

DEVELOPMENT AND APPLICATION OF DESIGN CONCEPT ONTOLOGIES FOR CONTEXTUAL CONCEPTUALIZATION

Imre Horváth
Joris S. M. Vergeest
György Kuczogi

Sub-Faculty of Industrial Design Engineering
Delft University of Technology

Abstract

The paper focuses on two specific problems of knowledge intensive computer aided design: (a) how can design concepts be modeled and represented in a form that is understandable for human beings and can be processed by computers, (b) how can they be arranged in structures that enables a computer-based functional design of products. First, a methodology for definition of very high level modeling entities based on the ontology theory is presented. It formalizes design concepts in terms of all concerned entities, phenomena and situations and describes them by attributes, parameters and descriptors, respectively. Validity and interactions of design concepts are governed by constraints. The very high level modeling entities are arranged into domain oriented design ontologies based on their contents and semantic relationships. The formalism used for logical specification of design ontologies is based on a library of declarative expressions called ACN-Code. A design ontology lends itself to a specific knowledge base called associative concept network (ACN). The inference engine that works on an application oriented ACN selects the appropriate design concepts against a set of user specified functional requirements. Due to the pre-defined associations incomplete functional specifications can be completed to result in a fully functional design solution. The paper also presents an application example.

1 Introduction

1.1. Issues of formalization of design concepts

In the case of previously not existing products, or design solutions, conceptual design should come up with novel contextual descriptions and configurations. Contextual description involves finding the appropriate *design concepts* for some functional specification and arranging them in a way that facilitates realization of a shape concept for the product. This intuitive process falls almost completely out of the scope of the existing CAD technology (Horváth, I., 1997). To provide effective computer support to this part of conceptual design, capturing and formalizing of design concepts are required. Although its importance is well known, formal specification of design concepts is almost in its infancy. In commercialized CAD/CAE systems, design concept formalization got stuck at the level of *application features* that have been introduced to allow capturing design intent (Shah, J, and Mantyla, M, 1995). Definition of features is

very often restricted to shape or form aspects, and handling of feature semantics is confined to validity constraint management (Dohmen, M., 1997). Since feature entities are associated with the final appearance of the design, rather than its evolving forms, feature-based computer-aided design is of a phenomenological nature (Soenen R. and Olling G. J., 1994).

Present feature technology is based mostly on conventional geometric modeling engines. Therefore, semantics of application features is typically described by some sort of additional symbolic representation (Horváth, I., 1996). The principal problem is that primary geometry definitions and the restricted functionality descriptions do not form a synergetic unity (Horváth, I., Kulcsár, P. and Thernesz, V., 1994). Because feature definitions are static there is a need for feature conversions. The lack of comprehensiveness results in functional limitations. The feature entities that are generally coded to be self-contained do not convey information about functional relations to other features.

Attempts have been made to extend the *feature paradigm* to a more intensive management of design concepts as well as to utilize it in product conceptualization. Umeda, Y. and Tomiyama, T., (1997) explain how functional reasoning adds functional concepts to the model of an artifact and specifies the functionality of the artifact. An explicit functional model can be used to reason out the behavior. Sasajima, M. et al., (1995) developed a function and behavior representation language that represents concepts at various levels of abstraction and maps behavior of a component to a term which represents its function. Some research papers reported about the use of *physical features*. Physical features have been introduced for qualitative and/or quantitative modeling and simulation of dynamic behavior of objects being conceptualized (Kiriya, T., Tomiyama, T. and Yoshikawa, H., 1990). The basis of this approach is qualitative physics that allows us to reason about the observable behavior of entities in themselves and in interaction with other components of a system. Physical features however were not intended to cover shape related aspects of design. To resolve some of the issues of representation and handling of design concepts the research reported in this paper started out of the *ontology theory* (Guarino, N. and Giaretta, P., 1995). The main advantage of this approach is that it allows us to formalize the knowledge both on domain-level and on object-level.

The authors are very well aware that the ontological approach is not the only possible way of formalizing design knowledge. Nevertheless, the principles and means offered by the ontology technology can be applied in the practice directly. This approach seems to have potential to forward present situation towards a more knowledge-intensive support of (conceptual) design. For instance, the implementation- and application-independent knowledge interchange format - KIF - provides a representation that can be understood by human beings as well as interpreted and processed by computers (Genesereth, M. and Fikes, R., 1992). Obviously, a lot of further work is needed to explore and utilize all possibilities offered by *design ontologies* (Eekels, J., 1997). An obvious advantage of defining design ontologies is that we can share knowledge with, and among functional agents - either human, or software (Grabowski, H., Rude, S. and

Pocsai, Zs., 1997). The next sub-sections give a brief overview on the notion of design concept and the fundamentals of ontology theory.

1.2. About design concepts in general

Although the term ‘design concept’ is commonly used in practice, its interpretation is somewhat vague and ambiguous. The proper understanding can be facilitated by considering a more general thing, namely human problem solving. Psychological and AI research suggest that heuristic problem solving utilizes intuitive or learnt *concepts*. A concept circumscribes a specific part of an existing or imagined reality. From the aspect of knowledge processing, concepts are meaningful individual logical units of reasoning. In communication about problem solving, a concept always implies some sort of modeling and representation. Thus, a formalized and documented concept is a representation of a communication unit of human cognizant knowledge.

Design concepts represent a part of the totality of general concepts and have relevance to design problems. Thus, they form a set on the universe of discourse, *UoD*, related to a design process. Due to the *associative nature* of human brain, design concepts typically do not appear as individual entities, but in association with each other. Association supports inclusion of novel concepts into an already configured set of concepts. Former research work has revealed that conceptual design needs considering concept topologies rather than single concepts (Tomiyama, T. and Yoshikawa, H., 1987), (Horváth, I. et al., 1995).

A specific characteristic of design concepts is how much they are concrete or abstract. An *abstract design concept* can be a generalization of one or more concrete concept. Design concepts may also vary in their coverage. In this respect we can distinguish *primitive* and *compound design concepts*. The engineering content of a design concept implies a particular materialization which corresponds to the expected functionality. The shape of a design concept however is not in all cases defined by its functions. Especially for consumer products (durables) the shape is also strongly influenced by aesthetics. This suggest to distinguish functioning oriented design concepts and shape oriented concepts. This paper concentrates mainly on the issue of computer-based representation and manipulation of functioning design oriented concepts.

1.3. Fundamentals of ontology theory

The notion of ontology tends to be too abstract for many people at the first time. To make the idea of ontologies clear and precise, first we define what is meant by an ontology. In philosophy ontology is the understanding of the existence. Principally, the notion of ontology denotes a system of fundamental concepts that allows us to reason about the world around us, or a particular domain of it, based on the acquired knowledge (Guarino, N. and Poli, R., 1995). These concepts can be about either natural things (e.g., objects, processes, phenomena, and situations) or artificial things (e.g., artifacts, technical systems and controlled processes).

The source of knowledge for an ontology, at the same time the field of application, is called *domain of discourse, DoD*. A *DoD* is a sub-set of an *UoD*, that is $DoD \hat{=} UoD$. Conceptualization gives birth to a simplified abstract view of a *DoD*. It results in an integral system of *knowledge chunks* that is a *semantic coverage* of the field of

application. The applied formalized representation is an *explicit specification* of the conceptualization. An ontology as such represents common understanding of the knowledge related to a conceptualization (Uschold, M. and Gruninger, M., 1996).

A *design ontology* has an intentional semantic structure that defines and arranges all related notions which can be either *generic notions* or *specific notions*. A specification is said to be *complete* when a *DoD* is exhaustively covered by design concepts. The specification is *conforming* when no counterexample occurs for a given definition of a design concept. Instance notions that are derived from generic notions are supposed not to violate generic definitions. Design ontologies may also play important role in formalizing the art of creating models of artifacts. On the one hand, design ontologies can be used by the developers of knowledge-intensive CACD systems as vocabularies for representation and processing of domain knowledge. On the other hand, design ontologies guide the system users in understanding and applying knowledge carried by the system. The need to integrate models of different design sub-processes into a unified and coherent framework also motivated the development of design ontologies (Olsen, G. R. et al, 1994). A well-elaborated ontology serves as a theory of content which enables us to discuss about a given design domain as formally as possible, but without losing the meaning of the related concepts.

1.4. The objectives of research and the paper

According to our understanding, the most fundamental issue related to computer aided conceptual design (CACD) systems is how to make them capable for providing contextual descriptions of designs. Principally what we need is:

- (a) a comprehensive definition of concepts related to a particular design application as very high level modeling entities,
- (b) a specific mechanism that selects and arranges these formalized design concepts based on semantical relationships (in short, associations) among them,
- (c) a transformation process that converts the alternative concept associations into working geometries with relevant design variables.

It is realistic to expect that these capabilities can be provided based on an ontology oriented specification of the requested engineering knowledge. Our research in the field of design ontologies decomposes into the following objectives:

- (a) development of a methodology for specification of design ontologies,
- (b) clarification of the content and structure of specific design ontologies,
- (c) working out a method of contextual conceptualization of designs,
- (d) programming a proof-of-ideas software tool, and
- (e) application of the tool/method for contextual conceptualization of a particular design.

A specific objective of the study was to find the most appropriate way of defining very high level modeling entities by ontological formalizing. An ontology centered definition augments the well known *object oriented approach* by the concept of *association* in

programming. The remained part of the paper summarizes the results of the research completed until now, discusses some of the main issues of design ontology specification and those that are related to its application for contextual conceptualization of designs.

1.5. About the methodology of design ontology specification

Conceptualization of a domain of discourse and specification of an ontology needs a specific methodology. According to the best knowledge of the authors, no dedicated methodology has been developed for specification of design ontologies. There are proposals however for specification of general ontologies (e.g., KIF, Ontolingua) and for development of ontologies for applications other than design (e.g., enterprise modelling - Fox, M. S. et al., 1996). The facts mentioned in Sub-section 1.2. imply that a design ontology has to merge two conceptualization levels into one. Namely conceptualization has to cover the level of design concepts and the level of system of design concepts simultaneously. This requirement is unique for specification of design ontologies. That is the reason why applying application-independent specification methodology at the development of a design ontology is not the best policy. The methodology that has been developed and applied by the authors integrates object level and object-structure level specification. The core of the methodology is presented in Figure 1. Further details about the practical application of the methodology will be given in Section 2.

Although this methodology has been delicately developed to meet the demands of specification of design ontologies, there are some issues related to its application that need proper consideration. Almost all phases of the ontology specification process are characterized by parallelism. It means that certain activities (e.g., contextual, geometric and functional specification of design concepts) have to be executed simultaneously. Obviously, it makes the processing more complicated. When a comprehensive coverage of a DoD is sought after, managing with the complexity also an issue. For practical reasons, a design ontology has to be simultaneously and exclusive

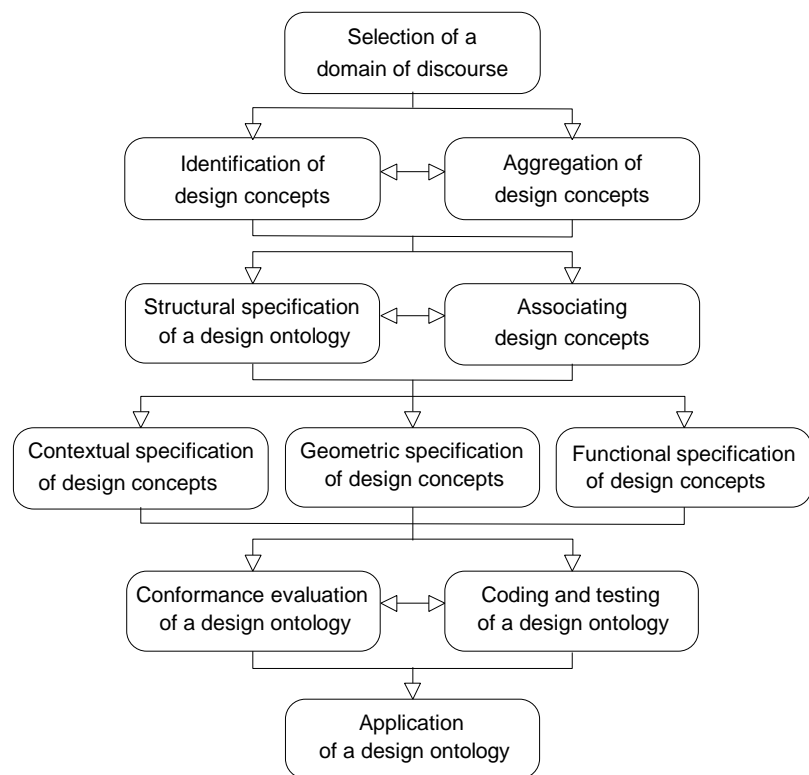


Figure 1 Methodology for specification of a design ontology

exhaustive. That is, it is not supposed to contain any concept that belongs to a different domain of discourse, but it is expected to cover all of the concepts that are relevant for the concerned domain of application. It is also anticipated that a design ontology (more precisely, its specification) should be capable of ensuring functionality and maintaining validity of the ontology as a whole and of all included design concepts in each relevant design situation. The content of the design ontology has to facilitate that the meaning of the specified design concepts be processed rather than the syntax of specification. Furthermore, the specification must not be coercive or compelling for a design ontology systematizes the process of conceptualizing/designing beyond formalizing the means of it.

1.6. Understanding of the contents of design concepts

In order to apprehend the contents of design concepts let's take the idea of *functionally coupled pairs* (FCP). The theory of FCPs originates in physics, but it has been also widely accepted in mechanical engineering. Functionally coupled pairs represent the most elementary building blocks in the mechanical engineering practice. From the family of mechanical coupled pairs a typical one is a *friction pair* that assumes

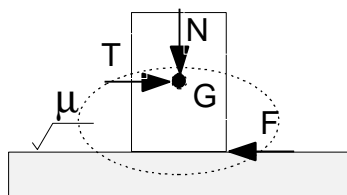


Figure 2 A friction pair

minimum two objects of some solid materials that are physically touching each other, and there is friction between them (Figure 2). If the objects are pressed together by a given normal force N , a friction force F is produced due to the friction μ between the objects. In this simple example three notions have been mentioned: (a) a set of (solid) objects, (b) a specific arrangement, and (c) a set of physical effects. Typically, these three types of constituents can be identified in any design concepts. Any

possible materialization can be reasoned out of the interactions of the constituents mentioned. Consequently, an abstraction of this example enables us to determine the necessary and sufficient constituents of a design concept also in other cases. The necessity of the constituents mentioned above can be proved in a pragmatic way. Evidently the basic nature of a design concept will change if just one of the constituents is altered. A sufficiently dominant change very often creates a different design concept. It is illustrated in Figure 3. In sub-figures a., b. And c. the objects are the same. The only variant is the position of the center of gravity of the white block relative to the gray block which results in a completely different behavior. The observable behavior is the outcome of the different functionality - as an interaction between the constituents of the design concepts.

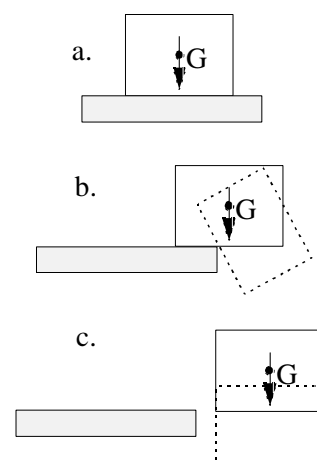


Figure 3 Interdependence between entities, situations and phenomena

The three fundamental constituents introduced above can be further generalized as (a) *entities, e*, (b) *situations, s*, and (c) *phenomena, p*. These will

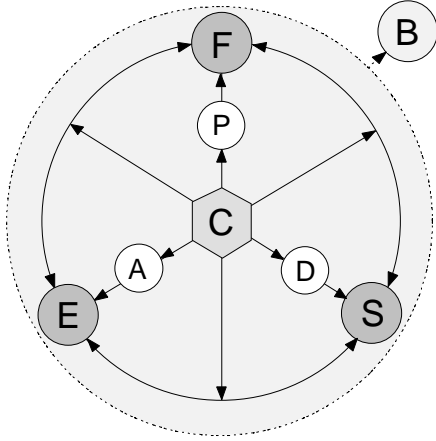


Figure 4 Formalized elements of a design concept

be referred to as *fundamental elements* of a design concept in the forthcoming part of the paper. A particular functionality is the basis of a discernible behavior b . The interdependence between entities e , situations s , and physical phenomena f is transitive and reflexive. The abstract scheme is depicted in Figure 4. This abstraction of the relationship between the fundamental elements supports us in formalizing the content of design concepts in general. As will presently become apparent, situations play a special role in how entities associate with physical phenomena.

The fundamental elements $\{e, s, f\}$ are depicted by qualifiers. They are *attributes*, a , *parameters*, p , and *descriptors*, d , in order. A design concept d materializes a specific domain D^3 of space E^3 . A functionally and morphologically separable domain D^3 of materialization is called an *entity*. An entity e , formed by a particular materialization, takes up its shape s_i and physical properties p_j according to the required functionality j . Therefore, entities can be characterized by their attributes a . A situation s associates entities and phenomena, as well as describes specific interactions between arbitrary number of entities and phenomena. The situations can be qualified by descriptors d . A configuration of entities relevant for a design concept is brought into functional connection by one or more physical phenomenon f . A phenomenon is a natural basis of a physical effect that can be described by physical laws and can be justified by common sense reasoning. Typically, a physical effect happens to occur as an interaction of either a field and an object, or two or more objects. A phenomenon is represented in terms of its process parameters p . Physical phenomena can be further classified based on their occurrence such as *internal* (when they appear inside the entities) or *external* (at the contacts of the entities).

The notion of constraint has been defined elsewhere as specification of a relation that must hold either on a variable or a set of variables (Tsang, E., 1993). In the context of design concepts, interdependencies of elements are governed by constraints. Thus, a constraint c_x may validate entity-situation, entity-phenomena and situation-phenomena relations. They will be denoted c_{es} , c_{ep} and c_{sp} constraints, respectively. Constrains also control the values of attributes, descriptors and parameters. These constraints are denoted c_a , c_d and c_p . The number of variables determines the arity (unary, binary, n-ary) of a constraint. Unidirectional and multi-directional constraints define asymmetrical relations, while bi-directional constraints are for symmetrical relations. A constraint assigns either a single value or a domain value to a value. An n -compound label (variable-value pair) represents a simultaneous assignment of n values to n -variables.

2. The process of design ontology specification

2.1. Selection of the domain of discourse

Mathematically, when we consider all concepts that are known for and/or can be generated by human beings, we create an *universe of discourse* (*UoD*). This is a set which in principle gives the basis of any design. In practice however only a specific part of the UoD has direct relevance to a given design process. These concepts are most probably related to: (a) the actual manifestation, functionality and appearance of a product (b) the technological circumstances in which a product is conceptualized, designed, manufactured, assembled, handled and so forth, and (c) the anticipated user environment. The amount of knowledge related to these aspects gives the contents of an ontology. The set formed by these pieces of knowledge constitutes the *domain of discourse* (DoD) of a design ontology. The domain of discourse is a structured set D of design concepts d so that $d \in D$, and (a) $d \neq f$, (b) $R(d, D)$, and (c) $P(d_1, d_2, \dots, d_N, \dots, d_N)$, where $N \neq \infty$. Here f denotes an empty concept, and R is the contextual relevance function for d on D . Possible values of R can be in the range of $0 < R \leq 1$, where 0 means irrelevant and 1 is the most relevant. The domain values of R assure that all concepts d even those that are seemingly not dedicated to a particular design application, can be incorporated to increase originality. P is a non-numeric function that assigns associations among design concepts based on their meanings.

There are two basic issues related to the definition of the domain of discourse. One of them is the issue of complexity of implementation, the other is the support of creativity. Although it is not evident by the first time they are complement. Complexity naturally arises when we endeavor to be exhaustive in conceptualization of an application domain and specification of an ontology. Then the development process lasts for long, needs tremendous efforts, results in overheads, and raises processing problems. An obvious action to cope with complexity would be limiting the domain of discourse. This pragmatic solution however negatively influences the usefulness of the design ontology. Typically this is what puts the issue of creativity forward. Any drastic decrease in the extent of the design ontology results in the closed world syndrome. It is well-known from AI research that a closed world reduces originality, creativity and flexibility of heuristic processes. Taking everything into consideration, construction of the domain of discourse for a design ontology is an optimization problem without clear-cut solutions. A proper solution supposes compromises in terms of the goals, requested coverage, available capacities and resources. Our experiences imply that it is expedient to apply a goal oriented modularity in exploration of the domain of discourse which in turn facilitates sparing time, labor and costs.

2.2. Identifying design concepts on the domain of discourse

Assume that selection, exploration and constraining of the domain of discourse D was successful. The next steps of ontology specification are (a) identification and naming all pertaining design concepts, (b) specifying the functional extent of the defined design concepts, and (c) compilation and structuring the pieces of knowledge that make a design concept explicit. In a more abstract way, specification of design concepts means

organizing the initially unordered knowledge particles into semantically complete *knowledge chunks*. In fact, each design concept \mathbf{d} corresponds to a design chunk.

The allocation of the pieces of knowledge to design concepts can be formalized as definition of a topology over a basis set. Let \mathbf{K} be the set of all pieces of knowledge \mathbf{k} that are meaningful to be identified for a particular \mathbf{D} . Due to our considerations regarding complexity, the set \mathbf{K} will have finite number of elements \mathbf{k} i.e. $card(\mathbf{k}) \neq \infty$. Now, let's specify subsets $\mathbf{d}^*, \mathbf{d}^* \hat{=} \mathbf{D}$ on \mathbf{K} , by applying a *selection criteria* \mathbf{G} for all \mathbf{k} so that:

$$\Gamma : \mathbf{k} \mapsto \{\emptyset, \mathbf{K}\} \begin{cases} \Gamma(\mathbf{k}) = \emptyset & \text{or} \\ \Gamma(\mathbf{k}) = \mathbf{k} \end{cases} \quad (1)$$

Then $\mathbf{d}^* = \hat{=} \mathbf{G}(\mathbf{k}), \mathbf{k} \hat{=} \mathbf{K}$. If no \mathbf{k} exists for a design concept \mathbf{d} either the scope of \mathbf{D} is defined improperly, or the selection criteria \mathbf{G} is specified inadequately. The selection criteria is a kind of meta-knowledge about whether a particular piece of knowledge \mathbf{k} is needed to describe a design concept \mathbf{d} or not.

Due to the facts that (a) design concepts can be identified at various levels of complexities, and (b) a particular piece of knowledge can be used to describe the content of more than one design concept \mathbf{d} different systems of subsets \mathbf{d}^* can be defined. A given system of subsets can be considered a topology \mathbf{t} on the set \mathbf{K} . The fact (a) above implies that coarser or finer ($\mathbf{t}_c, \mathbf{t}_f$) topologies can be defined depending on the construed complexity of the design concepts \mathbf{d} . Because $card(\mathbf{k}) \neq \infty$, both topologies \mathbf{t}_c and \mathbf{t}_f appear as topologies over finite number N of elements.

Fact (a) brings the *concept granularity* problem into the limelight. It is about what bestows the most efficient way of defining design concepts. The dilemma is that whether primitive design concepts or compound design concepts should be taken into consideration at specifying high level modeling entities, or both? Fact (b) implies the issue of *concept crystallization*. It relates to what pieces of knowledge \mathbf{k} should be included in a design concept \mathbf{d} to get a meaningful and, important enough, useful chunk of knowledge.

2.3. Aggregation of design concepts

Functionality of an artifact can be decomposed hierarchically to lower level functions. The decomposition terminates when the observed or supposed operation cannot be formulated without considering the aspects of materialization. The most elementary operations are implemented by coupled pairs of entities. Thus they will be considered also as *functioning primitives*. In order to support creativity it seems to be expedient to cover the domain of application by primitive design concepts. When this

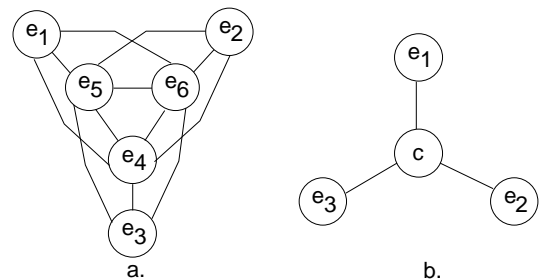


Figure 5 Aggregation of a sub-set of primitive design concepts into a compound: (a) before, (b) after.

highly granulated ontology is used to support conceptual design, most probable even the known higher-level design concepts must be re-synthesized from primitives. To avoid it, compound objects are to be defined for complex design concepts. The process of creating *functioning compounds* is called *aggregation*. It embeds a given number of primitive concepts into a compound concept and derives the total functions of the compound by extending the set of functions of the primitive concepts by those additional ones that are created by their interactions (Figure 5).

An accompanying problem of aggregation comes from the fact that certain primitive concepts will exist on their own and also as constituents of a compound concept. This is a surplus in the specification, but also demanding for the composition process based on the ontology. As far as conceptual design is concerned, functional primitives and compounds play equal roles. For ultimately low level operations functional primitives provide solution directly. At the same time a functioning compound may result in not requested functionality. Therefore, the redundancies in terms of functional primitives have to be accepted during specification, but reduced to the possible minimum during programming the design ontology.

2.4. Exploring interdependencies between design concepts

Design concepts are interdependent of each other based on their technical (semantic) content. The most important dependence concerns the manifestation of the dedicated functionality. From this point of view, four typical relationships can be identified among primitive design concepts.

The first form of interdependence appears when, to exist and/or to achieve a requested functionality, a design concept necessitates the functionality delivered by an other design concept. A curved door-lock handle is a good example of this kind of interdependence, since the concept of 'rotate due to torque' assumes the concept of 'converting force and movement into torque and rotation'. We call this relationship *for-connected*. The for-connected relationship is of 1 to N type, since more than once concept might be assumed to contribute to functioning of one single concept. An example for this from the field of physics is the concept of thrown object in space which assumes the concept of 'presence of gravity field' and 'move in space due to force effect', respectively.

Two design concepts are *equal-connected* if they are presenting the same functionality based on similar or dissimilar constituents (i.e., entities, situations, and/or phenomena). A piece of wire will be heated if it is exposed to either electric current, diffuse flame or inductive effect. From the point of view of a chosen concept, several alternative concepts might exist. Their interdependence can be described as equal-connected.

The third type of interdependence excludes simultaneous consideration of two design concepts when they demolish, limit or oppose each other functionality. This type of relationship is called *against-connected*. The design concepts of fluid lubrication and consuming kinetic energy by friction are against-connected. This means that due to their different functions they are not corresponding to a particular design solution.

The fourth type of relationship expresses the assertion that there is no interdependence between two or more design concepts. This relationship is termed *none-connected*. A

significant part of design concepts especially in mechanical engineering belongs to this category. Therefore, this functional relationship can be considered a default. The none-connected design concepts can be associated as it is governed by the pertaining constraints (c_{es} , c_{ep} , c_{sp} , c_a , c_d and c_p).

The four connections can be formalized mathematically. Let \mathbf{d} and \mathbf{d} two design concepts, $\mathbf{d}, \mathbf{d} \in \Delta$, and $\mathbf{d} \neq \mathbf{d} \neq \emptyset$. Let \mathbf{j}_i and \mathbf{j}_j the known function of \mathbf{d} and \mathbf{d} respectively. Then:

$$\begin{aligned}
 \text{for-connected:} & \quad \mathbf{j}_i \not\subset \mathbf{j}_j \ \checkmark \mathbf{j}_j \not\subset \mathbf{j}_i \text{ and } \mathbf{j}_i \Rightarrow \mathbf{j}_j \\
 \text{equal-connected:} & \quad \mathbf{j}_i \subset \mathbf{j}_j \ \checkmark \mathbf{j}_j \subset \mathbf{j}_i \text{ and } \mathbf{j}_i \Leftrightarrow \mathbf{j}_j \\
 \text{against-connected:} & \quad \mathbf{j}_i \not\subset \mathbf{j}_j \ \checkmark \mathbf{j}_j \not\subset \mathbf{j}_i \text{ and } \neg(\mathbf{j}_i \Rightarrow \mathbf{j}_j) \wedge \neg(\mathbf{j}_j \Rightarrow \mathbf{j}_i) \\
 \text{none-connected:} & \quad \mathbf{j}_i \not\subset \mathbf{j}_j \ \checkmark \mathbf{j}_j \not\subset \mathbf{j}_i \text{ and } (\mathbf{j}_i \mathbf{j}_j)
 \end{aligned}$$

Besides the four interdependencies among primitive design concepts discussed above, compound design concepts introduce two additional types of connections. The first one describes generalization and/or abstraction of a set of concrete design concepts. For instance, the design concept ‘cogwheel’ is a specialization of the design concept ‘wheel’, or, the other way round, the ‘wheel’ is a generalization of a ‘cogwheel’. The relationship between generic concepts and subordinate concepts is depicted by the term *class-connected*. This relationship is very useful for it enables us to reason with abstract concepts.

A relationship of different form exists between the primitive (or lower level) design concepts and a compound design concept that includes (embeds) them. This type of interdependence is called *piece-connected*. Obviously a weak overlap does exist with the class-connected relationship in that sense that the pieces might belong to an abstract category of design concept when it, in itself, represents a compound design concept. The notion ‘piece-connected’ has been introduced to specify ownership (possession) relations among compound and primitive design concepts.

2.5. Structural specification of a design ontology

Conceptualization of a domain of discourse results in a set of formalized design concepts and interdependencies among them. The interdependencies can be used for networking the design concepts. Arranging concepts based on their semantical relationships is the *structural specification* of the ontology. The result of the structural specification can be aptly represented as a network and portrayed by various graphical schemes. The aim of the graphical presentation is to visualize *inner structure* of the design ontology in order to support development and explanation of inferencing. There exist alternative forms of graphical representation, most of them is based on graph theory. Computer processing of design ontologies does not necessarily need graphical portraying.

The structure which is formed by the system of design concepts and their external connections has much common with semantic networks (known from artificial intelligence research). However, semantic relations are deduced from functional dependencies among the design concepts. In order to emphasize it as well as the other

differences, (a) the graphical representation shown in Figure 5, (b) the supporting mathematical constructs, and (c) the applied data management scheme have been collectively called *associative concept network* and acronymed as ACN.

Mathematically, an associative concept network (ACN) is defined as a finite, node- and edge-attributed graph and can be formalized as a 6-tuple:

$$ACN := \{ N, I, x, E, m, z \} \quad (2)$$

where:

- N is a finite, non-empty set of nodes, so as $N = C \cup \bar{C}$,
- C is a subset of primitive design concepts d_i ,
- \bar{C} is a subset of compound design concepts d_j ,
- I is a finite, non-empty set of attribute a and value u pairs assigned to the nodes C and \bar{C} as labels $I, I = \{ a, u \}$,
- x is the node labeling function, $x: N \rightarrow I$,
- E is a finite, non-empty set of edges, so as $E \subseteq N \times N$ and $E = E \cup \bar{E}$,
- \bar{E} is a subset of interdependencies between primitive design concepts d_i ,
- E is a subset of interdependencies between compound design concepts d_j and primitive design concepts d_k ,
- m is a finite, non-empty set of labels assigned to the edges E
- z is the edge labeling function, $z: E \rightarrow m$

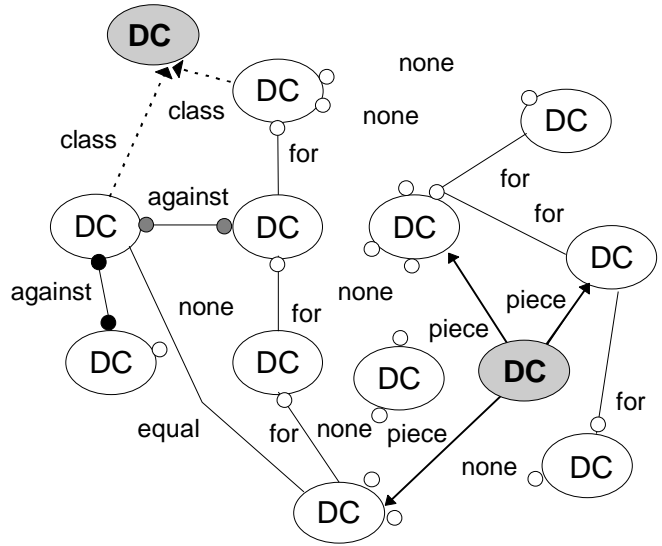


Figure 6 Graph representation of the inner structure of a design ontology

In the case of graph theory based visualization, a graph with marked nodes and edges is formed. The nodes of the graph represent the design concepts, either primitive or compound. In Figure 6, primitive concept nodes are empty ovals and compound concept nodes are shaded ovals. The edges and their labels indicate the four types of relationships introduced earlier. If there is no edge between two nodes it means that the type of connection has not yet been specified rather than that there is no connection between the concepts at all. The graph in Figure 6. is incomplete in this form, since there are several nodes whose connections have not yet been explicitly specified. Setting the connections between the primitive nodes to 'none-connected' as default is a good strategy to include all concepts in the specification. During the process of development of the ACN of a design ontology, defaulted labels can be redefined by other types of connections.

2.6. Specification of the contents of design concepts

A design concept contains either individual entities, pairs of entities, or a selection of entities. Each entity is identified by a unique name. The geometry of the design concepts, specific to the realized functions, is not fixed. The geometry of compound design concepts is made up by the included set of primitives. Therefore all included entities need generic (parametrized) geometry description. For generic modeling of the geometry of design concepts skeleton modeling has been applied. The entities are also modeled for their materialization and attributes. That is, besides shape data, all entities have geometry dependent, geometry related and geometry independent attributes. For the physical effects caused by a phenomena generally depend on the entity geometry, generic modeling of the geometry should support the treatment of physical effects.

Physical effects are caused by physical phenomena such us gravitational, mechanical, tribological, thermal, hydraulic, pneumatic, etc. The mechanical phenomena can sustain pressures, forces, moments, impulses, deformations, velocities and accelerations, etc. The full range of effects extends to, but is not limited to tribological, thermal, hydraulic, pneumatic, electric, magnetic, optical, etc., effects. Formalizaation of physical phenomena is based on functional and geometric *variables* and *coefficients* (constants) that are included in *expressions*. The expressions can be logical, numerical and/or symbolic.

The place of action of a physical effect has been called *port*. In the most general sense, they define the energy flow through the entities. Based on the direction of the energy flow (i.e. into the entity or out of the entity) contact ports has been classified as *in-ports* and *out-ports*. Certain physical effects occur inside the objects, the others between objects. Therefore, it is necessary to distinguish *mid-port effects* and *con-port effects*. Typical mid-port related effects are specific weight, deformation, heat capacity, center of gravity, internal pressure, strain energy, etc. Con-port effects occur between two or more physically touching ports or spatially separated ports. These effects are, for example, temperature, velocity, acceleration, normal force, shear force, bending moment, torque, surface pressure, friction force, liquid flow, fluid flow, gluing, welding, meshed connections, and so forth. Such features of the physical effects as initial conditions, change of coefficients, threshold values and limit values are characterized by constraints.

A situation describes a specific arrangement and interaction of entities and phenomena. It can be static and dynamic. A situation is considered static if the type and number of entities, phenomena and relations are constant. A situation is said to be dynamic, if the type and/or number of entities, phenomena and relations vary. This interpretation differs to that of Bardasz, T. and Zeid, I., (1992) that describes the circumstances under which an object best serves its goal. A situation has multiple views, therefore its comprehensive specification requires manipulation of all specific views. Descriptors of the situations are typical for the views. As most fundamental, structural, positional, morphological, kinematical, functional, dimensional and tense views have been identified.

2.7. Multiple views of a situation

The *structural view* of a situation describes which entities are involved in a design concept and whether they interact with each other. To specify a structural view descriptors ‘exist’, ‘not-exist’, ‘connected’ and ‘not-connected’ have been defined. Positional, morphological, kinematical and functional descriptors can be specified for connected entities only.

The *positional view* of a situation describes quantitatively how the entities are positioned and oriented in a three-dimensional space relative to each other. Positioning has both qualitative and quantitative aspects. In a hierarchical system of reference frames (systems of coordinates) a scheme of quantitative spatial relationships can be formulated. Qualitative positional descriptors can be systematically generated as Descartian products of the following three relation sequences: {*under, overlap, above*} {*left-to, overlap, right-to*} {*in-front, overlap, behind*}. Hence, if entity A is positioned for instance under, left-to and in-front of entity B, entity B is above, right-to and behind entity A. For entities that are topologically homeomorphic to a sphere, overlap has no meaning. But, for entities that are homeomorphic to a torus, two aspects of coincidence/inclusion can be identified. It is depicted by descriptors *inserted* and *surrounds*. A duality exists in linguistic naming of relative positions because referring and referred objects can be arbitrarily chosen. Thus, opposing descriptors can be formulated for alternative referencing of objects. The quantitative (dimensional) aspects of a positional view are described by descriptors of the dimensional view. Control over positional descriptors is introduced by constraints.

The *morphological view* interprets the situation in terms of the shapes of the entities and the morphological conditions of their physical connections. Contacts can be direct or indirect. In the case of indirect contacts the entities are not touching each other. Contacts can be specified among individual points on surfaces, domains on surfaces and/or complete surfaces. Only static contacts are called morphological, dynamic connections with certain degrees of freedom are termed kinematical. A morphological situation needs both qualitative and quantitative specification. Examples for qualitative descriptors can be seen in Figure 7 and 8. The contacts are characterized by (a) the type of surfaces in contact, (b) their spatial position and (c) orientation that is expressed by surface normals. Figure 7 illustrates the morphological descriptors for planar faces. Figure 8 shows the morphological descriptors for contacts of curved faces. Quantitative description involves specification of curvatures, distances and angles.

The *kinematical view* of a situation extends the morphological view with conceivable movements

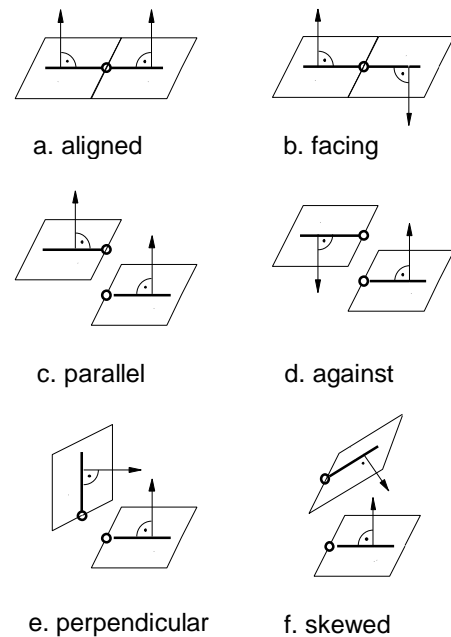


Figure 7 Morphological situations between planar faces

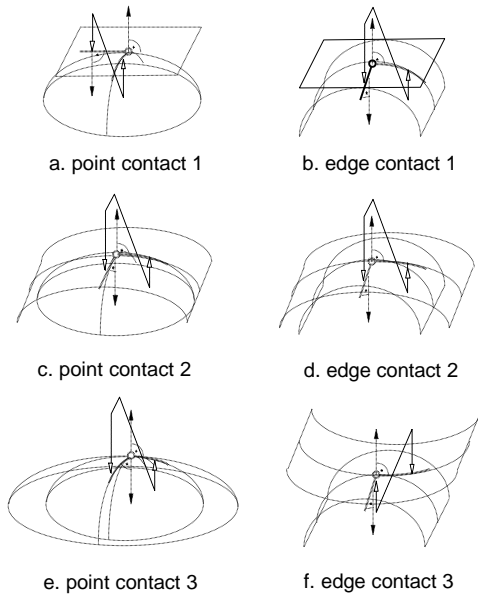


Figure 8 Morphological situations between curved faces

of the entities. Therefore, it specifies (a) the degree of freedom (*DoF*), (b) orientation of movement, and (c) the extent of relative movements of adjoining entities. Obviously, in an *xyz* orthogonal frame system, three translations $\{t_x, t_y, t_z\}$ and three rotations $\{r_x, r_y, r_z\}$ can be defined with alternate (+ or -) orientations. It gives 12 relative DoFs and needs 12 descriptors. For a mechanical coupled pair the possible movements have to be reasoned out from the knowledge provided by positional and morphological descriptors collectively. The degrees of freedom control possible relative motions between two or more entities and define transmissions (*translation - translation, translation - rotation, rotation - translation and rotation - rotation*). To describe all transmissions we need 4 more descriptors. Further descriptors are needed to specify transmitted directions which can be *same* or *counter*. Reversibility

needs descriptors *one-directional* and *bi-directional*. Finally, input-output exchangeability is to be defined by descriptors *exchangeable* or *non-exchangeable*.

The *functional view* of a situation describes the effects of a physical phenomenon on a given arrangement of entities. Either dependent or independent of time, physical effects can be represented by vector quantities. Vector quantities are specified by their reference, distribution, magnitude, direction and history, as descriptors. The *reference* relates an effect to its place of action. From the point of view of *distribution* an effect can be (a) concentrated to a point, (b) spread along a curve, and (c) spread over a surface. A *magnitude* specifies the measure of the effect. The *direction* gives the orientation of the vector. Finally, the *history* is a function that describes the time dependent profile of a quantity. Thus, each descriptor comprises one or more variable to be described. The coefficients of physical effects are generally scalar variables.

The *dimensional view* of a situation quantifies (a) dimensions of entities, (b) their absolute/relative positions and orientations, and (c) the places of actions of effects. Two descriptors have been introduced: (a) *distance*, and (b) *angle*. Distances and angles are valuated through metric variables, and can be absolute or relative.

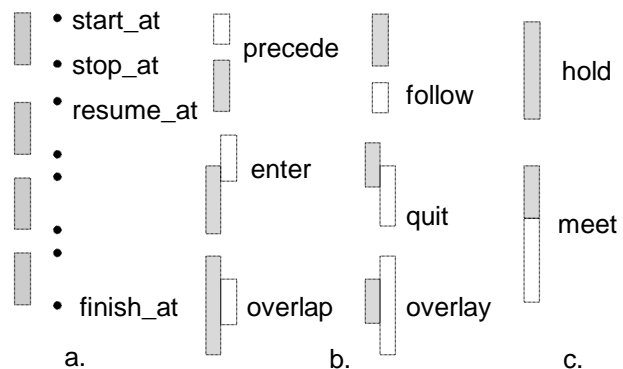


Figure 9 Tense descriptors (a) for the point of time of action, (b) for the duration of time, and (c) for the course of processes

The need for comprehensive treatment of physical effects hinted at consideration of the time. The processes related to design concepts imply a *tense* view of situations. The tense view adds extra descriptors that indicate points of time and periods of time for processes. There can be four descriptors introduced that specify point of time: (a) *start_at*, (b) *resume_at*, (c) *stop_at*, and (d) *finish_at* (Figure 9). There is a need for descriptors to indicate the relations between periods of time over which the physical processes (and/or sub-processes) come to pass. These are: (a) *precede*, (b) *follow*, (c) *overlap*, (d), *overlay*, (e) *enter*, and (f) *quit*. Furthermore, to characterize the process itself either at a point of time or in a period of time, descriptors (a) *hold* and (b) *meet* have been defined. These temporal relations are transitive, irreflexive and asymmetric. By using them all possible relations between processes can be adequately characterized.

2.8. Representation of the shape of entities

The materialized shape of the entities involved in a design should be represented by geometric means. The method of geometric representation is influenced by the following facts: (a) the shape is assumed to be defined by functional requirements, therefore the aspects of aesthetic design of shapes are neglected, (b) because during conceptual design the detailed shape is not yet known, vague representations can only be generated, (c) it is important to create at least some sort of initial geometric description that facilitates functional analysis and further detailing, and (d) shape representation can be direct (based on mathematical description of geometry) and/or indirect (based on alternative means such as symbolic language, verbal/textual circumscription, logic expressions). Because in the course of conceptualization the shape most probably evolves, it is wise to apply *generic shape description* for the entities rather than an instance oriented one. If the representation of shape relies on symbolic representation exclusively, difficulties may arise with the verbosity of the description and also with the efficiency of geometric calculations. As a result of the facts (b) and (c) *skeleton modeling* has been applied for geometric description of design concepts. For further details about the theory and methodology of skeleton modeling the reader is referred to (Horváth, I., Thernesz, V. and Bagoly, Z., 1995) and (Horváth, I. and Thernesz, V., 1995).

In order to illustrate the use of skeleton modeling for generic description of geometry, consider a simplified ball pen refill as a simple practical example. The arrangement is shown in Figure 10. The included entities are the ball, ball holder, ink tank, ink pool, and ink film. The paper is evidently not a part of the refill, it has been taken into account to assist functional specification.

The skeleton models of the entities can be seen in Figure 11, whereas the assembled entities (i.e., a skeleton model of the ball pen refill) can be seen in Figure 12. Mid-ports are as follows: 1 = mid-port of ink pool, 9 = mid-port of ball, and 10 = mid-port of

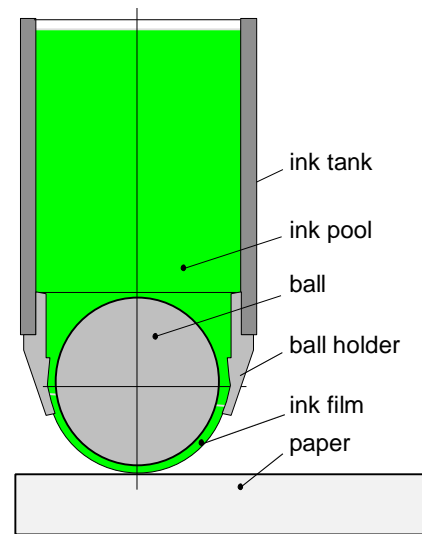


Figure 10 A simplified ball pen refill in use

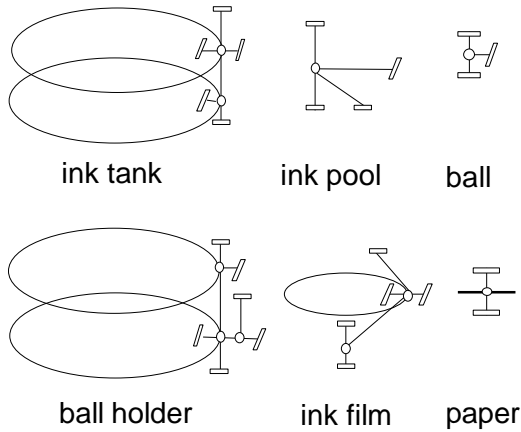


Figure 11 Skeleton models of the entities

paper. If there is more than one break point in the energy flow through the entities, mid-ports decompose into *sub-mid-ports*. For the skeleton model of the refill, ports 2-3 are sub-mid-ports of the ink tank, 4-5-6 are sub-mid-ports of the ball holder, and 7-8 are sub-mid-ports of the ink film. Sub-mid-ports 2, 3, 4, 5 and 7 are of enfolding type. The other mid-ports and sub-mid-ports are of branching type. The con-ports are: a = ink pool-air contact, b = ink pool-ink tank contact, c = ink pool-ink film contact, d = ink pool-ball contact, e = ink tank-air contact, f = ink tank-air contact, g = ink tank-ball holder contact, h = ink tank-ball holder contact, i = ball holder-air contact, j = ball holder-air contact, k = ball holder-ink film contact, l = ball holder-ink contact, m = ball-ink film contact, n = ball-ink film contact, o = ink film-paper contact, and p = paper-air contact.

In Figure 12 letters denote external contact ports, i.e., *con-ports*, and numbers denote internal ports, i.e., *mid-ports*. A con-port of the aggregated skeleton model which is under some physical effects (other than air pressure) but not in functional contact with a port of an other entity is either an *in-port* or an *out-port*. Interfacing functional design parameters appear generally on these ports. A con-port without functional contact and physical effects is said to be in *air contact*.

2.9. Assigning design parameters to skeleton structures

The geometry of con-ports is defined by surface patches. Besides their role in positional, kinematical and morphological definition, the surface patches dedicated to ports also define the shape of the entities in the vicinity of the reference points of contacts. The surface patches also stand for datum references for design variables and/or constraints. Thus, the aggregation of skeletons provide a basis (shape and structure) for geometric dimensioning of primitive and compound design concepts. This initial geometry is used to support reasoning about the behaviour and validity of the system formed by design concepts.

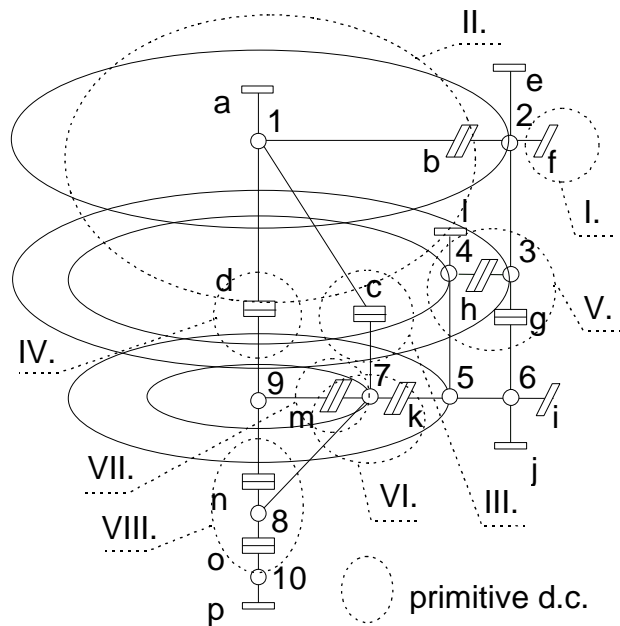


Figure 12 Skeleton model of the ball pen refill

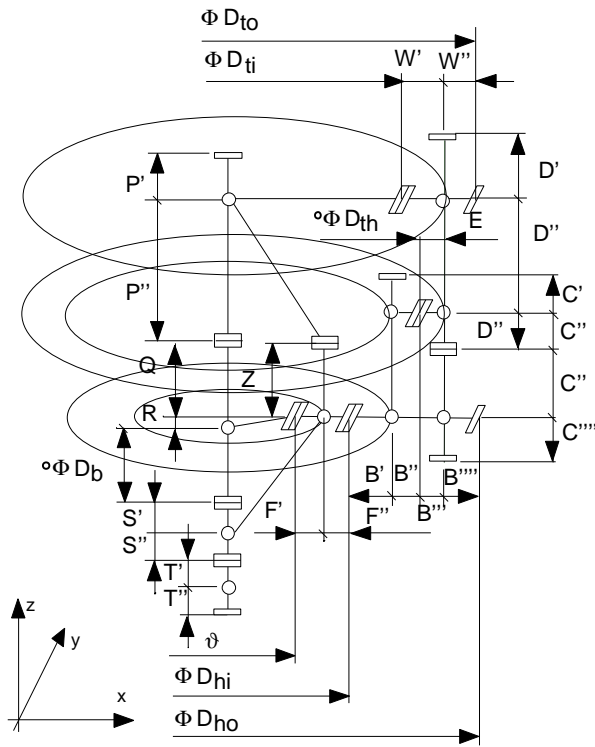


Figure 13 Geometric variables attached to the refill's skeleton

Figure 13 shows the *dimensional design variables* attached to the skeleton model of the refill. The chained piece dimensions (separated with a dash) play two roles simultaneously. For the entities, the piece dimensions specify the distances between ports. For the assembly of entities, the sum of the piece dimensions gives the gross dimensions of the included entities. Interpretation of the assigned design variables are as follows: ${}^{\circ}\mathcal{A}D_b$ = ball diameter, $\mathcal{A}D_{hi}$ = ball holder internal diameter, $\mathcal{A}D_{ho}$ = ball holder external diameter, $\mathcal{A}D_{ti}$ = ink tank internal diameter, $\mathcal{A}D_{to}$ = ink tank external diameter, $\mathcal{A}D_{th}$ = fitting diameter between the tank and the holder, B = total thickness of the ball holder, C = total height of the ball holder, D = total height of the tank, E = min. wall thickness of ink holder/ink tank, F = ink film thickness in the gap, P = height of the ink above the top of ball, Q = height of the ink above the center of the support, R = distance between the center of the ball and the center of the support, S = thickness of the ink film at the tip, T = thickness of paper, W = wall thickness of ink tank, and Z = separation height between the pool and ink. Geometric variables U (width of the ink print) and H (height of the ink print) are not shown.

The observable operation/performance of an object is called behavior. The function is the interpretation of the behavior under a desirable state which the object is expected to achieve. Functional specification of design concepts is based on variables, expressions, and constraints. Variables are associated with attributes, parameters and descriptors. They can be numerical, logical and/or symbolic. Geometric and functional variables are jointly named *design variables*. Expressions are mathematical relations between variables (and physical constants). Constraints can be typified as *functional* and *geometric constraints*. Functional constraints are associated with functional variables, and the geometric ones are defined over dimensional, positional, and morphological variables. In order to ensure the adequacy and validity of the situation in terms of permitted changes, simultaneous constraint management over both dimensional and functional variables is needed. The constraint solver differs from the conventional geometric ones in that it evaluates the expressions, and the functional and geometric constraints concurrently.

2.10. Functional specification of design concepts

To demonstrate the method of functional specification of design concepts we revisit the ball pen refill example. It has already been considered, that the refill comprises numerous design concepts. For an ultimate decomposition of the refill, we took all included functionally coupled pairs as primitive design concepts. The internal energy converters formed by the entities (i.e., the mid-ports of the skeletons) are also regarded as design concepts. The refill as an assembly, is a specific arrangement of these primitive design concepts. The refill as a whole can be characterized by a set of design variables, equations and constraints. To use the refill for writing on a piece of paper, only a force and a velocity is needed. Hence, the input functional variables are the force \underline{F}_w , the linear velocity \underline{v}_w , and time of writing t . The force \underline{F}_w is transmitted at in-port f , and the input velocity \underline{v}_w has also been applied to this in-port. Output variables are the amount of ink V left on the paper, and the compression force $\underline{F}_{wz,o}$ due to \underline{F}_w . The only out-port is port o , where the ink print sticks to the paper due to the pressure and adhesion.

Taking the implemented functions into consideration, we can decompose the refill into a set of primitive design concepts. Of course, a formal description of the included design concepts assumes the identification of effects of importance at con-port pairs, con-ports and/or at mid-ports, and the assignment of Functional variables are assigned as requested to provide a particular function. Thus, the design concepts, functional and geometric variables specifying the refill are as follows:

Design concept 1 Moving by constrained input force and velocity

- compression force $\underline{F}_{cz,f}$ due to \underline{F}_w , $\underline{F}_{cz,f} = \underline{F}_w \sin a$,
- traversal force $\underline{F}_{cx,f}$ due to \underline{F}_w , $\underline{F}_{cx,f} = \underline{F}_w \cos a$,
- velocity of move \underline{v}_w as applied, $\underline{v}_{x,f} = \underline{v}_w$.

Design concept 2 Storing low consistency liquid in a reservoir

- adhesive force $\underline{F}_{a,b}$ between the ink and the ink tank, $\underline{F}_{a,b} = c_a A_{a,b}$,
- hydraulic pressure $\underline{p}_{hx,b}$ on the tank, $\underline{p}_{hx,b} = \underline{r} \underline{g} P$,
- total volume of ink in the ink tank and in the ball holder, $V = (Q + P) D_{ii}^2 \underline{p} / 4 - PQ(3J^2 + Q^2) / 6$, auxiliary quantity $J = \sqrt{(D_b / 2)^2 - q^2}$.

Design concept 3 Material flow into a circular slot due to hydraulic pressure

- hydraulic pressure $\underline{p}_{hz,c}$ due to the height of the ink, $\underline{p}_{hz,c} = \underline{r} \underline{g} (P + Q - Z)$,
- ink flow V_c in the a ring slot, $V_c = v_c (D_{hi}^2 - D_b^2) \underline{p} / 4$.

Design concept 4 Material deposition on a rotating body

- total contact surface A_d for deposition, $A_{a,d} = A_{0,d} + v_d t l$.
- surface pressure $\underline{p}_{sz,d}$ due to the height of the ink, $\underline{p}_{sz,d} = \underline{r} \underline{g} (P + Q)$,
- adhesive force $\underline{F}_{a,d}$ between the ink and the rotating ball, $\underline{F}_{a,d} = c_a A_{a,d}$,

- material deposition $V_{d,d}$ from the ink pool on the ball, $V_{d,d} = V_{w,o}$,

Design concept 5 Axially supported radial fit between enclosing tubular bodies

- shear force $\underline{F}_{sx,h}$ due to the traversal force $\underline{F}_{cx,f}$, $\underline{F}_{sx,h} = \underline{F}_{cx,f}$,
- surface pressure $\underline{p}_{sx,h}$ due to shear force $\underline{F}_{sx,h}$, $\underline{p}_{sx,h} = 2\underline{F}_{sx,h} / C'' D_{th}$,
- normal force $\underline{F}_{nz,g}$ due to compression force $\underline{F}_{cz,f}$, $\underline{F}_{nz,g} = \underline{F}_{cz,f}$,
- surface pressure $\underline{p}_{sz,g}$ due to normal force $\underline{F}_{nz,g}$, $\underline{p}_{sz,g} = 4\underline{F}_{nz,g} / ((D_{th} + E)^2 - D_{th}^2) \mathbf{p}$.

Design concept 6 Shear stress in a liquid in adhesive contact with bodies of relative movement

- adhesive force $\underline{F}_{a,k}$ between the ink and the ball holder, $\underline{F}_{a,k} = c_a A_{a,k}$,
- adhesive force $\underline{F}_{a,m}$ between the ink and the ball, $\underline{F}_{a,m} = c_a A_{a,m}$,
- relative velocity $\underline{v}_{r,km}$ between the ball and the ball holder, $\underline{v}_{r,km} = \underline{v}_k - \underline{v}_m$,
- shear stress $\underline{\tau}_{r,km}$ in the ink film in the circular slot, $\underline{\tau}_{r,km} = G \underline{v}_{r,km} t / F$.

Design concept 7 Supporting a revolving object by a seat object through a mesh connection

- rolling of the ball with an angular velocity \underline{w} in the ball holder, $\underline{w} = 2\underline{v}_w / D_b$,
- radial force $\underline{F}_{rx,m}$ due to the traversal force $\underline{F}_{cx,f}$, $\underline{F}_{rx,m} = \underline{F}_{cx,f}$,
- axial force $\underline{F}_{az,m}$ due to the compression force $\underline{F}_{cz,f}$, $\underline{F}_{az,m} = \underline{F}_{cz,f}$,
- resultant force $\underline{F}_{R,m}$ due to the radial $\underline{F}_{rx,m}$ and axial force $\underline{F}_{az,m}$,
 $\underline{F}_{R,m} = \sqrt{\underline{F}_{rx,m}^2 + \underline{F}_{az,m}^2}$,
- surface pressure $\underline{p}_{sx,m}$ on the ball due to the compression force $\underline{F}_{R,m}$,
 $\underline{p}_{sx,m} = \underline{F}_{R,m} / D_b \mathbf{p} R$.

Design concept 8 Material deposition on a stationary object from a revolving object due to adhesion and compression force

- adhesive force $\underline{F}_{a,n}$ between the ink and the ball, $\underline{F}_{a,n} = c_a A_{a,n}$,
- adhesive force $\underline{F}_{a,o}$ between the ink and the paper, $\underline{F}_{a,o} = c_a A_{a,o}$,
- rolling of the ball on the paper with an angular velocity \underline{w} , $\underline{w} = 2\underline{v}_w / D_b$,
- moving of the ball on the paper with a linear velocity \underline{v}_o , $\underline{v}_o = \underline{v}_w$,
- volume V^* of ink film, $V^* = 2 / 3 ((D_b + 2S)^3 - D_b^3) \mathbf{p}$
- compression force $\underline{F}_{cz,o}$ between the ball and the paper, $\underline{F}_{cz,o} = \underline{F}_{cz,f}$,
- rolling resistance $\underline{F}_{r,o}$ between the ball and the paper $\underline{F}_{r,o} = \mathbf{r} \underline{F}_{cz,o}$,

- material deposition $V_{d,o}$ from the ink to the paper, $V_{d,o} = c_{vf}(v, f) U H v t$.

These concepts are indicated by dotted ovals and roman numbers in Figure 12. An associative concept network of the refill looks like as shown in Figure 14. In coding of design concepts entity names are further abstracted.

Generally, a situation does not specifies conditions under which the interrelations, properties and relationships, and causalities hold. We has to make it explicit by specifying constraints. For the refill case, to maintain proper functioning, geometric and functional constraints must also be specified for and amongst the design concepts. The primary geometric constraints are: $D_b > 0, D_{hi} > D_b, D_{ti} > D_b, E, F > 0$. The functional constraints are: $\underline{E}_{cx,f} > \underline{E}_{r,o}, \underline{p}_{sz,g} > \underline{t}_{r,km}, P > 0, \underline{V} t = V^*, V > 0, V_{d,d} = V_{d,o}$. The constraint network, a well-arranged set of constraints, can be *under-constrained* and *over-constrained*. A comprehensive management of constraints allows us to check not only for validity of entities, but also to account for causalities k related to a situation s . Causality can be traced by the sequence of processing of mathematical relationships.

2.11. Compilation of function specifications for applications

Any time when a design concept included in an ontology is to be selected, the selection can made based on the functional and/or attribute specifications. The basis of browsing and matching is the ACN graph of the design ontology. The aggregate functional specification for the design ontology can be collected and filtered out by the following procedure.

Let F^* be the set of individual functional specifications j_i that exist at all in the descriptions of design concepts d_j , so as $\Phi^* = \bigcup_{i=1..m} j_i$. Let F be the set of the aggregated functional specifications F^* , so as $\Phi = \bigcup_{j=1..n} \Phi_j^* = \bigcup_{j=1..n} \bigcup_{i=1..m} j_{j,i}$. Hence, F is

actually the superset of the sets of functional specifications for the design concepts.

Let's create one additional set $F = F \ominus \Phi$ by removing all but one of the redundant functional specifications $j_{j,i}$. That is, for any j,i and t,s , if $j_{j,i} = j_{t,s}$ then $F \ominus j_{j,i}$ else $F \ominus j_{t,s}$. Thus $F \ominus$ is the set of distinctive functional specifications. The contents of the set $F \ominus$ is the functional specification of the design ontology as a whole. It can be used to find design concepts by matching the functional specifications for the design concepts sought after.

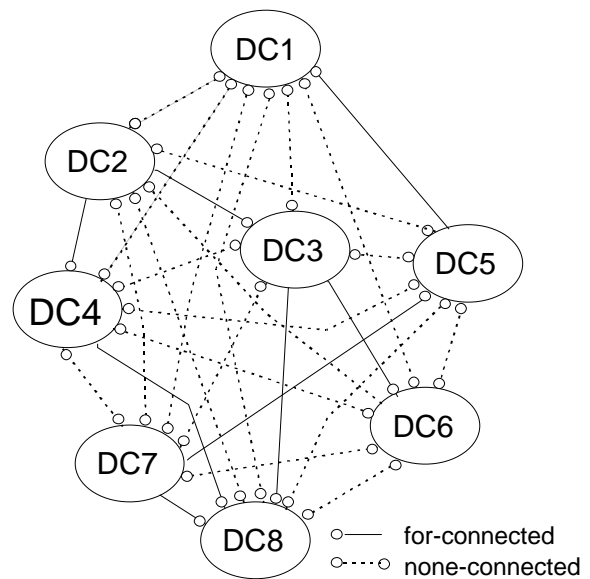


Figure 14 The ACN for the refill case

3. Implementation and application of design ontologies

3.1. Computer representation of design concepts

Implementation of a design ontologies has two levels and, thus, two phases. One is logical definition, the other is programming. The framework of logical definition is a library of declarative expressions called *ACN-Code*. From the declarative expressions schemes are created that represent the semantic elements of the design ontology. This formalism is a kind of intermediate language that bridges natural language representation and computer processable definitions. The schemes of the *ACN-Code* are instantiated by actualizing the declarations. Program level specification is based on a high level programming language. Practically, the vocabulary and constructs of any fourth generation programming language can be used if they are in harmony with the schemes of the *ACN-Code*. The authors used C++ programming language for programming. The two-step mapping of the conceptualization offers the advantage of keeping semantics independent of the final representation, coding and/or user.

Due to the various forms of knowledge related to the fundamental elements of design concepts, different representations, e.g., procedural, numeric and symbolic, are appended to the declarative schemes. The expressions, functional and validity constraints are defined in a symbolic form. The geometries of the entities are represented as numeric data structures corresponding to the STEP geometry specification. Miscellaneous descriptions are in textual form. A practical reason of using mixed representation was that declarations and rules are difficult to formalize when definition of design concept relies exclusively on procedural and/or numeric representation. On the other hand, when the specification is based exclusively on symbolic representation, difficulties arise with processing the geometry and quantitative relationships.

Below an example is presented for specification of a design concept. Numerical and textual data are in external text files and data files, respectively. The specification regards to *primitive design concept No. 8*. Due to space limitation, certain repeated parts of the specification have been deliberately omitted. The missing definitions are analogous with the presented ones.

```
Primitive_Design_Concept [#.8](Exist = 1, Connected = 0)
  Start_Identify_Primitive_Design_Concept
    Primitive_Design_Concept [#.8,..0](Name [#.8,..0]() &
      Functioning [#.8,..0]() &
      Description [#.8,..0]() &
      Reference_Point [#.8,..0]() &
      Primitive_Generic_Frame [#.8,..0]()),
    Name [#.8,..0]([String = Marking_objects_by_liquid]),
    Functioning [#.8,..0]([String = 'Material deposition on a stationary object from a
      revolving object due to adhesion and compression force']),
    Description[#.8,..0]([Text_File_Name {String = Void}],
    Reference_Point [#.8,..0](X_Coord [Single_Value {Real = 0}] &
      Y_Coord [Single_Value {Real = 0}] &
      Z_Coord [Single_Value {Real = 0}]),
```

```

Primitive_Generic_Frame [#.8,..0](Vector_X_Axis [#.8,..0]() &
  Vector_Y_Axis [#.8,..0]() &
  Vector_Z_Axis [#.8,..0]()),
Vector_X_Axis [#.8,..0](Reference_Point [#.8,..0]() &
  Magnitude [Single_Value {Integer = 1}] &
  Direction [Single_Value {Notion = X_Direction}] &
  History [{Function = Void}]),
Vector_Y_Axis [#.8,..0]( *** ),
Vector_Z_Axis [#.8,..0]( *** ),
End_Identify_Primitive_Design_Concept,
Start_Specify_Primitive_Design_Concept
Start_Specify_Object
Start_Entities
  Entity [#.8,..1] (Name [#.8,..1]() & Reference_Point [#.8,..1]() &
    Entity_Frame [#.8,..1]() & Geometry [#.8,..1]() & Attributes [#.8,..1]()),
  Entity [#.8,..2] ( *** ),
  Entity [#.8,..3] ( *** ),
  Entity [#.8,..4] ( *** ),
End_Entities,
Start_Names
  Name [#.8,..1]({String = Ball}),
  Name [#.8,..2]({String = Ink_Film}),
  Name [#.8,..3]({String = Paper}),
  Name [#.8,..4]({String = Support}),
End_Names,
Start_Reference_Points
  Reference_Point [#.8,..1](X_Coord [Single_Value {Real = Void}] &
  Y_Coord [Single_Value {Real = Void}] &
  Z_Coord [Single_Value {Real = Void}]),
  Reference_Point [#.8,..2] ( *** ),
  Reference_Point [#.8,..3] ( *** ),
End_Reference_Points,
Start_Frames
  Entity_Frame [#.8,..1](Vector_X_Axis [#.8,..1]() &
  Vector_Y_Axis [#.8,..1]() &
  Vector_Z_Axis [#.8,..1]()),
  Vector_X_Axis [#.8,..1](Reference_Point [#.8,..1] &
  Magnitude [Single_Value {Integer = 1}] &
  Direction [Single_Value {Notion = X_Direction}] &
  History [{Function = Void}]),
  Vector_Y_Axis [#.8,..1] ( *** ),
  Vector_Z_Axis [#.8,..1] ( *** ),
  Entity_Frame [#.8,..2] ( *** ), " " "
  Entity_Frame [#.8,..3] ( *** ), " " "
End_Frames,
Start_Geometries

```

```

Geometry [#.8,..1](Reference_List [{Mid_Port [#.8,..1]() & In-Port [#.8,..1]() &
    Out-Port [#.8,..1] & Shape-Port [#.8,..1]}]),
Mid_Port [#.8,..1](Type [Single_Value {Notion = Branching }] &
    Geometric_Points [Data_File_Name {String = Void}] &
    File_Pointers [Pointer_List {Pointers (1) = Void}] &
    Coordinates [Triple_Values {Double_Floating = Void, Void, Void}],
In-Port [#.8,..1](
    Geometric_Points [Data_File_Name {String = Void}] &
    File_Pointers [Pointer_List {Pointers (1) = Void}] &
    Coordinates [Triple_Values {Double_Floating = Void, Void, Void}] &
    Morphological [#.8,..1]({Void})),
Out-Port [#.8,..1](
    Geometric_Points [Data_File_Name {String = Void}] &
    File_Pointers [Pointer_List {Pointers (1) = Void}] &
    Coordinates [Triple_Values {Double_Floating = Void, Void, Void}] &
    Morphological [#.8,..2]()),
Shape-Port [#.8,..1](
    Geometric_Points [Data_File_Name {String = Void}] &
    File_Pointers [Pointer_List {Pointers (1) = Void}] &
    Coordinates [Triple_Values {Double_Floating = Void, Void, Void}] &
    Surface_Patches [Data_File_Name {String = Void}] &
    File_Pointers [Pointer_List {Pointers (x) = Void}] &
    Control_Data_Set [List_Value {Double_Floating = Void }]),

Geometry [#.8,..2] ( *** ),
Geometry [#.8,..3] ( *** ),
End_Geometries,

Start_Attributes
Attributes [#.8,..1](Material [Single_Value {String = Steel}] &
    Surface_Finish [Domain_Value {Max_Real = +0.8, , Min_Real = +0.4}] &
    Size_Tolerance [Domain_Value {Max_Real = +.02, Min_Real = -.02}] &
    Shape_Tolerance [Single_Value {Real = +0.01}] &
    Yield_Stress [Single_Value {Real = +320}],
Attributes [#.8,..2] ( *** ),
Attributes [#.8,..3] ( *** ),
End_Attributes,
End_Specify_Objects,

Start_Specify_Situations
Start_Structural
Structural [#.8,..1](Exist [Double_Value {Integer = 1, Entity[#.8,..1]}]),
Structural [#.8,..2](Exist [Double_Value {Integer = 1, Entity[#.8,..2]}]),
Structural [#.8,..3](Exist [Double_Value {Integer = 1, Entity[#.8,..3]}]),
Structural [#.8,..4](Non_Exist [Single_Value {Entity[#.8,..4]}]),
Structural [#.8,..5](Connected [Reference_List {Entity [#.8,..1], Entity [#.8,..2],
    Phenomena [#.8,..1]}]),
Structural [#.8,..6] ( *** ), .. .. .. ..
Structural [#.8,..13](Non-Connected [Reference_List {Entity#.8,..4}]),
End_Structural,

Start_Positional

```

Positional [#.8,..1](*Placement* [*Reference_List* {*Entity* [#.8,..1],
Primitive_Generic_Frame [#.8,..0]}] & *Dimensional* [#.8,..1]() &
Dimensional [#.8,..2]()),
Positional [#.8,..2] (***),
Positional [#.8,..3] (***),
End_Positional,
Start_Morphological
Morphological [#.8,..2](*Point_Contact_Type*%1 [*Reference_List* {*Entity* [#.8,..2],
Entity [#.8,..1]}],
Morphological [#.8,..5] (***),
End_Morphological,
Start_Kinematical
Kinematical [#.8,..1] (*Move_In_X* [*Reference_List* {*Entity* [#.8,..3],
Entity [#.8,..1]}] & *Direction* [*Double_Value* {*Notion* = *Positive_Direction*,
Notion = *Negative_Direction*}] &
Magnitude [*Domain_Value* {*Max_Real* = $+\infty$, *Min_Real* = 0}]),
Kinematical [#.8,..2] (***), " " " "
Kinematical [#.8,..7] (*Fused* [*Reference_List* {*Entity* [#.8,..2], *Entity* [#.8,..1]}]),
Kinematical [#.8,..8] (*Fused* [*Reference_List* {*Entity* [#.8,..2], *Entity* [#.8,..3]}]),
End_Kinematical
Start_Functional
Effect [#.8,..1](*Name* [{*String* = *Adhesive_Force* }] & *Effect_Variable* [#.8,..1]()
& *Vector* [#.8,..1]()),
Effect_Variable [#.8,..1]([*Double_Value* {*Symbol* = $\underline{E}_{a,n}$!, *Real* = *Void*}]),
Vector [#.8,..1](*Distribution* [*Reference_List* {*Surface_Patch* [#.8,..1]()}] &
Direction[*Reference_List* {*Surface_Normal* [#.8,..1]}] &
Magnitude [*Single_Value* {*Real* = *Void*}] &
History [{*Function* = *Void*}]),
Surface_Patch [#.8,..1]([*Data_File_Name* {*String* = *Void*}] &
File_Pointers [#.8,..1]([*Pointer_List* {*Pointers* (i) = *Void*}]) &
Control_Data_Set [#.8,..1]([*List_Value* {*Double_Floating* (j) = *Void*}])),
Surface_Normal [#.8,..1]([*Data_File_Name* {*String* = *Void*}] &
File_Pointers [*Pointer_List* {*Pointers* (k) = *Void*}] &
Coordinates [*Triple_Values* {*Double_Floating* = *Void*, *Void*, *Void*}])
Effect [#.8,..2] (***), " " " "
Effect [#.8,..8] (***),
End_Functional,
Start_Dimensional
Dimensional [#.8,..1](*Absolute_Distance* [*Length* {*Reference_Point* [#.8,..1],
Reference_Point [#.8,..0]}] & *Magnitude* [*Single_Value* {*Real* = *Void*}]),
Dimensional [#.8,..2] (***), " " " "
Dimensional [#.8,..6] (***),
End_Dimensional,
Start_Temporal
Temporal [#.8,..1](*Hold* [*Reference_List* {*Effect* [#.8,..1], *Effect* [#.8,..2],
Effect [#.8,..8,]}] & *Priod_Of_Time* [*Double_Value* {*Real* = *Void*,
Real = *Void*}]),

```

    Temporal [#.8,..2] ( *** ),
    Temporal [#.8,..3] ( *** ),
    End_Temporal,
    End_Specify_Situations,
    Start_Specify_Phenomena
    Phenomena [#.8,..1] (Name [{String = 'Adhesive force between the ink and the
        ball'}]) & Effect_Variable [#.8,..1] & Coefficient [#.8,..1]() & Factor [#.8,..1]()
        & Expressions [{Symbol = !Fa,n = caAa,n!}],
    Coefficient [#.8,..1]([Double_Value {Symbol = !ca!, Real = 0.35}]),
    Factor [#.8,..1]([Double_Value {Symbol = !An!, Real = 0.016}]),

    Phenomena [#.8,..2] ( *** ), .. .. .. ..
    Phenomena [#.8,..8] ( *** ),
    End_Specify_Phenomena,
    Start_Specify_Constraints,
    Constraints[#.8,..1](Meaning [{String = Void}]) &
        Pertain_To [Reference_List {Factor [#.8,..4]}] &
        Content [Expressions {Symbol = !Db > 0!}],
    Constraints[#.8,..2] ( *** ), .. .. .. ..
    Constraints[#.8,..7] ( *** ),
    End_Specify_Constraints,
    End_Specify_Primitive_Design_Concept
    Primitive_Design_Concept [#.8]

```

Examples of specifications of compound design concepts and a design ontology as a whole cannot be included here. Interested readers are referred to the web-site <http://www.io.tudelft.nl/research/ica/>.

3.2. How to use an ACN for contextual conceptualization of designs

Contextual conceptualization of a design (product) consists of the steps of (a) searching for design concepts, (b) selection based on functional specification, and (c) combination and arrangement of appropriate design concepts. This process can be supported by design ontologies. Actually this form of conceptualization has much common with the conceptualization of an ontology for a particular DoD. Nevertheless, clear distinction can be made between them because of the different objectives. Conceptualization of an ontology is knowledge oriented, while contextual conceptualization is artifact oriented. Conceptualization of a brand new product, or a part of it, can be based on the associative concept network of the design ontology of the DoD concerned. When used for contextual conceptualization, information about functionally adequate design concepts can be recalled by browsing the ACN in response to a more or less complete functional specification. The ACN implies its own rules of inference that can also be used to reason about the possible interfacing of design concepts. The principle of application for contextual conceptualization is shown in Figure 15.

To demonstrate the process of selection and association of design concepts, assume that an ontology O is available in the form of an associative concept network ACN. Furthermore, assume that the compiled set of functional specification $F C$ is also available.

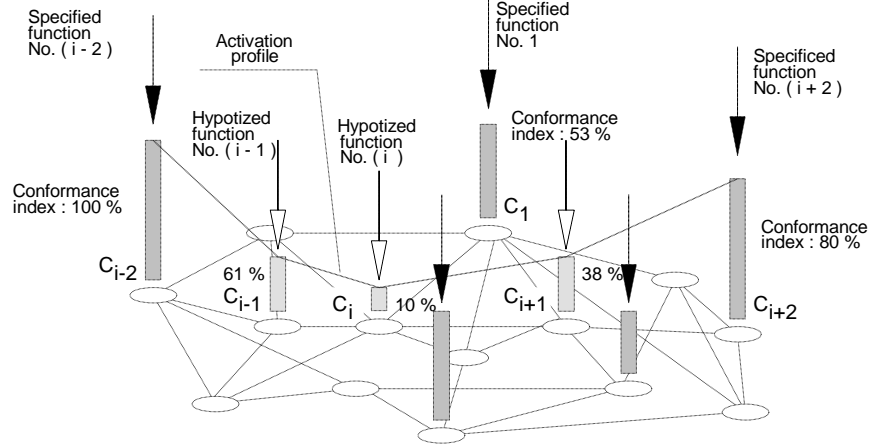


Figure 15 The principle of contextual conceptualization

From the set $F C$ select a set F° of the requested functions, where $F^\circ = \bigcup_{i=1..p} j_i$. Then, map all j_i functions, $i = 1..p$, of the set F° onto the equivalent functional specifications $j_{j,k}$ of F so as $j_i = j_{j,k}$. For all j_i that has been matched successfully, find the design concept d_j . Compute the conformance index of matching c_j for all design concepts d_j by dividing the number s of found functions by the total number t of functional specifications in d_j . Store the values of c_j . Activate those design concepts d_j that has the highest conformance values c_j for the given set functions. For each find the 'for-', 'against-', 'equal-' and 'none-connected' neighbors. Neglect all design concepts $d_{n,k}$ that are in none-connected associations. Deactivate (exclude) all design concepts $d_{u,l}$ that are directly (by one edge) against-connected to d_j . Find, indicate and store those design concepts $d_{u,m}$ that are equal-connected neighbors. These concepts will be alternatives for d_j . Lastly, find and activate all design concepts $d_{j,n}$ that are directly for-connected to d_j .

If there is no for-connected neighbor, then ask the user to select further functional specifications. If a for-connected neighbor is found, activate it. Hypotize its related functional specification F^* as a requested one, and present it to the user for approval. If approved, label and store this $d_{j,n}$. Repeat this process for all for-connected associations. If a design concept d_j has been reached from more than one node of the ACN, increase its conformance index. The concepts activated by for-connected associations form alternative paths in the ACN. Among these paths having different activation profiles the one with the highest aggregated conformance index is the best solution.

3.3. An example for ACN application

A design ontology has been developed for conceptualization of microcomputer systems. The tasks are (a) to find the most appropriate components that meet the functional requirements, and (b) to configure and arrange the components based on their functional connections. In this case the associative concept network ACN helps to explore the space of design concepts (sub-solutions) and stimulates a creative composition. This application is featured by the predominating existence of compound design concepts. To conceptualize the application field more than 300 compound design concepts have been included into the design ontology. The number of out-of-compound primitive design concepts thus a slightly more than 50. The associative network manager provides windows for input the data needed to actualize the schemes of the ACN-Code. It presents the graph of the ACN during its editing and browsing in graphical windows. Figure 16 shows the graphical representation of a part of the ACN graph worked out for conceptualization of microcomputer architectures. The vertical bars in the graphical symbols of the nodes of the ACN graph refer to compound and primitive design concepts, respectively. The smaller blocks on the edges between the node-blocks describe the types of connections. The horizontal bars above the node-blocks show the conformance value c_j of the concerned design concept for a given function. The inference engine that works on the application oriented ACN selects the appropriate design concepts against a set of user specified functional requirements. During the composition process, the inference engine also checks the interface features (contact surfaces at ports, dimensional parameters, physical parameters, etc.) to exclude concepts that are physically not, or only partly, matching. The output of the ACN modeller is a structural description of the arrangement of design concepts for the required functionality.

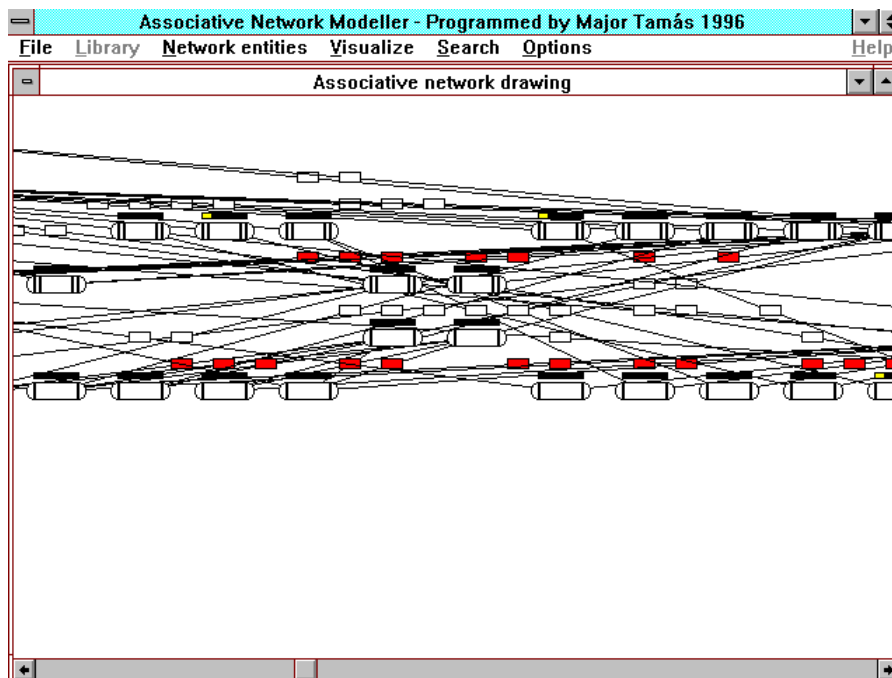


Figure 16 Graphical presentation of the ACN for conceptualization of microcomputers

A unique feature of the ACN structure is that when an adequate design concept has been selected the related concepts become also pointed at due to the pre-defined associations. It means that the ACN is able to work with incomplete functional specifications and to suggest initially

not conceived sub-solutions. This form of composition of design concepts is very close to the intuitive conceptualization by human beings. Although using ACNs introduces some sort of automatism in functional and structural design, it assumes high level interaction with the user. Nonetheless, it needs the expert knowledge of the user to judge proposals in the lack of all-embracing objective functions.

4. Conclusions

The novelty of the work presented in this paper is in that it applied the ontology paradigm to formalize design concepts. Also it presented a well-suited application where the advantages of formal specification of associations can be utilized. For specification of design concept ontologies a dedicated methodology was applied. Its elements are: (a) selection and limiting the domain of discourse, (b) identifying the related design concepts, (c) aggregation of design concepts, (d) exploring external interdependence of design concepts, (e) structural specification of the design ontology, (f) contextual specification of design concepts, (g) representation of the shape of entities, (h) functional specification of design concepts, (i) validating the design ontology, and (j) coding design concepts as very high level modeling entities. The main difference between traditional (feature oriented) and ontology-based definition of modeling entities is that the former captures object related knowledge only, while the latter extends to domain knowledge too. Our conclusions are:

1. Implemented as a very high level modeling (VHLM) entity, a design concept object acts as if it had the knowledge and reasoning capabilities to achieve its requested functionality in various design applications. It lends itself to an agent-like performing.
2. The ACN graph representation of the design ontology has been applied for contextual conceptualization. The domain knowledge embedded in the design ontology is used to improve the way of obtaining solutions for a given set of functions, rather than the object knowledge related to design concepts alone.
3. Future research is supposed to explore a more theoretical way of design concept definition than the presented one. It is empirical in that it followed a speculative way of finding out the content and contextual dependencies of a design ontology and the design concepts. It is also for future research how to make concept generation unique and unambiguous.
4. It is an absolutely open issue how to create a population of materializable design concepts quasi automatically, if possible at all. As well, it needs further research how to further articulate possible interdependencies and conceivable interactions of design concepts, especially in unknown circumstances (situations).

5. Acknowledgment

This project has been partly supported by grants No. T12814 and T017598 of the National Scientific Research Foundation (OTKA). The authors wish to thank to Mr. Tamás Major for his contribution to the first implementation of the ACN software.

6. References

- Bardasz, T., Zeid, I.: "Cognitive Model of Memory for Mechanical-Design Problems", *Computer Aided Design*, Vol. 24, No. 6, 1992, pp. 327-342.
- Dohmen, M.: "Constraint-Based Feature Validation", Ph.D. Thesis, Delft University of Technology, Delft, 1997.
- Eekels, J.: "On the Ontology of Simulation in Engineering", in *Proceedings of the 11th International Conference on Engineering Design*, Vol. 2, ed. by Riitahuhta, A., Tampere University of Technology, Tampere, 1997, pp. 455-460.
- Fox, M. S., Barbuceanu, M., Gruninger, M., "An Organisation Ontology for Enterprise Modelling: Preliminary Concepts for Linking Structure and Behaviour", *Computers in Industry*, Vol. 29, 1996, pp. 123-134.
- Genesereth, M., Fikes, R.: "Knowledge Interchange Format Reference Manual - Version 3", Stanford University, CSD Tech-Report Logic 92-1, 1992.
- Grabowski, H., Rude, S., Pocsai, Zs.: "Ontology Technology: The Key for Intelligent Migration and Retrieval of Information in Engineering Networks", in *Proceedings of the 11th International Conference on Engineering Design*, Vol. 2, ed. by Riitahuhta, A., Tampere University of Technology, Tampere, 1997, pp. 231-236.
- Guarino, N., Giaretta, P.: "Ontologies and Knowledge Bases - Towards a Terminological Clarification", in *Towards Very Large Knowledge Bases - Knowledge Building and Knowledge Sharing*, ed. by Mars, N. J., IOS Press, 1995, pp. 25-32.
- Guarino, N., Poli, R.: "The Role of Formal Ontology in the Information Technology", *International Journal of Human-Computer Studies*, Vol. 43, No. 5/6, 1995, pp. 623-624.
- Horváth, I., "A Workbench Architecture for Object Oriented Handling of Features", in *Proceedings of The 1996 ASME Design Engineering Technical Conferences and Computers in Engineering Conference*, ASME, New York, 1996, (CD-ROM).
- Horváth, I., Thernesz, V., Bagoly, Z.: "Conceptual Design with Functionally and Morphologically Parametrized Feature Objects"; in *Proceedings of the Computers in Engineering Conference*, ed. by Busnaina, A. A., ASME, New York, 1995, pp. 507-516.
- Horváth, I., Thernesz, V.: "Morphology-Inclusive Conceptual Modelling with Feature Objects"; in *Proceedings of The Fourth International Conference on CAD/CG*; ed. by Yang, S., Zhou, J., Li, C., SPIE, Wuhan, 1995, pp. 804-813.
- Horváth, I., "Design Concept Formalization Based on Ontologies", in *Proceedings of the 11th International Conference on Engineering Design*, Vol. 3, ed. by Riitahuhta, A., Tampere University of Technology, Tampere, 1997, pp. 219-224.
- Horváth, I., Kulcsár, P., Thernesz, V.: "A Uniform Approach to Handling of Feature-Objects in an Advanced CAD System", in *Proceedings of Advances in Design Automation, DE-Vol. 69-1*, ed. by Gilmore, B. J., Hoeltzel, D. A., Dutta, D., Eschenauer, H. A., ASME, New York, 1994, pp. 547-562.

- Kiriyama, T., Tomiyama, T., Yoshikawa, H., “Qualitative Reasoning and Conceptual Design with Physical Features”, in *Proceedings of the 4th International Workshop on Qualitative Physics*, Lugano, 1990, pp. 153-160.
- Olsen, G. R., Cutkosky, M., Tenenbaum, J. M., Gruber, T. R.: “Collaborative Engineering Based on Knowledge Sharing Agreements”, in *Proceedings of the 1994 ASME Database Symposium*, ASME, New York, 1994.
- Sasajima, M., Kitamura, Y., Ikeda, M., Mizoguchi, R.: “FBRL: A Function and Behavior Representation Language”, Web-site: <http://www.ei.sanken.osaka-u.ac.jp/pub/all-publications.html>
- Shah, J. J., Mäntylä, M., “Parametric and Feature-Based CAD/CAM”, Wiley and Sons, Inc., New York, 1995.
- Tomiyama T., Yoshikawa, H.: “Extended General Design Theory”, in *Design Theory for CAD*, ed. by Yoshikawa, H., Warman, E. A., Elsevier Science Publishers (North Holland), IFIP, 1987, 95 - 124.
- Tsang, E., “Foundations of Constraint Satisfaction”, Academic Press, San Diego, 1993.
- Umeda, Y., Tomiyama, T.: “Functional Reasoning in Design”, *IEEE Expert*, March/April 1997, pp. 42-48.
- Uschold, M., Gruninger, M.: “Ontologies: Principles, Methods and Applications”, *The Knowledge Engineering Review*, Vol. 11, No. 2, 1996, pp. 93-136.
- Soenen R., Olling G. J. (eds.): “Advanced CAD/CAM Systems, State of the Art and Future Trends in Feature Technology”, Chapman & Hall, New York, 1994.