PR_2). Following [Polia 1965], we distinguish a "*problem solution*" (determined $D\in PR_1$ and $P\in PR_1$) and an "*answer to the problem*" (an output of PR_1 performance). Then any problem (*PrB*) can be represented by its scheme when "what is needed" has indirect presentation – by a providing process:

$$\begin{tikzcd} PrB_1=\langle\langle SD, SP\rangle\langle PR_1\rangle\rangle \\ PR_2 \arrow[u, "p"] \end{tikzcd} \qquad (2)$$

The output of PR_2 enters the input of PR_1. Having restored upon PR_2 its problem scheme, we obtain the structure out of problem schemes ($n=2$; PR_2 is supposed to be provided; relation between structures is the relation between their cores):

$$\begin{tikzcd} PrB_1=\langle\langle SD, SP\rangle\langle PR_1\rangle\rangle \\ PrB_2=\langle\langle SD, SP\rangle\langle PR_2\rangle\rangle \arrow[u, "p"] \end{tikzcd}$$

Not going into elucidation, make only an example of the structure with $n=3$ – the so-called *super*- or *S*-tree (Figure 1). *S*-tree is an arc-bichromatic tree where each *S*-node is an ordinary tree – the third order structure.

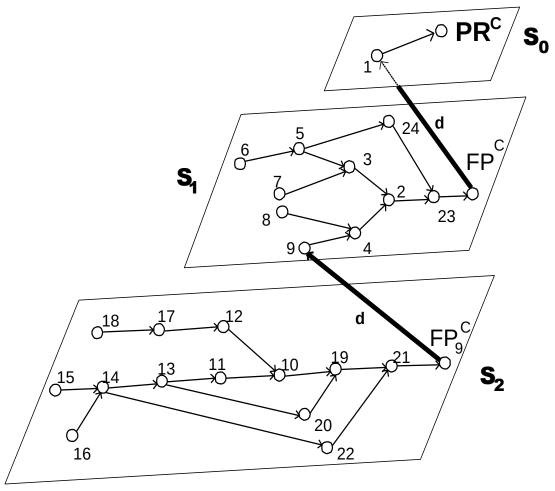

Figure 1. The fragment of the third order structure out of processes

3. Design problem quasi-realization

Let $DP=\langle\langle SD, SP\rangle\langle DPR\rangle\rangle$ be design problem scheme, where *DPR* is a design process. In the general case, *DPR* input is shaped by needs and requirements. On the one hand, *DP* is *insoluble* ($D\in DPR$ does not exist): the needs and requirements cannot be transformed into a goal design – the former and the latter are different conceptual worlds [Meijers 2000]. On the other hand, *DP* has the answer (a design). As *DP* has no solution, we call it *quasi*- or *q*-realizable.

3.1 What is a conjugate problem for the design problem?

Thus, there should be another problem actually realized instead *DP* – call it *conjugate* with respect to *DP*: CP^{DP}. The latter should be soluble and the answer to it coincides with the sought-for answer to *DP*. There is nothing for it but to find and realize CP^{DP}. (Note that problem realization consists of two steps: problem solving – the performance of processes <*SD, SP*>, and the answer inference – the performance of <*PR*> or, in case of *DP*, <*DPR*>*).*

CP^{DP} is obvious from the Axiom 1 [Sedenkov 2008]:

> "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)".

This axiom defines both the process conjugate to *DPR* and this process input. We call the former *meta-* or *mDPR* (because it has concurrently to weaver a priori non-existent *DPR* and to perform it) and the latter – *DPR design* (3).

$$CP^{DP} = << SD, SP >< \underset{\underset{DPR\ \text{design}}{\uparrow}}{mDPR} >> \qquad (3)$$

Due to (2), to determine *mDPR*, we have first to construct its input – *DPR* design (Section 3.3).

3.2 Preliminary discussion on design representation in general

The objective of design process is to produce a product design via implementation some product design progress concept (DPC^P). The declared DPC^P is "evolution of individual" (in contrast to "evolution of population" [Gero 1996]): adaptation of the current design state to a new state of a product life cycle environment (*LCE*). This presupposes evolving (designing) of both a product and its *LCE* ($DPC^{LCE} = DPC^P$). In other words, we have to deal in the course of *DP q*-realization with the three kinds of designs – for a product, *LCE* and *DPR*.

In the case of that, why do not we construct some generic *pre-design* – a unified base for shaping any design from the triplet? To be feasible, such a base has to meet the precondition – to be domain-, task- and processor-independent (call it 3D independence or, for short, 3Di). Besides, our evolutionary *DPC* implies representation of a pre-design dynamics. What should it be (see Section 3.2.2)? Static representation of a design (drawings, for instance) is necessary for analysis, modeling, optimization and physical embodiment of a design. But those do not fit to design synthesis.

The starting point for the generic base search gives Axiom 2:

> "Just as an object (product, process, environment) is obtained through implementation of an object design, so an object design should be obtained through the implementation of an object's design of design:
> *object design of design* $\rightarrow$ *object design* $\rightarrow$ *artifact (physical object)*" (4)

An extension of string (4) 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 sequence is called *diachronic* (*dh*) or "historical" structure (*St*) of design-of-design, while 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 the *approximation model* of the design-of-design (and a design as well): *AM=(St, Sm)*. The opposite case, when semantics is abstract (*Sm**), is referred to as *quasi-approximation model*: *qAM*=(*St*, *Sm**) or *design platform*.

3.2.1 The structure of design platform

It is just the time to consider the development of $St \in qAM$, which consists of three steps: revealing design *dh*-structures for *LCE*, product and *DPR*. First, try to identify *LCE* and construct a primary *dh*-structure for its design. *LCE* 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 2*a*). Hence, we can break $\{PR_q\}$ up to three constituent sets: $\{PR_{q1}\}$, $\{PR_{q2}\}$ and $\{PR_{q3}\}$ (Figure 2*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 *restrictions* (*Rs*) 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. Having taken the members of each set for the vector of 3D space, we shaped *dh*-structure of *LCE* (Figure 2*c*). The latter is referred to as *&-cube*.

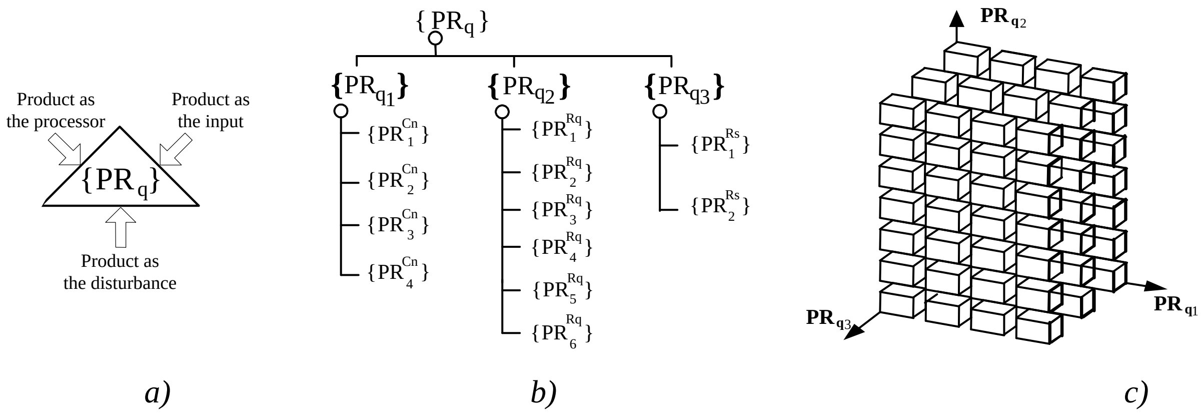

Figure 2. *LCE dh*-structure construction

Next, following evolutionary *DPC*, construct primary *dh*-structure for a product design. We distinguish for the latter four sequentially attainable design states and call them *design goals*: *Prototype* (PRT), *Market version* (ITM), *Manufacturing version* (COM) and *Artifact* (ART).

Each design goal involves four *subgoals*: *quasi-system* or *q*SYS (a minimal set of product units), *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). Transform the obtained hierarchy into *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 3.

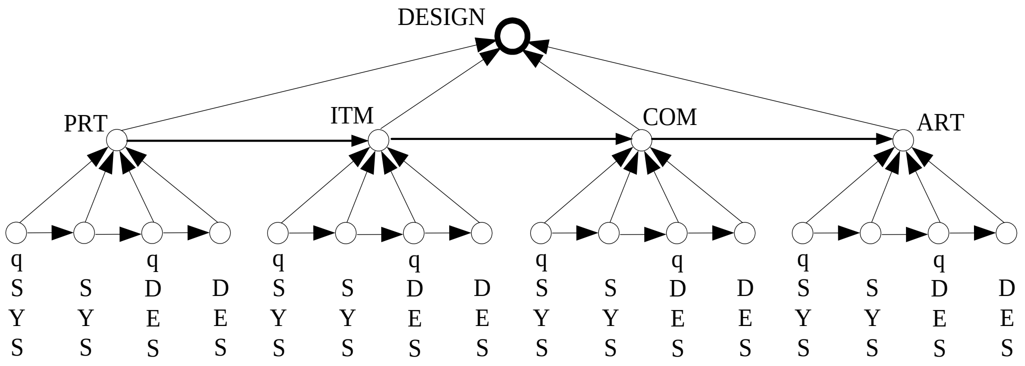

Figure 3. Primary *dh*-structure for a product design

Taking into account that the *DPR* under design would deal concurrently with both *dh*-structure of a product and *LCE*, *dh*-structure of the *DPR* design is obtained by substitution of *&*-cube for terminals in the above *q*-hierarchy. Then *DPR* design *dh*-structure is borrowed for both other designs – product and *LCE*, taking the status of $St \in qAM$. Whereupon, it takes only to refine $Sm^* \in qAM$ to come from *qAM* to AM^P, AM^{LCE} or AM^{DPR}.

3.2.2 Design platform semantics

Diachronic structure of the design platform imparts the property of quasi-dynamics to design representation if synchronous view of the structure items remains static. We may try to make representation completely dynamic via dynamic presentation of *qAM* semantics. Dynamics implies a process, that is *sh*-content of each *dh*-structure item should be presented by a process or structure out of processes. Hence, $Sm^* \in qAM$ is generally a conditional (undetermined) process, determination of which means the *Sm** refinement.

3.3 *DPR* design construction

As any other design, *DPR* design matures through evolving. 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 the new state of a 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 for *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* – 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 *LCE* design state synthesis (SPR^{LCE}). This twain is assigned to each item (*&*-cube cell) of *DPR* design *dh*-structure.

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 knowledge management system.

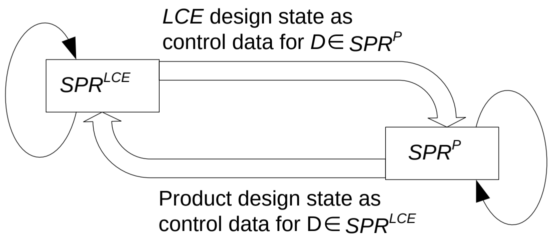
LCE design state as control data for $D \in SPR^P$
SPR^{LCE}
SPR^P
Product design state as control data for $D \in SPR^{LCE}$

Figure 4. AM^{DPR} semantics

3.4 Conjugate problem solving

3.4.1 2Di solution of CP^{DP}

Now when the CP^{DP} input (*DPR* design) has been identified, we can proceed to problem solving, that is $mDPR \in CP^{DP}$ determination. But let us first obtain the solution when *DPR* design *ML* entering the problem input is ML_{2D}^{DPR}. In this case, we obtain not *mDPR* but 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 traverse with triggering in each cell the twain of processes (Figure 4), while $P \in qDPR$ is determined as diprocessor [Sedenkov 2006] (H^H, H^C, C^H or C^C, where *H* is a human being and *C* – a computer, for instance) or action system platform for *mDPR*. At the same time, the facilities of *qDPR* implementation (a special purpose OS) have gotten the name of domain-independent *Design Machine*

[Sedenkov 2004] considered as the platform for domain-specific design system – the facilities that support *mDPR* implementation.

3.4.2 1Di solution

Determination of *mDPR=(D, AS)* upon subject gives the *AS* configuration. Determination upon object *D* results in $D \in qDPR$ added with abilities to interact with the knowledge management system from ML_{3D}^{DPR}. As *mDPR* implementation (Axiom 1) entails concurrent inference and performance of domain-specific *DPR*, the former is called *design engine*.

4. The outcomes of design problem *q*-realization

4.1 Platform approach in "design industry"

Overall, the technology of design problem *q*-realization shifts the focus from design process to meta-design process and from representation of a product to representation of a design (product design, life cycle environment design or any process design) – particularly dynamic representation. This can be easily observed while analyzing the four platforms distinguished in the course of design problem *q*-realization: for a design (*qAM*), for meta-*DPR* (*qDPR*), for *mDPR* action system (*d*P), and for design system (*DM*). Each of those brings its own contribution to design theory and practice.

Thus *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 life cycle environment and design process. In the case of product, *Sm** specification means *q*-realization of the *structure synthesis problem* (via realization of its conjugate problem – product operation process determination [Sedenkov 2000]).

The major role of 3Dindependent 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). Thus, meta-*DPR* plays the part of design engine – it weavers the goal *DPR*, implements it and delivers the sought-for product design. In the end, meta-*DPR* platform serves the purpose of *DPR* designing. Shifting the focus from design process (a priori nonexistent by nature) to the constantly available meta-design process allows also to eliminate the problem of design process complexity: the complexity problem loses its sharpness in point of regular meta-*DPR* since the structure of the latter does not critically depends on product structure.

While addressing to *mDPR* action system platform ($d\text{P}=C^H$, where *C* is a master processor – a computer, and *H* – a human processor providing control data for *C*), we can note that it enables to refine both the matter of design automation and its subject (meta-*DPR*). Thereby the present intuitive notion of design automation gives place to the more accurate one.

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 it with domain-specific SW and knowledge management system.

In turn, the bulk of facilities used for platform technology description may shape, in our view, design language (*DL*) platform, which should allow to change from the current substantially intuitive notion base of designing to more strict one and thereby to serve for design coordination over distance and across professions. Concurrently, this will not prevent from making platform-based domain-specific design languages as a universal *DL* is hardly possible.

4.2 Platform approach to Design science

The above stated entails the issue of Design Science platform as well. What is it all about? To date we, evidently, have no Design Science – there are only numerous fragments of the amorphous building remaining up in the air as it has no foundation bed; call it "virtual Design Science". It was observed [Sedenkov 2008] that real Design Science does not need to be created but needs activation via underpinning the hanging construction with the third constituent of Design Science – Science *for* design[1] (Science *in* design, which goes back to [Simon 1970], and Science *of* design [Gasparsky 1990] are the rest and available constituents of Design Science). Science for design identification consists in attribution of the triple of its paradigmants –"design representation", "*DPC*" and "*DP* problem realization".

But this way of Design Science structuring is fraught with a lot of interpretations and arguing. To reduce those, we propose instead of three "sub-sciences" the three layers of a single real Design Science (Figure 5):

- *system* or generative layer (the former Science *for* design),
- *application* or technological layer (the former Science *in* and *of* design or, otherwise, "set of the knowledge needed for designing" [Hubka 1996]),
- *integrative* layer (links the two first layers in a manner of "knowledgemation of design problem *q*-realization").

Each layer has its own paradigm. The first or generative layer, responsible for design problem *q*-realization, plays the part of Design Science platform (fundamentals). The twain of two first layers presents designing as a "discipline in its own right", while another twain – the second and third layers – provides it with interdisciplinary perception.

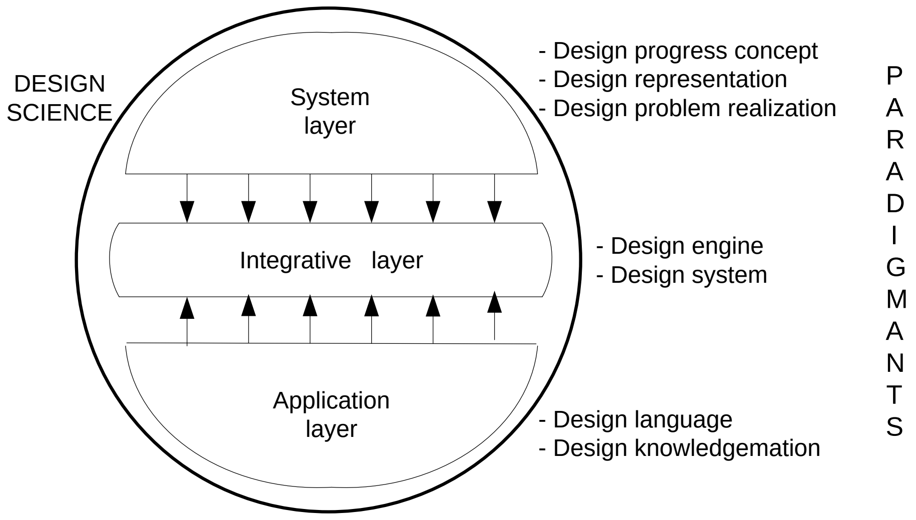

Figure 5. The proposed three-layered structure of Design Science

5. Conclusion

Design problem *q*-realization crosses the positivism of H. Simon (designing as problem solving, [Simon 1970]) and constructivism of D. Schön (the study of designer's activity) [Schön 1983, 1992]. In addition to the Simon's concept, we have changed the coined characteristic of design problem ("ill-defined") for the more promising one – "unsolvable of the second kind". This resulted in realization of the conjugate problem relative to the phantom *DP* – the problem of design process design implementation. In addition to the Schön's concept, there has been proposed the proactive model of interaction between a product design and product life-cycle environment ("evolution of individual") embodied in

[1] Our treatment of Science for design quite differs from [Krippendorff 2006].

DPR design. Thus, the things have gone to platforms. On the whole, the platform-based reasoning paves the way for domain-independent approach to theory and research in design.

Individual remark should be made on Design Science platform. There is an impressive number of questions induced by design theory and practice that remain without answers. For instance, "How it is that designers can commence designing with incomplete information and before all the relevant information is available?" [Gero 2006] or "How do we register and transform needs into a product goal specification?" [Andreasen 2001]. Layerwise structure of Design Science and the substance of its generative layer enable to field these questions and eventually to clear up the situation with calls for "engineering design science consolidation". On the evidence of the above stated we prefer the appeal to (consolidated) Design Science activation and its further development.

References

[Andreasen 1998] Andreasen, M., "The role of artefact theories in design", *Proceedings of the Workshop Universal Design Theory*, Karlsruhe, Germany, 1998, pp 57-71.
[Andreasen 2001] Andreasen, M., "The contribution of design research in industry – reflections on 20 years of ICED conferences", *Proceedings of the ICED'01*, Glasgow, Scotland, UK, 2001, pp 3–10.
[Birkhofer 2006] Birkhofer, H. and Weber, B., "The consolidation of engineering design science – an utopian dream?", *Proceedings EDIProD 2006*, Rohatynski, R. and Poslednik, P. (eds), Zielona Gora, Poland, 2006, CD.
[Gasparsky 1990] Gasparsky, W. and Strzalecki, A., "Contributions to Design Science: Paradoxical Perspective, Design Methods and Theories", *Journal of DMG*, Vol.24, No.2, 1990, pp 1186–1194.
[Gero 1996] Gero, J., "Creativity, Emergence and Evolution in Design", *Knowledge-based Systems*, Vol.9, No.7, 1996, pp 435–448.
[Gero 2006] Gero, J., "Design prototypes: a knowledge representation schema for design", *AI Magazine*, No.11, 2006, pp 26-36.
[Hubka 1996] Hubka, V. and Eder, W., "*Design Science*", Springer-Verlag, London, 1996.
[Krippendorff 2006] Krippendorff, K., "*The Semantic turn: a new foundation for design*", Taylor and Francis CRC Press, Boca Raton, Florida, USA, 2006.
[Meijers 2000] Meijers, A., "*The relational ontology of technical artifacts*", In: "The empirical turn in the philosophy of technology", Kroes, P. and Meijers, A. (eds.), Elsevier Science, Oxford, 2000.
[Nauman 2004] Nauman, T. and Vajna, S., "Adaptive system management", *Proceedings of the TMCE 2004*, Horvath, I. and Xirouchakis, P. (eds), Millpress, Rotterdam, 2004, pp 1183-1184.
[Polya 1965] Polya, G., "*Mathematical discovery*", V.II, John Wiley&Sons, New York–London, 1965.
[Rittel 1973] Rittel, H.and Webber, M., "Dilemmas in a General Theory of Planning", *Policy Sciences*, Vol.4, 1973, pp 155-169.
[Schön 1983] Schön, D., "*The reflective practitioner: How professionals think in action*", Basic Books, NY, 1983.
[Schön 1992] Schön, D., "Designing as reflective conversation with the materials of a design situation", *Knowledge-based systems*, Vol.5, No.1, pp 3-14.
[Sedenkov 2000] Sedenkov, V., "Product structuring and synthesis in evolutionary design", *Proceedings of the TMCE 2000*, Horvath, I., Medland, A. and Vergeest, J., (eds), Delft, The Netherlands, 2000, pp 183-196.
[Sedenkov 2006] Sedenkov, V., "Design process: holistic view", *Proceedings EDIProD 2006*, Rohatynski, R. and Poslednik, P. (eds), Zielona Gora, Poland, 2006, pp 227-237.
[Sedenkov 2004] Sedenkov, V., and Guziuk, Z., "Design machine: theory and implementation", *Proceedings of the T MCE 2004*, Horvath, I. and Xirouchakis, P. (eds), Millpress, Rotterdam, 2004, pp 1119-1120.
[Sedenkov 2008] Sedenkov, V., "An attempt to answer perennial design questions", *Proceedings EDIProD 2008*, Rohatynski, R. and Poslednik, P. (eds), Gdynia-Zielona Gora, Poland, 2008, pp 21-30.
[Simon 1970] Simon, H., *"The Science of the artificial"*, MIT Press, Cambridge, UK, 1970.

Dr. Vladimir M. Sedenkov
Belarus State University, SW Engineering Department
4, Nezavisimosty Ave., Minsk, 220030, Belarus
+375 17 2723344 (voice)
+375 17 2265548 (fax)
sedenkov@bsu.by