1-1-2001

Languages for Engineering Design: Empirical Constructs for Representing Objects and Articulating Processes

Clive L. Dym
Harvey Mudd College

Philip Brey
Universiteit Twente

This is a preprint version of the following article:
Dym, C. and Brey, P. (2001). 'Languages for Engineering Design: Empirical Constructs for Representing Objects and Articulating Processes'. In: P. Kroes and A. Meijers (eds.), *The Empirical Turn in the Philosophy of Technology. Research in Philosophy and Technology 20*. Londen: Elsevier/JAI Press, 119-148.

Languages for Engineering Design: Empirical Constructs for Representing Objects and Articulating Processes

Abstract

Design knowledge incorporates knowledge and information about designed objects and their attributes, as well as about methods and means for undertaking the design process. Such design knowledge is articulated in several different representations or languages. This paper presents a typology of the languages of engineering design, emphasizing the representation of designed objects and the articulation and representation of the cognitive processes of design. Design languages include verbal or textual statements, drawings and graphics, formulas, and numbers. Still other design languages follow from computational styles. The languages of design and their computer-based implementations are empirical in origin, since observation reveals that these languages are derived not from an overarching theory, but from our experience in trying to understand what we do when we: talk about designed objects, articulate design processes, and teach computers how to do these things as well.

Next to presenting a typology of the languages of engineering design, and discussing the role of these languages in design activity, the paper also discusses the possibility of automating design activity through the design and manufacture of expert systems for product design. We will be looking at one of the most advanced systems of this sort, the PRIDE system, and use our study of PRIDE to discuss the possibilities and limits of automating design through the use of expert systems.

1. Introduction

This paper arises from a shared interest in the kinds of knowledge employed by engineers when doing design. We are interested in a better understanding of engineering design, and we believe that such an understanding should centrally involve an analysis of design knowledge and its the cognitive role in the design process. We hold that the act of designing can be analyzed as a cognitive process, and more specifically as a problem-solving process, in which designers go through a series of cognitive steps to solve design problems in order to arrive at a design that meets a sets of objectives while adhering to a set of constraints. An analysis of design as a cognitive process should, we claim, identify the various sorts of knowledge used in design (which we call *design knowledge*), and the cognitive processes and procedures by which such knowledge is employed by designers.

It is not our purpose in this paper, however, to offer a full-blown account of design knowledge or the cognitive structure of the design process. We focus instead on two more specific, interrelated issues that we believe can be helpful in elucidating both design knowledge and the cognitive structure of design processes. The first topic of our concern is the nature and the role of external representational media, or *languages*, as we will call them, in design activity. By external representational media (or representations) we mean representational forms (e.g. symbols, images) that can be found outside the human mind. They contrast with the inner mental representations that cognitive scientists have argued exist in the human mind. Engineers use various sorts of external representations in design activity, such as verbal descriptions, mathematical equations, sketches, and graphs. We want to arrive at a typology of these sundry representational forms, and assess their role in design.

The second issue concerns the possibility of automating design activity through the design and manufacture of expert systems for product design. More and more design activity relies on the use of computers, and there have been efforts to actually automate design processes through the use of so-called knowledge-based expert systems (KBESes) (Mittal, Dym and Morjaria, 1986; Dym, Henchey et al., 1988; Dym, Summers et al., 1995). We will be looking at one of the most advanced systems of this sort, the PRIDE system (Mittal, Dym

and Morjaria, 1986), and use our study of PRIDE to discuss the possibilities and limits of automating design through the use of expert systems.

We would like to emphasize the ways in which these two topics are related. For us, studying external representations in design and studying expert systems for design constitute different ways in which both design knowledge and the cognitive structure of design processes can be elucidated. Moreover, they are approaches that may benefit each other. An analysis of the role of external representations in design is likely to be helpful in an assessment of possibilities and limitations of automating design through the use of expert systems, because an expert system is itself an external representational medium that embodies computational representations of design knowledge. Thus, any general considerations on the role of external representations in design *ipso facto* also apply to the computational representations found in expert systems. In other words, understanding the role of external representations in design may help us better assess the possibilities and limitations of automating design.

Conversely, we believe that our analysis of expert sytems such as PRIDE can help us gain a better understanding of the role of external representations in design, because such systems are intended as models of actual design processes. We will try to assess how successful they are in this capacity, and what this implies for the role of exernal representations in design.

Before we embark on a discussion of these two topcs, we want to explain what moves us to study design knowledge and design processes, and what we take the status and relevance of our joint project to be. We write this paper as an engineer and a philosopher with a shared interest in engineering design. We both seek to arrive at better theoretical analyses of engineering design that describe how design processes are structured and explain how designers use their expertise to arrive at new designs. We both believe that an analysis of design knowledge and of cognitive processes and procedures in design is central to such an undertaking. We believe that the study of design requires an interdisciplinary effort. The result of our joint effort may be classified as a study in the newly emerged interdisciplinary field of design studies, to which engineering, the social sciences, and the humanities all contribute. However, we also each have our own disciplinary interests, and we would now like to describe some of these in order to explain how this research may be seen as contributing to our respective disciplines.

One of us, Dym, is an engineer, and has been motivated to study design because of a feeling, shared by many in the engineering community, that design

is a misunderstood activity that is not well represented in engineering education or in engineering research. He believes that a more rigorous and scientific study of design can contribute to better design education as well as better practice in design research (cf. Dym, 1994a). Dym, moverover, has a particular interest in artificial intelligence (AI) models of design, both as a vehicle for gaining greater understanding of the cognitive process of design, and for their practical use as design tools. Some of Dym's work is described in Sections 2 and 3 of this paper wherein we propose a typology of design representations and discuss the PRIDE expert system.

The other author, Brey, is a philosopher with a research emphasis in the philosophy of technology. He is interested in studying design because it has been a neglected topic in the philosophy of technology, which has traditionally focused on the consequences of technology while neglecting the engineering practice from which technology stems. He has been concerned with the relation between design and technological impacts (Brey, 1998a, b, 1999) and believes that an analytical understanding of design knowledge and design processes is required for the analysis and normative evaluation of the role of engineering design in the coming about of the societal consequences of technological innovations. Brey also believes, however, that a philosophy of design engineering constitutes a topic for philosophy in its own right that can do for engineering what the philosophy of science does for science: clarify and evaluate methods, analyze and set standards for progress, and explore ontological and epistemological issues within the discipline (Brey, 1996).

The outline of the remainder of the paper is as follows. In Section 2, a typology of representational formats used in design engineering is proposed and argued for, and a role in the design process is tentatively identified for each type of representation. This section paves the way for a discussion (in Section 3) of PRIDE, an example of a successful knowledge-based expert system used in design. Section 4 briefly discusses some practical implication of the previous sections. In Section 5 we address several issues that relate design knowledge and cognition to applications of artificial intelligence and their validity.

2. The languages of engineering design

Design engineering, as practiced by designers, is analyzed by us as a cognitive process, and more specifically, a problem-solving process. Our analysis of design as a cognitive process is influenced in part by the successful use of programming constructs derived from artificial intelligence (AI) to elucidate design knowledge

and formulate a theory of design (Dym and Levitt, 1991a; Dym 1994a). Designers, we claim, normally find themselves working to meet a number of goals or objectives while satisfying a number of constraints in the end product of their design activity, which may be a blueprint, a list of fabrication specifications, a prototype, or some other type of design plan. Their attempt to satisfy such constraints may be analysed as a problem-solving activity in which designers try to find design solutions that meet several objectives while satisfying multiple constraints. In this process they employ design knowledge the specialized knowledge that defines their expertise. Much, if not all of this knowledge is representational in nature: it represents artifacts (designed objects) and their attributes, as well as design processes and procedures.

In analyzing engineering design as a cognitive process, we do not mean to deny the role of noncognitive factors in design. Certainly, the inner mental processes of designers and their interaction with representational media constitute only some of the factors that determine design outcomes. We are aware, in particular, that designs are commonly explained in technology studies as largely, if not exclusively determined by social factors, including the interactions between different social groups, such as designers, corporations, government agencies, and users (Bijker, Pinch & Hughes, 1987; Bijker & Law, 1992). A complete explanation of individual designs should certainly include such factors. However, our purpose is not to explain design outcomes, but to assess the cognitive structure of design activity. We hold that sociological or economic analysis of technological innovation are well suited to explain why certain specifications came to prevail over others in a design. However, such analyses do not explain how it is that designers are able to turn a set of specifications into a design plan. We believe that this latter process can only be understood through the analysis of cognitive processes in design, even if a further explanation of these cognitive processes may again require reference to social factors.

A cognitive process is, in our analysis, not necessarily just a mental process. Certainly, design activity is made up in part by inner mental processes (e.g., thought, concept formation, and reasoning) that take place within the minds of designers. However, just as important as these mental processes is the systematic interaction of designers with various sorts of external representational media that aid them in their design activity. Designers write down their ideas on paper, using various kinds of symbolical languages, they make various kinds of graphical representations, they use computers to process information, and they

take in information by consulting various sorts of information sources. An account of the cognitive structure of mathematical modeling, for example, would probably be incomplete if it would only refer to mental processes in the head of designer and made no reference to his interactions with calculators, computers or other mathematical tools that directly affect the steps taken in modelling. A proper cognitive analysis of design should therefore combine accounts of the inner mental processes of designers with accounts of their interactions with these various representational media.

In this section, we will analyze the external representational media used in design. Our aim is to present a typology of the various types of external representations used in design and to describe their role in the design process. We will call these representational media the *languages of design* because they consist of interconnected systems of representations (symbols, icons, etc.) that function like languages in which design ideas are expressed and communicated. It will become clear that external representations used by designers do not just refer to objects and their attributes. They are also used to represent design processes and procedures. We believe that this is true because designers do not just try to prescribe what a designed artifact should look like, they also continually think of the functions served by the artifact and its attributes, and of subsequent steps in their design and manufacture. For example, there is evidence that designers think about design processes when they begin to they create sketches and drawings to represent the objects they are designing (Ullman, Wood and Craig, 1990).

Thus, a complete representation of designed *objects* and their attributes requires a complete representation of design *concepts* (e.g., design intentions, plans, behavior, and so on) that are more difficult to represent than are physical objects. Since the endpoint of engineering design thinking is most often a set of fabrication specifications for objects, it is important to discuss the *languages* or *representations* for both objects and processes (Dym, 1994a).

In order to describe an object, whether physical or conceptual, in detail or abstractly, there must exist a language within which that description can be written. There are several languages in which design information is cast (Dym, 1994a; Dym 1994b):

Verbal or **textual statements** are used to articulate design projects; describe objects; describe constraints or limitations, especially in design codes (see below); communicate between different members of design and

manufacturing teams; and document completed designs.

Graphical representations are used to provide pictorial descriptions of artifacts. These visual descriptions include sketches, renderings, and engineering drawings, are often interpreted within CADD systems.

Mathematical or **analytical models** are used to express some aspect of an artifact's function or behavior, and this behavior is in turn often derived from some *physical* principle(s). Thus, this language could be viewed as a **physical representation language** (cf. (Pahl and Beitz, 1984)).

Numbers are used to represent design information in several ways. Discrete values appear in design codes and constraints (and as attribute values, e.g., part dimensions). Numbers also appear as continuously varied parameters in design calculations or within algorithms wherein they may represent a mathematical model.

We also add to this list two text-based representations that derive from programming constructs used in symbolic computing and form core ideas of AI-based programming (Dym and Levitt, 1991a). These two representations or languages have highly stylized and specific syntaxes or grammars:

Rules prescribe specific *action* to be taken in a given *situation*. Normally written in sets of IF–THEN clauses, the left-hand (IF) sides of the rules define situation(s) that must obtain before a rule is applied, while the right-hand (THEN) sides of the rules define actions to be taken. Rules are used to represent design heuristics or rules of thumb and design codes.

Objects or **frames** are structures that can be used to represent *objects* and their *attributes*. Frames are elaborate data structures that can be linked to other frames through a variety of logic-based and/or procedural calls to other data or attributes or to calculation methods. Objects, the heart of *object-oriented programming*, are interesting because, together with their networks of links and procedures for storing and passing data, they can describe how physical and conceptual objects relate to each other.

We recognize that it is perhaps unusual to classify these two representation schemes—rooted as they are in symbolic computation—as design languages. However, *our experience has shown* that these particular constructs offer effective means for stating and applying design knowledge. We have also found that these structures can be used to organize and integrate design knowledge represented in one or more of the other design languages, thus laying the

groundwork for developing integrated computational environments for design (Dym, Garrett Rehak, 1992; Dym and Levitt, 1991b).

We could be extract other constructs from symbolic programming, e.g., logic programming, neural networks, and genetic algorithms. However, we do not identify them as design languages or representations for the following reasons. *Logic programming* is really an implementation of predicate and propositional calculus that has severe limitations in terms of its expressive power. We could also argue that rules, as an extension of logic programming, serve design representation purposes quite well. *Neural networks* attempt to imitate the learning and knowledge transmission characteristics of the human brain. They function by recognizing and classifying patterns that are in turn collections of discrete numbers representing different aspects or characteristics of a problem. This representation cannot represent more structured concepts, so neural networks are limited in their ability to mimic and simulate human thought and reasoning processes. *Genetic algorithms* are used to extract local and global optima in discrete optimization problems. Here, problems are represented by strings of attributes representing potential solutions to a problem, fitness functions used to evaluate potential solutions that reflect problem constraints and objectives, and a process by which generations of potential solutions are generated by allowing reproduction and mutation of fit solutions from previous generations. Genetic algorithm representations are limited to being variations of a given topology, so their potential for representing reasoning is limited.

Clearly, we employ different languages to represent design knowledge at different times. We often cast the same knowledge in different languages in order to serve different purposes, which in turn requires that we find translations between the different languages in order to represent different aspects of an artifact or different phases of the design process. In the domain of structural engineering, for example, a complete design of a structure requires the use of many kinds of knowledge expressed in different representations (viz., Table 1). In fact, we identify experts in part by their ability to select the right language at the right time to solve a given problem. We recognize that different languages may be employed, that different representations offer different insights and utility, and that it is desirable to link these different languages in order to seamlessly model a designed artifact and the design process.

as a *rule*:

IF a structural element has one dimension much thinner than the other two

AND it is loaded in that direction

THEN it will behave as a plate in bending

mathematically, the partial differential equation that governs the deflections of bent, thin plates is $D\nabla^4 w(x,y) = q(x,y)$

verbally and *numerically*, the deflection of a floor in a residential building should not exceed its length (in feet) divided by 360

Table 1. Alternate expressions of fundamental knowledge about plate structures.

2.1 Graphical representations

Graphical languages or representations include sketches, freehand drawings, plots and graphs, and CADD models extending from simple wire-frame drawings through elaborate solid models. Engineers have long engaged in putting "marks on paper" while doing design, and they add supporting text, lists, dimensions, and calculations to their sketches of objects and their associated functions, as well as related plots and graphs (Ullman, Wood and Craig, 1990). The support notes and other marks (and the drawings) facilitate a parallel display of information as they can be surrounded with adjacent notes, smaller pictures, formulas, and other pointers to ideas related to the object being drawn and designed. One illustrative example is a sketch made by a designer working on the packaging—to consist of a plastic envelope and the electrical contacts—to accept the batteries that provide the power for a computer clock (cf. Figure 1). Here the designer has jotted down some manufacturing notes adjacent to the drawing of the spring contact. And, while not evident here, the designer might have scribbled modeling notes (e.g., model the spring as a cantilever of stiffness?) or suggestions (e.g., calculate the spring stiffness as if it were a cantilever beam) or other design-related information.

Marginalia of all sorts are commonly experienced by anyone used to working in an engineering environment. Engineers doing both analysis and design often surround their pictures with text and equations. Conversely, sketches are often drawn in the margins of documents, perhaps to elaborate a verbal description,

perhaps to indicate more emphatically a coordinate system or sign convention. Thus, it should come as no surprise that sketches and drawings are as essential to engineering design as any other representation.

Experience also suggests several major issues in graphical representation, as evidenced in a listing of how drawings are used in the design process. Drawings provide a permanent record of the shape or geometry of a design; facilitate the communication of ideas among designers, and between designers and manufacturing specialists; support the analysis of an evolving design; simulate the behavior or performance of a design; and ensure that a design is complete, as a picture and its associated marginalia could serve as reminders of still-undone parts of that design. In fact, experience with graphical communication is so compelling that it has been suggested that drawing is the method preferred by mechanical engineering designers to represent external data (Ullman, Wood and Craig, 1990).

Graphical representations of all kinds—whether done by hand or on a computer, whether informally sketched or detailed in complex sets of blueprints, whether bare or annotated with marginalia—represent a great deal of design information. Clearly, there is much leverage to be gained by integrating these representations with other kinds of representations (Dym and Levitt, 1991b).

2.2 Feature-based descriptions

We know that designers work in terms of aggregated information and concepts, for example, shapes of surfaces and volumes, holes, fit, interference, tolerances, and so on—even when they are thinking about geometry and topology. Thus, we argue that a richer, more powerful representation is needed to enhance our ability to reason about objects being drawn and designed, even if the reasoning is about spatial issues. CADD (and originally CAD) systems have been used for some time to draw pictures and plans of objects as they are being designed, but designers clearly think about artifacts in ways that are much more encompassing than the points, lines, and surfaces used in CADD systems. Feature-based representation, as it is called, has emerged from just such attempts to enable reasoning about objects for which data is stored in CADD systems (Cunningham and Dixon, 1988; Finger and Dixon, 1989).

Features were originally thought of as volumes of solids that were to be removed, typically by a machining process, because of the interest in relating a part to some part of a process plan. More recently features have been taken to include devices such as gears, bearings, and shafts, in the contexts of (1) relating

an artifact's intended function to a *form feature* in the CADD representation and (2) representing the physical features of an artifact so that they can be evaluated. Examples of feature-based descriptions include: windows, corners, and tongues for injection molding; walls and fillets for extrusion; and walls and boxes for casting. Thus, the use of the term feature has broadened to include both form and function, and the context has been extended to manufacturing and life-cycle concerns of designed shapes and objects in which both geometric and behavioral issues figure prominently. This is largely due to the fact that, as noted above, designers tend to think in terms of particular forms that are intended to serve a function imposed by the designer.

The part shown in Figure 2 is a locator for a piece that is formed within a mold and is itself made through the process of injection molding (Dixon, Libardi and Nielsen, 1989). In order to keep the molded piece at a desired location, the shaded surfaces are required to remain parallel and at a specified distance from one another. The tongue thus serves to locate the piece within the die. Changes in any properties of the tongue must be consistent with the locating function. Thus, modifications in the tongue's geometry must be followed by corresponding changes in the rest of the part that do not inadvertently defeat the intended purpose. Similarly, the window over the tongue makes it easier to mold the piece being molded within a two-plate die whose parting direction is normal to the tongue, and it makes it easier to extract the part from the two-plate die. Any changes in the nature of the window must be such that they do not defeat the two purposes that the window serves in the present design.

We note that the design thinking just outlined is expressed in terms of the *tongue*, the *window*, and the purposes these features will serve. We see that this thought process is *not* about the coordinates that define the window opening or the precise location and orientation of the face at the end of the tongue. The thought process is done in terms of features and what features do in the final design. Experienced designers thus aggregate design information in terms of an artifact's features because they are readily available, perhaps even "intuitively obvious" choices as means to organize and apply design knowledge. Features are also a bridge between graphical representations and the object-oriented descriptions discussed in Section 2.4 because feature hierarchies are easily and naturally represented in object-oriented descriptions.

2.3 Rules

The form of a rule is well-known. However, in order to show how rules are used

to express design information, one brief rule example is taken from a KBES called DEEP that was built to assist designers in configuring electrical service for residential plats (Dym, Summers et al., 1995). (The design process in DEEP is described in Section 3.) The rule describes how separate building lots on a street should be clustered or grouped so that they can be efficiently configured for service. The rule combines heuristic design knowledge (i.e., a judgment about how close lots on a street are, one to another) with some basic electrical engineering knowledge (i.e., the transformer size needed to meet a specified service demand):

IF the number of lots on the street is very close or equal to the maximum number of services a 50 kVa transformer can serve
THEN cluster all of the lots on that street and serve them with a 50 kVa transformer.

2.4 Object-oriented descriptions

It is now appropriate to delineate object-oriented representation as a prime language of design. It clearly has evolved from the experience of both computer scientists and designers as a means of organizing or clustering information about a central core, while maintaining the ability to declare common attributes and departures from defaults, as well as to call for data or calculations that are related to the identity of the object who's central core is thus described.

The frame shown in Table 2, also taken from DEEP (Dym, Summers et al., 1995), illustrates how particular examples of a kind of object, called *instances*, can be described in relationship to a *class* of objects that share common attributes, in this case an instance of the class of transformers. This object description contains *slots* for particular attributes (the left-hand column), and the specific *values* for those attributes that distinguish the particular transformer in question, that is, XFR50_1 (in the right-hand column). The slot for the attribute DefaultType is *inherited* or passed down from the class Transformers, here as a PADMOUNT transformer, unless it is overridden at the local level either by a rule application or through intervention by the system user. In addition, the links (not shown here) can be used to establish and maintain various kinds of relationships, for example, a CONCRETE_PAD may be *part-of* a PADMOUNT transformer, whereas a BURD is a *kind-of* transformer.

Figure 3 shows the *inheritance lattice* or object tree of all the objects in the DEEP system. The figure shows how the class Transformers and its instances

SLOT/ATTRIBUTE	VALUE
SuperClass	Components
Class	Transformers
InstanceName	XFR50_1
Type	BURD
DefaultType	PADMOUNT
Rating (kVa)	50
Size (ft^2)	48
Structure	Concrete
MaxConnections	6
Coordinates	(X1, Y1)
MaxCustomers	21

Table 2. The instance XFR50_1 of the class Transformers of the DEEP system. (There is more complexity than is visible because the links between the instance and its classes are not shown.)

(e.g., XFR50_1) fit into the overall structure of all the objects involved in this particular design project. In fact, the classes Transformers and Handholes (that are, typically, small underground structures containing electrical connection and relay devices) are themselves subordinate to a more abstract class or "superclass," Structures, which is in turn part of the top-level class that contains all the objects in the DEEP knowledge base. Note the heterogeneity of this lattice; that is, it represents a mix of different kinds of physical structures and of more conceptual objects, such as the Configurations, that represent recommendations for the spatial layout of cables and other electrical equipment.

What may also be evident from our discussion—if not from the figure, is that the branches of the object tree are not independent of one another, even though they may appear to be in the pictorial representation of the tree. For example, the transformer instance XFR50_1 has associated with it a specific location (the slot Coordinates in Figure 1), and a set of lots that it serves (not shown in that figure, although the total number of lots served is given in the slot MaxCustomers). The location of this particular transformer and the lots it serves are identified in appropriate slots in objects elsewhere in the tree; that is, in the classes Configurations and Lots, respectively.

3 Modeling a design process

We now turn our discussion away from the languages of design to the modeling of design processes using knowledge-based expert systems (KBESs). A KBES is computer system that facilitates solving problems in a given field by drawing inferences from a knowledge base, which is represented in symbolical form, and which has been developed from human expertise. Work on KBESs is based on the assumption, held by many artificial intelligence researchers, that a computer, which is a universal symbol-manipulating device, has the capacity to generate intelligent action that matches that of experts in a given field (cf. Newell and Simon, 19xx). It is sometimes even assumed that KBESs can actually model or mimic human thought processes, because intelligence is essentially a matter of manipulating symbols according to formal rules, whether done by humans or by computers.

In this section we describe the design process encapsulated in PRIDE, a knowledge-based configuration design system for the mechanical design of paper handling systems in copier machines (Mittal, Dym and Morjaria, 1986). PRIDE is a particularly successful design system that was designed under the assumption that human design intelligence could indeed be modeled on a computer. PRIDE was also a pioneering demonstration of an empirical, AI-based approach to capturing the experiential knowledge that is applied to the performance of a design task, modeling real design processes, and furthering the understanding and vocabulary of design as a discipline (Dym and Levitt, 1991a; Dym 1994a; Mittal and Araya, 1986; Morjaria, 1989). It is even now a "real" system, used daily as a design tool by designers doing feasibility studies for new copiers. It was not just an academic exercise, but a convincing realization of the kinds of representation and reasoning described herein. Our present discussion focuses on a few of the key points of modeling the design process.

If we looked inside a copying machine, we would see paper moving rapidly along a very complicated paper path, past various components and physical processing elements, and under rather stringent constraints. While there are several kinds of paper-handling systems, the PRIDE system focuses on transport systems that use pinch rolls to "grab" and move pieces of paper. The design requirements are tremendous and include geometrical properties (e.g., paper entrance and exit locations and angles (see Figure 4)), timing constraints, allowable skew with respect to the path, tolerances on engineering parameters, and the ability to adapt to a variety of paper properties, including size, weight, stiffness, and curl. Designers are required to make qualitative and quantitative judgements while reasoning geometrically, algorithmically, and heuristically

about both physical objects and more abstract concepts.

We can decompose the design of a paper transport system into subproblems, including designing a smooth path between the input and output locations, deciding the number and location of pinch-roll stations to be placed along this path, designing a "baffle" to be placed around the paper path to guide the paper, designing the sizes of various pinch rolls (drivers and idlers), selecting the proper materials for the pinch rolls and baffle, making decisions about paper travel speeds and the forces on the paper produced by the pinch rolls, calculating the time needed to move the various sizes of paper, and calculating the various performance parameters and ensuring that they satisfy the requirements. However, even with a very effective decomposition, the range of design tasks we face is very diverse as it involves making decisions about geometry, spatial layout, timing, forces, jam clearance, and so on, the totality of which are often beyond the scope of a single engineer. The same transport has to be able to handle different sizes and weights of paper, which often presents conflicting constraints. For example, if the lengths (or widths) of the different sizes of paper are far apart, then the constraint on the maximum separation of neighboring roll stations for the smallest paper conflicts with the constraint on not having more than two stations guiding the paper for the longer papers. The design of the paper path is further complicated by obstructions that have to be avoided, as well as adherence to strict requirements on the smoothness, continuity, and manufacturability of the baffle in which the paper travels.

Paper-transport design is clearly a complex task that requires: *spatial reasoning* about the geometry and routing of a paper path, and the placement of particular components; *component selection* of the particular components or subsystems (from a large array) needed to perform specified functions; and *component configuration* of various subsystems in just the right way to obtain some specified behavior. Given this complexity, what does the PRIDE system do to be so helpful, either as a stand-alone design system or as a designer's assistant? In brief, PRIDE:

1. facilitates the designer's choice of a planar path that avoids obstructions caused by equipment items within the copier and lies between specified input and exit points (see Figure 4);

2. automatically checks that all constraints on the path geometry (e.g., smoothness, minimum radii of curvature, etc.) are satisfied;

3. identifies a physical device (the baffle) in which the paper will be carried along the path; and

4. identifies, designs, and locates along the path the pinch-roll pairs that grasp and move the paper along the chosen path.

In fact, PRIDE simulates rather closely the actual design process that experienced copier system designers have used for years. One of the results is that feasibility studies for preliminary copier designs are now completed and evaluated, with PRIDE's assistance, in hours rather than weeks.

PRIDE uses several representation schemes to incorporate heuristic, relational, and algorithmic aspects of the design problem, as well as several inference schemes at different levels of abstraction. PRIDE also has a powerful graphics interface that facilitates a rather complete simulation of the way human designers actually design paper-handling subsystems for copiers. Figure 5 shows a small part of an inheritance lattice that describes a paper-transport system designated as Trans5. In this object-oriented representation (cf. Section 2.4), Trans5 is an object with several attributes, some of which are linked to other objects, some of which are physical (e.g., Roll2, Driver1), and some of which are conceptual (e.g., Spec1). It can be seen that Trans5 has components that are connected to it by one or more *SubPart* links (e.g., Driver1). In order to to determine which specifications govern the input point, the *InputSpecs* link is exercised to define the attributes of the design constraints at the point where the paper enters this subsystem (e.g., Specs1).

Thus, a designer can reason about the device being designed (Trans5) because questions can be asked of the system, for example, What are the output specifications that govern this design? and, How many and what roll stations are there in Trans5? And as is seen in Figure 5, the answers to these two questions are, respectively, Specs2 and 4: Roll1, Roll2 . . . . With this representation for devices and parts in place, attention can be turned to the design process in PRIDE.

The knowledge base in PRIDE represents a design plan structured as a top-down process of identifying and satisfying design goals and subgoals (Figure 6). The design plan decomposes design goals into simpler steps. Figure 6 shows a top-level goal of designing the paper transport, as well as goals for subproblems such as deciding the number of roll stations and deciding the diameter of the driver at station 4. This design plan will work only if PRIDE provides the

knowledge needed to order the steps, to perform each step, to detect failures in the design requirements, and to suggest fixes for the failures.

The design process in PRIDE can be thought of as one in which a very complex system is being configured. The geometry of a path along which the paper will be transported must be established first (again, Figure 4). After the path is established, the designers must choose components and establish their configuration (e.g., size, location, materials, etc.) so that different kinds of paper can be smoothly moved along the path, without jamming. The parts are chosen from a database of parts in normal use. Thus, in terms of standard design terminology [5], the process includes elements of both preliminary and detailed design. The process can also be characterized as *routine design* because the designers know how to decompose the design (cf. Figure 6), how to design the subsystems and components, and what to do when a constraint is violated.

PRIDE's problem-solving strategy may be thought of as *generate–test–analyze–advise–modify*. It also makes very effective use of *decomposition* (as is evident in Figures 4 and 5) and, although the details of this are not elaborated here, *constraint satisfaction*. A few elements of the methods by which designs are generated, and failures are analyzed and fixed, are now described.

A design goal in PRIDE is responsible for designing (and sometimes redesigning) a small set of design parameters that describe some part or aspect of the artifact being designed. Some of the design parameters in this domain are paper path segments, paper path length, number of roll stations, diameter, width, and material of each pinch roll, baffle gap, baffle material, and time taken by each size of paper during transport. We show in table 3 a simplified representation of the goal "Decide number and location of roll stations." The variables Descriptor and Name are used to describe the goal to the (human) users of PRIDE. DesignMethods is an ordered list of all the alternate methods for achieving the goal. In this example there is only one method for carrying out this goal, that is, four subgoals be must achieved. Constraints contains the verification knowledge about the acceptability of a design.

Where does the "generate" come into the picture? In fact, one of the values of the slot DesignMethods could be a design generator, and the approach to generating designs could itself vary. It should be noted that, from the point of view of capturing a lot of design alternatives, the design generators are among the most powerful design methods. These methods are all capable of generating different values for the same (or a small set of related) design parameter(s). For example, heuristic knowledge for making "good" guesses about initial values to

ATTRIBUTE	VALUE
Type	SimpleGoal
Name	Goal5
Descriptor	"Decide number and location of roll stations"
Status	INIT
AnteGoals	"Design Paper Path"
InputPara	"Paper Path," "length of PaperPath"
OutputPara	"Number of RollStations," "location of AllRolls"
DesignMethods	(SubGoals Goal51: "Decide min number of rollStns" Goal52: "Decide Abstract Placing" Goal53: "Generate Concrete Location" Goal54: "Build RollStn Structure")
Constraints	(Constr8: "First Stn <= 100 mm." Constr17: "Dist. between adj. stn <= 160mm." Constr24: "Dist. between adj. stn >= 50mm.")

Table 3. The goal "Decide number and location of roll stations" from the PRIDE system.

be generated could be attached to these methods. The generators in PRIDE also specify the ranges of possible values and increments. In Table 4 we show a design generator for the goal "Design driver diameter" that generates diameters for the drivers (in a pinch-roll pair) from a known database of acceptable driver diameters.

The particular type of generator shown in Table 4 belongs to the class InstanceSetGenerator because the database is composed of instances of different classes of objects. The generated objects are instances of DriverDiameter. It can also be seen that this method specifies that 10 mm is a good starting value for the diameter, probably because the experts have found this to be a good default choice. Finally, it specifies that this instance object becomes the value of the design parameter driver diameter.

If the current design runs into trouble, if some requirement is not satisfied, the PRIDE problem solver analyzes the current partial design and tries to come up with suggestions to overcome any violations. These modifications may be heuristics reflecting a designer's experience in fixing similar problems, or they may be based on a more general problem-solving approach which analyzes

ATTRIBUTE	VALUE
type	InstanceSetGenerator
name	SetGen1
descriptor	"Generate standard driver diameters"
assignTo	(DesignObject defRollPair driver diameter)
initValue	"Find a diameter of 10mm"
classes	DriverDiameter
soFar	NIL
status	INIT

Table 4. The generator "Design driver diameter" from the PRIDE system.

dependencies between different parts of a design to suggest modifications that go beyond knowledge directly represented in its knowledge base. Figure 7 shows how advice is provided in PRIDE. In this example, the design goal "Decide number of roll stations" calculates a number of roll stations which produces a violation of the constraint on the maximum separation between roll stations. Advice—in this case based on a built-in heuristic—is provided to say that the number of roll stations should be larger than the number calculated.

PRIDE has many features beyond those discussed here, among which is the capacity to maintain multiple designs simultaneously and to switch between different partial designs so that designers can explore different options in parallel. A designer can also selectively undo a design or impose additional constraints. In fact, PRIDE's many features make it very useful as a designer's assistant, because designers working with PRIDE often develop suitable designs faster than either the system or the designer would have done alone.

Our snapshot of the design process in PRIDE illustrates how a complicated configuration task can be described and analyzed with the aid of symbolic representation and concomitant problem solving. Further, the stylized symbolic descriptions we present contrast sharply with the numerical representations used in procedural programs. This clearly opens the door to detailed and structured discussions of design, both at the blackboard and on the workstation screen, because both a vocabulary and a structure for talking about the design process can clearly be realized.

4. Some application-oriented thoughts

We have reviewed the several languages of design, including verbal and textual statements, graphical representations and images, mathematical and analytical models, numbers, rules, and structured descriptions of objects. The last two languages in particular, derived from current computational paradigms, offer empirically-based support for representing and applying various kinds of design knowledge. Inasmuch as these kinds of languages also offer a dual promise of integrated design environments and of capturing designers' intentions, we feel that a brief discussion of these two issues is warranted.

There has long been a gap between the designers of artifacts and their makers. Whether a natural evolution of the engineering profession or a reflection of an economic model of specialized labor as a driving force in capitalism (although we see it in societies built on vastly different economic premises), this gap is nonetheless a source of continuing concern. One of its most common expressions is the desire to tear down the legendary "brick wall" that is said to exist between designers and manufacturing engineers.

There are many interesting issues about design communication. One is related to delineating various aspects of integrating graphical and other representations (e.g., object or device representations, analysis programs, codes, documentation, and so on) within an integrated computational design environment. A second issue is concerned with the concurrent use of design software by many users in large design and manufacturing organizations. For example, in addition to the technical issue of integrating different representations, we can easily identify the need to handle different kinds of information that result from the social and gepgraphical distribution of cognition.

We must also confront some other issues, for example, how do we maintain control of an evolving design, understand how and why design decisions are made, and understand how these decisions are propagated and enforced? Wanting to understand *why* a design decision was made prompts a question: What advantage can we gain by articulating, preserving, and communicating the intentions that designers have for their design?

Designs are realizations of their designer's intentions (Dym, 1994a). However, design intentions are often subtle in their expression, or are masked by the complexity of the designed artifact. The Kansas City Hyatt Regency failure was due to the fabricator changing the structural connections for the second-floor atrium walkway because he couldn't hang it as originally designed (Pfrang, 1982; Petroski, 1982). The fabricator found that the long hanger rods were

unavailable, and he didn't know that the designer intended to hang the second-floor walkway directly from the roof truss, not from the fourth-floor walkway. Had the designer been able to convey his intention automatically and unambiguously, without waiting (in vain) for the question to be asked, this tragedy might well have been avoided. Thus, one wonders whether the representations described above could help convey a designer's intentions to whomever makes the final design.

Capturing the designer's intent by conjuring up a description of both the artifact and its intent is rather easy. Getting this information into the hands of the fabricators is much harder. This view of capturing design intent leaves the problem of transmitting design intent at the same abstract level as the discussion of integration. Further, it ignores some recent attempts to use design intent as a point of departure for reasoning about design. Now, the design process can be viewed as one of refining abstract goals and objectives until a fabrication specification emerges at the end of the process (Dym, 1994a). In this context, *design intent* can be viewed as a recorded history of the design process in which the *reasons* that design decisions were made are tracked as they are implemented. That is, were design intent captured systematically, design decisions could be reviewed and the consequences of revising them or undoing them altogether would be better understood. Design intent becomes a reasoning tool to help guide verification, modification, or reuse of designs, rather than simply a retrospective glance at the process.

A representation of function and its links to form and its representations is also a lively area of debate. Does the specification of function dictate a specification of form? Can one infer function from form? There is thus a need to interpret, refine, and represent at different levels of abstraction the intentions that are embodied in a design. The question is, Are issues of intention for and function of an artifact inextricably linked to the issue of representing the artifact? That is, can function be thought of independently of the representation of the device that performs that function? In fact, the examples indicate that the details of the representation of the artifact can be chosen apart from the representation of that function, subject only to the proviso that in the final analysis the designers have to produce a set of complete and unambiguous specifications for the manufacture of that artifact, which means that the representation must be acceptable to the manufacturer.

5. Cognition, design, and artificial intelligence

So far, we have presented a typology of languages of design, together with a brief description of their role in design processes, and we have discussed PRIDE, a KBES that is intended to model a design process. Now we address some hard questions about these two topics that we have previously postponed. First, we have postponed the question of how the languages of design function in the cognitive process of design, and specifically, how they relate to the inner mental processes of designers. Second, we have postponed the question of whether PRIDE provides a realistic model of the cognitive process of design, and, more generally, of identifying the possibilities and limitations of KBESs as tools for design and design modeling.

As for the first question, we have managed thus far to resist equating cognitive processes to (inner) mental processes. Cognitive processes may take place in the mind, in which case they are mental processes, but they may also take place partially outside of the mind, as when they involve external representations and manipulations performed with those representations. For example, when someone uses pen and paper to calculate a sum, a full account of the cognitive process by which the sum is calculated should not merely describe the mental processes involved. Rather, it should also analyze the interactions that take place with this medium. After all, this medium functions as a place for inscription, storage and retrieval of information, just like human memory does. Likewise, an analysis of design activity as a cognitive process should analyze both the mental processes of designers *and* their interactions with various representational media.

At one level, we want to claim, languages of design function just like any ordinary language like English or French: they are symbolical (or iconic) forms that are decoded by designers in their effort to construct inner mental representations that are used in the cognitive process of design activity. Often, the result of this cognitive activity is one or more new external representations (e.g., a blueprint, or mathematical model). For instance, a series of textual statements that express design constraints may be read by a designer who uses them to construct a mental image of an object that satisfies these constraint, which in turn causes him to draw a graphical representation of such an object. These cognitive activities evidently require special cognitive abilities. A professinal designer has interpretive abilities that enable him or her to read and understand languages of design (i.e., construct inner mental representations that denote the meaning of the external symbols or images under consideration), and cognitive abilities and background knowledge that enable him or her to translate

these mental representations into new representations that move one closer to a complete design solution.

Cognitive scientist Donald Norman (1991, 1993a) has suggested that we should think of external representations as *cognitive artifacts*: tools that serve to aid and abet cognitive activity. External representations like texts and images help us in the acquisition, storage and retrieval of information. To be good cognitive tools, Norman suggests, external representations should adhere to three criteria: They (1) "capture the important critical features of the represented world while ignoring irrelevant ones," (2) "are appropriate for the person, enhancing the process of interpretation," and (3) "are appropriate for the task, enhancing the ability to make judgments, to discover relevant regularities and structures." (1993a: 52). These, then, are three outstanding criteria for the evaluating the use of external design representations in the design process.

We hence propose that (external) design representations should be understood as cognitive artifacts used by designers as information retrieval and storage vehicles in the cognitive process of design, for the generation of mental representations and (ultimately) novel external representations. What is needed at this point is a more detailed account of the role of external representations in cognition. This is still an ongoing research topic for cognitive science (see Norman, 1993b; Winograd and Flores, 1986; Clark, 1997; Suchman 1987). For them to be *good* cognitive artifacts, they must be in a representational format that enables designers to encode only relevant features of a design, and that can be decoded or 'read' well by the intended user of the representation, presenting him or her with information that is, both in its form and in its content, immediately relevant to his or her goals. Evidently, the quality of design and design communication suffers when not all three of Norman's criteria are met.

The other issue to be considered in this section is the status of KBESs as tools for design and design modeling. We are both optimistic that KBESs can play an increasingly important role in design. We are, however, divided over the question of whether they can also be understood as adequate tools for design modeling: Dym holds that they can provide good models of design processes, whereas Brey believes that this is not the case. We agree that whether KBESs can function as adequate models of the design process depends on whether they are able to adequately model the cognitive processes involved in design. This implies that there must be a structural similarity between the structure and operations of KBESs and the cognitive processes of human designers.

It is widely agreed that KBESs, like other programs run on computers, are

physical symbol systems. That is, they are systems in which physically realized symbols (e.g., zeros and ones, realized as the presence or absence of an electrical current) are able to combine into symbol structures (structures composed of multiple symbols related in some physical way) and are operated on through processes of creation, modification, reproduction, and destruction. In this way, evolving collections of symbol structures are produced over time. If KBESs are to function as models or theories of design processes, it must be because the symbol structures of KBESs, and operations performed on them, mimic the representations and cognitive operations used in design by human designers. This, in turn, implies that human designers (perhaps along with the external representations used by them) are (at least by approximation) also physical symbol systems. That is, the cognitive activity of human designers must be analyzable as a process or creating, modifying, reproducing and destroying symbol structures.

In fact, one influential account of human cognition that has been proposed in cognitive science makes exactly this assumption. This approach to cognitive science, that dates back to the very beginning of the field, features the *physical symbol systems hypothesis* proposed by Newell and Simon (1963, 1972): "A physical symbol-system has the necessary and sufficient means for general intelligent action." According to this hypothesis, intelligence is not only something that can be exhibited by a symbol-processing system like a modern digital computer. Intelligence is also essentially a matter of symbol processing according to formal rules, and therefore the human mind is *also* a physical symbol system. Mental processes are then rule-governed operations defined over symbols or strings of symbols. This model is sometimes called *cognitivism*, and when used in AI it is also called *symbolic AI*.

The usefulness of KBESs as models of the design process hence depends on the plausibility of the central assumption of cognitivist cognitive science, i.e., the physical symbol system hypothesis. We do not here want to repeat all the arguments that have been proposed for and against cognitivism (e.g., Newell and Simon, 1963, 1972; Fodor and Pylyshyn, 1988; Haugeland, 1981; Dreyfus, 1992). We merely want to point out the fact that the usefulness of KBESs as models of the design process centrally depends on the truth of the physical symbol systems hypothesis, and point out that this hypothesis is still hotly debated.

We will now turn to the more practical question of how useful KBESs can be expected to be in taking over design processes. Evidently, if cognitivism is true,

then one may expect future KBESs that adequately model design as it is done by humans to automate an ever increasing part of design, with no principal limit to what can be automated. If cognitivism is false, it does not follow immediately that computers cannot be as good at design as human designers. Remember that chess programs are able to display a significant level of expertise, while they are widely believed to process information very differently than do chess grandmasters. So perhaps KBESs can be designed that follow cognitive procedures very different from human designers, but that are still successful at design.

The question is then *how successful* they may be. In answering this question, it useful to realize that design activities encompass a spectrum that ranges from *routine* design of familiar parts and devices, through *variant* design that requires some modification in form and/or function, to truly *creative* design of new artifacts. We hold that existing KBESs, such as PRIDE, have excelled at (very complicated) routine design, and that there is evidence that they can replicate human expertise in this regard. This implies that for routine design, KBESs may not just be used as *intelligent assistants* (e.g., PRIDE), but also as automated designers (e.g., DEC's R1 system, that routinely configures computer systems on the factory floor (McDermott, 1982),a task previously performed by human sales engineers).

KBESs successful at variant and creative design have yet to be constructed. To the extent that it can guide the construction of KBESs, psychological research on expertise suggests that this would require a new generation of KBESs that are better capable of transfering knowledge to new knowledge domains and to integrate knowledge so as to create new cognitive procedures. In a review of psychological theories of expertise, cognitive scientist Keith Holyoak (1991) argues for a distinction between *routine* and *adaptive* expertise. Routine expertise involves the ability to quickly and accurately solve problems in a limited problem domain. Experts who have routine expertise have developed increasingly specialized, skilled knowledge to solve problems in local problem domains. Adaptive expertise, however, includes the ability to transfer learned skills to new domains, and to integrate different skills so as to create new procedures out of routine expert knowledge (cf. Dörner & Schölkopf, 1991; Hatano & Inagaki 1986). Genuine experts, and hence also, perhaps, expert systems capable of adaptive and creative design, are able to adapt their skills so as to go beyond routine expertise.

While we disagree on the usefulness of KBESs as models of design, we agree

that it cannot be prejudged whether KBESs are capable of variant and creative design. The boundary between routine, variant and creative design, and between routine and adaptive expertise, is a moving one, that is open to negotiation. The possibilities and limitations of KBESs for variant and creative design will reveal themselves in future generations of KBESs.

6. References

Brey, P. (1996). "Philosophy of Technology: A Time for Maturation," *Metascience: An International Review Journal for the History, Philosophy and Social Studies of Science*, 9: 91-104.

Brey, P. (1998a), "New Media and the Quality of Life," *Society for Philosophy and Technology Quarterly* 3, 1-23.

Brey, P. (1998b), "The Politics of Computer Systems and the Ethics of Design," In M. J. van den Hoven (Ed.), *Computer Ethics: Philosophical Enquiry. ACM/SIGCAS Conference*, Rotterdam University Press, Rotterdam, The Netherlands, p. 64-75.

Brey, P. (1999). "Design and the Social Ontology of Virtual Worlds," In A. Nijholt, O. Donk and B. van Dijk (Eds.), *Interactions in Virtual Worlds. Proceedings of the XVth Twente Workshop on Language Technology*, 5-12.

Bijker, W., and Law, J. (Eds.) (1992). *Shaping Technology/Building Society: Studies in Sociotechnical Change*. Cambridge/London: MIT Press.

Bijker, W., Pinch, T., and Hughes, T., (Eds.) (1987), *The Social Construction of Technological Systems: New Directions in the Sociology and History of Technology*. Cambridge, MA: MIT Press.

Clark, A. (1997). *Being There: Putting Brain, Body and World Together Again,* MIT Press, Cambridge, MA.

Cunningham, J. J. and Dixon, J. R. (1988), "Designing with Features: The Origin of Features," In *Proceedings of the ASME Computers in Engineering Conference*, ASME, San Francisco, CA.

Dörner, D., and Schölkopf, J. (1991). "Controlling Complex Systems: Or, Expertise as 'Grandmother's Know-How'," In K. Ericsson and J. Smith (Eds.), *Towards a General Theory of Expertise: Prospects and Limits,* Cambridge, Cambridge University Press.

Dreyfus, H. (1992), *What Computers* Still *Can't Do: A Critique of Artificial Reason*, MIT Press, Cambridge, Massachusetts.

Dixon, J. R., Libardi, E. C. Jr. and Nielsen, E. H. (1989), "Unresolved Research Issues in Development of Design-With-Features Systems," in Wozny, M. J., Turner, J. and Preiss, K. (Eds.), *Proceedings of the 1989 IFIP WG 5.2 Second Workshop on Geometric Modelling*, North-Holland, Amsterdam.

Dym, C. L., Summers, M. D., Demel, C. T. and Wong, C. S. (995), "DEEP: A Knowledge-Based (Expert) System for Electric Plat Design," *Computing Systems in Engineering*, 6 (6): .

Dym, C. L. (1994a), *Engineering Design: A Synthesis of Views*, Cambridge University Press, New York.

Dym, C. L.(1994b), "Representing Designed Artifacts: The Languages of Engineering Design," *Archives of Computational Methods in Engineering*, 1 (1): .

Dym, C. L., Garrett, J. H. Jr. and Rehak D. R. (1992), "Articulating and Integrating Design Knowledge," in *Workshop on Preliminary Stages of Engineering Analysis and Modeling*, Second International Conference on Artificial Intelligence in Design, Pittsburgh, PA, June.

Dym, C. L., Henchey, R. P., Delis, E. A. and Gonick, S. (1988), "Representation and Control Issues in Automated Architectural Code Checking," *Computer-Aided Design*, 20 (3): . ..

Dym, C. L. and Levitt, R. E. (1991a), *Knowledge-Based Systems in Engineering*, McGraw-Hill, New York.

Dym, C. L. and Levitt, R. E. (1991b), "Toward an Integrated Environment for Engineering Modeling and Computation," *Engineering with Computers*, 7 (4): .

Finger, S., and Dixon, J. R. (1989), "A Review of Research in Mechanical Engineering Design. Part II: Representations, Analysis, and Design for the Life Cycle," *Research in Engineering Design*, 1: . ..

Fodor, J., and Pylyshyn, z. (1988). "Connectionism and Cognitive Architecture: A Critical Analysis." *Cognition* 28: 3-71.

Hatano, G., and Inagaki, K. (1986). "Two Courses of Expertise," in H. Stevenson et al. (Eds.), *Child Development and Education in Japan*, Freeman.

Haugeland, J. (1981). "The Nature and Plausibility of Cognitivism," in J. Haugeland (Ed.), *Mind Design*, MIT Press, Cambridge, MA.

Holyoak, K. (1991). "Symbolic Connectionism: Toward Third Generation Theories of Expertise," In K. Ericsson and J. Smith (Eds.), *Towards a General Theory of Expertise: Prospects and Limits,* Cambridge, Cambridge

University Press.

McDermott, J. (1982), "R1: A Rule-Based Configuration of Computer Systems," *Artificial Intelligence*, 19 (1): …..

Mittal, S. and Araya, A. (1986), "A Knowledge-Based Framework for Design," in *Proceedings of AAAI-86*, AAAI, Philadelphia, PA.

Mittal, S., Dym, C. L. and Morjaria, M. (1986), "PRIDE: An Expert System for the Design of Paper Handling Systems," *IEEE Computer*, 19 (7):

Morjaria, M. (1989), "Knowledge-Based Systems for Engineering Design," in *AUTOFACT '89 Conference Proceedings*, Detroit, MI.

Newell, A. and Simon, H. A. (1963), "GPS: A Program that Simulates Human Thought," in Feigenbaum, E. A. and Feldman, J. (Eds.), *Computers and Thought*, McGraw-Hill, New York.

Newell, A. and Simon, H. A. (1972), *Human Problem Solving*, Prentice-Hall, Englewood Cliffs, NJ.

Norman, D. (1991). "Cognitive Artifacts," In J. M. Carroll (Ed.), *Designing Interaction: Psychology at the Human-Computer Interface* (pp. 17-38), Cambridge University Press, New York.

Norman, D. (1993a). *Things that Make Us Smart: Defending Human Attributes in the Age of the Machine*, Addison-Wesley, Reading, MA.

Norman, D. (Ed.) (1993b). *Special Issue on Situated Action. Cognitive Science*, 17, 1.

Pahl, G. and Beitz, W. (1984), *Engineering Design*, Design Council Books, London.

Petroski, H. (1982), *To Engineer Is Human*, St. Martin's Press, New York.

Pfrang, E. O. (1982), "Collapse of the Kansas City Hyatt Regency Walkways," *Civil Engineering*, 52 (7): …

Simon, H. A. (1990), Personal Communication, 29 October.

Suchman, L. (1987). Plans and Situated Actions: The Problem of Human-Machine Communication, Cambridge, Cambridge University Press.

Ullman, D. G., Wood, S. and Craig, D. (1990), "The Importance of Drawing in the Mechanical Design Process," *Computers and Graphics*, 14 (2): …

Winograd, T., and Flores, F. (1986). *Understanding Computers and Cognition: A New Foundation for Design*, Ablex, Norwood, NJ.

Figure 1. Design information adjacent to a sketch of the designed object (after [7]).

Figure 2. An injection-molded part and some of its features (after [14]).

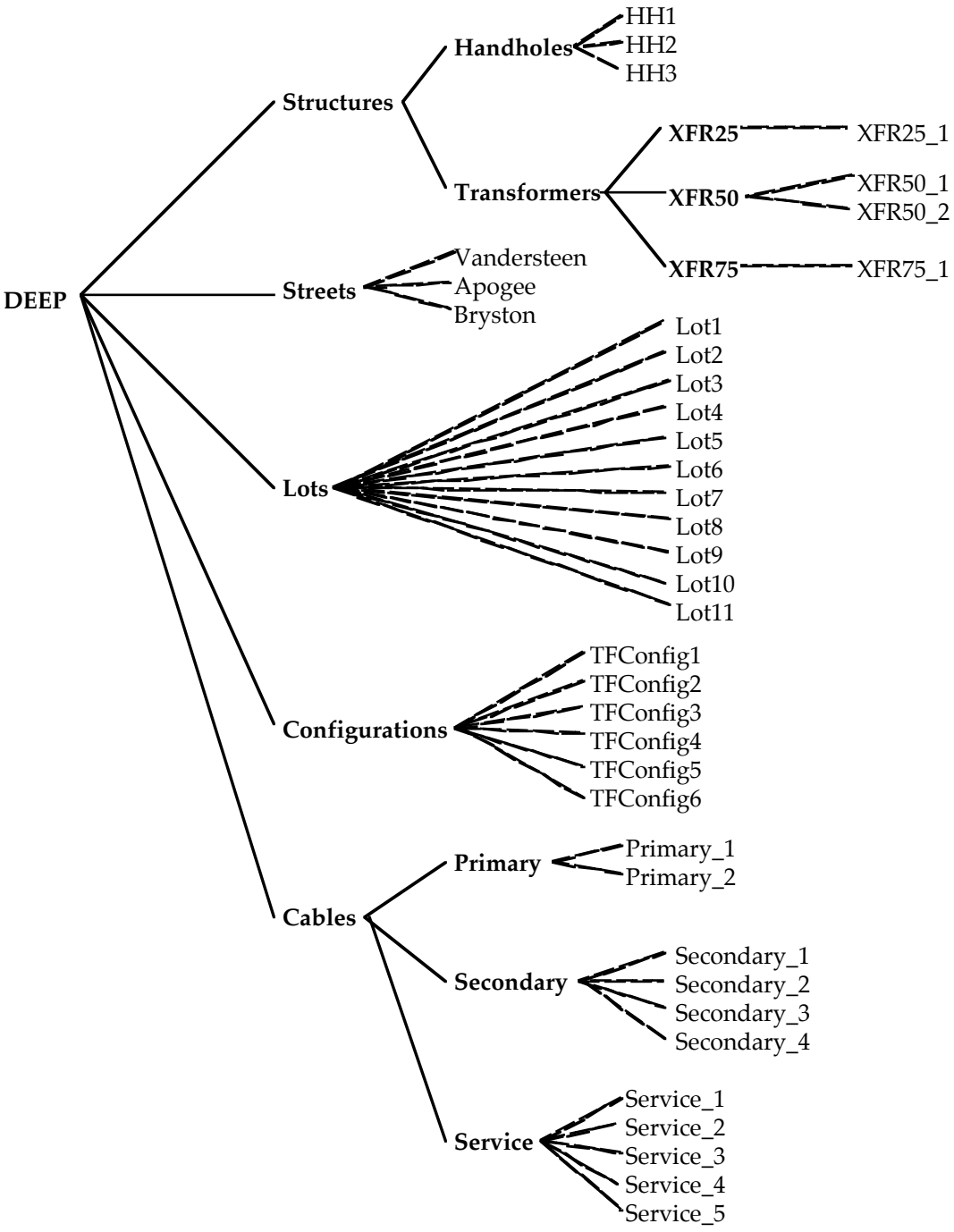

Figure 3. The inheritance lattice (or object tree) in the DEEP KBES (Dym, Summers et al., 1995).

Figure 4. A snapshot of a sample paper path in PRIDE, including roll stations, input and output points, and obstructing regions to be avoided by the transport (after [1]).

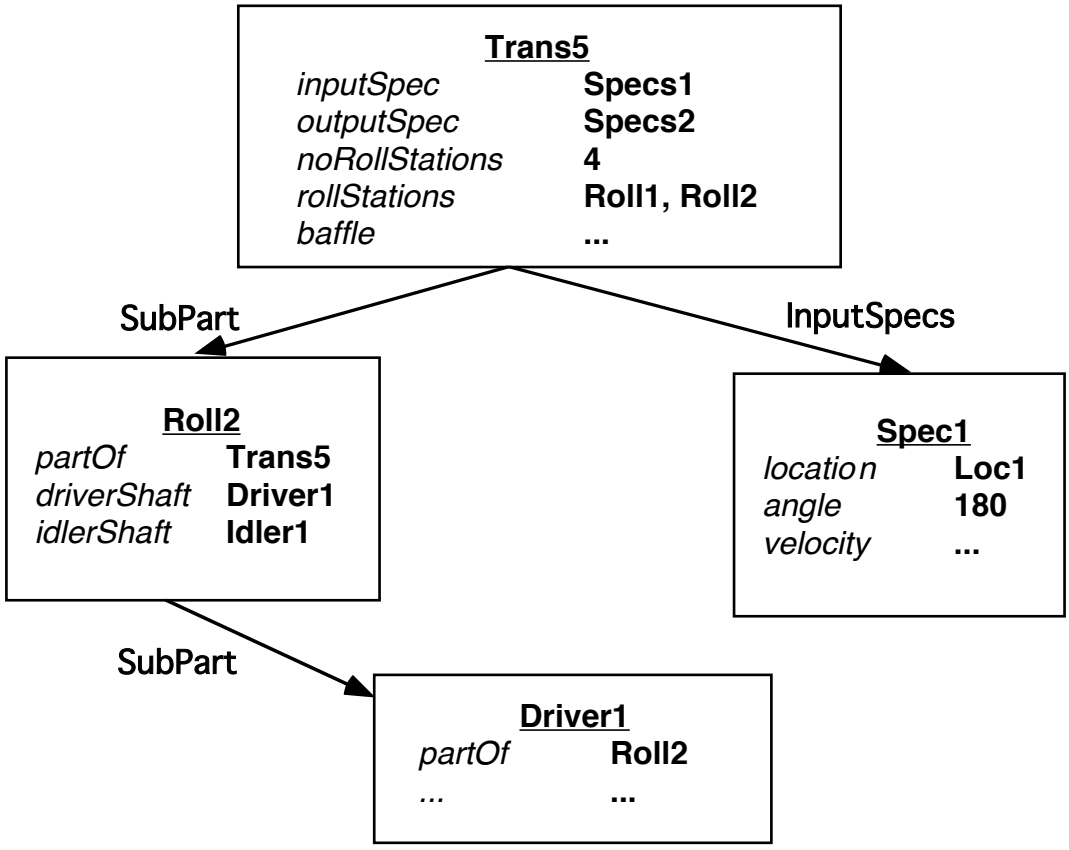

Figure 5. A stylized version of a small portion of the inheritance lattice that makes up PRIDE's knowledge base (after Morjaria, 1989).

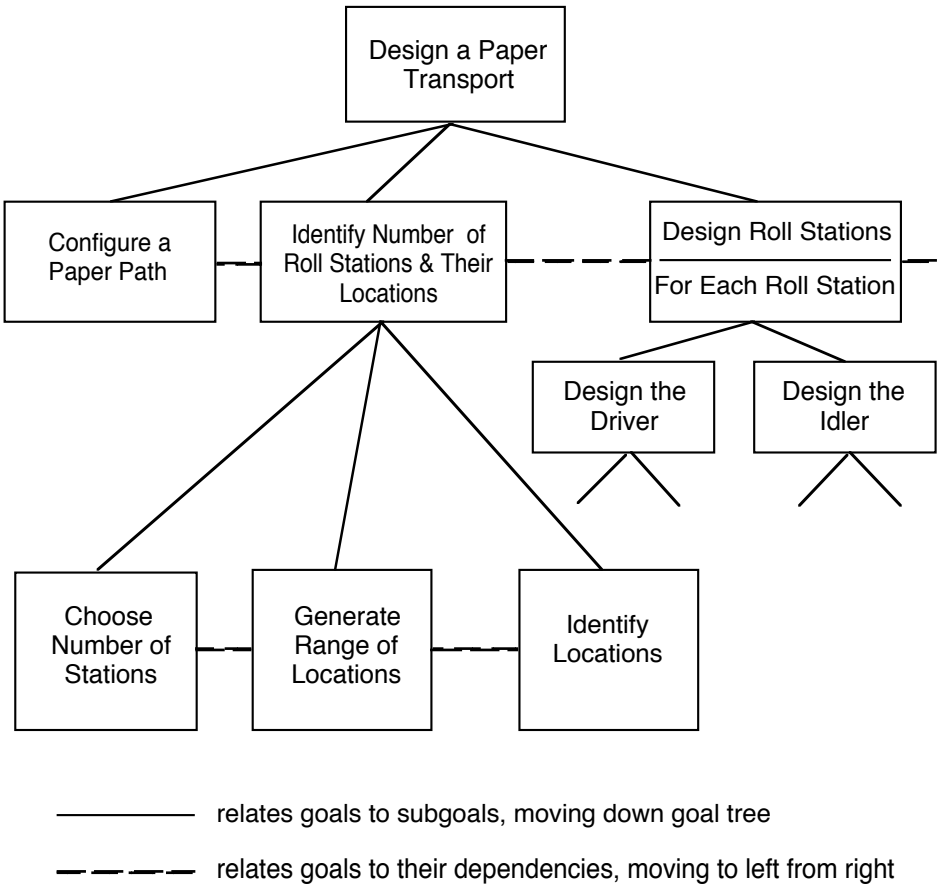

Figure 6. A stylized version of the goals and methods that make up PRIDE's design plan (after [1, 3]).

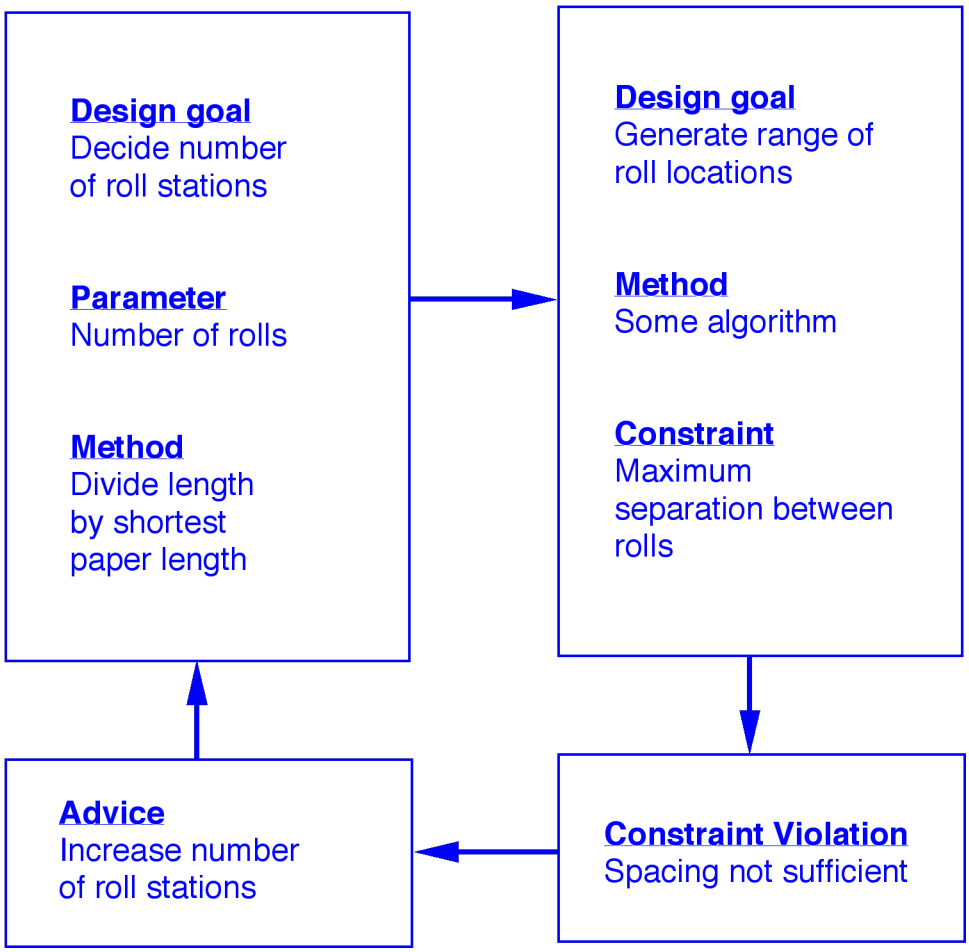

Figure 7. A stylized view of how design advice is handled in PRIDE (after (Mittal and Araya, 1986)); see also (Morjaria, 1989)).