The development of BRIDGER: A methodological study of research on the use of machine learning in design

Yoram Reich
Engineering Design Research Center
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213, USA

Artificial Intelligence in Engineering, 8(3):217-231
Special issue on Machine Learning in Design

Keywords: research methodology, generic learning tasks, multistrategy learning, concept formation, bridge design, scientific method, design knowledge acquisition, design rationale.

Running head: Research methodology of learning in design

Abstract: Research is a design activity whose decisions involve the ways in which research is carried out and its results interpret. These activities comprise what is referred to as *research methodology*. This paper brings the concepts of capturing design rationale and machine learning to bear on the design of research itself. Therefore, design decisions concerning research must be recorded to allow for understanding feedback and updating of research strategies. In addition, successes as well as failures of research decisions must be reported to facilitate learning about research. This requires a shift in the way research is carried out and reported. This paper illustrates this shift in the context of a specific project on machine learning in design.

1 Introduction

There are two senses of science according to McMullin (McMullin, 1987). In the first sense, denoted by S_1, science is a collection of hypotheses, theories, observations, data, etc. that are the end products of a long process. The extent to which details relating to these end products are discussed is to allow fellow scientists to evaluate and replicate results. In this sense, science does not contain failures such as fruitless research paths pursued by scientists, albeit some argue (e.g., Popper(Popper, 1965)) that science is all about conjectures and refutations.

In the second sense, denoted by S_2, science contains as many more details as possible about scientists' activities. In effect, science is a historical description of research. Science is broader and vaguer than S_1, and sometimes may even be embarrassing. Decisions about research are made not necessarily based on profound reasons but may be responses to circumstances such as proximity to a required source. Research decisions embedded in the actual exercise of research have significant impact on scientific progress. According to Kuhn (Kuhn, 1962) and Feyerabend (Feyerabend, 1975), it is what scientists do that accounts for scientific progress, rather than, the use of any principled, universal methodology (see also (Pickering, 1992) for an elaborate philosophical discussion on this topic, (Clancey, 1986; Pople, 1985) for two rare examples of historical descriptions of research in AI, and (DeMillo et al, 1979) for a demonstration of the same ideas in mathematics).

In arguing for adopting S_2, I will apply terms borrowed from two topics in design research: the capture of design rationale (CDR) and machine learning (ML), to the research activities researchers pursue. There are three reasons for adopting S_2: methodological, moral, and substantive.

Methodological reason. Research is about the identification of needs, creation of ideas, their formalization, testing, dissemination, and refinement. *Research is a design activity.* As such, research can benefit from ideas developed in design research such as CDR. The rational of the design of a research activity contains precisely what S_2 includes: all the contextual information about the research activity. Moreover, as argued by researchers on CDR, it is this rationale that allows other researchers to "replay" research, thereby testing the foundations of a scientific work and extend it while avoiding previous pitfalls. It is in this sense, that as a community, researchers can build principles, tools, and understanding about a particular scientific subject matter.

This understanding directly relates to the second concept of design research mentioned and to the focus of this special issue: machine learning. In studying the effectiveness of different ML techniques in design, we understand the importance of accurate information that allow learning to operate effectively. Similarly, if we expect that learning of concepts will guide meaningful scientific progress through the accumulation of knowledge, we must detail all aspects involved in any research activity, including, of course, "positive" (i.e., successful) and "negative" (i.e., failure) examples of research.

In summary, principles from both CDR and ML demand that we adopt S_2, thus leading to a better methodology for design research.

Moral reason. In design research, we researchers invent many concepts and expect others to use them in practice. It is a *moral* obligation to use the concepts we develop in our own work where applicable. The capture of design rational of research and the reporting of accurate data for learning about research projects must therefore take place to set example for prospective users of these concepts in design.

Substantive reason. If $\mathbf{S}_2$ is practiced, research results can come closer to practice. The information gathered about design choices in research can inform the selection of the research products in practice. The accumulation of knowledge through learning is particularly relevant to one of the end products of this research: a method for selecting ML techniques for executing specific learning tasks called $\mathbf{M}^2$**LTD**. The *substantive* reason for choosing $\mathbf{S}_2$ in this research project is that $\mathbf{M}^2$**LTD** relies on knowledge gathered from detailed feedback from using it to assign ML techniques to learning problems. This feedback must include the reasons for selecting particular ML techniques and their (un)successful operation. This study shows that in $\mathbf{S}_1$, often the true reasons are hidden behind careful re-constructions that may render the use of $\mathbf{M}^2$**LTD** fruitless in practice.

After detailing the methodological, moral, and substantive reasons for using $\mathbf{S}_2$, it is time to demonstrate the first and last reasons. Therefore, this study focuses, on the *processes* involved in the research reported (methodological reason) including its end products (substantive reason).

This paper discusses the history of a research project including: the issues and circumstances involved in the conception, design, testing, and redesign of **Bridger**, a system that may assist in the design of cable-stayed bridges. This paper also discusses some of the fruitless paths explored. While the latter rarely appear in publications, being a testimony of partial failures, they are, in fact, crucial facilitators of progress and learning.

This paper also details the products of the research and their partial evaluation; it does so following $\mathbf{S}_1$, as it would be reported according to practiced conventions. The description of the end products has two methodological functions. First, it demonstrates what I view as a minimal evaluation of an end product. Second, a comparison of this description with the historical description of the project (i.e., $\mathbf{S}_2$) shows that $\mathbf{S}_1$ hides important data that is inaccessible by fellow researchers for future reference. This comparison provides a data point in favor of choosing $\mathbf{S}_2$.

One note before starting the journey. This paper does not present the design rationale as coded *while* doing the research, nor is it reported by an unbiased researcher. Rather, the design rationale reported is a subjective rational reconstruction of the events that took place. Furthermore, the description does not escape the goal of prescribing how decisions should have been made or should be made in the future.

They were two reasons for not recording the design rationale during the course of the project reported. The one that will probably be used more often by researchers states that the lack of tools appropriate for recording and organizing design decisions prevented serious effort of coding. This reason, while stating a correct fact: the lack of tools, is inaccurate. The second and true reason is that this rational was not considered important at the beginning of the project. Since then, I have changed my views to arguing that the process of research is no less significant than its end products.

Plan of the paper. The remainder of this paper is organized as follows. Section 2 provides a detailed description of two end products of this research as reported following S_1. Section 3 discusses another end product of the research that triggered much of the shift from S_1 to S_2. Section 4 puts sections 2 and 3 into a historical perspective. Section 5 discusses the methodological lessons extracted from the research project and Section 6 summarizes the contribution of this paper.

2 S_1: The end products of the research

This section provides a description of ECOBWEB and BRIDGER as it would appear following the perspective of S_1. This description is not comprehensive. It is meant only to provide details sufficient to contrast it with S_2 and to demonstrate several methodological aspects in carrying research in S_1 as well.

2.1 ECOBWEB

ECOBWEB is a system for learning synthesis knowledge for a type of routine design problems. This section presents a rational reconstruction of the assumptions and the implementation and testing of ECOBWEB's mechanisms.

2.1.1 Assumptions about design

This research made the following assumptions about design.

> ASSUMPTION 1 (**Artifact representation**): Artifacts and their specifications are described by (finite) lists of property-value pairs.

A property-value pair may specify the length of a bridge or its particular type (e.g., arch or suspension). This assumption restricts the scope of designs to those with fixed structures. Therefore, the design of artifacts that are described via graphs, such as layouts, cannot be supported under this assumption. If the restriction on the finite number of properties is removed, the representation becomes general, but of course, unreasonable to implement computationally.

> ASSUMPTION 2 (**Structure of design knowledge**): Knowledge about types of designs is represented as a hierarchical classification tree.

This assumption does not impose any restrictions on the potential application.[1] In Section 2.1.3 we show that such structure does not restrict design to the selection from existing designs; rather, it can give rise to different processes depending on *how* it is used.

> ASSUMPTION 3 (**Quality of design knowledge**): The quality[2] of design knowledge is determined by the quality of the hierarchical classification tree representing it. The quality of a hierarchical classification tree is determined based on whether it classifies (dis)similar designs (dis)similarly. The terms "similar" and "classify" are left vague.

The knowledge quality assumption approximates the more desired quality measure stating that a classification is 'good' if the description of a design can be guessed with high accuracy, given that it belongs to a specific class, or even the better measure, that a classification is 'good' if it results in good design performance.

> ASSUMPTION 4 (**Design process**): Design is a (*direct*) mapping from a specification to an artifact description.

Defining design as a mapping is very general unless the nature of the mapping is restricted to be *direct* as it is in the above assumption. This assumption almost excludes the use of explicit knowledge in design. On the other hand, information can be compiled into implicit knowledge embedded in the mapping.

> ASSUMPTION 5 (**Nature of design knowledge**): The structure of design knowledge is static. It is altered incrementally if new design knowledge warrant this operation.

This assumption states that knowledge is assumed to be correct, unless additional information becomes available. In this research, the nature of the additional information is restricted to the description of new designs or procedures that evaluate the knowledge quality. Another consequence of this assumption is that knowledge is built incrementally and continuously since it is always incomplete.

Assumption 5 requires that a system supporting design be able to learn and modify its knowledge incrementally. Assumption 2 specifies the structure of the knowledge that needs to be generated and Assumption 3 determines how this knowledge should be evaluated and therefore hints at how it may be built. Assumption 1 restricts the scope of artifacts discussed, thereby allowing available learning techniques to be used for the knowledge generation.

All these assumptions lead to the selection of COBWEB (Fisher, 1987) as a potential tool. Since COBWEB has several limitations in the context of design (Reich and Fenves, 1991), a new system was developed, called ECOBWEB, that supports the five assumptions.

[1] Note that a hierarchical classification tree is not a decision tree. One significant difference is that a decision tree specifies a process: an ordered sequence of design choices, whereas a hierarchical classification specifies how concepts are organized in a class/sub-class structure.

[2] In this paper, we are using the terms design knowledge and its quality rather loosely; elsewhere, we try to address these issues in more detail while pointing at some of the difficulties in being precise about them (Reich, 1995).

Reflections. As a rationale reconstruction, it is evident that the above assumptions were geared towards the solution proposed. This will probably be the case for most re-constructions of assumptions. Nevertheless, such acknowledgment is outside the scope of $\mathbf{S}_1$, which only requires that the implementation is consistent with the assumptions. While this criticism downplays the significance of the assumptions, they still have a role in $\mathbf{S}_1$. The listing of the assumptions states explicitly the foundations of the implementation, thus providing a concrete basis for a dialogue, rather than discussing over lengthy, detailed text.[3] In any case, it is the responsibility of a study to show that the implementation follows the assumptions and that the study validates them, even if partially.

The next two subsections describe the realization of these assumptions in **Ecobweb** learning and synthesis processes.[4]

2.1.2 Ecobweb's learning algorithm

Ecobweb learns design knowledge incrementally (based on Assumption 5). It creates a classification hierarchy (based on Assumption 2) from design examples represented by lists of property-value pairs (based on Assumption 1). **Ecobweb** uses several operators to build the hierarchy. Its aim is to improve the knowledge quality continually (based on Assumption 3). Therefore, learning operators are selected to maximize the knowledge quality. A statistical function, called *category utility*, approximates knowledge quality by evaluating a classification of a set of designs into mutually-exclusive classes.

When a new design is introduced, **Ecobweb** tries to accommodate it into the existing classification hierarchy starting at the root by performing one of five operators (see (Fisher, 1987) for a detailed description of these operators) that maximizes the value of the category utility function. The process terminates when the new design has reached a leaf node.

2.1.3 Ecobweb's synthesis process (based on Assumption 4)

Ecobweb's synthesis methods can be described along two dimensions: the refinement dimension which can be *extensional* or *intensional*; and the generation dimension which can be *case-based* or *prototype-based.* Figure 1 illustrates these dimensions. In the extensional approach, refinement classifies a new specification with a new subclass starting from the top class (class 1 in Figure 1) until the process terminates at class 3 when the property values of the specification have been matched by characteristic property values in the classes traversed (classes 1, 2, and 3 in Figure 1). Intuitively, characteristic property values of a class are those property values that are very common in the class and rarely appear in the other classes of the same level. In

[3] This can be contrasted by the lengthy, detailed description in S_2. It is almost certain that S_2 cannot be compressed to a list of assumptions although its style may take different shapes after some experience is accumulated.

[4] Many properties of Ecobweb discussed here are inherited from its predecessor Cobweb. Since the purpose of this paper is not the evaluation of Ecobweb contribution but the methodological aspects of the research, we do not emphasize the distinctions between the two learning systems. Such evaluations appear elsewhere (Reich, 1991b; Reich and Fenves, 1991; Reich and Fenves, 1992).

the intensional approach, while classifying the new specification, characteristic property-values of the classes traversed are assigned to the new design. Therefore, at the time the refinement terminates, the new specification would be augmented with several design description properties.

In the generation stage, the case-based approach views a class as representing the set of all its leaves. Therefore, leaves 4, 5, and 6 are selected as candidates in conjunction with the extensional refinement approach, or are used to complete the properties missing in the new design description in conjunction with the intensional refinement approach. The prototype-based generation approach takes the current class (class 3 in the example) as the prototype whose description is composed of *all* the property values of its leaf nodes with their associated frequencies. In conjunction with the extensional refinement, a sequence of candidates can be generated from the prototype, starting with the best candidate having property values that are the most frequent. In conjunction with the intensional refinement, the property values assigned in the refinement are maintained by all the candidates.

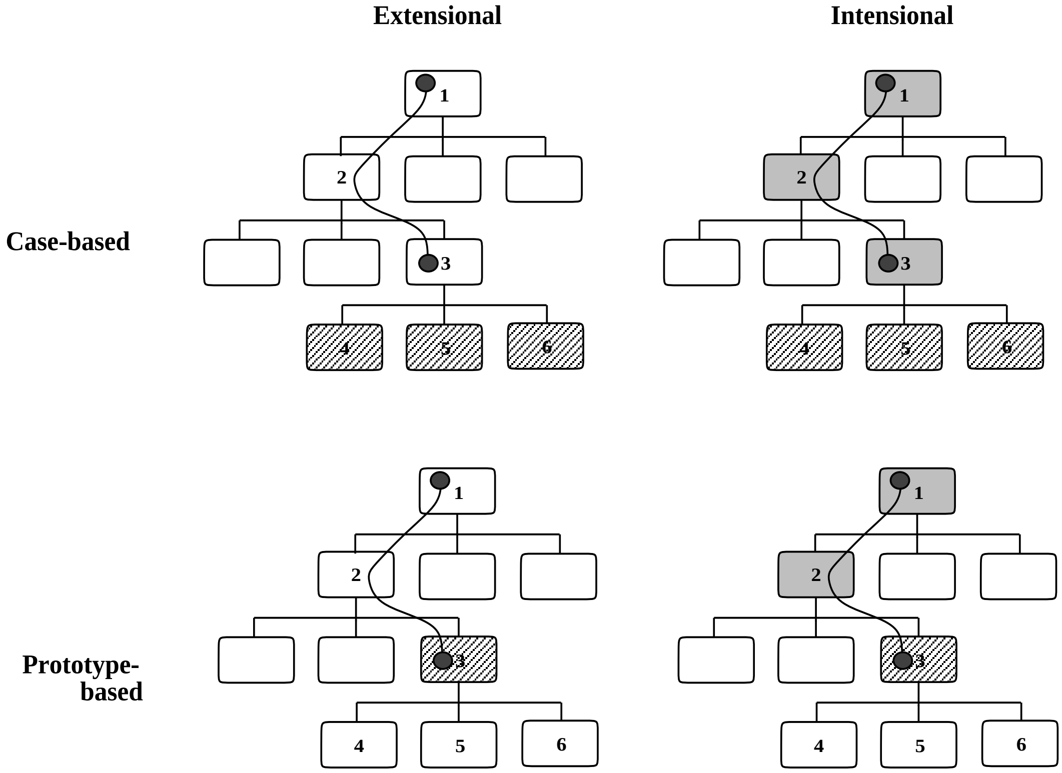

Figure 1: Synthesis methods

There are two observations emerging from experiments with ECOBWEB. First, ECOBWEB generates several alternatives that not always satisfy all the requirements and it generates alternatives that *did not exist* in the original set of potential designs (when using any but the extensional/case-based approach). Second, in deep hierarchies generated by many examples, it is observed that the path traversed by the guidance of the category utility function can be *interpret* as a progressive matching of the specifications or even as a design derivation. It is important to acknowledge that this observation does not approximate in any way the explicit intent and domain knowledge on the order in which design derivations are to proceed. Such concerns may be supported when domain knowledge is incorporated in the learning or synthesis processes. Such knowledge will alter Assumption 4.

2.1.4 Evaluation

ECOBWEB satisfies the assumptions laid out in section 2.1.1; therefore, its evaluation is their validation. In any research, it is the systematic testing of the consequences of assumptions that establishes confidence about them until some evidence arises that renders them inappropriate. It is unfortunate that many research projects in ML in design do not exercise serious testing, thereby leaving their assumptions largely unsupported.

This section illustrates one type of quantitative evaluation that was performed with ECOBWEB. Additional quantitative and qualitative evaluations are detailed elsewhere (Reich, 1991b; Reich and Fenves, 1991; Reich and Fenves, 1992; Reich, 1995). In general, ECOBWEB demonstrated performance comparable to and sometimes better than other learning programs in classification tasks. But more important, it demonstrated good performance in design domains. To illustrate this performance, we review some results of ECOBWEB's evaluation in the domain of cable-stayed bridge design.

Table 1 illustrates the performance of ECOBWEB by providing the mean values of one performance measure generated from 48 test problems. The measure, called *scaling*, calculates how close was the length of a bridge synthesized by the *case-based/extensional* method to the length required by the new specification, i.e., *scaling* equals the ratio between the new length requirement and the length of the retrieved bridge; the closer the value to 1, the better is the match between the new requirement and the retrieved design. The scaling measure was selected since it has a significant influence on the design. In the table, the measure is provided for four knowledge hierarchies generated by learning: K_1, K_2, K_3, and K_4. Hierarchy K_1 was generated from the original set of 96 bridge examples created in a study on bridge design. Hierarchy K_2 was generated from the 96 examples after their analysis and redesign; therefore it contained higher quality examples. Hierarchies K_3 and K_4 were generated from 144 (48 in addition to the 96 in K_2) and 192 (48 in addition to the 148 in K_3) good quality examples, respectively. Since K_1 was built from lower-quality examples it did not participate in the statistical analyzes performed.[5] From looking at the table, one may be inclined to infer that indeed the more examples ECOBWEB learns, the better it performs. Such statement is not yet warranted. One has to demonstrate that the results cannot be a result of chance, and moreover, one must seek for (competing) patterns in the behavior to uncover some insight about it.

Two competing models, beside the chance hypothesis, were hypothesized for explaining the change in the scaling measure: (1) the scaling measure depends linearly on the number of examples learned, or (2) the logarithm of scaling depends on the logarithm of the number of examples. The former is rather *ad hoc*, while the latter relies on the power law of practice so

[5] See also (Reich, 1995) for an elaborate explanation based on measurement theory.

prominent in human learning (Newell and Rosenbloom, 1981).

Table 1: Scaling statistics of candidates: Average scaling value from 48 test cases

Knowledge	Scaling
K_1	3.07
K_2	2.15
K_3	2.09
K_4	1.32

A General Linear Model (GLM) procedure with a MANOVA[6] (Hays, 1988) analysis was performed to assess the differences between the scaling values observed according to the two models. In both models, the scaling values satisfied: $K_2, K_3 >_{0.01} K_4$; where the $>_{0.01}$ indicates that K_2 and K_3 are greater than K_4 with statistical significance at the $p < 0.01$ level and that the difference between K_2 and K_3 was not statistically significant. Note that the second model was better than the first, although the statistical significance of this statement was not assessed. In conclusion, *the more knowledge* ECOBWEB *has, the more relevant are the synthesized candidates to a new problem.*[7] In addition, ECOBWEB learning behavior seems to obey the power law of practice.

The tests performed support the claim that if a design domain can be approximated with the assumptions detailed in Section 2.1.1, then, its knowledge can be learned and used by the kind of techniques implemented in ECOBWEB.

2.2 BRIDGER

BRIDGER is a system built for assisting in the conceptual design of cable-stayed bridges. This section presents a rational reconstruction of the assumptions underlying the design of BRIDGER and its implementation and testing.

2.2.1 Assumptions

Since the major component of BRIDGER is ECOBWEB, the following remark holds.

REMARK: BRIDGER inherits the assumptions of ECOBWEB discussed in Section 2.1.1

ASSUMPTION 6 **(Structure of design process)**: Design is a sequence of five tasks, problem analysis, synthesis, analysis, redesign, and evaluation, executed sequentially with one feedback loop (Reich, 1991c).

[6] A MANOVA was used instead of an ANOVA analysis because (1) there were two dependent variables analyzed at the same time: the scaling measure and the quality measure (see Table 2); and (2) the independent variable had more than two values.

[7] Actually, these candidates are existing designs since the case-based/extensional method was used. Additional tests must be executed to generalize to the others synthesis strategies ECOBWEB has.

This assumption admits Assumption 4 as referring to the synthesis, rather than the whole design process.

2.2.2 Bridger's architecture

Bridger's architecture is based on assumption 6. Bridger's main emphasis is on mechanisms for synthesis knowledge acquisition and therefore, the complete architecture is intended to support this task. Figure 2 shows an overall view of Bridger's architecture which consists of two main sub-systems: synthesis and redesign.

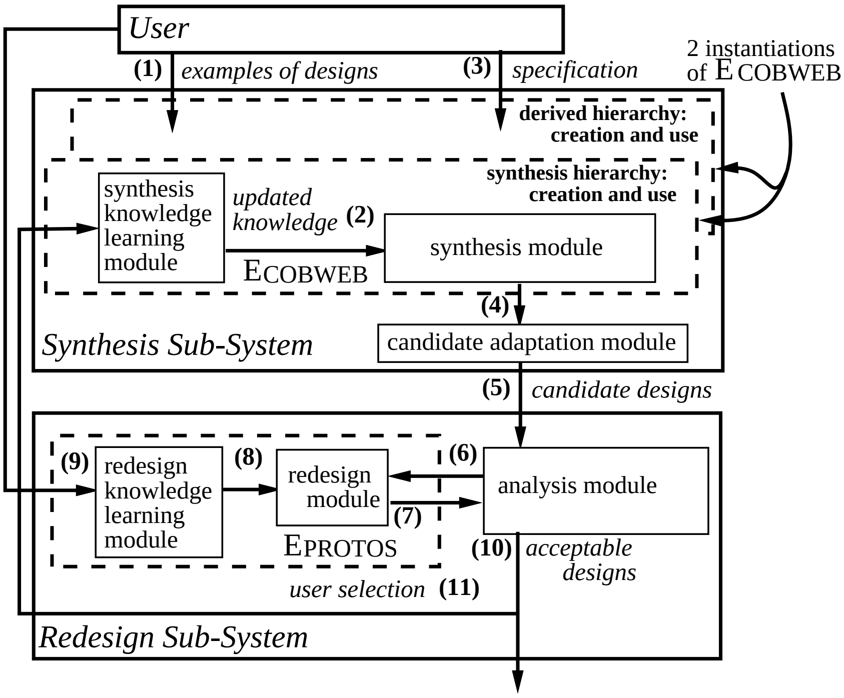

Figure 2: Overview of Bridger's architecture

The synthesis sub-system is responsible for synthesizing several candidates from a given specification (branch 5). Synthesis knowledge is generated by learning from existing designs (branch 1) and from successful design examples that are selected by the user (branch 11). The synthesis module contains two instantiations of Ecobweb.

The first instantiation creates a classification hierarchy (branch 2) from the original bridge examples. This hierarchy is subsequently used to synthesize bridges (branch 4) as discussed in Section 2.1. When dealing with specifications that are expressed by real numbers, rarely will a synthesized design match the specification. This problem can be remedied by performing various scaling operations to fit the synthesized design to the specification. In order for performing sensible scaling, relevant scaling values are retrieved from a *second instantiation* of Ecobweb. This instantiation builds a classification hierarchy by learning from proportions of various components of bridges. The process of retrieving scaling values and using them to modify the design is called *adaptation* (see (Reich, 1993; Reich, 1991b) for additional details). Since the creation of candidate designs relies on heuristic synthesis processes, candidate designs are usually inadequate in some aspects even after the adaptation process. The redesign sub-system resolves this problem.

Candidate designs are transferred to a module that analyzes them and submits them to a redesign module, if necessary (branch 6). The redesign module is responsible for modifying designs after their analysis (branch 7). On receiving the analysis results, this module retrieves the best design modification for the bridge. The user can override the redesign modifications and supply explanations that enhance redesign knowledge (branch 9). The results of the redesign sub-system are acceptable designs (branch 10). The designer evaluates the results and can submit a subset of them to the synthesis sub-system for further training (branch 11) and subsequent modification of the synthesis knowledge (branch 2).

2.2.3 Evaluation

BRIDGER was partially evaluated qualitatively and quantitatively (Reich, 1991b; Reich, 1995). This section details the quantitative evaluation only. The quantitative evaluation was mainly concerned with the ability to acquire synthesis knowledge. The first part of the evaluation was demonstrated by ECOBWEB in Section 2.1.4 and is not discussed further. The adaptation part was tested by calculating the *quality* of bridges generated by the synthesis module of BRIDGER, where the quality of a bridge is a function of the weighted summation of the squares of the functional constraints that a candidate bridge violates; lower values mean higher quality. These violations were calculated by the analysis module.

The redesign part was not assessed quantitatively. At the time the project ended, it was ready for demonstration purposes only. Therefore, no conclusion can be drawn about the appropriateness of the approach to learning redesign knowledge, or its lack thereof. This is an unfortunate but correct statement. It is based on the premise that a simple demonstration is not a test, although this has become common practice in AI research in design.

Table 2: Quality statistics of candidates: Average quality measured from 48 test cases

Knowledge	Quality
K_1	278.36
K_2	50.19
K_3	2.89
K_4	1.20

Table 2 illustrates the performance of BRIDGER by providing the mean values of the quality measure generated from 48 test problems. The same models and statistical test that were used to test the scaling measure in Section 2.1.4 were used again: (1) a linear model, and (2) a power law model. According to the first model, the quality values satisfy: $K_2 >_{0.01} K_3, K_4$; but according to the second: $K_2 >_{0.01} K_3 >_{0.01} K_4$. The second model was better in term of explaining the data, but no statistical test was performed to differentiate between the two models. Both results say that *the more knowledge* BRIDGER *has, the better the quality of* *candidates it generates.* The second model describes the behavior more conclusively and more favorably. Furthermore, the reduction of quality values when additional examples are learned shows that BRIDGER converges to good design competence in absolute terms. Future evaluations may include the assessment of the redesign and the evaluation of the complete system.

The testing of data through the use of two models changed the results slightly. One could state that this was a redundant effort. In contrast, we maintain that the testing of alternative interpretations of data is a prerequisite for getting plausible results. Often, in AI work, no significant testing is performed, and when one is performed, it often consists of testing a single model.

Reflections. There are additional avenues for conducting evaluations beside testing the performance of the computational model. In particular, the relevance of the work to the practice of engineering, a topic that is almost never assessed in design research, including the study reported in this paper (see (Piela et al, 1992) for a good positive example), can be assessed by deploying a program in a workplace and analyzing its usability. Such studies must be informed by methodologies developed in the human and social sciences to be considered adequate.

3 M^{2}LTD: The road from S_1 to S_2

The third product of this research is M^{2}LTD, a method for selecting ML techniques to perform learning tasks (Reich, 1994). M^{2}LTD consists of *knowledge* describing the issues and tradeoffs that inform the selection and a sequence of 5 steps that uses this knowledge (see Figure 3).

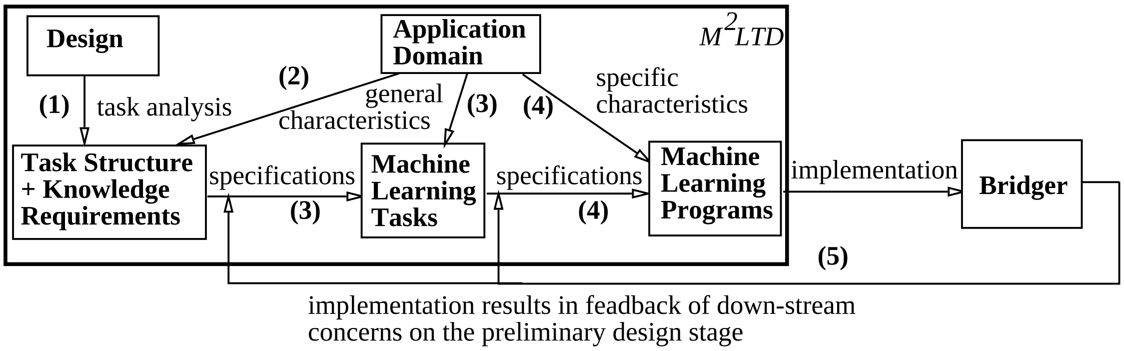

Figure 3: M^{2}LTD: Matching Machine Learning programs To Design tasks

The evaluation of M^{2}LTD is presently limited. In includes the reconstruction of its use to design, test, and evaluate BRIDGER and few other uses such as (Milzner and Harbecke, 1992), where M^{2}LTD was used to select STAGGER (Schlimmer and Granger, 1986) for building a system that learns evaluation knowledge for circuit design. The discussion in (Milzner and Harbecke, 1992) about the use of M^{2}LTD and the evaluation of the system created enhances the knowledge used in executing M^{2}LTD. That discussion, however, does not convey many subtle issues such as those discussed in this paper; it may be a "filtered" or a rational reconstruction of the decision process involved in using M^2LTD. Therefore, it may provide partial feedback on the usefulness of M^2LTD.

M^2LTD is intended to be revised continually and incrementally by adding information from experiences. A major revision, in the form of changing the structure of the selection process presented in Figure 3 can also take place. The further development of M^2LTD requires its use for selecting ML techniques in many projects and the incorporation of the feedback from its use back into its recommendations.

M^2LTD serves as the substantive reason for adopting S_2. As a tool for advancing the research and practice of ML in design, it assumes the responsibility for recording and analyzing the rationale for selecting between different ML techniques for specific tasks. This rationale can then inform future choices by researchers and practitioners.

4 S_2: A historical overview of the project

This section provides a historical overview of the learning project by reviewing the events leading to the creation of the end products discussed in Sections 2 and 3. Since the purpose of this paper is methodological, this section provides only some details that are needed to support S_2 rather than a comprehensive overview.

4.1 Soar

The first experiment with ML techniques I[8] performed involved studying the suitability of **Soar** as a vehicle for design knowledge acquisition. **Soar** is a problem solving architecture with an integral learning capability (Laird et al, 1987). The decision to select **Soar** was simple, **Soar** was developed in-house at Carnegie Mellon University and seemed a natural first candidate. The reconstructed reason I offered later was that a successful implementation in **Soar** will be powerful because **Soar** allows for less degrees of freedom to be exercised in design choices of any system implemented. The tight coupling of learning and performance would force taking an approach that is more constrained and hopefully “natural” to any domain of interest.

A simple problem of design by parameters selection was chosen and experimented with (Reich and Fenves, 1988). It was observed that **Soar** was an excellent problem solving architecture that enforced a specific discipline on the structuring of domain knowledge, but that its learning mechanism, when used, was too demanding of this structuring. The design of problem structures that resulted in learned rules that transferred between problems was difficult. This difficulty, as well as other limitations, were discouraging. After conducting a few additional experiments, the use of **Soar** was suspended.

Reflections. The assumption that after structuring domain knowledge learning “will take on

[8] Notice the change from “we” to “I.” First, the use of “I” conveys less authority than the use of “we.” Second, the use of “I” does not remove the contribution of Steven Fenves who acted as my advisor in this PhD research project, it is simply meant to put the burden of all the pitfalls on me.

from there" was mistaken. When learning was expected to work, the naturalness of structuring the domain knowledge had to be distorted. Also, the assumption (and a major reason for using SOAR) that mechanisms can be shared across projects was premature. For example, mechanisms for abstractions or analogy that were developed, were incompatible and their use required changing the representation and problem structure to fit their design. Only recently, were several high level mechanisms such as data-chunking (Rosenbloom et al, 1987) made available for sharing among different SOAR projects.

4.2 The goal of learning

After the initial experiments with SOAR, it was time to formulate an attainable goal for the learning project. I then decided to explore prototype systems that could simulate the functionality I was expecting to get from SOAR. I approached such simulations while waiting for a new version of SOAR to be released.

The learning task as posed after the initial studies with SOAR was as follows. Given examples of designs, including their specifications, derivations, and final descriptions, create three concept hierarchies: specifications, design derivations, and design descriptions (see Figure 4). In addition, learning had to create indices between the hierarchies such that a node in the specifications hierarchy was connected to a node in the derivations hierarchy that heuristically could create an appropriate design (the dashed line in the figure), and connected to the design descriptions hierarchy for heuristically selecting designs that may satisfy the requirements (the dotted line). After learning, design could be executed in one of two ways. One way would be to take a new specification and classify it with the specifications hierarchy (node 1). The (generalized) node in the specifications hierarchy would be connected to a node in the derivations hierarchy that reflects, in a general sense of course, the actions needed to design an artifact for the specification given (node 2). This derivation would be adjusted to the new situation and replayed to result in a design (node 3 in the design description hierarchy). Another possibility for designing would be to move directly from the specifications hierarchy to the design descriptions hierarchy (node 4). Note that there was no guarantee that these two methods would result in the same candidate designs.

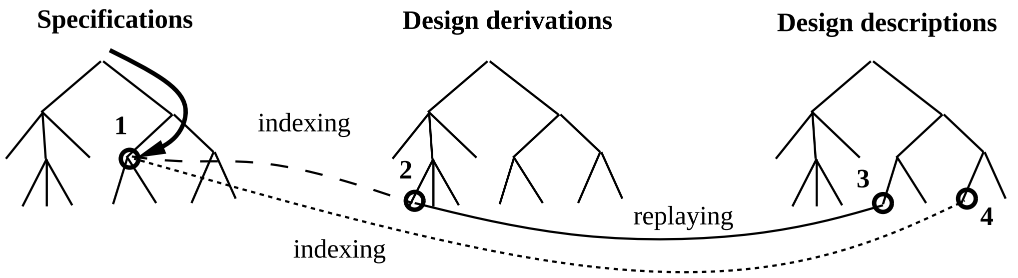

Figure 4: Design as a mapping

Reflections. The functionality discussed above drove the development of a prototype system.

This functionality was a reconciliation of what may be needed to support real design and what ML techniques could offer. While the goal was informed by both, it simplified design and stretched the capabilities of learning techniques.

4.3 Focussing

The second attempt of this research involved a test of the **Focussing** algorithm (Bundy et al, 1985). **Focussing** is a symbolic supervised concept learning algorithm comparable to the version spaces algorithm (Mitchell, 1978), but one that operates on concepts described by hierarchies. The **Focussing** algorithm contained extensions that solved several problems, such as the emergence of contradictions during the course of learning (Wielemaker and Bundy, 1985; van Someren, 1987). These extensions dealt with changing the hierarchies representing concepts and examples, and therefore, appealed to the functionality sought from the learning task in this research. The algorithm and some of its extensions were implemented and experimented with. It was apparent that two significant problems were manifested. First, the (symbolic) nature of the process led to fast learning that assumed more than warranted from the evidence presented in the design examples (this is a problem with all purely symbolic learning), leading to contradictions between learned concepts and subsequent training examples. Second, the mechanisms available to solve or prevent these contradictions presented multiple choices that could not be resolved easily. These resolutions required having knowledge about the functionality of the learning mechanisms involved in the conflicts. The need to collect such information has led to the inception of the concept of *generic learning task* (Reich and Fenves, 1989).

Reflections. In retrospect, the path that was selected later in the research attempted to address the two issues raised by the **Focussing** algorithm. In reality, the first issue seemed more fundamental and its solution was perceived to answer partially the second issue.

4.4 Generic learning tasks

In the use of any learning program it is necessary to know (1) what is the input and output of the learning program, (2) what is the representation of knowledge used and learned, and (3) how are operational parameters for running the program selected. This information may be used to select a learning program for performing a specific learning task. Similar information requirements emerge from Chandrasekaran's work on generic tasks as high level building blocks for expert systems design (Chandrasekaran, 1986). The concept of *generic learning tasks* is the application of the generic task idea to ML except that now the task is learning a specific type of knowledge that will be used by some generic (problem-solving) task. Generic learning tasks can be used as building blocks in the creation of complex learning programs.

Reflections. At that time, and still today, the concept of generic learning tasks remains only partially understood and complex to implement. Primarily, this results from the fact that the functionality of a learning task must be determined experimentally and evolve continually through feedback from its use. This requires following $\mathbf{S}_2$. The number of learning tasks that have been formulated in a preliminary fashion is small and no experiments are available that shed light on the respective functionality of these tasks, certainly not in a way that can be used to select between them automatically. Some projects are underway to improve this situation (Morik et al, 1991).

The concept of generic learning task can also force useful report of ML techniques since it contains necessary information required to select a particular program for particular uses. As such, this concept is the foundation of M^2LTD.

4.5 ECOBWEB

As discussed in Section 4.3, the first attempt at solving the questions raised by the FOCUSSING algorithm was to change the purely symbolic nature of FOCUSSING. In involved the exploration of a probabilistic representation of learned knowledge. The COBWEB algorithm (Fisher, 1987) supported this requirement, in addition to being incremental. While the latter was viewed initially just as a good feature it later became a fundamental requirement for supporting design.

Another characteristic of COBWEB that initially did not seem necessary but proved useful was its ability to perform flexible prediction. This ability eliminates the need to specify *a priori* what are the specification or the design description properties. Rather, a design problem could be defined by any partial description of properties and the solution would be the complete description. Once the importance of this characteristic was observed, it also became a fundamental requirement. It was conjectured that unsupervised concept learning could support such prediction, whereas supervised concept learning could not (Reich and Fenves, 1991). One deficiency of supervised concept learning is that one property must be selected *a priori* as the predicted property (i.e., the class property). This observation was later adopted as a guiding principle by other projects (Bailin and Henderson, 1992; Garrett and Ivezic, 1992).

COBWEB was re-implemented and its functionality tested using a historical data set of examples of bridges constructed in Pittsburgh, a personal choice directed by the expertise of Steven Fenves in the subject. The results demonstrated that COBWEB learned gradually from examples and reconstructed (i.e., design) the bridges, given their specifications only (Reich, 1991b; Reich and Fenves, 1992).

The promising results from the initial experiment and from others that followed were analyzed, trying to relate the behavior of COBWEB to characteristics of real design. The characteristics put forward were: design is a multiple-objective and constrained process, design makes use of a variety of knowledge sources and operates in different levels of abstraction, and design employs multiple strategies. Finally, the nature of everything relating to design may change over time. These characteristics informed the extensions to COBWEB, but were only addressed partially (Reich, 1991b).

Reflections. The hypothesis that concept formation (unsupervised incremental concept learning) techniques are more suitable than supervised concept learning for the acquisition of synthesis knowledge was articulated at length (Reich and Fenves, 1991), but an experimental support for it still sought. The adoption of this hypothesis by others and its testing may provide the necessary experimental support.

The flexible prediction ability of COBWEB resulted in a change in the research plan. Instead of learning three hierarchies and the relationships between different nodes in these hierarchies as discussed in Section 4.2, a single hierarchy seemed sufficient to capture design knowledge (Reich and Fenves, 1991). This was an experimental observation coupled by the parsimony principle: if a desired result can be obtained with a simple system, do not attempt to build a more complex one for the same purpose. This change demonstrates the evolutionary nature of problem definition in response to insight from experience.

At this time, a firm decision was made to follow this path of research instead of using SOAR. The investment in this path and the interesting and promising results were too encouraging to risk them for the difficult path of using SOAR.

4.6 General Design Theory

During that time, I accidentally came across Yoshikawa's paper on General Design Theory (GDT) (Yoshikawa, 1981). It was an accident, because although Yoshikawa's initial work and its later extensions (Tomiyama and Yoshikawa, 1986; Tomiyama et al, 1989) are insightful, its complex mathematical foundations prevented researchers in the West from paying any attention to its assumptions and their consequences for design. Therefore, GDT is hardly ever referenced in the literature.

From first glance it was clear that GDT and COBWEB had many similarities. A preliminary analysis detailed some of these similarities (Reich, 1990). While COBWEB was able to approximate the initial formulation of GDT, it lacked the ability to address the issues raised in its extensions. The extensions presented difficulties that were also manifested in the study of the domain discussed in the next section.

Reflections. It took several years, a comprehensive study of GDT, and a reflection on the complete project to arrive at a detailed comparison between GDT and the end products of this research (Reich, 1991d; Reich, 1991e). This comparison illuminated several aspects. (1) GDT assumes that knowledge about designs is represented by mathematical topology(Sutherland, 1975), whereas ECOBWEB represents it by a classification hierarchy, a very degenerate approximation of a topology. (2) Based on the structure of design knowledge, GDT guarantees perfect convergence to design solutions, whereas ECOBWEB can never be perfect and performs depending on the examples it has learned and its synthesis mechanisms. Nevertheless, even when using a considerably less expressive knowledge structure than a topology, ECOBWEB demonstrated an increasingly competent design ability in various evaluations.

This comparison has two implications. First, ECOBWEB's performance might be improved, if its knowledge representation is made closer to a topology. (A step towards this end is the generation of dynamic graph structures of knowledge that is currently under development.) Second, if GDT is to be more relevant to real domains, the topological structure of knowledge it assumes must be relaxed. This may cause proving less precise statements about design, but still statements that can guide the design of ML programs and their evaluations.

This comparison between a theory and the predictions it makes and an implemented system with its performance evaluation is fundamental to scientific progress. It can be perceived as a test of the theory. On the one hand, the positive results of this test in the form of good design performance of Ecobweb provide support for the theory, and on the other hand, the theory helps guiding the future development of the implementation (see also Section 5.1).

4.7 Bridge design

Parallel to observing the similarity between GDT and Cobweb, there was the need to start addressing design problems that were more complex than those experimented with to that point. A domain was selected based on personal preference, approval of Steven Fenves, and preliminary observation that it will lead to successful results[9]: the design of cable-stayed bridges. The study of the domain, including the search and construction of bridge data was time consuming. The study started with reviewing old texts on bridge design (Waddell, 1916) and ended with recent books and publications on the subject (Fox, 1979; Podolny and Scalzi, 1986; Spector and Gifford, 1986; Troitsky, 1988).

The study revealed that the process of bridge design can be roughly decomposed into three phases: conceptual, preliminary, and detailed. The first phase was adopted as the focus of the study because of its difficulty and importance. The conceptual design could be roughly decomposed into five tasks: problem analysis, synthesis, analysis, redesign, and evaluation.

It was clear from this study that Cobweb alone could not support the complete conceptual design process. It was clear that additional components had to be incorporated and integrated into a larger program. This surfaced again the concept of generic learning tasks and triggered the inception of M^2LTD (Reich, 1991c).

Reflections. While the study made many pieces of knowledge explicit and identified the phases and tasks discussed, it did not include any observation of actual designers in their work. This was rationalized as though the goal of the research was to identify the limits of building a support system with ML techniques without engaging in the time consuming process of "knowledge engineering." The study obviously simplified the actual design problem.

It is clear from the evaluation of the work that building a system for automating bridge design is a flawed goal; rather, the development of a support system, where designers can interact with, monitor, and, if so desired, override system operations, is a more important and attainable goal.

It is not clear at all that the generic task approach will succeed in supporting the implementation of a truly practical design support system because such a system must incorporate significant detailed domain knowledge that was mostly abstracted in this study. Furthermore, it is not guaranteed that an assembly of generic tasks created *a priori* can accommodate the peculiar features of a new domain knowledge as it unfolds during the accumulation of large amounts of knowledge. In fact, experience with the development of large systems demonstrate that substantial evolution often takes place in such projects (see (Pople, 1985) for a description

[9]It must not be forgotten that the work was in the context of a PhD thesis, perceived as basic research, but one that had to result in a thesis in a limited time frame.

of the evolution of a system in response to the collection of significant domain knowledge over several years).

4.8 Bridger

After identifying the various tasks that needed to be implemented, the ML techniques that could support their construction had to be identified. The search for learning programs that could support the conceptual design was limited to the redesign task. The first task, problem analysis, was performed *a priori* in a simplified form, the analysis involved coding a finite element procedure, the evaluation remained unsupported computationally, and synthesis remained based on Ecobweb, although it now included two instantiations with a candidate adaptation module (Figure 2).

At that time, I reviewed the book by Bareiss on Protos (Bareiss, 1989) for Machine Learning Journal (Reich, 1991a). During the review, I also experimented with a newer version of Protos (Dvorak, 1988). It seemed that Protos had the necessary ingredients for supporting the acquisition of redesign knowledge. First, it supported a diagnosis task; therefore, it could identify the cause of a problem in the bridge behavior revealed by analysis. Second, it could accommodate domain knowledge in the form of causal relations that could enhance bridge redesign. Third, it was incremental.

Several extensions were made to Protos, leading to a new system called Eprotos. The important extension allowed Eprotos to handle continuous values. Other necessary extensions to the mechanisms that manipulate domain knowledge were not performed due to lack of time. The limited effort spent on Eprotos directly influenced the functionality of the resulting redesign sub-system. No serious testing was possible. By the time the research terminated, Eprotos was capable only of demonstrating simple redesigns.

The choice of Protos remains a hypothesis. It is clear that the reason for its selection was based on partial information only. The redesign task was found to be more complex than envisioned initially and Protos was missing functionality, such as handling numerical relationships, that could not be incorporated without significant changes to the original mechanisms. Since, Protos' mechanisms were more complex than those employed by Cobweb and since significantly less effort was available for extending it compared to that spent on Cobweb, the required modifications were implemented only partially.

Reflections. The availability of the well written Protos code had an impact on its selection for the task of implementing the redesign sub-system. Again, as the results show, proximity to a resource is not a success-guaranteeing heuristic.

In the review of Protos, I stated that its evaluation, as appeared in publications, was insufficient. This is true also for most large AI programs. No serious testing could be performed to examine various aspects of the program. The only way remaining for testing was to distribute the system and expect additional researchers to test in on their problems. The use of Protos in this research provides such an additional test, albeit one that outlines significant limitations.

The design of Bridger was definitely biased by a reductionist approach. This bias simplified the original design problem. One could no longer anticipate to solve the original, real design problem automatically with BRIDGER. Rather, BRIDGER was assumed to be controlled by a user that oversaw its operations.

5 Methodological ramifications

This section elaborates on the methodological issues raised in this paper. The first part discusses these issues in relation to S_1 while the second relates the issues to S_2.[10]

5.1 Closing one scientific cycle

The "story" of this research can be reconstructed as follows. GDT, a formal theory of design, proved properties of design following some assumptions, mainly the one stating that knowledge structure is a mathematical topology. GDT was missing a crucial aspect: an explanation of how design knowledge arises or, how can one create such knowledge structure to benefit from its proven ability to generate designs. This study attempted to provide an answer to this question, thereby serving as a test of GDT's assumptions. Some of GDT's assumptions were relaxed, in particular, the knowledge structure became a hierarchical classification rather than a topology, leading to the development of ECOBWEB and BRIDGER.

The evaluation of these systems demonstrated that there was some discrepancy between the predictions of GDT and the performance of these systems, but that this difference was not significant given the significant relaxation of the assumptions.

This led to the formulation of future research consisting of two parallel activities. The first activity would aim at relaxing GDT's assumptions and attempting to prove theorems similar to the original ones, thereby refining GDT in response of the experimental results. The second activity would attempt to account for the discrepancy between the experimental results and the prediction of the theory by building a system that learns graph structures instead of a hierarchy, the former better approximating a topology. This "story" completes a single scientific cycle as illustrated in Figure 5.

Most researchers accept S_1. To fully adhere to its methodology, the end product of their research must include: the theory that served as the hypothesis of the research, the detailed description of the implementation of the theory, the precise description of the testing set-up, its results and their analysis, the conclusions of the research, and how they guided the refinement of the theory. The omission of any of these parts casts doubts about the credibility of the research.

This research had several end products. The first is a theory of learning, appropriate for a class of routine design problems consisting of: (1) *A theory of learning in design* which accepts

[10] In this emphasis, the paper does not deal with other methodological issues relevant to AI, such as the role of programs as theories or experiments. For an extensive discussion on these issues see (Partridge and Wilks, 1990).

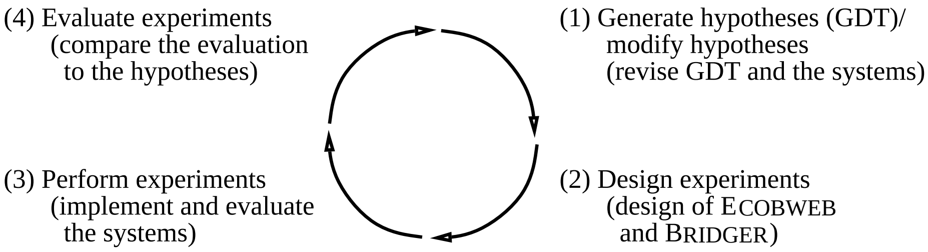

Figure 5: The scientific method cycle

GDT's formulation of design and adds the hypothesis that concept formation is the means for creating design knowledge. (2) *An experimental set-up*, starting from the six assumptions approximating the theory, leading through the detailed implementation of Ecobweb and Bridger, to the description of the experiments to be conducted with these systems. (3) *Experiments execution* and (4) *results analysis*, including their impact on the original theory. All these parts collectively constitute the theory developed in this research.

Given the information provided for this theory, one can easily challenge it. For example, one can argue that one of the assumptions is incorrect. This may render the theory inappropriate for a domain whose properties are different than assumed by the theory. In contrast, if a domain can accept the assumptions, it is subjected to the predictions of the theory until they are proven wrong. Consequently, a discussion on these theories can focus on the assumptions, rather than on irrelevant details.

5.2 From theory to practice

We have seen that even a reconstructed historical description of a research ($\mathbf{S}_2$, Section 4) deviates from its reconstructed report in the first sense of science ($\mathbf{S}_1$, Section 5.1). A reconstruction tends to be an analysis process, namely, given the existing implementation, what could be a set of assumptions that is consistent with it and still makes some sense. In contrast, design rationale captures the synthesis of the development with all the subtle and crucial issues.

Two ways exist for advancing the recommendation of subscribing to $\mathbf{S}_2$. This paper exemplifies the first way; it puts forward a moral, methodological and substantive reasons for selecting $\mathbf{S}_2$.

The second way attempts to *demonstrate experimentally* that $\mathbf{S}_2$ leads to better practice of design. The development of $\mathbf{M}^2$**LTD** (which was the substantive reason) is a step in this direction. It is based on one principle: evolutionary development through user participation (Reich et al, 1992). $\mathbf{M}^2$**LTD** can only be developed through its use by engineers for synthesizing learning systems to solve engineering problems and from the accumulation of its users' feedback. Each feedback loop involves making hypotheses about ML techniques by using $\mathbf{M}^2$**LTD** and testing them continually in fine grained "scientific cycles," similar to those discussed before (see Figure 6). In order that this feedback be comprehensive it must follow $\mathbf{S}_2$. If $\mathbf{M}^2$**LTD** will lead to better implementations of learning systems it will serve as a test supporting the theory that S_2 is superior to S_1. We can observe that these tests involve a shift from research-based or prototype tools to practical tools.

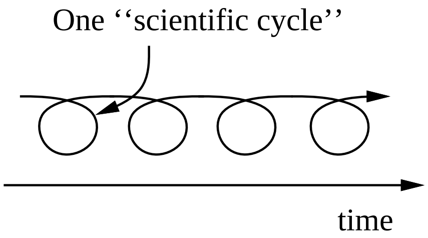

Figure 6: Evolutionary development of theories and artifacts

5.3 The insight from S_2

After detailing the two senses of science one can ask what does S_2 tell us that S_1 could not? or how does S_2 help us understand the limitations of S_1? First, it is clear that S_2 is more verbose, less rigid, more personal, and more comprehensive than S_1. S_2 presents a story, rather than a concise set of hypotheses, tests and evaluations, although it *contains* them as an integral part, that is, S_2 contains S_1. Therefore, one does not compromise the "quality" of results one obtains by S_1, one simply attributes them to their proper cause. For example, in S_2, a reconstructed set of assumptions or hypotheses will be stated as a valid set of assumptions that can be supported by the tests, instead of the product of profound observations—a belief that the report of S_1 would like us to accept. If a researcher is imaginative, the assumptions will be the *maximal* set of assumptions supported by the tests.

There are some observation that can be drawn from the comparison between S_1 and S_2.

1. *Richness of information.* S_2 is perfect for detailing shifts in thinking. For instance, S_2 illustrated the shift from design as a mapping between three hierarchies (Figure 4) to design as a refinement and generation (Figure 1). By no means is this shift a conclusive evidence that the latter view is better than the former; but it constitutes some support for this argument. This argument can be contrasted or augmented with future arguments to result in a richer context for making choices in future implementations.

   Note that part of the reason for the above shift was the inability to provide rich knowledge for the automatic selection of generic learning tasks. Later, while facing the complex problem of preliminary design of cable-stayed bridges, the necessity for integrating techniques arose. It was past experience that enforced a two-level treatment of this integration: macro, a manual selection of ML programs, and micro, a collection of techniques with pre-defined control. In both levels, the control was defined *a priori* due to lack of knowledge about generic learning tasks.

2. *Inevitability of failures.* S_2 makes us less optimistic about fast and easy progress. There

are fruitless paths to be explored, and they will be visited again unless reported as such and learned from.

Failures are probably the best source of progress (Petroski, 1989). Failures will be those that are acknowledged as such after researchers tried to salvage whatever "positive" results they could for their publications. Therefore, failures reported will be close to "near misses" which are excellent triggers of learning (Winston, 1975). Therefore, beyond subscribing to S_2, the documentation of failures must be legitimized as a service critical for progress.

3. *Mundane aspects of research.* S_2 makes us aware of the rudimentary but unavoidable issues influencing research (e.g., context of research is thesis or the availability of cheap resources).

4. *Value of longitudinal studies.* Long and integrative studies are critical for accumulating information about research questions. Such studies allow filtering intermediate conclusions by the researcher, rather than by peers. Such studies provide rich context required to understand the problem addressed, the solution paths attempted, and the reasons for the reported successes.

5. *Difficulty of reporting.* It is difficult to report studies in S_2. It requires revealing unsuccessful activities and exercising constant reflection on the research activity. S_2 reports are more difficult to read; they require paying attention to a long description of events, some of which are less technical than others.

6. *Role of experiments.* Experiments are critical for progress; to illustrate, without them **EPROTOS** would have been pronounced a success in this study when such statement was not warranted. Scaling up to practical applications provides the best test of research questions. In fact, practice is the experimental set-up for research questions such as: "what does it take to select ML techniques for design applications?"—a question which **M^{2}LTD** attempts to answer.

There are also some heuristics that can be extracted from the reconstruction of S_2.

1. The initial steps of the study were rather unconstrained, with relatively no theoretical grounding. Part of these steps were meant to understand the capability of ML techniques. Such method may be justified in the beginning of an exploratory research but may be too costly when research progresses. The adoption of S_2 can lead to learning within the course of research (e.g., see item 1 in the previous list).

2. It is hard or rather impossible to formulate tight research hypotheses *a priori*, but if one follows a sensible S_2, such hypotheses can be constructed and evolve in subsequent cycles which benefit significantly from the feedback on the first cycle.

3. The cheapest resource (e.g., an available program) is not necessarily the best one to select as the core of a research project.

4. S_2 suggests that research results should be interpret with great care. In particular, there may be other sets of assumptions that are supported by the experimental evidence, and one can never prove that the hypotheses are correct.

6 Summary

In this paper I have argued for improving the methodology of doing research on ML applications for design in general. I have presented two ways of looking at science that are prevalent in philosophy and argued that S_2 is the only one acceptable from the particular perspective of researchers on design, in general, and researchers working on ML in design, in particular.

In the course of the argument, I presented a research project whose end products are: a theory of learning in a class of design problems, a set of computer tools, and a method for selecting ML programs for executing specific learning tasks. In doing so, I have discussed the evaluation of the computer tools and argued that similar evaluations must be carried out by researchers even if they choose to remain subscribed to S_1.

Acknowledgments

This work has supported in part by the Engineering Design Research Center, a National Science Foundation Engineering Research Center. Steven Fenves had significant input to the process (i.e., S_2) of this research and therefore to its results (i.e., S_1). The ideas about methodology have benefited also from discussions with Suresh Konda, Ira Monarch, and Eswaran Subrahmanian. I thank the reviewers for their constructive comments on a draft of this paper.

References

Bailin, S. C. and Henderson, S. (1992). "Applying machine learning to software engineering." In *Proceedings of the AID'92 Workshop on Machine Learning.*

Bareiss, R. (1989). *Exemplar-Based Knowledge Acquisition*, Academic Press, Boston, MA.

Bundy, A., Silver, B., and Plumer, D. (1985). "An analytical comparison of some rule-learning programs." *Artificial Intelligence*, 27(2):137–181.

Chandrasekaran, B. (1986). "Generic tasks in knowledge-based reasoning: high-level building blocks for expert system design." *IEEE Expert*, 1(3):23–30.

Clancey, W. J. (1986). "From GUIDON to NEOMYCIN and HERACLES in twenty short lessons: ONR final report 1979-1985." *AI Magazine*, 7(3):40–60.

DeMillo, R. A., Lipton, R. J., and Perlis, A. J. (1979). "Social processes and proofs of theorems and programs." *Communication of the ACM*, 22:271–280.

Dvorak, D. L. (1988). "Guide to CL-Protos: An exemplar-based learning apprentice." Technical Report AI88-87, Artificial Intelligence Laboratory, The University of Texas at Austin, Austin, TX.

Feyerabend, P. K. (1975). *Against Method*, New Left Books, London, UK.

Fisher, D. H. (1987). "Knowledge acquisition via incremental conceptual clustering." *Machine Learning*, 2(7):139–172.

Fox, G. F. (1979). "The Dame Point Bridge main spans-superstructures." In *Long Span Suspension Bridges: History and Performance*, American Society of Civil Engineers, New York.

Garrett, J. and Ivezic, N. (1992). "A neural network approach for acquiring and using synthesis knowledge." In Dagli, C. H., Burke, L. I., and Chin, Y. C., editors, *Intelligent Engineering Systems Through Artificial Neural Networks, Vol. 2*, pages 767–772, New York, NY, ASME Press.

Hays, W. L. (1988). *Statistics. Fourth edition*, Holt, Rinehart & Winston, New York.

Kuhn, T. S. (1962). *The Structure of Scientific Revolution*, The University of Chicago Press, Chicago, IL.

Laird, J. E., Newell, A., and Rosenbloom, P. S. (1987). "Soar: an architecture for general intelligence." *Artificial Intelligence*, 33(1):1–64.

McMullin, E. (1987). "Alternative approaches to the philosophy of science." In Kourany, J. A., editor, *Scientific Knowledge: Basic Issues in the Philosophy of Science*, pages 3–19, Belmont, CA, Wadsworth.

Milzner, K. and Harbecke, A. (1992). "Incremental learning for improved decision support in knowledge based design systems." In Gero, J., editor, *Artificial Intelligence in Design'92, Proceedings of The Second International Conference on Artificial Intelligence in Design, Pittsburgh, PA*, pages 719–738, Dordrecht, The Netherlands, Kluwer Academic Publishers.

Mitchell, T. (1978). "Version spaces: An approach to concept learning." PhD thesis, Stanford University, Palo Alto, CA.

Morik, K., Causse, K., and Boswell, R. (1991). "A common knowledge representation integrating learning tools." In Michalski, R. S. and Tecuci, G., editors, *Proceedings of the First International Workshop on Multistrategy Learning*, pages 81–96, Fairfax, VA, Center for Artificial Intelligence, George Mason University.

Newell, A. and Rosenbloom, P. S. (1981). "Mechanisms of skill acquisition and the power law of practice." In Anderson, J. R., editor, *Cognitive Skills and Their Acquisition*, Erlbaum Associates, Hillsdale, N.J.

Partridge, D. and Wilks, Y., editors (1990). *The Foundations of Artificial Intelligence: A Sourcebook*, Cambridge University Press, Cambridge, UK.

Petroski, H. (1989). "Failure as a unifying theme in design." *Design Studies*, 10(4):214–218.

Pickering, A., editor (1992). *Science as Practice and Culture*, The University of Chicago Press, Chicago, IL.

Piela, P., Katzenberg, B., and Mckelvey, R. (1992). "Integrating the user into research in engineering design systems." *Research in Engineering Design*, 3(4):211–221.

Podolny, W. and Scalzi, J. B. (1986). *Construction and Design of Cable-Stayed Bridges. Second edition*, John Wiley and Sons, New York.

Pople, H. (1985). "Evolution of an expert system: From Internist to Caduceus." In De Lotto, I. and Stefanelli, M., editors, *AI in Medicine*, pages 179–208, Amsterdam, Elsevier.

Popper, K. (1965). *Conjectures and Refutations: The Growth of Scientific Knowledge*, Harper and Row, New York.

Reich, Y. and Fenves, S. J. (1988). "Floor system design in Soar: A case study of learning to learn." Technical Report EDRC-12-26-88, Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, PA.

Reich, Y. and Fenves, S. J. (1989). "Integration of generic learning tasks." Technical Report EDRC 12-28-89, Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, PA.

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

Reich, Y. and Fenves, S. J. (1992). "Inductive learning of synthesis knowledge." *International Journal of Expert Systems: Research and Applications*, 5(4):275–297.

Reich, Y., Konda, S., Monarch, I., and Subrahmanian, E. (1992). "Participation and design: An extended view." In Muller, M. J., Kuhn, S., and Meskill, J. A., editors, *PDC'92: Proceedings of the Participatory Design Conference (Cambridge, MA)*, pages 63–71, Palo Alto, CA, Computer Professionals for Social Responsibility.

Reich, Y. (1990). "Converging to "Ideal" design knowledge by learning." In Fitzhorn, P. A., editor, *Proceedings of The First International Workshop on Formal Methods in Engineering Design*, pages 330–349, Colorado Springs, Colorado.

Reich, Y. (1991). "Book review: Exemplar-Based Knowledge Acquisition, by Ray Bareiss. Academic Press, 1989." *Machine Learning*, 6(1):99–103.

Reich, Y. (1991). "Building and improving design systems: A machine learning approach." PhD thesis, Department of Civil Engineering, Carnegie Mellon University, Pittsburgh, PA. (Available as Technical Report EDRC 02-16-91).

Reich, Y. (1991). "Design knowledge acquisition: Task analysis and a partial implementation." *Knowledge Acquisition*, 3(3):237–254.

Reich, Y. (1991). "Design theory and practice I: A critical review of General Design Theory." Technical Report EDRC 12-45-91, Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, PA.

Reich, Y. (1991). "Design theory and practice II: A comparison between a theory of design and an experimental design system." Technical Report EDRC 12-46-91, Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, PA.

Reich, Y. (1993). "A model of aesthetic judgment in design." *Artificial Intelligence in Engineering*, 8(2):141–153.

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

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

Rosenbloom, P. S., Laird, J. E., and Newell, A. (1987). "Knowledge level learning in Soar." In *Proceedings of AAAI-87*, pages 499–504, Seattle, WA, Morgan Kaufmann.

Schlimmer, J. C. and Granger, R. H. (1986). "Incremental learning from noisy data." *Machine Learning*, 1(3):317–354.

Spector, A. and Gifford, D. (1986). "A computer science perspective of bridge design." *Communications of the ACM*, 29(4):268–283.

Sutherland, W. A. (1975). *Introduction to Metric and Topological Spaces*, Oxford University Press, Oxford, UK.

Tomiyama, T. and Yoshikawa, H. (1986). "Extended general design theory." Technical Report CS-R8604, Centre For Mathematics and Computer Science, Amsterdam.

Tomiyama, T., Kiriyama, T., Takeda, H., Xue, D., and Yoshikawa, H. (1989). "Metamodel: A key to intelligent CAD systems." *Research in Engineering Design*, 1(1):19–34.

Troitsky, M. S. (1988). *Cable-Stayed Bridges: An Approach to Modern Bridge Design. Second edition*, Van Nostrand Reinhold, New York.

van Someren, M. W. (1987). "Knowledge based learning: reducing the description space for rule learning." In B. Du Boullay, D. H. and Steels, L., editors, *Advances in Artificial Intelligence-II*, pages 53–59, North-Holland, Amsterdam.

Waddell, J. A. L. (1916). *Bridge Engineering, Volume I*, John Wiley & Sons, New York.

Wielemaker, J. and Bundy, A. (1985). "Altering the description space for focussing." In Merry, M., editor, *Expert Systems 85*, pages 287–298, Cambridge, December, Cambridge University Press.

Winston, P. (1975). "Learning structural descriptions from examples." In Winston, P., editor, *The Psychology of Computer Vision*, pages 157–209, McGraw Hill, New York.

Yoshikawa, H. (1981). "General Design Theory and a CAD system." In Sata, T. and Warman, E., editors, *Man-Machine Communication in CAD/CAM, Proceedings of The IFIP WG5.2-5.3 Working Conference 1980 (Tokyo)*, pages 35–57, North-Holland, Amsterdam.