New Roles for Machine Learning in Design

Yoram Reich
Engineering Design Research Center
Suresh L. Konda
Software Engineering Institute
Sean N. Levy
Engineering Design Research Center
Ira A. Monarch
Center for Design of Educational Computing
Eswaran Subrahmanian
Engineering Design Research Center
Carnegie Mellon University, 5000 Forbes Ave.
Pittsburgh, PA 15213, USA
Tel +1 412 268 5221, Fax +1 412 268 5229
email: yoram@cmu.edu, slk@cmu.edu, snl@cmu.edu, iam@cmu.edu, sub@cmu.edu

In *Artificial Intelligence in Engineering*
Special issue on Machine Learning in Design 8(3):165-181, 1993

Keywords: Machine learning, design, computational models, design practice, flexible modeling, shared memory, natural language processing, multistrategy learning.

Running head: New roles for machine learning in design

Abstract: Research on machine learning in design has concentrated on the use and development of techniques that can solve simple well-defined problems. Invariably, this effort, while important at the early stages of the development of the field, cannot scale up to address real design problems since all existing techniques are based on simplifying assumptions that do not hold for real design. In particular they do not address the dependence on context and multiple, often conflicting, interests that are constitutive of design. This paper analyzes the present situation and criticizes a number of prevailing views. Subsequently, the paper offers an alternative approach whose goal is to advance the use of machine learning in design practice. The approach is partially integrated into a modeling system called n-dim. The use of machine learning in n-dim is presented and open research issues are outlined.

1 Introduction

In order to situate our argument, we begin with a declaration of our perspective and orientation: we care about design first and only about computational models or tools in so far as they support design. In contrast, the prevalent approach displayed by most of those interested in machine learning (ML) in design is to care first about the creation of learning programs and only second, typically with scant attention, to their utility and usability in real design tasks.

Whether this downplaying of practice is intended or not, it most often results from one crucial overriding assumption: computational tools can be built that exhibit intelligent behavior, especially insofar as they can learn. This overriding assumption unfortunately blinds researchers into neglecting the pragmatics of research (1).

Recently, however, some early optimistic proponents of developing intelligent learning programs have become skeptical. For example, Wilkes (2) has recently commented upon forty years of work in AI programming stimulated by Turing's famous paper "Calculating Machinery and Intelligence." Initially, Wilkes' enthusiasm led him to explore various simple learning programs, and then to work on generalized learning programs. His initial hopes were fulfilled neither by his own work nor by that of his colleagues. Wilkes summarized his reflections by suggesting that "we take as a working hypothesis that intelligent behavior in Turing's sense is outside the range of digital computer" (p. 20). The aim of computational research with respect to learning should not be to produce learning algorithms that mimic human learning behavior, but rather should be to produce computational environments that fit and enhance human practice.

Fruitful design practice is collaborative involving different perspectives and knowledge from diverse disciplines requiring the creation of shared meanings of the designed artifact (3; 4; 5). Designs are produced, used, and evaluated in rich and varied environments. Processes are needed to capture the rich contextual information for application to future designs. The essence of design is the reconciliation of multiple disciplines, perspectives, knowledge sources, and modes of legitimacy. Moreover, the traditional approach of assessing user needs through non-participative approaches such as surveys, marketing research, and indirect feed-back from the marketplace via product failure, are insufficient (6).

An example might crystallize these points. Consider the design of the Golden Gate Bridge (7). In addition to the usual need for physiographical, geological, functional, and technical consideration, political, demographic, social, and environmental consideration are needed as well. These intervened at various stages of the project requiring, for example, the creation of an administrative unit, the Golden Gate Bridge and Highway District, to manage the bridge. It also led to two long litigations about the power of the district to tax, and even to the issuing of a permit from the Department of War (resolved through political supporters). It took almost 16 years from the first proposal (of a rather ugly bridge), to the completion of the construction. This included several failures, one of which was the construction of the fender ring requiring four revisions to the design. To condense an important design story, the Golden Gate required significant negotiations between different, incommensurate, and often conflicting interests in order to result in the design of a bridge which today symbolizes a major metropolis.

To ignore the problem of multiplicity of interests and concomitant is to miss the point of all engineering design practice: that the improvement of design is a pragmatic activity that includes time-to-market, usability, quality, producability, disposability, and other considerations.

In short, to improve design, design must essentially be approached as multi-dimensional and heterogeneous.

Consequently, learning in a design context cannot be reduced to a consideration of generating abstract data structures over sets of representations as is done in inductive machine learning, or of compiling operators as in explanation-based learning. If such computational tools are expected to play a role in design (though precisely what role is still an open question), their utility, inevitably limited and supportive in nature, has to be established by their applicability in real heterogeneous design tasks. One should keep in mind that such learning techniques are only a small part of the design process and only create a niche insofar as designers find the techniques to be usable and useful. Designing requires the resolution of terminology, misunderstandings, and correlations of factors spanning multiple perspectives, all of which are pervasive in design. Learning in design must anticipate, allow for, or incorporate, this multiplicity. An approach that isolates learning into a fixed set of enumerable techniques begs the question of relevance.

What must be studied is how meaning is shared among multiple disciplines, groups, and group members. Shared meaning, and its persistent form as shared memory (3), always requires careful mutual translation or linking of terms and concepts across groups because members of design groups working on the same artifact do not share the same experiences, concepts, perspectives, exemplars, methods, or techniques. This problem is readily demonstrated in several case studies (8; 4; 5). Shared memory not only concerns sharing among variegated professions but also among members of the same profession. The need for sharing *within* a profession arises from the differences in functioning context (as, for example, among electrical engineers in academe, in manufacturing plants, in design shops, in systems integrators, aerospace, etc.). In short, design is not only inescapably concurrent, but also, to use other currently fashionable terms, collaborative or participatory.

Design is an evolving process in which all the participants continually learn about the problem from the perspective of their discipline as they interact with other team members from different disciplines. When a design is repeatedly revised, what is learned also needs to be re-negotiated. Moreover, rapid changes in technology often results in patterns becoming obsolete, hence further complicating the learning and negotiation processes.

In what follows, we provide a brief tour of the ML landscape, situate learning in the design context, draw out the potential role of ML in such a context, and offer a few notes indicating what an approach that advances learning in design might look like.

2 An overview of current approaches to machine learning in design

Most developers of machine learning (ML) techniques operate within certain rigid assumptions about the nature of the input to ML programs, especially as to the ease with which viable, unambiguous representations can be created and statically maintained throughout the learning process. Experience, however, shows that these assumptions are rarely if ever valid in the context of the design situation which is inherently multi-perspective and heterogeneous. Each of these multiple interactive perspectives could potentially have its own representation. The multiplicity of design cannot typically be reduced to the perspective of a single participant, since any one participant does not fully understand the problem, nor can any single participant complete the design alone. Hence, one cannot take a technique effective for a single designer and conflate it for the multi-perspective design problem.

Our position is that what is central to both the development of computational design tools and their subsequent uses in real design situations is their capacity to support multiple design participants (9; 10; 11; 6). This view should not come as a surprise; we take the position that development of ML tools for design is as multi-perspective as is design itself.

2.1 Using ML programs

In what follows, we describe six steps in the development and use of ML programs based on the tacit assumption in prevailing research practice, each of which is followed by our comments and criticism.

(1) *Formulation of the learning problem in a particular design context.*

> ASSUMPTIONS: The design problem and what needs to be learned are already understood. The formulation of design and learning problems is not difficult and once a formulation is specified, it tends not to change.
>
> COMMENTS/CRITICISM: Given such assumptions, a learning tool would not be viable when the user has only a partial or evolving understanding of the design situation — something that is quite common, especially in the early phases of design. The formulation of the design problem and what needs to be learned in a given design context has to be supported iteratively and continually.

(2) *Preparing input for ML programs.*

> ASSUMPTIONS: The integration of multiple sources of information and the processes for integrating them, e.g., selecting the sources, reconciling their terminology, and selecting the information that can best aid learning, insofar as they are considered, are either ignored or assumed to require minimal effort.
>
> COMMENTS/CRITICISM: Interesting and useful design data are not easily identified, gathered, and integrated into a form that is amenable to being translated into computer programs (see also step 3). For one thing because design cases are the products of multiple perspectives, this multiplicity and its reconciliation must be captured in their description. Such a descriptive task is difficult in itself and becomes even more difficult when technology advances and a particular reconciliation becomes obsolete.
>
> Moreover, in order for designers to turn past cases into useful sources of information, (12; 13; 14) these cases have to be described meaningfully relative to the present state of design knowledge. Even when this is done, descriptions are always from the point of view of those recording them and tend to incorporate only a partial understanding of the overall design problem. Assumptions can be easily hidden in design cases. For example, bridges constructed in the U.S. as compared to those constructed in Britain for similar sites may be different simply because their respective design standards give rise to the tendency to select different structural elements for similar functions (15). Such factors may be implicit in

the minds of those describing these cases and therefore not be expressed in their descriptions, let alone be made available for learning programs.

(3) *Devising a description schema for representing information to be used as input to the learning program.*

ASSUMPTIONS: Once cases are collected from various sources, a schema for representing them must be devised. A list of property-value pairs is the traditional representation for most learning programs. It is supposed to capture the properties used to describe cases, their values, and their relative importance.

COMMENTS/CRITICISM: While fixed formalisms have their value in organizing information, they do not solve (with step 2) the problem of encoding the knowledge in the schema. In real world design situations, this can require a knowledge representation of a breadth and depth that cannot be satisfied by the techniques and methods used on toy or demonstration design problems. Moreover, it is the interaction between the schema and the information content that creates the problem of learning bias which is only partially understood even for trivial artificial cases (16; 17).

Moreover, the above assumption presupposes that representational forms are sufficient to encompass the needs of the design problem. Instead of looking for representations that are appropriate for design (18; 19), ML in design has shoe-horned design problems to fit, *a priori*, representational forms.

(4) *Selecting the learning program.*

ASSUMPTIONS: In principle, the selection of the learning program is based on the properties of the learning problem created after simplifying the design problem. It is unnecessary to assess the relevance of this selection by testing the learning program in the context of the original design problem.

COMMENTS/CRITICISM: In most cases, programs are selected because they are readily available and not because they are actually the best for the task (20). In many cases, the representation "supposedly natural" to the domain, is selected to fit the learning program available. Even if programs are selected based on steps 1 to 3, it is rare that the reasons underlying the selection are well articulated or that their validity is reviewed based on a test of the learning system in real design situations.

(5) *Selecting operational parameters for the learning program, and testing them.*

ASSUMPTIONS: The selection of operational parameters is trivial. Furthermore, parameters are often built into learning programs in an *ad hoc* manner to improve performance on particular simplified learning tasks. It is assumed that these parameters can then generalize to new learning contexts.

COMMENTS/CRITICISM: It is common practice among researchers to use operational parameters to tune the performance of their learning programs to obtain the best possible results through extensive experimentation on demonstration problems. These selections, however, do not easily generalize to new contexts. The reason is that different parameters control different aspects of the program behavior (e.g., the complexity of the schema, noisy data, or the amount of searching allowed) and it is not clear which ones contribute to good or bad performance.

In consequence, the selection of parameters that result in good performance in a particular learning context remains non-trivial and not clearly understood.[1] It is only recently, and then only in simple cases, that comparative studies on the selection of operational parameters were performed (21; 22).

(6) *Analyzing the results.*

ASSUMPTIONS: Analysis of learning is often one-dimensional. A common dimension for evaluation of the quality of learning is the reduction of errors in arriving at correct solutions. As a direct consequence of this assumption, it is implicitly assumed that such analysis is sufficient to deal with evaluation from the multiple dimensions that characterize design problems.

COMMENTS/CRITICISM: Analysis of knowledge, whether created by ML or not, is inherently multi-dimensional. Content and functional properties of the learned knowledge must be assessed to provide a more comprehensive view of its quality (23). The selection of particular evaluation metrics and their interpretations is relative to the multiplicity of dimensions and perspectives that are embodied in a design.

The multi-perspective nature of evaluation must be addressed by testing the relevance of learned knowledge in design practice. It is only through user participation in the development, use, and evaluation that the multiplicity of perspectives are permitted to arise in its fullest form to evaluate the relevance of what has been learned. This observation applies with even greater force in the context of ML (9; 10; 6).

In the accepted, and optimistic, view of ML in design, activities 1-3 are perceived as preparatory and minor and the remaining activities as relatively simple. Namely, (1) the creation of a description schema is straightforward — that is, the consequences of choosing a learning bias are unimportant; (2) the meaning of terminology is clear and is easily discerned; and (3) learning can be carried out with little regard to the need for knowledge and purpose but purely as a manifestation of the algorithmic nature of almost all learning programs (24). Therefore, the difficulties of preparing design input information are ignored or thought to be under control. On the basis of this assumption, it is expected that ML techniques will learn all the rest of domain knowledge autonomously and even learn how to perform new tasks as well.

This is a misguided view: learning bias has major importance and impact on learning (25), terminology is critical and non-trivial (26), and knowledge about what is to be learned is crucial. Without addressing these factors, the use of ML is reduced to performing pragmatically irrelevant algorithmic tasks.

In the next section, we briefly discuss two types of learning using only selected techniques that generate new knowledge structures (e.g., inductive learning techniques), rather than techniques that improve the efficiency of problem solving (e.g., explanation-based learning techniques). We have chosen to illustrate the deficiencies of inductive learning methods as an example of how these methods fail to address the criticisms of learning systems described above. Explanation based learning techniques are really directed at automated techniques themselves rather than design procedures.

---

[1] The selection of parameters, is therefore, itself, a full-fledged learning problem.

2.2 Inductive learning

Inductive learning techniques consist of: *supervised* and *unsupervised concept learning*. These do not differentiate between techniques that are implemented differently (e.g., symbolic vs. neural learning), an issue orthogonal to the functionality of learning viewed at the knowledge level (27). A brief summary is provided of each.

Supervised concept learning can only support the task of classifying new objects into a set of pre-defined classes. For most learning programs, design examples are represented by a list of specific property-value pairs and are classified into a set of classes that can represent a *single* design descriptor. The task is to predict this design descriptor based on these specific properties. To illustrate, Arciszewski, Mustafa, and Ziarko (28) used a supervised concept learning technique (a descendant of a program previously developed by the third author) to differentiate between feasible and infeasible designs. They simplified the description of artifacts to several properties and restricted the task to classifying artifacts into two classes, feasible and infeasible, essentially extracting *evaluation*, rather than *synthesis* knowledge.

Unsupervised concept learning provides a better basis for acquiring synthesis knowledge than supervised concept learning (29). The principal idea is that design specifications and solutions (i.e., design descriptions) are correlated; certain combinations of the chosen properties give rise to corresponding combinations of design descriptions that satisfy design specifications. A clustering based on this correspondence allows the retrieval of appropriate designs given a new specification similar to an existing one. To illustrate, Lu & Chen (30) described an approach for treating multiple properties in the design specification. The representation of artifacts they used was again restricted to a short list of property-value pairs. The ML techniques they used were CLUSTER and AQ15. In their approach, the design specifications are clustered into a finite set of classes using unsupervised learning.

The classification generated by an unsupervised learning program can then be used by a supervised concept learning program to generate evaluation rules as before. The results of supervised learning are rules that assign the description of a design to the right class, i.e., approximately identify the design specification satisfied by the design solution. Thus used in conjunction with each other, these two types of learning programs allow for the generation of synthesis and evaluation knowledge.

In spite of the seeming power of these programs to learn design knowledge, the simplifications that underlie both of them (and their examples) are subject to the same criticisms enumerated earlier of learning programs. Such simplifications occur in many ML projects in design (including one by a co-author of this paper (31; 32)). Note that our criticism about the simplifications made by these projects is not meant to dismiss their contribution. In fact, they constitute the foundation that enabled further understanding of the issues involved in the use of ML in design. Our criticism is directed against those perceiving that the situation today corresponds to that of 10 years ago, or those persisting in believe that working on simplified problems will lead to significant progress in addressing the critical problems of learning in design.

2.3 Multistrategy learning

To overcome the limitations of existing learning techniques, machine learning researchers postulated that the solution to diversity in learning situations requires the use of multiple ML techniques. This would enable a variety of information available for learning to be taken into account. The use of a multiplicity of techniques was called *multistrategy learning* (33).

The idea of multistrategy learning in not new. Several proposals incorporating knowledge or heuristics in selecting machine learning techniques have been presented in the past. Salzberg (34) described heuristics to be used in inductive learning such as usualness, conservatism, ambivalence, and proximity. While these heuristics may improve the use of ML techniques, they do not resolve the issue of how to choose among the infinite ways in which these heuristics can be used. More recently, similar ideas have been discussed by Stepp, Whitehall, and Holder (35). Their view is that intelligent ML techniques can be improved by embedding fixed interpretations of levels into an algorithm.

In general, two levels can be identified within the multistrategy approach to learning (36): the *macro* and the *micro*. The macro level deals with the use of a collection of learning programs each addressing a separate learning problem even though they interact. However, it is the *non-trivial* task of the user to assemble these techniques and resolve their interactions. As was pointed out earlier, ML should be seen as integrated with a suite of computational tools that support design participants in various tasks including those which may assist them in using ML tools.

The micro level deals with the development of learning programs that employ a variety of fine-grained learning strategies for solving a specific learning task. In this case, the learning program is expected to automatically select its own strategies without user intervention; an expectation that is questionable in light of criticism already made.

BRIDGER, an experimental system developed to explore the extent to which learning can aid in the creation of design support systems, illustrates the use of these levels in building a learning system. At the macro level, BRIDGER's learning task was manually decomposed into two subtasks, learning synthesis knowledge and learning redesign knowledge, with pre-defined interaction scheme. Each of these tasks was assigned to a different learning program (37). At the micro level, each of these systems uses several learning operators to accomplish its subtask. The control over these operators was fixed as well. The designs of the macro and micro levels were independent of each other.

In the multistrategy approach, once a system is constructed by integrating the macro and micro levels, it is expected to operate completely within the scope originally defined by the user. Thus, such a system does not deal with perspectives different from those initially embedded in it, oversimplifying the design task.

In conclusion, our criticisms of ML research may seem to indicate that we are against the development of techniques such as those briefly reviewed. On the contrary, our task here is to situate these techniques in the context of the practice of design. This means that we do not set our goals for ML research to be that of "creating systems that pass the Turing test" but to explore the critical role they can play in design insofar as it is evolutionary, negotiated, and multi-perspective.

3 ML for design practice: research areas and issues

There is significant potential for ML in design. This potential can be realized if the criticisms in Section 2 are addressed. We posit that this can be done by revising the role of ML from an *automatic* process to one that is part of a design *support* system aiding humans in performing the tasks required in designing.

In what follows we briefly describe learning activities that occur in design and discuss how these can guide the design of better ML programs for design practice.

3.1 What is learned in Design?

In order to explore the role of ML in design, we need to discuss the kinds of things that are learned in the design process. Learning activities can take several forms and assume several roles.

First, designers learn technical (in most cases analytical) knowledge; in fact, over the years, it became the primary, if not the sole, function of the professional education system (38; 39; 40). For example, designers learn how to use finite-element programs for calculating the strength of structures; these programs calculate the strength of *models* of the actual structures, while leaving the responsibility of creating the models and interpreting the analysis results to designers. Such models creation and interpretation are the points where multiple perspectives must be negotiated and reconciled. If they are not, the analysis may be incorrect and subsequent failures of designs may occur (41).

Second, in a study of learning and design, Cross and Nathenson state: "in the course of designing, designers learn about the problem, its solution, and their relationship" (42). Such learning has been observed in empirical studies of human designers (43). While Cross and Nathenson concerned themselves with the single designer, the context of actual design generally has multiple design participants learning about their part in the problem and their peers' perspectives; additionally, since all participants modify their understanding, the target knowledge about the problem is never static.

Third, designers assimilate experiences for use in future design problems. These experiences are what differentiate expert and novice designers (44). Designers must always be aware that new design situations prevent the "as-is" application of previous experiences. Designers need to learn the similarities as well as the differences between current and previous problems and adapt old solutions to new situations. There are currently no algorithmic solutions to this problem. What is needed is research that records and evaluates how current automated techniques can support such learning.

Fourth, from user feedback or from the failures of artifacts, designers learn about the viability of certain design beliefs, judgments, decisions, or practices in certain situations (45; 12; 14). Feedback from users (customers) about design arrives after a product is released to the market. When a product fails, designers attempt to analyze the reasons underlying the failure, sometimes, for example, to find them rooted in simple inattention to customers' concerns (46).

One of the central roles of ML techniques should be to amplify the ability of designers to perform these four learning activities (47). Further, the third and fourth points above, about the role of learning in design, suggest the need for developing data or text bases of design cases and information management tools for maintaining, searching, and learning from, these.

3.2 How can we design better ML programs?

The design of "good" ML programs suited even for simplified design tasks is complex. For example, the design of decision trees that are optimal in the number of tests is NP-complete (48); furthermore, the wealth of information in numerous reviews on the design of decision trees suggests that many different approaches are being and need to be tried. We, as designers of ML programs, cannot know in advance how to build ML programs because we are still learning about what the problem is. While we collect information about relevant disciplines, we have not even started to accumulate experiences from past uses, nor have we accumulated meaningful users' feedback. The wealth of information about ML programs, their testing, and experiences using them, must be documented and managed effectively. Therefore, our approach to the design of ML tools relies on the following premises:

(1) The study of design must accompany the development of ML tools for design; they need to be interlinked.
(2) ML tools must be developed in a context in which information about the success and failure of previous design practices is accessible from multiple points of view. Keeping the history of tools development in an accessible form, for example, can be as beneficial as keeping design rationales. The more organized such repositories are the better. Such an organization requires a framework similar to the one we outline in Section 4.
(3) ML tools must be tested in actual design settings.

We next describe a particular approach that facilitates addressing the issues raised in this section. We defer the argument that this approach can address the criticism raised in Section 2 to Section 4.3.2.

Note that although we will detail a specific implemented system, it is not the particular implementation that is the focus; rather, the critical characteristic is the *approach that admits, facilitates, and records arbitrary and evolutionary modifications in response to actual practice.*

4 *n*-dim: Information modeling and creation of shared memory

The motivation for developing *n*-dim (*n*-dimensional information modeling) emerges from the need to address issues observed in actual design practice. As learning is integral to design, *n*-dim must provide facilities for learning. In addition, since our approach involves the design of ML programs, *n*-dim becomes a candidate for aiding in this activity.

4.1 An overview of *n*-dim

The space of objects in *n*-dim is conceptually flat; that is, objects do not contain other objects, *per se*. Instead, multiple structures can be imposed on this flat space by means of *models*, which are comprised of *links*, or relationships between objects (models themselves being objects). In this way, the same object may participate in many models.

n-dim is implemented in a prototype-based object system called BOS, the Basic Object System (49). Since it is prototype-based, there are no classes, *per se*; rather, any object is a potential prototype for another object. For more information on prototype-based object systems, see (50).

There is a basic cleavage in the space of *n*-dim objects between *atomic* and *structured* objects. As the name indicates, *atomic* objects cannot be broken down any further, e.g. an integer, a link, a piece of text, an image, an audio bitstream, etc. One could think of atomic objects as things that have *values* of some sort.[2]

The primary form of *structured* object is the model. A model is a set of links, which are, themselves, atomic objects. The value of a link object is a 3-tuple, ¡*source, target, type*¿ where *type* is merely a label for the link; link types are given their meaning(s) by the modeling language(s) in which they occur.[3] All objects, whether structured or not, are constructed using another model acting as their *modeling language*. Typically, modeling languages specify what objects can be in a model and what relations they can have to one another. Such specifications can be thought of as grammars.

Three main features of *n*-dim form a sort of *critical mass*, in which the whole is greater than the sum of the parts.

(1) *Flat Space* captures the fact that an individual object can be situated in multiple contexts. There is a special link called a `part` link, which is canonically represented as a box inside of a box; i.e., objects "inside" models. All other links are shown as directed and labeled lines. It must be stressed that due to the flat space of *n*-dim, a model does *not* (in any physical sense) contain objects that appear inside it. Also, things can be found *in the context(s)* in which they are used (referred to, referenced by, placed in relationship to other things, etc.). Hence, rich contexts can be created with virtually zero overhead. Also, things can be found *in the context(s)* in which they are used (referred to, referenced by, placed in relationship to other things, etc.).

(2) *Generalized Modeling* allows people to operate on *things* and *kinds of things* interchangeably, and move freely from one level of abstraction to the other. Every unpublished (see item 3) model is mutable, and can have operations and attributes defined on it as an individual without affecting any other model. This means that models (information structures) can serve as *prototypes:* potential ancestors of other models.[4] Any model can serve as the starting point for another model, in the sense that it can be copied and the copy modified in ways independent of the original (not as with class/instance relationship). Models can also serve as *modeling languages*: schemas for structuring information. If, for instance, one were to ask *n*-dim to use an

---

[2] The creation of new atomic object types generally requires some programming, since new types of values often indicate new types of fundamental operations.

[3] It is quite possible to have the same link type mean totally different things in different contexts; we view the meaning of links as something to be *negotiated* by users of the system over time. Operationalizing the semantics of particular interpretations of links is considered an open-ended process; n-dim provides mechanisms for doing so, but does not require it to be done in order to use a link type.

[4] We will use the terms "instance" and "prototype" somewhat interchangeably in what follows, since, in a prototype-based system, the two concepts coincide; the different connotations are useful in distinguishing various *uses* of a model, however.

`Integer` object[5] with the value 1 as a modeling language, one would get an object in the language 1, which could only have as its value the number 1. This grammar has only one legal sentence. Modeling languages are represented as models themselves (i.e., as things to be designed, negotiated, etc.). In this sense, models can act as types for the creation of other models. In a fundamental way, the notion of prototypes is more basic than of modeling language, in that one modeling language can serve as a prototype for another modeling language.

(3) *Publishing* is a mechanism for making models formally exchangeable and persistent following the library as a metaphor. Hence, *traces* of the evolution of information and its structure over time can be found in the growing repository of published objects.

One can map a structure of an *n*-dim model onto multiple *projections*, which discriminate between possible views of that structure. Any projection can be mapped onto multiple *displays*, which fix the characteristics of a projection *vis a vis* its rendering.[6] Projections are models, as are renderings. The system merely interprets these models appropriately when needed.

Modeling languages can also specify semantic, as well as syntactic, information about models. One way this can be done is to include *rules* in modeling languages. There are two broad classes of rules: rules on structure and rules on content. If a rule's predicate takes as arguments only objects that appear in the model itself, and are not the contents of parts of the model, then it is a rule on structure, e.g., it operates only on the *visible* structure of the model. For example, a rule that limited how many of a certain kind of object could appear in a model would be a rule on structure. If a rule's predicate takes any arguments that are contents of the parts of the model, then the rule is a rule on content. If one were creating a model to be used as a task-assignment modeling language, one might want a rule which restricted the total number of hours of work assigned to an individual to be less than 40. Such a rule would not operate on tasks assigned to individuals but on the hours of work contained as part of the model for individuals.

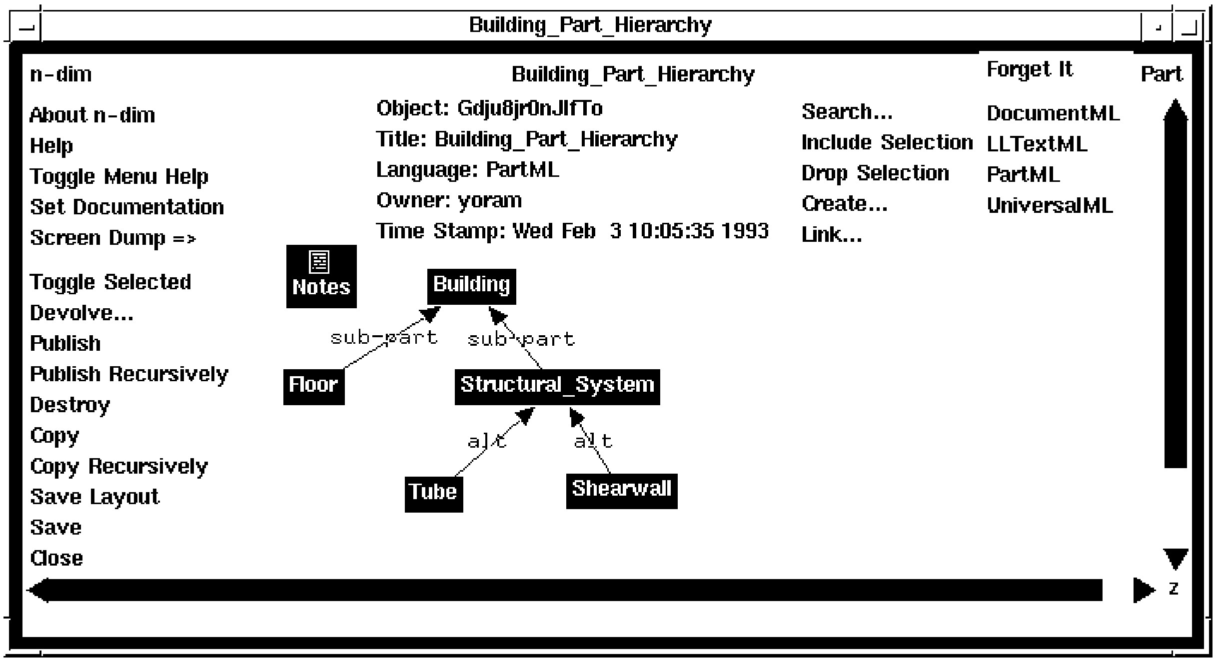
To illustrate the above concepts, consider the model called `Building_Part_Hierarchy` in Figure 1. The model's display is used to illustrate several of *n*-dim's facilities. Three menus and one overlay are displayed in addition to the model (the latter in reversed video). The left menu appears when clicking on the word `n-dim` in the left hand side of the top banner; it can be used to obtain help about *n*-dim, publish objects, as well as to copy and save objects. The middle overlay provides information about the model and appears when clicking on the model's name in the middle-top banner. The data displayed include the unique model identifier, title, language used to create the model, the creator and the time the model was created. The left menus include the facilities for creating the model in addition to searching for objects; they appear when clicking on the modeling language name in the right-top banner. When the `Create...` is clicked, the most right menu appears, allowing to create new objects that are permitted by the `Part` modeling language. The model itself contains five part objects linked with either `sub-part` or `alt`(ernative) links in addition to a text object containing notes about this model.

We now discuss how *n*-dim supports the four learning activities introduced in Section 3.1.

---

[5] Note that Integer objects are atomic.

[6] A rendering of a model can be something like a window presented to the user for interaction, a printed file, etc.



Object: Gdju8jr0nJlfTo
Title: Building_Part_Hierarchy
Language: PartML
Owner: yoram
Time Stamp: Wed Feb 3 10:05:35 1993

Figure 1: A simple *n*-`dim` model

4.2 Coding design knowledge

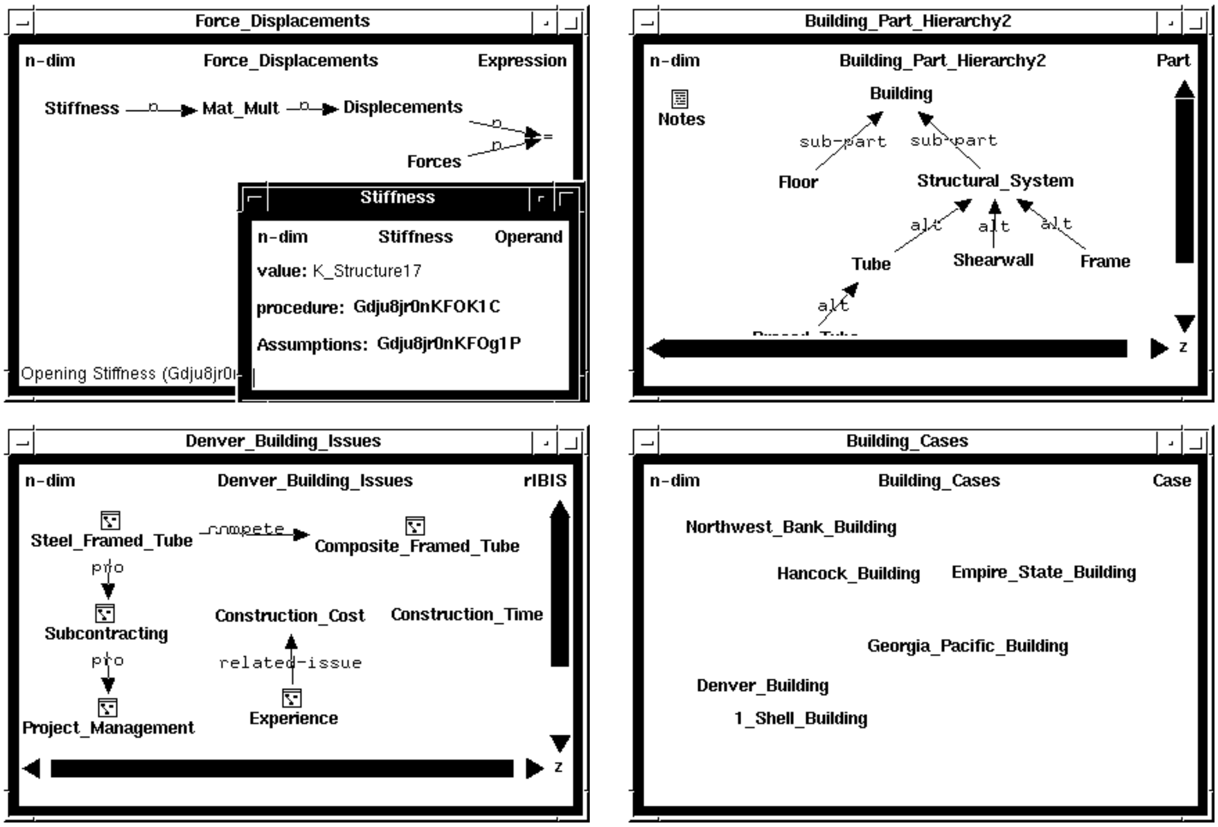
This section addresses the first learning activity in section 3.1. Design knowledge includes formal and informal components.[7] Figure 2 consisting of four separate models shows how *n*-`dim` facilitates the coding of knowledge of both types.[8] The top-left model, written in the `Expression` modeling language, shows the relationship between the stiffness matrix of a structure, its displacements, and the external forces acting on the structure. This model can be used to calculate the displacements given the external forces or the external forces given displacements. The small overlay window, displays the model `Stiffness` as a frame with slots describing: its value, procedure of calculation, as well as the assumptions underlying the modeling of the structural elements as finite elements. The assumptions provide the context for assessing the viability of the modeling. Note that the `Stiffness` object is written in the `Operand` modeling language, which was prototyped using the `Frame` modeling language; it requires that the three slots displayed will be provided for each operand in the `Expression` modeling language.

The top-right model, called `Building_Part_Hierarchy2` written in the `Part` modeling language, shows an elaborated version of the the part/sub-part hierarchy that appeared in Figure 1. The lower-left model, called `Denver_Building_Issues` and written in the `rIBIS` modeling language (i.e., a recursive variant of issue-based language (53)), describes issues related to the design of a building in Denver. Some of the objects are primitive issues while the others are `rIBIS` models themselves, demonstrating how models can be embedded in other models. Finally,

---

[7] Elsewhere we have demonstrated that even the most "formal" component involves, and is based on, informal components (51).

[8] The examples below contain brief data from analysis of protocol interviews of engineers and architects collected in a knowledge acquisition study of tall building design conducted by Steven Meyer and Steven Fenves (52). We thank them for allowing us access to these protocols.



value: K_Structure17

procedure: Gdju8jr0nKFOK1C

Assumptions: Gdju8jr0nKFOg1P

Figure 2: Knowledge of structural design

the lower-right model, called `Building_Cases` and written in the `Case` language, depicts several cases of building designs, including the `Denver_Building` which includes the aforementioned issues.

One may ask what is the significance of this knowledge coding for ML? The answer comes from the bi-directional relationship between contextual information and learning. First, information situated in a rich context can focus learning on missing information that is needed in that design context and might be learned. Second, information always is created, modified, and evolves by learning, for instance, the `Building_Part_Hierarchy2` represents part of the product of learning from interview data of designers.

4.3 Learning with a toolbox of ML programs

This section addresses the second and third learning activities in section 3.1. Learning design knowledge has to be responsive to the multiplicity of perspectives and the volume of information. As discussed in Section 2.1, the prevalent way of handling these difficulties, one we disagree with, *reduces the multiplicity and the volume at the source*. All information and perspectives, are analyzed, organized, and represented before starting to develop a design; no new perspectives are permitted once the initial representation is determined, and therefore, no conflicts can arise.

In contrast, our approach *supports compressing the volume of information in context and* *according to one or another perspective.* We will use ML and natural language processing techniques in order to facilitate the compression of information into a usable form only in context and according to a selected perspective.

The type of data in engineering databases and the diversity of potential designers' background knowledge and preferences, require adopting an interactive approach to learning (26; 10). Designers and computers will be inextricably involved in the learning process. Our approach aims to provide designers with facilities to model the learning problem and its solution in an incremental manner; it differs from the development of learning toolboxes. The latter are limited to providing a framework for interacting and managing a collection of learning tools by providing formalisms that facilitate transfer of results from one system to another (54). Being fixed *a priori*, such formalisms restrict the techniques available to the designer. In contrast, our approach does not restrict the kind of techniques used or the way they are used.

We view learning as any activity of "information growth." For example, asking a question and receiving an answer is learning. In this sense, we consider even communication and search facilities learning tools. In fact, *n*-dim provides both facilities. Nevertheless, we now focus on the use of ML techniques to support the second and third learning activities discussed in Section 3.1. In this focus, we mainly describe one particular learning mechanism that provides natural language processing for *n*-dim. It will be used to demonstrate the kind of flexible integration of ML programs in *n*-dim and the functionality expected from such integration.

4.3.1 Natural language processing

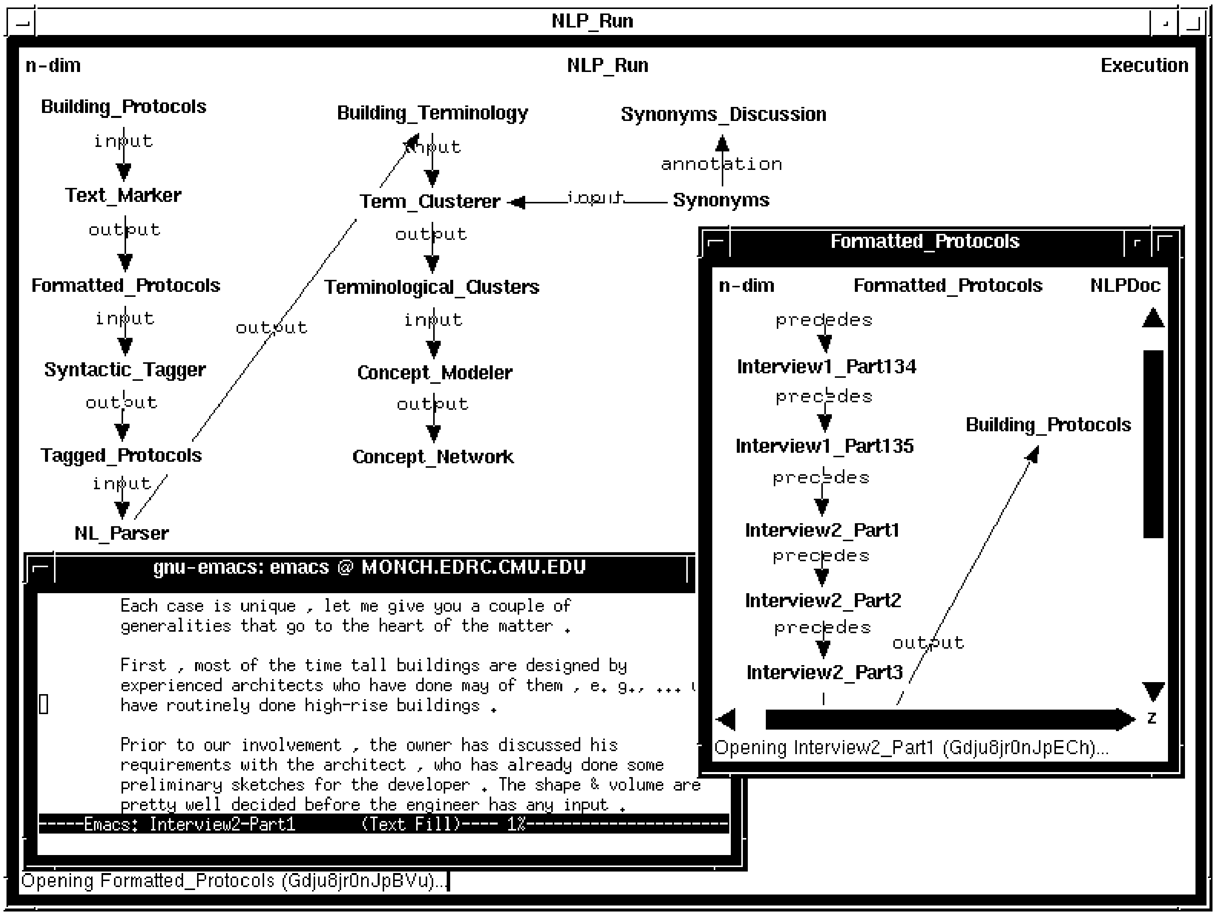
One of the learning capabilities being integrated into *n*-dim is natural language processing (NLP). NLP enables the discovery of terminological patterns implicit in large text corpora. The discovered patterns can act as a basis for building multiple conceptual networks in various sub-domains. The example that follows is taken from the domain of architectural and engineering design of tall buildings.

The information structuring techniques described will allow different participants on the same design team to: (1) retrieve efficiently information relevant to their current tasks and decisions, (2) support fast introduction of new team members to on-going projects, and (3) support the construction of shared meaning via links among conceptual networks as specified and negotiated by the participants (3).

To illustrate this capability, consider the model `NLP_Run` created in the `Execution` modeling language and displayed in Figure 3. It is an example of how *n*-dim can facilitate the use of NLP and how NLP can help establish shared vocabulary and meaning for various activities. Applicability to activities such as indexing and building multi-perspective representations of artifacts is shown. Applicability to other activities such as query formation and knowledge acquisition, also facilitated by use of NLP, is discussed in (55).

The model shown in Figure 3 contains a sequence of steps, not all automated, that lead from text to various kinds of NLP assisted analysis and information structuring. Everything that is shown is either implemented or in various stages of implementation in *n*-dim.

The process shown starts with marking up a file containing text (`Building_Protocols`) that is then tagged, parsed, analyzed, and structured by NLP tools (`Syntactic_Tagger`, `NL_Parser`,



```
Each case is unique , let me give you a couple of
generalities that go to the heart of the matter .

First , most of the time tall buildings are designed by
experienced architects who have done may of them , e. g., ...
have routinely done high-rise buildings .

Prior to our involvement , the owner has discussed his
requirements with the architect , who has already done some
preliminary sketches for the developer . The shape & volume are
pretty well decided before the engineer has any input .
-----Emacs: Interview2-Part1       (Text Fill)---- 1%--------------------
```

```
Opening Interview2_Part1 (Gdju8jr0nJpECh)...
```

```
Opening Formatted_Protocols (Gdju8jr0nJpBVu)...
```

Figure 3: Integrating NLP in n-dim

Term_Clusterer, and Concept_Modeler). These contribute to the building of an n-dim model. The Formatted_Protocols model contains transcriptions of interviews that are marked for later indexing via operations available in the Text_Marker. A small section of one of the transcriptions, Interview2_Part1, appears at the lower part of the figure. The Building_Terminology model contains syntactic well-formed phrases that represent meaningful chunks of information in the building domain. These chunks have indices into the original document according to the previous text mark-up operations. Information from Building_Terminology and an object containing synonym information are used by the Term_Clusterer to produce another n-dim object called Terminological_Clusters. The Synonyms object is annotated by an rIBIS model (Synonyms_Discussion) representing why terms were considered synonymous. The Terminological _Clusters are further structured via the use of concept modeling tools (Concept_Modeler) to produce a concept network which still maintains indexing links back to the original document. Concepts can therefore be viewed in the context in which they occur.

Figure 4 displays a very small portion of the Terminological_Clusters object in a particular rendering. The position from left-to-right indicates the relative importance of the terms as they appear in the protocols analyzed. The figure shows terms from the largest clusters and indicates some relationships among them. These relationships need to be evaluated, refined, and labeled through negotiation by design participants. The term building has several sub-terms. These

The figure appears at the end of the paper (page 27)

Figure 4: Terms discovered by NLP

can be organized according to categories decided on by the design participants. For example, the `building` sub-terms can be organized along the following lines: particular building (e.g., `hancock` and `empire state`), properties of building (e.g., `form` and `width`)[9], and types of buildings (e.g., `office` and `apartment`).

The `Concept_Modeler` provides tools for manipulating the `Terminological_Clusters`, including their presentation according to various perspectives and for using the indexing links to access contexts in the original document. Besides viewing terms in context, the indexing links provide the basis for iterative processing of documents, such as resolving the referents of vague words like 'thing', dividing the document into smaller or larger chunks, mapping synonymous words into an agreed upon single term, and correcting errors.

Terms created by the NLP analysis can serve as labels for other *n*-dim models. They can also be used in creating various structured representations of domain concepts from a single discipline or different disciplines. These representations can be used as a baseline for negotiating shared meaning and thereby facilitate communication and interdisciplinary learning. The models described and their associated facilities are embedded in modeling languages such as `Execution`. These languages provide for interactive learning in a rich context.

While NLP tools provide facilities for capturing textual context, *n*-dim's modeling facilities also allow for capturing other forms of context such as drawings, photographs, audio, and video. Future advances in various fields may allow the extraction of information from these forms as well.

In addition, the NLP techniques described above are not the only kind of techniques that can facilitate the establishment of consensus over terms. Other tools, such as KSS0 (56), that are developed to be used in multi-expert elicitation process, can be used as well. In fact, such tools can be integrated into *n*-dim.

4.3.2 Contextualized use of ML

ML techniques can also be integrated into *n*-dim in a manner similar to the NLP programs, via the `Execution` modeling language. Figure 5 shows the model `ML_Run` which enable users to manage the use of different ML programs. We will use this model to illustrate how *n*-dim improves on present use of ML techniques by addressing the issues raised in Section 2.1.

We address the first three issues raised in Section 2.1 together:

(1) formulation of the learning problem in particular design contexts;

---

[9] The notation ":" stands for a preposition.

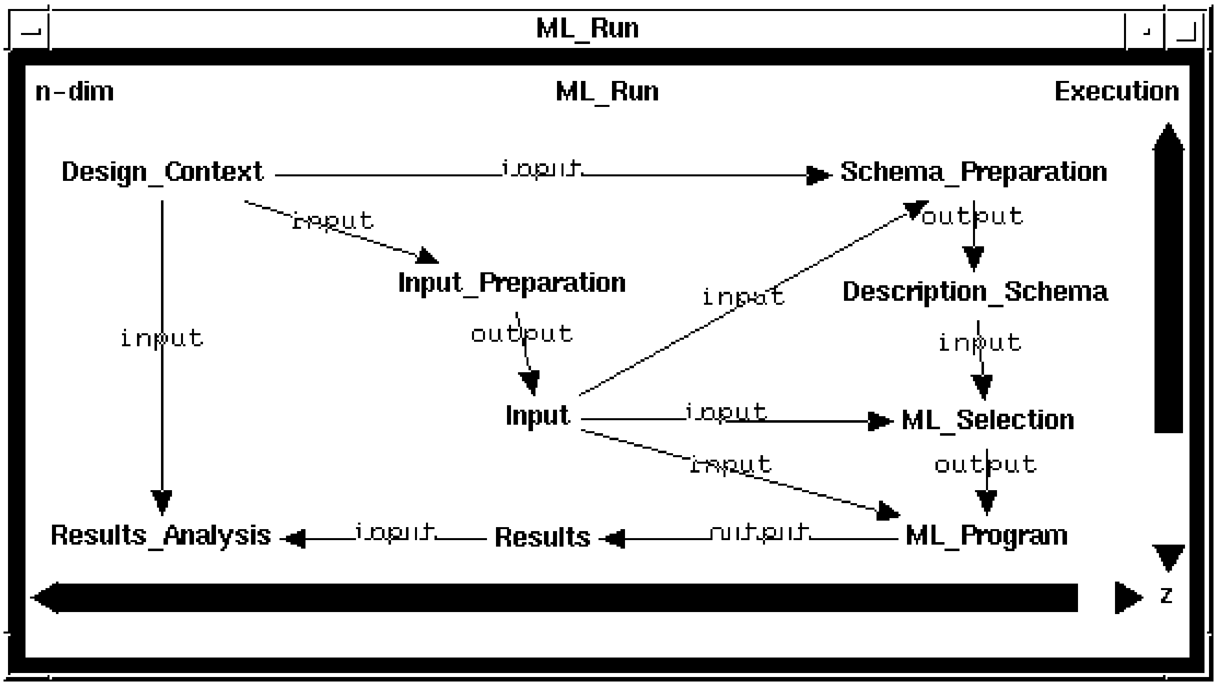


Figure 5: Integration of a ML program

(2) preparing input for ML programs; and
(3) devising a description schema for representing information to be used as input to the learning program.

In response to these issues, we have provided, in the previous section, an example of how n-dim enables a user to deal with changing context by the interactive use of NLP programs. n-dim enables the use of ML techniques in contexts that may change over time (`Design_Context` in Figure 5). Once various portions of the context of a design problem are modeled in n-dim, ML techniques can make use of this context. As the context changes, the changes are accounted for in the model.

The `Input_Preparation` and `Schema_Preparation` contain operations for mapping items in the context into structured input suitable for ML programs. A mapping can be elaborated by `rIBIS` models that record the assumptions behind the mapping operations and their origin in the design context.

Creating the modeling languages for interactive use of NLP programs to support learning was relatively easy since: (1) the NLP programs accept unstructured text as their input; and (2) at least some important aspects of a new design context are reflected simply by the availability of new texts, such as available project memos, manuals, reports, etc., as well as literature from disciplines relevant to the design project.

The creation of languages for interactive learning of most ML programs is more difficult than the one needed for NLP programs since they require structured data, such as lists of property-value pairs, as their input. The input is not simply marked-up text, but a mapping from new design contexts into the structured input required by the ML program. By in large, the operations provided by such languages are manual, rather than automatic. These languages need to evolve in response to better understanding of design problems and the mapping operations.

The next two issues raised in Section 2.1 are handled separately.

(4) Selecting the learning program.
Gradually, we intend to build models of uses of different learning techniques (included in the model `ML_Selection`). Initially, a model can be specified according to basic dimensions such as: supervised/unsupervised, incremental/non-incremental, similarity-based/explanation-based learning. Later, more detailed characteristics of the methods employed such as: divide-and-conquer/covering can be employed (see Figure 6).[10] New models will be gradually created that incorporate the specific context of using these techniques.

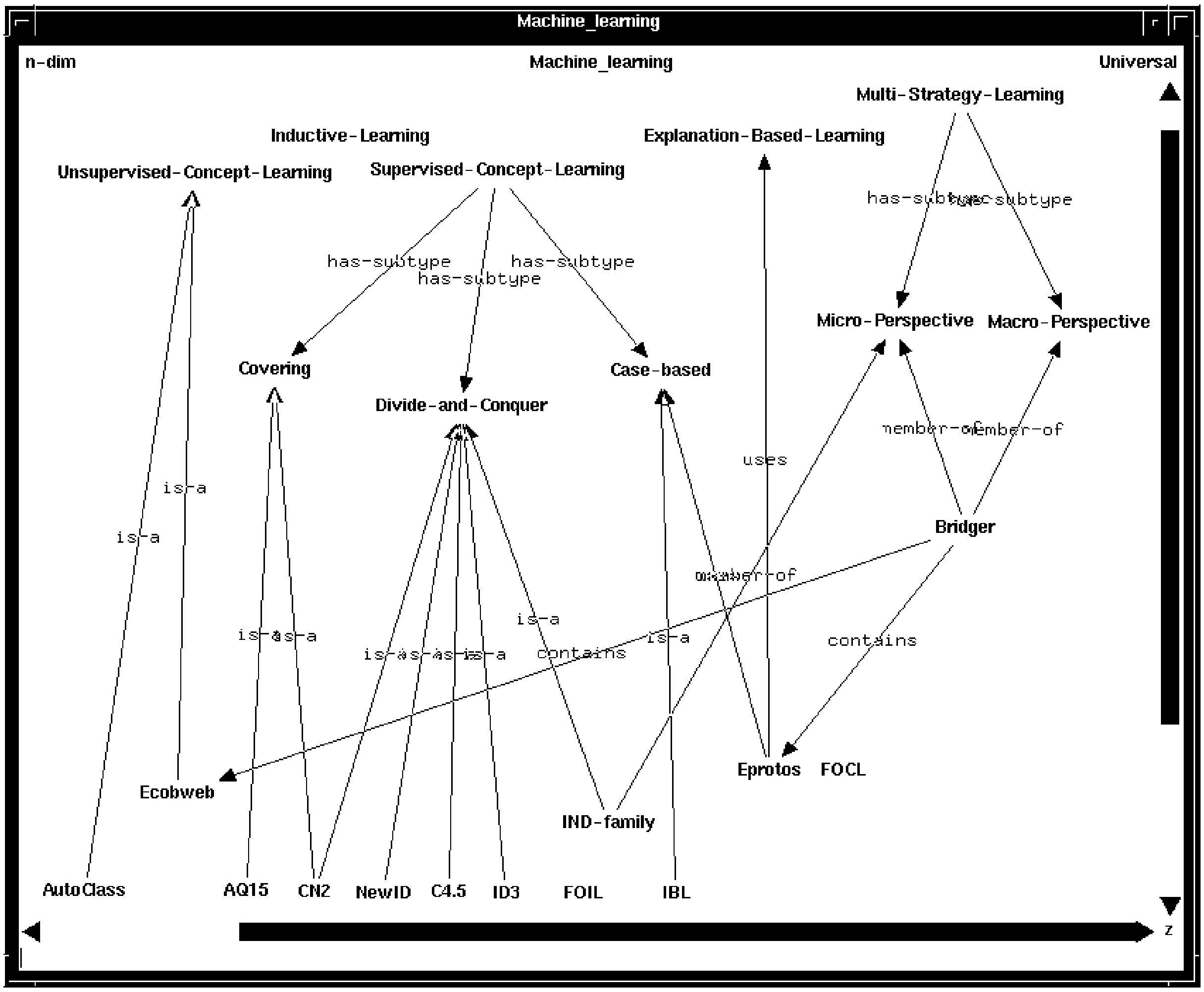


Figure 6: Classification of ML techniques

Such models can provide information for selecting learning programs in particular situations. The selection process, including its pro and con arguments, can be recorded

---

[10] Most of the objects in Figure 6, (i.e., AutoClass, Bridger, CN2, Ecobweb, Eprotos, FOCL, FOIL, IBL, IND-family, and NewId) are not merely "empty" boxes, but contain information about the programs including documentations, source code, data files and results.

by using the `rIBIS` modeling language. When experience about the use of particular ML program grows, the reasons documented in the selection model, can be reflected upon and corrected.

(5) Selecting operational parameters for the learning program, and testing them.
This activity is part of the `ML_Program` model. The problem of selecting operational parameters for learning programs can be addressed in a way similar to addressing the issue of program selection. Each program can have its own model evaluating different uses of the program in different design contexts, different operational parameters, and different feedback on performance. These evaluations can be used to select operational parameters for new learning problems.

Addressing the last issue,

(6) analyzing the results, is a culmination of addressing the first five. The results of learning programs always refer to the design context and to the choices made in the five preceding steps. It is these analyzes that close the loop and allow experience to accumulate and better inform future uses of ML programs.

4.4 Validation—learning from feedback

This section addresses the fourth learning activity in Section 3.1. We believe that design practice can be enhanced by the use of computer tools that help professionals in their task. The development of these tools must be done in collaboration with the professionals who are expected to use the tools (57; 58; 59). Our initial conjecture, based on the study of many approaches, is that improved practice can be achieved in this manner (6; 60). The participation of designers means that the tools must be flexible enough to accommodate their needs and the regulations of their organization much in the same way as *n*-dim is able to accommodate the needs of the ML techniques expected to be integrated with it. We propose that a significant part of this flexibility is achieved by providing designers with an ability to adapt their computing environment to their needs. The aforementioned flexible modeling feature of *n*-dim, provides this functionality to designers.

This practical goal, is also the key to validation of computer tools aimed at improving practice. The development of tools in collaboration with practitioners means that practitioners use the tools and provide input as to how it can be further improved. This approach provides the basis for validation of computer tools in the context of practice.

n-dim contains some facilities to collect and organize feedback from users. Modeling languages such as the `rIBIS` can be used to record and elaborate issues raised by users. Feedback in the form of text can be analyzed by the NLP procedure and subsequently, models that organize this information can be created. Improvements in *n*-dim can then refer directly to issues and feedback models, thereby maintaining their rationale and preventing recurring mistakes.

5 Summary and Future work

Starting with a critical examination of the received view of developing and using ML techniques in (computational models of) design, we arrived at describing an approach that is aimed at supporting design practice. This approach relies on several principles:

(1) The goal of the research is to advance design practice.
(2) Design is multi-faceted and heterogeneous, therefore, any technique that supports design must address these properties. Most computational techniques are limited to a simple subtask of design or one that operates on a fixed formalized model of the designed artifact.
(3) The creation of shared meaning and the management of shared memory is fundamental to the success of design. It is through this evolving body of information that successful design is possible.
(4) Flexible modeling facilities can aid in the creation, management, and use of shared memory. These facilities can be significantly enhanced by the creation of indexing mechanisms that use the evolving conceptual networks of the particular domain.

Based on these principles we propose an approach for using machine learning in design. First, research must be built on top of a flexible modeling facility to allow for impacting practice. Second, the research must address the development of a toolbox of techniques that can be used by designers depending on their particular learning needs. Third, research on flexible modeling and ML must be performed in parallel.

We described an implemented system called *n*-dim that provides the integration framework for our approach and illustrated how it can support four primary learning activities in design. We also discussed how this approach circumvents critical problems faced by current ML approaches in design, but nevertheless, creates its own research agenda.

Our proposed approach addresses the criticism raised in section 2 but creates a new set of issues that must be addressed for attaining its stated goal: improved use of ML programs in design practice. Two issues are critical to our approach: usability and validity in practice. These issues open a host of other issues to be addressed.

The issue of *usability* means that the activities described in the description of *n*-dim must be easy enough to execute so as not to discourage potential users. The activities include:

(1) *The creation of modeling languages.*
A critical part of the activities discussed in Section 4 involved creating modeling languages with the functionality required to carry out interactive learning in context. Currently, such languages are created by programming. While this is sufficient for the initial development and it also provides a basic set of languages that can be used by any *n*-dim user, it cannot serve as a mode of use for design practitioners.
The critical issue is not having the ability to create languages with fancy syntax and functionality, but the provision of facilities to create such languages without programming, that is, via modeling: the natural way of doing things in *n*-dim.
(2) *The presentation of learned information.*
Most ML programs are limited in the way they present learned information, the most common ways being decision trees, rules, or, even less comprehensible, weights in a neural network. Learning from the enormous amount of information expected to accumulate in using *n*-dim requires additional ways of inspecting information. We propose to do it by creating models on top of learned information and providing multiple ways of viewing these models. This separation of content and presentation of models is in the process of being integrated into the next version of *n*-dim.

(3) *The learning from unstructured data.*
The data presented in *n*-dim models are unstructured. While at some level, one sees objects and links with labels, these objects and links can represent arbitrary complex *n*-dim information, including text, figures, and voice bitstreams. The use of this information goes beyond concepts such as multistrategy learning. We address such learning by providing manual mappings from unstructured to structured data but, by in large, this item remains completely open to research initiatives.

The issue of *validity* means that the benefits from using ML techniques in our approach must outweigh the benefits currently available from the use of ML. Furthermore, we would like to demonstrate that it improves design practice. Therefore, addressing usability becomes mandatory for addressing the validity issue. Two of the issues related to validity are:

(1) *How do we accumulate information on learning experiences?*
Since we would like to impact practice, feedback must be collected from practitioners. Experience will be collected directly through practitioners creating models as part and parcel of doing design. This information must then be analyzed which is where ML techniques are expected to play a significant role.
One conceptual issue to be addressed is the development of methods that learn from semi-structured information stored in relational databases, where *n*-dim model structures are stored. It seems possible that this process will bootstrap itself: the more the approach is used, the easier it will become to design and integrate these techniques.
In addition, the deployment of computer programs for their evaluation and further development — technology transfer in a broad sense — presents a critical problem which we try to address through participatory design (6).

(2) *Creation of shared memory.*
Implicit in the use of ML techniques by different practitioners is the creation of a shared memory for organizations (3). We conjecture that such memory can be created by the approach outlined in this paper, and that subsequently, this evolving body of shared memory can improve design practice. In keeping with our general approach, this conjecture must itself be tested constantly to further guide the development of our approach.

Acknowledgments

This research has been supported in part by the Engineering Design Research Center, a National Science Foundation Engineering Research Center. We would like to thank Robin King for comments on an earlier draft, to Steven Meyer and Steven Fenves for allowing us access to their study protocols, and to the reviewers for their constructive comments.

References

[1] E. Subrahmanian and J. Davis, "Validation of expert systems: Two perspectives," Tech.

Rep. EDRC-05-11-87, Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, PA, 1987.

[2] M. V. Wilkes, “Artificial intelligence as the year 2000 approaches,” *Communications of the ACM*, vol. 35, no. 8, pp. 17–20, 1992.

[3] S. Konda, I. Monarch, P. Sargent, and E. Subrahmanian, “Shared memory in design: A unifying theme for research and practice,” *Research in Engineering Design*, vol. 4, no. 1, pp. 23–42, 1992.

[4] L. L. Bucciarelli, “Reflective practice in engineering design,” *Design Studies*, vol. 5, no. 3, pp. 185–190, 1984.

[5] L. L. Bucciarelli, “An ethnographic perspective on engineering design,” *Design Studies*, vol. 9, no. 3, pp. 159–168, 1988.

[6] Y. Reich, S. Konda, I. Monarch, and E. Subrahmanian, “Participation and design: An extended view,” in *PDC’92: Proceedings of the Participatory Design Conference (Cambridge, MA)* (M. J. Muller, S. Kuhn, and J. A. Meskill, eds.), (Palo Alto, CA), pp. 63–71, Computer Professionals for Social Responsibility, 1992.

[7] J. B. Strauss, *The Golden Gate Bridge.* San Francisco: Golden Gate Bridge and Highway District, 1937.

[8] S. Hales, *Analysis of The Engineering Design Process in an Industrial Context.* PhD thesis, Department of Engineering, University of Cambridge, Cambridge, UK, 1987.

[9] C. Floyd, H. Züllinghoven, R. Budde, and R. Keil-Slawik, eds., *Software Development and Reality Construction.* Berlin: Springer-Verlag, 1992.

[10] I. A. Monarch and E. Subrahmanian, “Artificial intelligence and interactive learning: A decision support exemplar,” in *Proceedings of AISIG ’90 Research Workshop on Full-Sized Knowledge Based Systems (Washington, D.C.)*, 1990.

[11] P. Piela, B. Katzenberg, and R. Mckelvey, “Integrating the user into research in engineering design systems,” *Research in Engineering Design*, vol. 3, no. 4, pp. 211–221, 1992.

[12] D. Kaminetzky, *Design and Construction Failures: Lessons From Forensic investigations.* New York, NY: McGraw-Hill, 1991.

[13] M. H. Magued, M. Bruneau, and R. B. Dryburgh, “Evolution of design standards and recorded failures of guyed towers in Canada,” *The Canadian Journal of Civil Engineering*, vol. 16, no. 5, pp. 725–732, 1989.

[14] H. Petroski, *To Engineer is Human.* New York: Vintage Books, 1992. (First Edition, 1982).

[15] P. G. Buckland, “The inherent beauty of cable-stayed bridges,” in *Esthetics In Concrete Bridge Design* (S. C. Watson and M. K. Hurd, eds.), (Detroit, MI), pp. 233–246, American Concrete Institute, 1990.

[16] D. Haussler, “Quantifying inductive bias: AI learning algorithms and Valiant’s learning framework,” *Artificial Intelligence*, vol. 36, no. 2, pp. 177–221, 1988.

[17] P. E. Utgoff, *Machine Learning of Inductive Bias.* Boston, MA: Kluwer Academic Publishers, 1986.

[18] L. J. Leifer, "Instrumenting the design process," in *Proceedings of ICED-91 (Zurich)*, 1991.

[19] E. Subrahmanian, A. W. Westerberg, and G. Podnar, "Towards a shared information environment for engineering design," in *Computer-Aided Cooperative Product Development, MIT-JSME Workshop (Nov., 1989)* (D. Sriram, R. Logcher, and S. Hukuda, eds.), (Berlin), Springer-Verlag, 1991.

[20] Y. Reich, "The development of BRIDGER: A methodological study of research on the use of machine learning in design," *Artificial Intelligence in Engineering*, vol. 8, no. 3, pp. 217–231, 1993. Special issue on Machine Learning in Design.

[21] J. Mingers, "An empirical comparison of selection measures for decision-tree induction," *Machine Learning*, vol. 3, no. 4, pp. 319–342, 1989.

[22] J. Mingers, "An empirical comparison of pruning methods for decision-tree induction," *Machine Learning*, vol. 4, no. 2, pp. 227–243, 1989.

[23] Y. Reich, "Measuring the value of knowledge," *International Journal of Human-Computer Studies*, 1995. (in press).

[24] J. Doyle, "On rationality and learning," Tech. Rep. CMU-CS-88-122, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, 1988.

[25] T. M. Mitchell, "The need for biases in learning generalizations," Tech. Rep. CBM-TR-117, Rutgers University, New Brunswick, NJ, 1980.

[26] W. Buntine and D. Stirling, "Interactive induction," Tech. Rep. TIRM-88-030, The Turing Institute, Glasgow, 1988.

[27] A. Newell, "The knowledge level," *Artificial Intelligence*, vol. 18, no. 1, pp. 87–127, 1982.

[28] T. Arciszewski, M. Mustafa, and W. Ziarko, "A methodology of design knowledge acquisition for use in learning expert systems," *International Journal of Man-Machine Studies*, vol. 27, no. 1, pp. 23–32, 1987.

[29] Y. Reich and S. J. Fenves, "The formation and use of abstract concepts in design," in *Concept Formation: Knowledge and Experience in Unsupervised Learning* (D. H. J. Fisher, M. J. Pazzani, and P. Langley, eds.), (Los Altos, CA), pp. 323–353, Morgan Kaufmann, 1991.

[30] S. C.-Y. Lu and K. Chen, "A machine learning approach to the automatic synthesis of mechanistic knowledge for engineering decision-making," *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing*, vol. 1, no. 2, pp. 109–118, 1987.

[31] Y. Reich and S. J. Fenves, "Inductive learning of synthesis knowledge," *International Journal of Expert Systems: Research and Applications*, vol. 5, no. 4, pp. 275–297, 1992.

[32] Y. Reich, "Design knowledge acquisition: Task analysis and a partial implementation," *Knowledge Acquisition*, vol. 3, no. 3, pp. 237–254, 1991.

[33] R. S. Michalski and G. Tecuci, eds., *Proceedings of the First International Workshop on Multistrategy Learning.* Fairfax, VA: Center for Artificial Intelligence, George Mason University, 1991.

[34] S. Salzberg, "Heuristics for inductive learning," in *Proceedings of The Ninth International Joint Conference on Artificial Intelligence*, (Los Angeles, CA), pp. 603–609, Morgan Kaufmann, 1985.

[35] R. E. Stepp, B. L. Whitehall, and L. B. Holder, "Towards intelligent machine learning algorithms," in *Proceedings of the 8th European Conference on Artificial Intelligence*, (Munich, W. Germany), pp. 333–338, Pitman, 1988.

[36] Y. Reich, "Macro and micro perspectives of multistrategy learning," in *Machine Learning: A Multistrategy Approach, Vol. IV* (R. S. Michalski and G. Tecuci, eds.), (San Francisco, CA), pp. 379–401, Morgan Kaufmann, 1994.

[37] Y. Reich, *Building and Improving Design Systems: A Machine Learning Approach.* PhD thesis, Department of Civil Engineering, Carnegie Mellon University, Pittsburgh, PA, 1991. (Available as Technical Report EDRC 02-16-91).

[38] E. S. Ferguson, *Engineering and the Mind's Eye.* Cambridge, MA: MIT Press, 1992.

[39] A. D. Kerr and R. B. Pipes, "Why we need hands-on engineering education," *Technology Review*, vol. (October), pp. 36–42, 1987.

[40] J.-L. Le Moigne, "The paradoxes of the contemporary engineer," *European Journal of Engineering Education*, vol. 6, pp. 105–115, 1981.

[41] E. Subrahmanian, S. L. Konda, S. N. Levy, Y. Reich, and A. W. Westerberg, "Modeling and analysis in design," in *Proceedings of the AID'92 Workshop on Preliminary Stages of Engineering Analysis and Modeling*, 1992.

[42] N. Cross and M. Nathenson, "Design methods and learning methods," in *Design: Science: Method, Proceedings of the 1980 Design Research Society Conference* (R. Jaques and J. A. Powell, eds.), pp. 281–296, Guildford, England: Westbury House, 1981.

[43] C. M. Eastman, "On the analysis of intuitive design processes," in *Emerging Methods in Environmental Design and Planning* (G. T. Moore, ed.), pp. 21–37, Cambridge, MA: MIT Press, 1970.

[44] M. T. H. Chi, R. Glaser, and M. J. Farr, eds., *The Nature of Expertise.* Hillsdale, N.J.: Lawrence Erlbaum Associates, 1988.

[45] B. R. Gaines, "Positive feedback processes underlying the formation of expertise," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 18, no. 6, pp. 1016–1020, 1988.

[46] J. A. Bermingham, "Why product development at sony is driven by the engineering and manufacturing groups rather than marketing." Lecture given at The Graduate School of Industrial Administration, 24th September, 1991.

[47] Y. Reich, R. Coyne, A. Modi, D. Steier, and E. Subrahmanian, "Learning in design: An EDRC (US) perspective," in *Artificial Intelligence in Design'91, Proceedings of The First International Conference on Artificial Intelligence in Design, Edinburgh, UK* (J. Gero, ed.), (Oxford, UK), pp. 303–321, Butterworths, 1991.

[48] L. Hyafil and R. L. Rivest, "Constructing optimal binary decision trees is NP-complete," *Information Processing Letters*, vol. 5, no. 1, pp. 15–17, 1976.

[49] S. Levy, E. Subrahmanian, S. L. Konda, R. F. Coyne, A. W. Westerberg, and Y. Reich, "An overview of the n-dim environment," Tech. Rep. EDRC-05-65-93, Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, PA, 1993.

[50] D. Ungar and R. B. Smith, "SELF: the power of simplicity," *LISP and Symbolic Computation*, vol. 4, no. 3, pp. 187–205, 1991.

[51] E. Subrahmanian, S. L. Konda, S. N. Levy, Y. Reich, A. W. Westerberg, and I. A. Monarch, "Equations aren't enough: Informal modeling in design," *Artificial Intelligence in Engineering Design, Analysis, and Manufacturing*, vol. 7, no. 4, pp. 257–274, 1993.

[52] S. Meyer and S. J. Fenves, "Structural design of tall buildings, knowledge acquisition study report," Tech. Rep. EDRC 12-58-93, Engineering Design Research Center, Pittsburgh, PA, 1993.

[53] J. Conklin and M. L. Begeman, "gIBIS: A hypertext tool for exploratory policy discussion," *ACM Transaction on Office Information Systems*, vol. 6, no. 4, pp. 303–331, 1988.

[54] K. Morik, "Balanced cooperative modeling," in *Proceedings of the First International Workshop on Multistrategy Learning* (R. S. Michalski and G. Tecuci, eds.), (Fairfax, VA), pp. 65–80, Center for Artificial Intelligence, George Mason University, 1991.

[55] E. Subrahmanian, S. L. Konda, S. N. Levy, I. A. Monarch, Y. Reich, and A. W. Westerberg, "Computational support for shared memory in design," in *Automation-Based Creative Design: Current Issues in Computers & Architecture* (A. Tzonis and I. White, eds.), Elsevier Science Publishers, 1993.

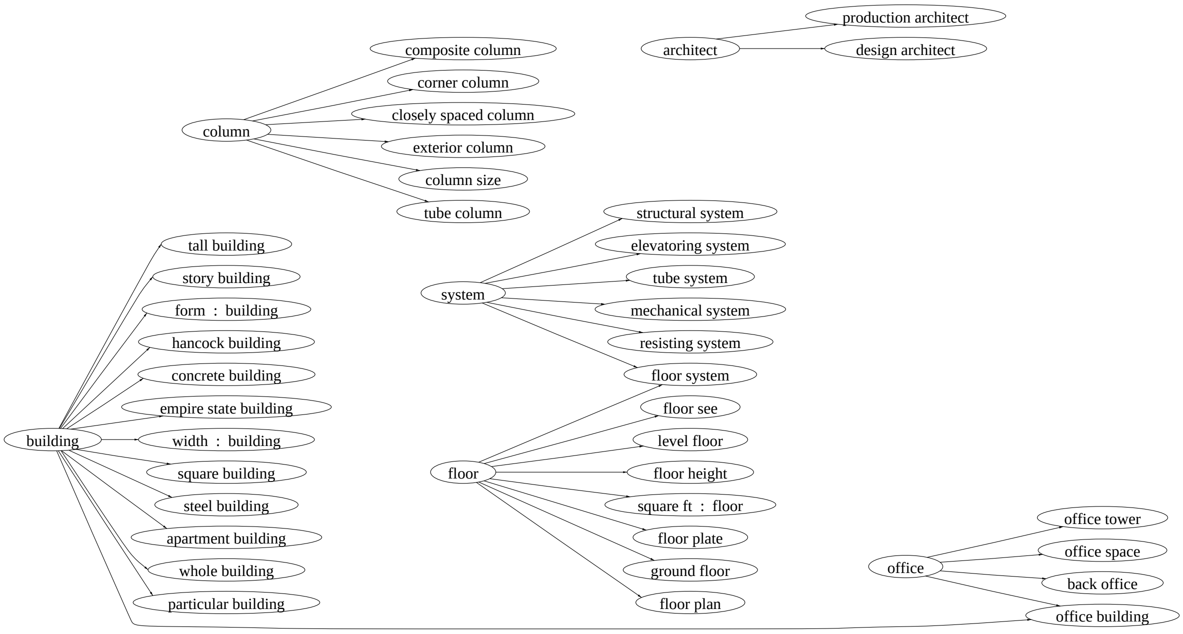
[56] B. R. Gaines and M. L. G. Shaw, "Comparing the conceptual systems of experts," in *Proceedings of The Eleventh International Joint Conference on Artificial Intelligence*, (Detroit, MI), pp. 633–638, Morgan Kaufmann, 1989.

[57] H. Barki and J. Hartwick, "Rethinking the concept of user involvement," *MIS Quarterly*, vol. 13, no. 1, pp. 53–63, 1989.

[58] A.-M. K. Baronas and M. R. Louis, "Restoring a sense of control during implementation: How user involvement leads to system acceptance," *MIS Quarterly*, vol. 12, no. 1, pp. 111–123, 1988.

[59] P. Tait and I. Vessey, "The effect of user involvement on system success: A contingency approach," *MIS Quarterly*, vol. 12, no. 1, pp. 91–108, 1988.

[60] G. Salaway, "An organizational learning approach to information systems development," *MIS Quarterly*, vol. 11, no. 2, pp. 245–264, 1987.


production architect
architect
design architect
composite column
corner column
column
closely spaced column
exterior column
column size
tube column
structural system
elevatoring system
system
tube system
mechanical system
resisting system
floor system
tall building
story building
form : building
hancock building
concrete building
empire state building
building
width : building
square building
steel building
apartment building
whole building
particular building
floor see
level floor
floor
floor height
square ft : floor
floor plate
ground floor
floor plan
office tower
office space
office
back office
office building