Design Theory and Practice II: A Comparison Between a Theory of Design and an Experimental Design System

Y. Reich

EDRC 12-46-91

Design Theory and Practice II: A comparison between a theory of design and an experimental design system.

Yoram Reich
Engineering Design Research Center
Carnegie Mellon University
Pittsburgh, PA 15213

Keywords: design theory, scientific method, machine learning, knowledge acquisition.

Abstract: This paper is Part II of a study advocating the use of the scientific method for conducting design research. Part I critically reviewed General Design Theory (GDT)—a formal theory of design—and discussed its influence on the construction of design support systems. This part presents an experimental system built based on the guidelines set by GDT. It evaluates the experimental system and compares the theory and the experimental approaches. The comparison generates recommendations for enhancements of the experimental system and refinements to the theory. These recommendations complete a single, successful scientific cycle of this research. The recommendations that are grounded in testings show the benefits from subscribing to the scientific method of conducting design research.

1 Introduction

This study advocates to, and demonstrates, the use of a scientific method in design research, illustrated in Figure 1. This method consists of an essential recurring process of hypotheses generation, experiment design and execution, results evaluation, and hypotheses refinement. The first part of the study (Reicji, 1991c) discussed three methods for conducting research: the theoretical, empirical, and scientific. These methods were briefly reviewed in the context of design research. To prepare the argument that the scientific (i.e., theory plus empirical) method is best, the first part of the study concentrated on one theoretical approach to design, called General Design Theory (GDT) (Yoshikawa, 1981; Tomiyama and

Yoshikawa, 1986). It critically reviewed it and discussed its implications to building design systems.

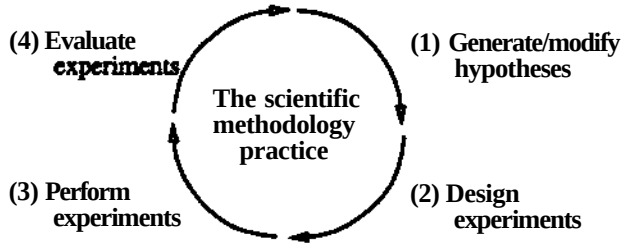

Figure 1: The scientific method cycle

Since GDT is a mathematical theory, it can be evaluated based on "aesthetic" or parsimony principles: if two theories prove the same theorems, the one that makes less assumptions is better. The review of GDT, which discussed potential revisions to the assumptions, can be viewed as a "thought[9]* experiment that suggested revisions to the theory. Therefore, the scientific method was exercised already. This exercise, however, is not sufficient. GDT needs to be tested in actual experiments. This part of the study completes the Argument favoring the scientific method by discussing such an experiment, its evaluation and implications. The next two paragraphs briefly preview the observations of the study.

GDT suggested guidelines for building design systems. The experimental design system discussed in this study, BRIDGER, incorporates some of these guidelines, but also relies on the task analysis of the application domain. Overall, BRIDGER makes considerably weaker assumptions about the nature of design knowledge than GDT makes. BRIDGER was implemented and evaluated successfully (Reich, 1991b). It has shown that a competent design behavior can be obtained with assumptions less restrictive than GDT's. Therefore, the results challenge GDT's scope. Although the performance was good, it was not, and never will be, perfect. This is different from the perfect performance predicted by GDT's theorems. Could this be the result of the large discrepancy between the knowledge structures of GDT and BRIDGER? These two observations suggest two lines of future research. First, some of GDT's assumptions can be relaxed to result in a better theory. Alternatively, a new theory that can explain the results obtained by BRIDGER needs to be formulated. Second, the provision of a more elaborate knowledge structure may give BRIDGER the opportunity to achieve better design performance.

These two lines of research demonstrate the usefulness of subscribing to the scientific method. Instead of creating *ad hoc* design theories and experimental systems, the interleaved theoretical and experimental investigation focuses research on the crucial aspects of the inquiry. Observing current

trends in design research, this conclusion advocates for balancing the effort expended in design research towards additional studies on theory as opposed to experimental research; both theory and experimental studies are necessary ingredients of progress in this field.

Plan of the paper. The remainder of this paper is organized as follows. Section 2 reviews ECOBWEB, an experimental system for learning synthesis knowledge. Section 3 describes BRIDGER, a system that employs ECOBWEB and other procedures for learning to design cable-stayed bridges. Section 4 discusses the relationships between GDT (state of *ideal* knowledge) and ECOBWEB, and GDT (state of *real* knowledge) and BRIDGER. Section 5 summarizes the implications of this study.

2 ECOBWEB

ECOBWEB is a system that learns and uses knowledge for synthesis. It is the core part of BRIDGER, the system that will be discussed later. ECOBWEB is an extension of the concept formation learning program COBWEB (Fisher, 1987) to synthesis tasks. Therefore, many properties of ECOBWEB discussed here are inherited from its predecessor. Since the purpose of this paper is not the evaluation of ECOBWEB, we do not emphasize the distinctions between the two learning systems. Such evaluations are described elsewhere (Reich, 1991b; Reich and Fenves, 1991a). Rather, this section reviews the assumptions underlying ECOBWEB and illustrates its operation using the chair domain which was described in Part I of the study.

2.1 Assumptions

ECOBWEB makes the following assumptions about design that are believed to be appropriate for the preliminary design of a large class of design problems.

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

This assumption restricts the scope of designs to those with fixed topologies. 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, impossible to implement computationally.

ASSUMPTION **2 (Structure of design knowledge):** Design knowledge is represented in a hierarchical classification **tree.**

This assumption does not impose any restrictions on the potential application. The crucial issue is how the knowledge structure is being used.

ASSUMPTION 3 **(Quality of design knowledge):** The quality of design knowledge is recursively determined by a syntactic function called category utility.

To elaborate the assumption, ECOBWEB evaluates a classification of a set of designs into mutually-exclusive classes $C\backslash, C2, \ldots, C_n$ by a statistical function called *category utility (CU):*

$$CU = \frac{\sum_{k=1}^{n} P(C_k) \sum_{i} \sum_{j} P(A_i = V_{ij} | C_k)^2 - \sum_{i} \sum_{j} P(A_i = V_{ij})^2}{n} \quad (1)$$

where *Ck* is a class, A; = V,j is a property-value pair, *P(x)* is the probability of *x,* and *n* is the number of classes. The first term in the numerator measures the expected number of property-value pairs that can be guessed correctly by using the classification. The second term measures the same quantity *without* using the classes. Thus, the category utility measures the expected *increase* of property-value pairs that can be guessed *above* the guess based on frequency alone. The measurement is nonnalized with respect to the number,of classes. The higher is the value of *CU,* the better the quality of the knowledge is.

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. Although the quality assumption compromises quality, it supports the use of an efficient algorithm for building the knowledge[1].

ASSUMPTION **4 (Design process):** Design is a *direct* mapping from the specification to the 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 can only be altered incrementally if new design knowledge warrants it.

[1]**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, 199Id).**

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

Assumption 5 determines that a system that supports design must be able to learn and modify its content incrementally. Assumption 2 establish the structure of the knowledge that needs to be generated and Assumption 3 determines how this knowledge should be evaluated. Assumption 1 restricts the scope of artifact discussed, thereby allowing available learning techniques to be used for the knowledge generation. Of course, 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, 1991b; Reich and Fenves, 1991a), a new system was developed, called ECOBWEB, that will be able to support all the assumptions in design domains.

The next two subsections describe the realization of these assumptions in ECOBWEB learning and synthesis processes.

2.2 ECOBWEB'S learning algorithm

ECOBWEB learns from a sequence of design examples. Examples need not be classified as feasible, optimal, or by any other classification scheme. However, any *a priori* classification can be assigned to an example and treated as any other property. When a new design is introduced, ECOBWEB tries to accommodate it into the existing classification hierarchy starting at the root. The system performs one of the following operators (see (Fisher, 1987) for a detailed description of these operators):

(1) expanding the root, if it does not have any sub-classes, by creating a new class and attaching the root and the new design as its sub-classes;
(2) adding the new design as a new sub-class of the root;
(3) adding the new design to one of the sub-classes of the root;
(4) merging the two best sub-classes and putting the new design into the merged sub-class; or
(5) splitting the best sub-class and again considering all the alternatives.

If the design has been assimilated into an existing sub-class, the process recurses with this class as the top of a new hierarchy. ECOBWEB uses the category utility function *(CU)* to evaluate the results of the operator applications and selects the operator that results in the highest *CU* score.

Figure 2 shows the hierarchy generated from the eight chairs (see Part I of the study (Reich, 1991c) for a detail description) by ECOBWEB. The classes are described with all their properties. The properties

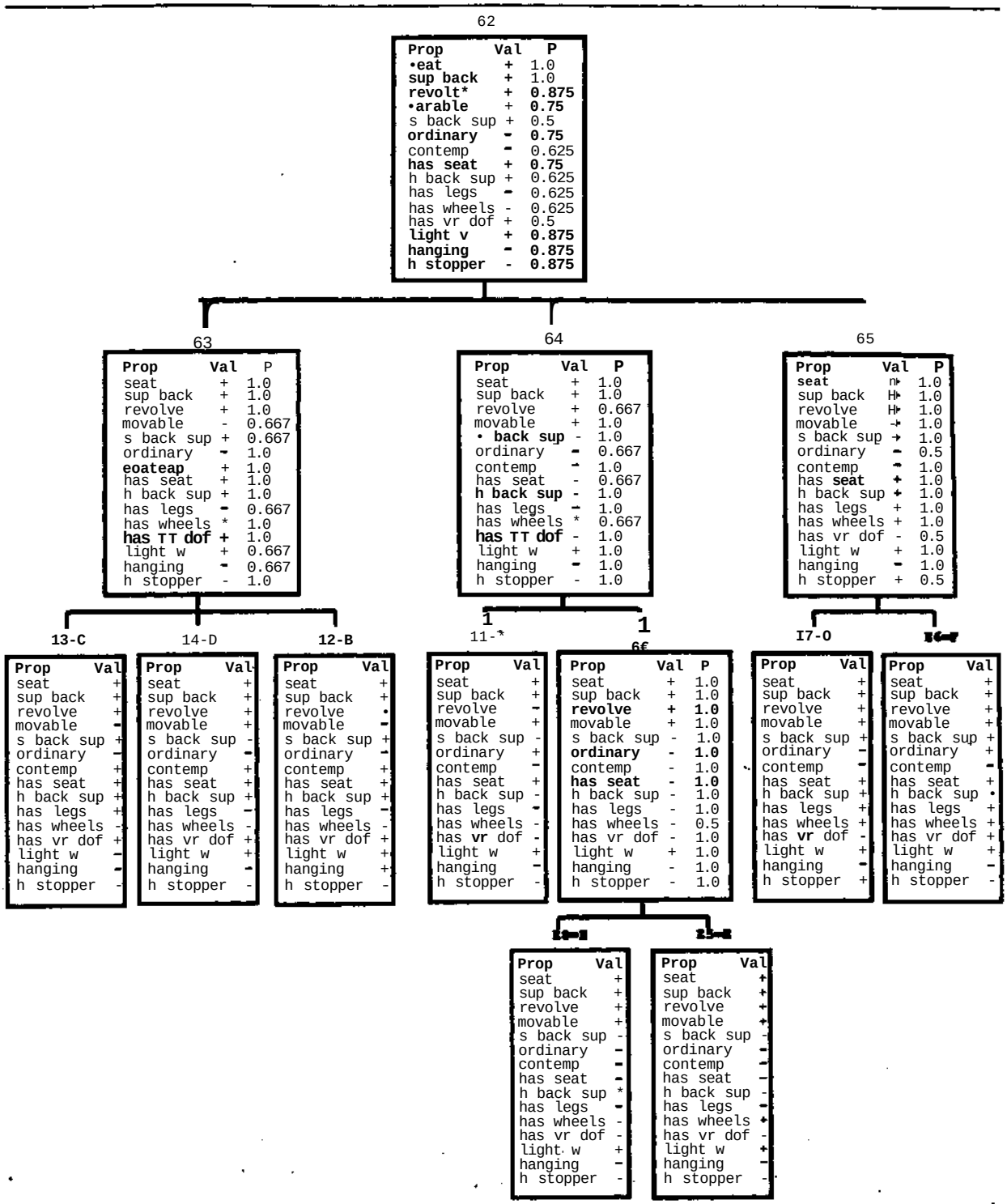

Figure 2: A classification hierarchy of the chair domain generated by ECOBWEB. The property names are obvious abbreviations of those appearing in Tables 1 and 2 of Part I of the study.

that are shown in **bold** font **are** the characteristic properties. 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. Translating this intuition into probability terminology, characteristics are property **values that satisfy *P(Ai = V(;|C*) ≥ threshold* and *P(Ck\Ai = Vij) ≥ threshold,* where *threshold* is a** pre-detennined fixed value that potentially, can be learned for each domain. The figure also shows the name of each class and the value of *P(Ai = Vy|C*)*, denoted by P, for each property value of an abstract concept.

23 ECOBWEB'S **synthesis process**

ECOBWEB has several synthesis methods which can be described along two dimensions: the refinement dimension which can be *extensional* or *intentional;* and the generation dimension which can be *case-based* or *prototype-based.* Figure 3 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 3) until the process terminates at class 3. In the intentional approach, while classifying the new specification, characteristic property-values of the classes traversed (classes 1, 2, and 3 in Figure 3) are assigned to the new design. In the generation stage, the case-based approach views a class as representing the extension 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 missing properties in conjunction with the intentional refinement approach. The prototype-based generation approach takes the current class (class 3 in the example) and uses its property-value pairs to create candidate designs in conjunction with the extensional refinement approach, or to complete several descriptions in conjunction with the intentional refinement approach.

Three design scenarios from the chair domain illustrate the synthesis techniques. The design scenarios assume that the current state of knowledge is as appears in Figure 2.

EXAMPLE 1: The first design scenario deals with designing a chair that is *movable, contemporary* and *stably support back* (Reich, 1991c; Example of Definition 7). Even though, the set of examples that satisfy this specification is empty, ECOBWEB still proceeds to synthesize a candidate. In the *case-based/extensional* method, ECOBWEB starts by selecting G5 for further refinement. It is interesting to note that the selection between classes G5 and G3 is almost arbitrary since both are almost equally good for the first refinement step. The final candidate is a random choice between chairs *F* and G, which both satisfy two of the three specification properties. In the *case-based!intentional* method, the characteristic property values of G2 is used to refine the specification (revolve, seat, support back)

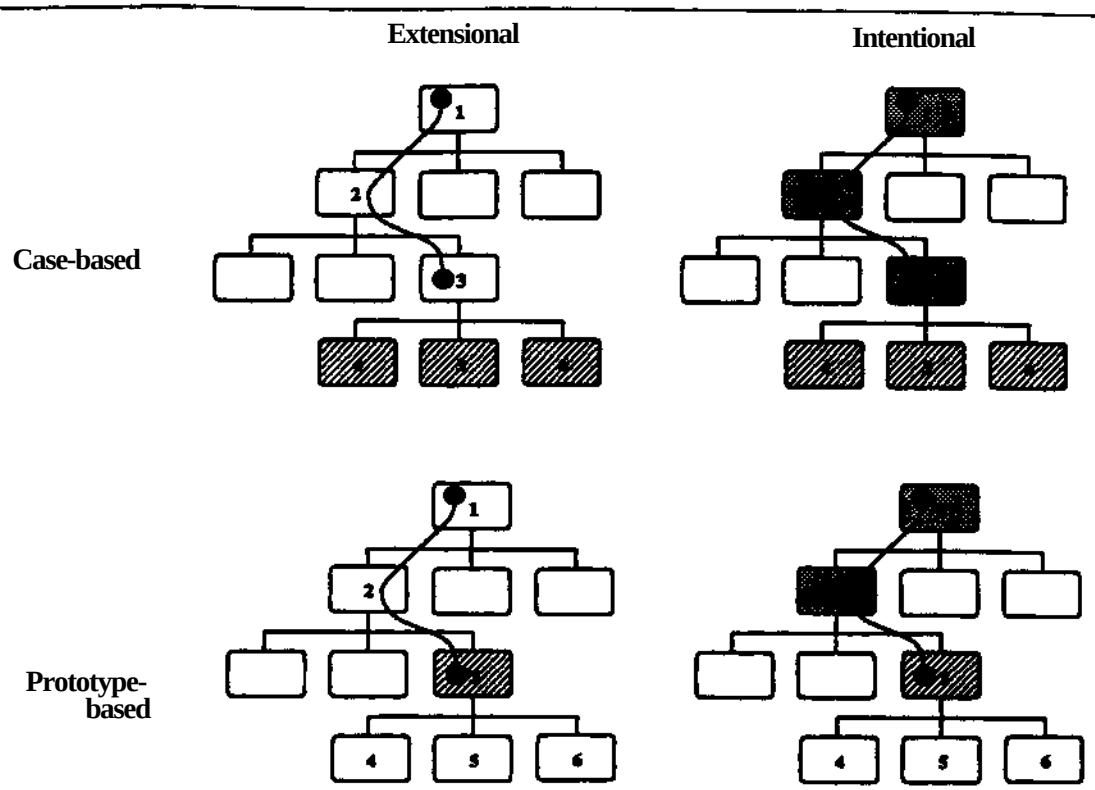

Figure 3: Synthesis methods

and the design (light-weight, not hanging, no stopper). Again, G5 is selected for further refining the design and the final result is the addition of the properties of *F* to the current partial design description. The two *prototype* approaches yield the same behavior since the process does not end at an intermediate class; therefore there is no "true" prototype that can generate several alternatives.

EXAMPLE 2: The second design scenario deals with designing a chair that *revolves* and *is movable* (Reich, 1991c; Example of Theorem 2). In the *case-based/extensional* method, ECOBWEB stops at class G2 since both the specification properties are matches by characteristic values. Therefore, all the eight chairs are candidate designs. In the *case-based/intentional* method, the characteristic property values of G2 are used to refine the specification (ordinary, seat, support back) and the design (has seat, light-weight, not hanging, no stopper). The eight chairs that arc again candidates arc only used to *complete* the partial description accumulated thus far from the characteristic values. In the *prototype!extensional* method, ECOBWEB generates 14 candidates from the various combinations of property values represented in class G2. The *prototype/intentional* approach yields 14 candidates from variations on property values that are not characteristics. As a reference, for the same problem GDT outputs $\{D,E_yF,G\}$ as the set of candidate designs.

EXAMPLE 3: The third problem is to design a *contemporary* chair. The first refinement step chooses class G3. In the *case-based/extensionalmethod,* ECOBWEB selects chairs fl, C, *D* as candidates. This is also the solution that GDT produces. In the *case-based/intentionalmethod,* ECOBWEB generates 6 candidates from class G3. As in example 1, The two *prototype* approaches yield the same behavior since the process does not end at an intermediate class.

There are two observations from the three examples and other examples not described here. First, ECOBWEB is always less conservative than GDT. It generates more alternatives that not always satisfy all the requirements and it generates alternatives that *did not exist* in the original set of potential designs. 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 interpreted as a progressive matching of the specifications or even as a design derivation[2]. This behavior is desirable, even though the coherence of the knowledge structure generated is not conceptualized as a criterion for the success of the learning approach.

2.4 Performance

ECOBWEB was tested in many domains (Reich, 1991b) and demonstrated performance comparable and often better than other learning programs in classification domains. But more important, it demonstrated good performance in design domains (Reich, 1991b). To illustrate the performance, we review some results of ECOBWEB[f]s evaluation in the domain of cable-stayed bridge design (Reich, 1991b).

Table 1: Scaling statistics of candidates

Knowledge	Scaling
Kx	3.07
K_2	2.15
K_3	2.09
K_4	1.32

Table 1 illustrates the performance of ECOBWEB by providing the values of one performance measure generated from statistics of 48 test problems. The measure, called *Scaling,* calculates how close was the span of the synthesized bridge to the required span. The measure is provided for four knowledge hierarchies, *K\,Ki,K^,* and *K4,* generated by learning. Hierarchy *K* was generated from the original 96 bridge examples. Hierarchy *Ki* was generated from the 96 examples after their analysis and redesign;

[2]It is important to acknowledge that the sequence of design description property-value assignments 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.

192 good quality examples, respectively. Since $K\backslash$ was built from lower-quality examples it does not participate in the statistical analyses performed. Two models were hypothesized: (1) scaling linearly depends on the number of examples in a hierarchy; and (2) The logarithm of scaling depends on the logarithm of the number of examples. The latter reflect the power low of practice (Newell and Rosenbloom, 1981).

A General Linear Model (GLM) procedure with a MANOVA analysis was performed to assess the differences between the scaling values observed according to the two models. In both models, the scaling values satisfy: $K\%, K3 >_{o}.oi K^{*}\backslash$ where the >o.oi indicates that K_2 and AT_3 are greater than K_4 with statistical significance at the $p < 0.01$ level and that the difference between K_2 and AT_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. Therefore, *the more knowledge* ECOBWEB *has, the more relevant are the retrieved candidates.* The improvement is not a smooth function, but occurs in steps. This performance evaluation shows that ECOBWEB captures knowledge and gradually uses it more effectively in design.

3 BRIDGER

BRIDGER is a system that assists in the preliminary design of cable-stayed bridges. It contains ECOBWEB as its core part but adds on additional modules that support analysis, redesign and evaluation. These additions are the result of initial studies with BRIDGER when it only contained ECOBWEB (Reich, 1990a). These studies tried to provide a framework in which examples could be assimilated to results in knowledge that highly approximates the state of ideal knowledge, namely, design could be obtained with high accuracy for a given specification. This behavior would require learning many design examples. In reality however, good design performance is expected after learning a moderate number of examples. In this case, the hierarchy would be insufficient to perform as accurately as required. Consequently, additional mechanisms that conect the synthesized designs, such as analysis, redesign, and evaluation, would have to be introduced. This observation confirms the understanding that GDT-IDEAL has a limited scope and that a better model of design (i.e., GDT-REAL) must be devised to account for the imperfections of real design knowledge.

The introduction of analysis, redesign, and evaluation as sub-tasks of design redefines the task of ECOBWEB as being synthesis, rather than design. Therefore, the assumptions stated in Section 2 as underlying design, now relate to synthesis. This section, describes the assumptions underlying BRIDGER development and reviews its architecture.

3.1 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**

ASSUMPTION **6 (Structure of design process): Design is a sequence of five tasks executed sequentially with one feedback loop (Reich, 1990b).**

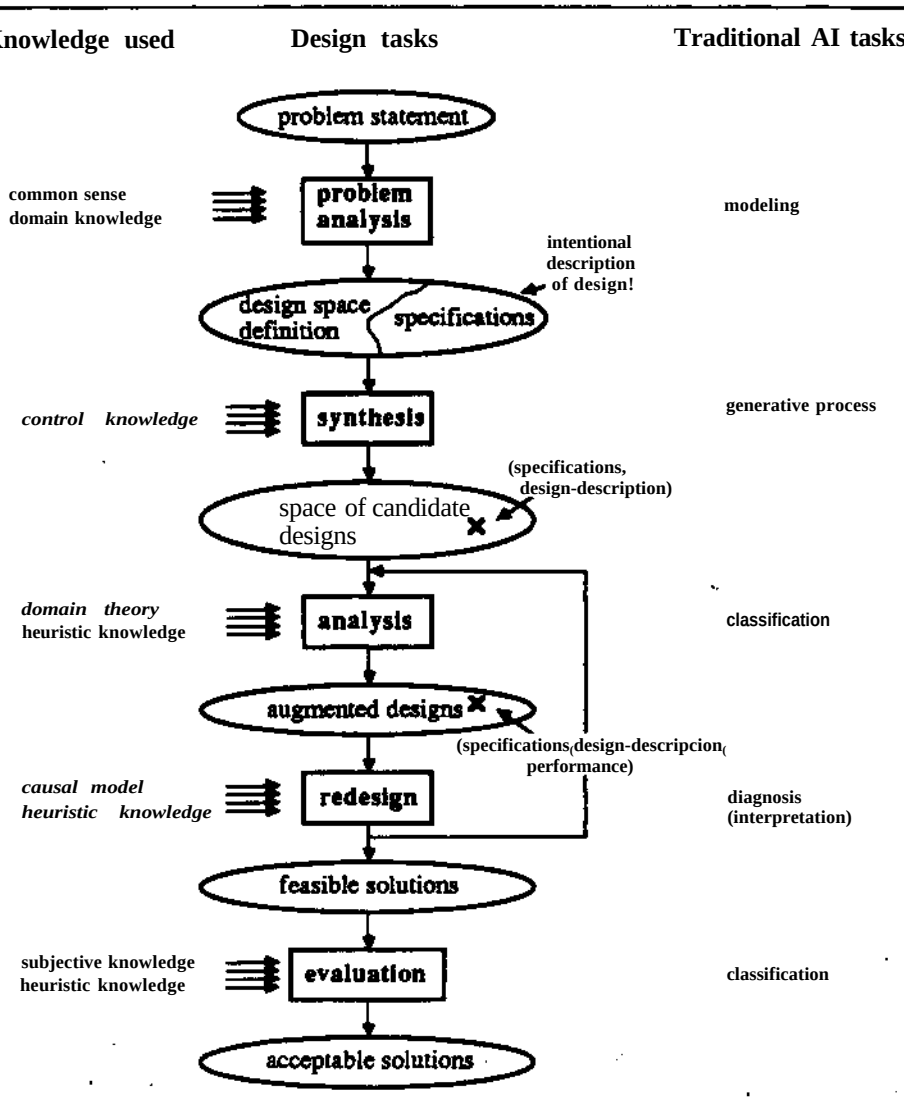

Figure 4: The structure of design

Figure 4 illustrates Assumption 6. The design tasks are denoted by rectangles, and ellipses denote the. information flowing in and out from the tasks. The type of knowledge used in each of the tasks

is shown on the left of each task; only the italicized items are addressed by BRIDGER. The traditional artificial intelligence (AI) task corresponding to each design task is given to the right of each task. This view is a simplification of real design but is sufficient for the intended domain of preliminary design of cable-stayed bridges.

3.2 BRIDGER's architecture

BRIDGER's architecture is based on the design task analysis. Furthermore, BRIDGER's main emphasis is on mechanisms for synthesis knowledge acquisition and therefore, the complete architecture is intended to support this task. Figure 5 shows an overall view of BRIDGER's architecture which consists of two main systems: synthesis and redesign.

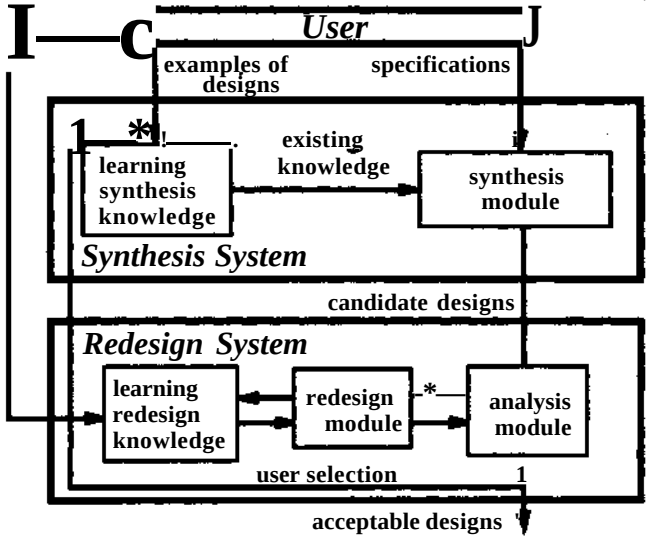

Figure 5: Overview of BRIDGER's architecture

The synthesis system is responsible for synthesizing several candidates from a given specification. Synthesis knowledge is generated by learning from existing designs and from successful design examples that are selected by the user. The synthesis module contains two instantiations of ECOBWEB.

The first instantiation creates a classification hierarchy from the original bridge examples. This hierarchy is subsequently used to synthesize bridges as discussed in Section 2. When dealing with specifications that arc 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 onler for performing sensible scaling, relevant scaling values are retrieved from a second instantiation of ECOBWEB. This instantiation builds the 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, 1991a; Reich, 1991b) for additional details). Since both classification hierarchies represent heuristic knowledge, candidate designs are usually inadequate in some aspects even after the adaptation process. The redesign system resolves this problem.

Candidate designs are transferred to a module that analyzes them and submits them to a redesign module, if necessary. The redesign module is responsible for modifying designs after their analysis. 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. The results of the redesign system are acceptable designs. The designer evaluates the results and can submit a subset of them to the synthesis system for further training.

There are many possible interactions between the different modules and within their internal mechanisms. BRIDGER exercises a simple sequential control strategy. At the beginning of any design session, BRIDGER loads the knowledge base from permanent storage and waits for an input problem. When a new specification is obtained, BRIDGER generates a list of candidate designs. Each of the candidates is submitted to an iterative cycle of analysis and redesign. When this cycle terminates, the human user evaluates the results and submits a subset of them to the synthesis system for further training. At the end of the session, BRIDGER stores the current knowledge base.

3.3 Performance

BRIDGER's performance was partially assessed quantitatively. 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 and is not discussed further. The adaptation part was tested by calculating the *Quality* of the bridges, where the quality is a weighted summation of the squares of the constraints that a candidate bridge violates.

Table 2: Quality statistics of candidates

Knowledge	Quality
K_X	278.36
K_2	50.19
K_3	2.89
K_4	1.20

Table 2 illustrates the performance of BRIDGER by providing the values of the quality measure

generated from statistics of 48 test problems. The same models that were used to test scaling in the previous section were used again: (1) a linear model, and (2) a power law model. According to the first model, the quality values satisfy: *Kz* >o.oi *Kz,K** but according to the second: *Kz* >o.oi £3 >o.oi £4. The second model was better but no statistical test was performed to differentiate between the two models. **Both results says that *the more knowledge* BRIDGER *has, the better the quality of candidates it generates.*** The second model describes the behavior more conclusively. This demonstrates the importance of hypothesizing a model and testing it even in this problem.

This evaluation shows fast convergence of BRIDGER to good design competence. Future evaluations may include the assessment of the redesign and the evaluation of the complete system.

4 A comparison

The comparison between GDT and BRIDGER is done in two stages. First, both GDT-IDEAL and GDT-REAL, denoting GDT in the state of ideal knowledge and in the state of real knowledge, respectively, are compared with ECOBWEB. Second, GDT-REAL is compared with BRIDGER.

4.1 Comparison between GDT and ECOBWEB

This section establishes a correspondence between GDT and ECOBWEB. First the assumptions underlying both approaches are compared. Second, the mapping of GDT's concepts into ECOBWEB's knowledge representation scheme is discussed; and third, the mapping is used to explain the learning processes in terms of improvement in real knowledge. Figure 6 reflects the mapping between GDT and ECOBWEB as discussed below.

4.1.1 Assumptions

Artifact representation.

GDT. GDT determines that entities can be described intentionally and abstract concepts extensionally. Also, when specifying design solutions, the attribute representation is used and with potentially infinite number of attributes. GDT-REAL relaxes the number of attributes to be finite with the use of models to describe entities.

ECOBWEB. Assumption 1 states that artifacts are represented by finite lists of property-value pairs. The representation of artifacts is intentional which is in correspondence with GDT. The finiteness of

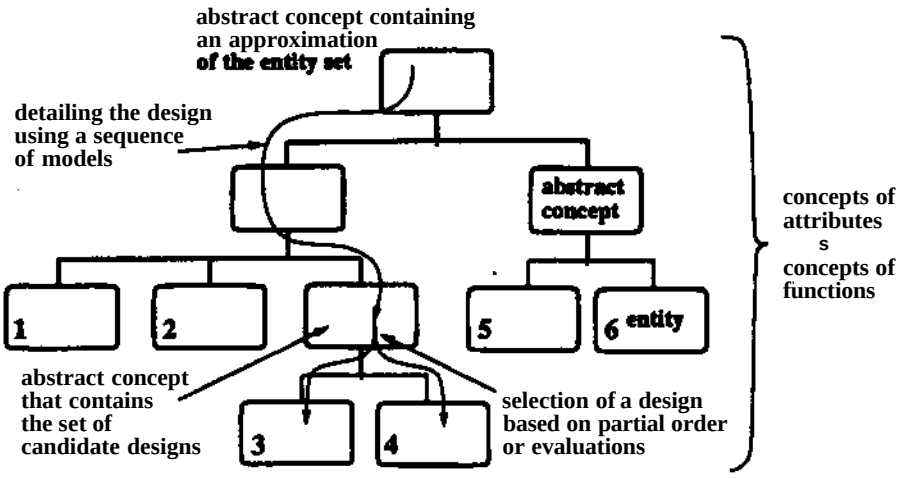

Figure 6: Mapping GDT onto ECOBWEB

the representation is in correspondence with GDT-REAL.

Structure of knowledge and its quality.

GDT. Axiom 3 states that design knowledge is a topology. Since knowledge is a topology and ideal knowledge can separate well between entities, the quality of knowledge is perfect. It therefore never changes. Hypothesis 3 suggests that in reality, however, knowledge structure is hierarchical.

ECOBWEB. Assumption 2 states that design knowledge is represented by a hierarchy. The hierarchy is heuristically organized toward good design performance as implicitly embedded in *CU*. The classes in the hierarchy are extensions of their entities. They represent abstract concepts in the domain. Therefore, although entities are originally represented in an intentional form, knowledge structure is extensional. Each class has also an approximate intentional representation by the characteristic properties. This representation alleviates the difficulty of understanding the complete meaning of extensional descriptions. The hierarchical structure only approximates topology by explicitly defining which unions of abstract concept are part of the knowledge.

Design process.

GDT. Theorem 2 says that a specification can be described by the intersection of abstract concepts. Definitions 12 defines design as a designation of a domain in the attribute space that corresponds to the domain used to describe the specification. Theorem 9 determines that this process terminates

with a design solution when the specification is given. GDT-REAL relaxes the nature of design to an evolutionary process that terminates with a set of candidate designs.

ECOBWEB. A specification is described by abstract concepts. Assumption 4 says that the mapping from the specification to the design is direct. This is in correspondence with Theorem 9. The design terminates when the specification is described completely, but with a set of candidate designs, not with a single solution. This was demonstrated by the design examples in Section 2.3. Later we see that the same process can be viewed as in correspondence with GDT-REAL as well.

Nature of design knowledge.

GDT. Since design knowledge is perfect, it is static. Knowledge remains static in GDT-REAL in spite of its inherent imperfections.

ECOBWEB. Design knowledge only approximates perfect knowledge. Design knowledge evolves by the assimilation of new information. Therefore, knowledge increasingly becomes more refined and closer to ideal knowledge through learning. Learning is the main difference between ECOBWEB and GDT; it does not correspond to any process in the framework of GDT. Learning consists of the addition of abstract entity concepts that modifies the topology of the domain; thereby, leading to enhanced correspondence between the specifications and the design descriptions, and to a better design process. While learning, abstract concepts may be changed (i.e., by the add-to-existing-class operator), constructed (i.e., by the merge or add-new-class operators), or eliminated (i.e., by the split operator). Only useful concepts are generated and retained.

Learning gradually enhances real knowledge by increasing its scope and refining its granularity. The scope of knowledge expands by learning additional property-value pairs that describe artifacts. Additional similar designs add information that refines the existing abstract concepts such that better designs can be obtained for each specification. These two enhancements advance real knowledge towards "ideal" knowledge.

4.1.2 Performance versus prediction

This is the most crucial comparison between GDT and ECOBWEB. It evaluates the purpose of building ECOBWEB by contrasting its performance with the predictions produced by the theory. The discrepancies between the performance and the predictions are then discussed in light of the differences in the assumptions between GDT and ECOBWEB.

Design process.

GDT. There is always a solution to *a feasible* specification, and it is found when the specification is provided. GDT-REAL relaxes this prediction to guarantee finding a set of candidate designs for a feasible specification.

ECOBWEB. ECOBWEB finds a set of candidate designs for any specification: feasible or infeasible. It finds the solutions by trying to satisfy heuristically as many requirements as possible. The candidate designs may have unexpected behavior as suggested by Theorem 34 of GDT-REAL: some candidate may violate a given specification requirement. In addition, not all the possible good designs are generated by ECOBWEB. The behavior described becomes increasingly better as more knowledge is accumulated. Statistical tests show that good performance can be obtained with a relatively moderate number of examples. Therefore, ECOBWEB gradually approximates better the predictions of GDT about design performance.

Nature of knowledge.

GDT. Topological structure of knowledge is the key to obtain a perfect design performance. In GDT-REAL, compactness is also required to facilitate the mapping between the specification to the candidate designs.

ECOBWEB. Design knowledge improves with additional information. The knowledge structure is restricted to a hierarchy. This structure is tuned to capture the specifications that are "most likely"[9] to arise in the future, based on past experience. Although there is no chance that the hierarchy will ever be a topology, the more examples are assimilated the better is the quality of knowledge.

ECOBWEB can be compared also to GDT-REAL. In this case, the abstract concepts of the hierarchy are perceived as models that mediate between the specification and the final design. These models do not contain any semantic information. This may be acceptable theoretically but not practically. An elaboration on this subject appears in the comparison between BRIDGER and GDT-REAL.

It can be observed that the ideal topological representation of knowledge cannot be generated by ECOBWEB. This precludes making precise predictions about the ability to design. Nevertheless, ECOBWEB demonstrates increasingly competent design ability when it learns, using a considerably less restrictive knowledge structure. This has two implications. First, if we wish that ECOBWEB will design better, we need to have it generate a knowledge structure that is closer to topology. A step towards this end is the generation of dynamic graph structures of knowledge, currently under development. Second, if GDT is

to be more relevant to real domains, the topological structure of knowledge must be relaxed. This may cause making less precise statements about design, but still statements that can further guide experimental research.

The predictions of the revised GDT can aid in measuring BRIDGER's performance (i.e., how far is its knowledge from the ultimate "ideal" knowledge?). This measurement is critical if we wish to have some measure of confidence in BRIDGER's solutions other than the indication from experimental studies. Also, such analysis will allow the prediction of the number of design examples necessary for producing a given desired level of design performance.

4.2 A comparison between GDT-REAL and BRIDGER

Since BRIDGER design knowledge is captured by ECOBWEB, this section only compares the design process in both BRIDGER and GDT-REAL. This section illustrates how Assumption 6 that is illustrated in Figure 4 and implemented in BRIDGER corresponds to the design process in GDT-REAL.

Figure 4 matches GDT-REAL design process via the use of models (Figure 6 in (Tomiyama et al., 1989)). A refinement in the terminology of GDT corresponds to the synthesis task, and local redesign mechanisms correspond exactly to the redesign task. Meta-models are the mechanisms that generate the different points of view of the artifact and allow the application of analysis and specific redesign methods.

The above correspondence refers to the state of real knowledge. In addition, all tasks re-iterate when different meta-models arc used to generate different point of views of the artifact. In the presence of ideal knowledge, the mapping is different. In ideal knowledge, only straightforward synthesis occurs.

In the previous section, the synthesis module was described as implementing GDT's knowledge representation and processes. This analysis may be viewed as imputing considerable functionality and capability to the simple synthesis mechanisms currently implemented in BRIDGER. This section shows how BRIDGER as a whole can be viewed as an instantiation of the metamodel view of design in the context of GDT.

Reiterating on the convergence process with the aid of models, in real knowledge there is no direct mapping from specifications to design descriptions. Models of the artifact serve to iteratively and incrementally detail the design. Each model is a representation of the artifact based on some theory (e.g., linear behavior theory). GDT states that there is a convergent sequence of models that is bound to converge at candidate design solutions.

Figure 7 shows a possible sequence of models that mediates the convergence from specifications to

design descriptions. BRIDGER can be viewed as the instantiation of a single model: the linear behavior model. Bridges arc detailed by synthesis to a level that allows for their analysis according to the model. Backtracking is used to modify the design until it satisfies the behavioral constraints associated with the model (e.g., stresses and deflection constraints). Additional models can be implemented and used to complete the transformation from the specification to the final design description.

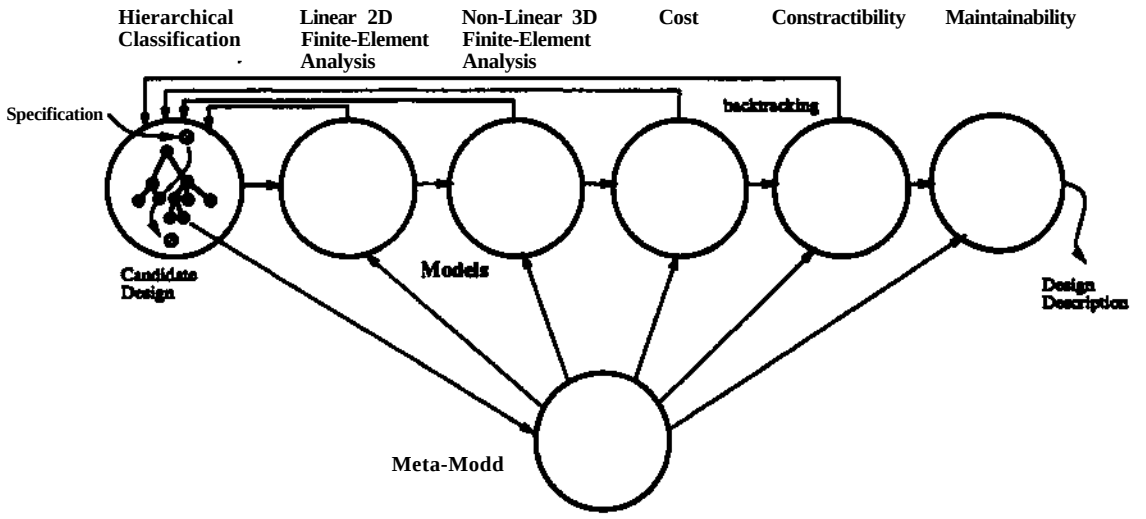

Figure 7: A comparison between GDT (state of real knowledge) and BRIDGER

Since there is no intermediate/approximate analysis in BRIDGER, every design must be detailed to the same level. This is not a desirable situation. For example, it prevents the output of abstract designs for given abstract specifications. This situation can be remedied by the addition of approximate models and by modifying the control of BRIDGER to sequentially use these models. Such a sequence of approximate models is illustrated in Figure 8. It expands the model of the generation of candidate designs shown in Figure 7. The mechanisms shown make use of a single synthesis hierarchy.

To illustrate the use of such a sequence of models we use the chairs domain. The design problem is to detail a chair that is *movable, contemporary,* and that *stably support back.* Assume that based on domain knowledge the first specification property to be satisfied is the *stably support back.* The specification is sorted through the hierarchy until a characteristic value of *stably support back* is found (point 2 on Figure 8). At this stage, a simple analysis (e.g., 2D stability) is performed. This analysis provides feedback to the path from 1 to 2. Then, the classification of the specification, now augmented by the *stably support back* property, continues to 3 where the *movable* property is determined by matching it with a characteristic values. The analysis performed at this stage is visual (e.g., determine whether the chair can move), or physical (e.g., determine whether the chair can be carried). The feedback from this analysis may only

change the path from 2 to 3. Last, the style specification is satisfied by a similar process.

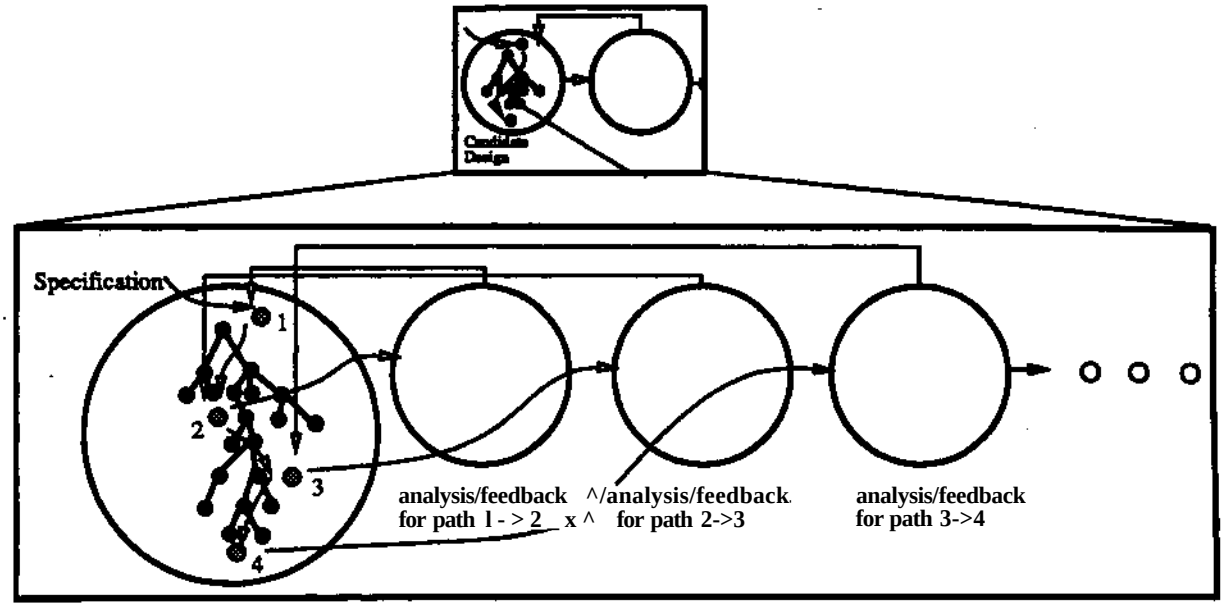

Figure 8: A sequence of approximate models

The above procedure would allow BRIDGER to exercise intent and domain knowledge and essentially follow a specific design derivation. Learning such derivations may be a task for future research. No change needs to be made to the synthesis module when approximate models are used, since it fully supports the generation of abstract designs. A different implementation may include a separate hierarchy for each path. This, however, is not expected to provide better results if sufficient number of examples is provided.

The use of a separate hierarchy for each path has an advantage over a single hierarchy for representing flexible artifact rcpicsentations. In fact, it can better support extensional descriptions of artifacts than the single hierarchy. Using a separate hierarchy for supporting the refinement associated with each model used, does not present any difficulty. In fact, although not detailed here, the synthesis module of BRIDGER makes use of two instantiations of ECOBWEB that each provide different information for the synthesis process (Reich, 1991a; Reich, 1991b): one hierarchy is used to synthesize candidates and the other to retrieve scaling values that modify the candidates to fit the specification.

To summarize, the comparison between BRIDGER and GDT-REAL shows how BRIDGER can be extended to account for additional models and design concerns and still remains within the framework ofGDT.

5 Summary

Part I of the study discussed three methods for conducting design research and focussed on one instance of a theoretical approach to design. That part critically discussed the ramifications of some of GDT's crucial assumptions and how they can be relaxed. This critical review is the first contribution of this study. Since, the review suggested ways in which the assumptions of the theory can be potentially relaxed thereby expanding the scope of the theory, the review can be treated as a simple form of executing the scientific method. This method is the subject of Part II of the study.

Part II shows how an experimental system for design, called BRIDGER, is built on the foundations of GDT. This constitutes step 2 in the scientific cycle. Therefore, GDT serves as the theoretical foundation of BRIDGER. For example, GDT supports the use of concept formation as a method for synthesis knowledge acquisition since concept formation gradually better approximates topology. The implementation of BRIDGER (step 3 in the cycle) was tested in experiments and found to produce successful results. These results provide experimental support for GDT as a theory of design.

A careful analysis of the discrepancies between GDT and BRIDGER shows that GDT assumptions are too restrictive. These assumptions are necessary for *guaranteeing* a perfect design performance. In reality, a lesser quality design is acceptable, and this is obtained gradually through learning, by an experimental system that makes considerably less restrictive assumptions about design.

This analysis contributes to both the theory and the experimental system. The contribution to both studies demonstrates the benefits from executing the scientific method. By illustrating these benefits, this study advocates for subscribing to the scientific methodology. It is our belief that following this method is also essential for generating high quality design research.

The last contribution of this study is the presentation of learning as a necessary ingredient of design systems; learning generates and refines knowledge structures that gradually better approximate topologies. BRIDGER has demonstrated that the introduction of learning is not a burden but a necessary ingredient that can be integrated naturally with the design activity. Beside this, *learning implicitly forces the use of an adequate research method since it requires the use of experiments to show improvement in design performance.* This necessitates generating some performance metrics, even if overly simplistic, and using them to test performance.

Acknowledgments

This work has supported in part by the Engineering Design Research Center, a National Science Foundation Engineering Research Center.

References

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

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, NJ.

Reich, Y (1990a). 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 (1990b). Design knowledge acquisition: Task analysis and a partial implementation. *Knowledge Acquisition,* in Press.

Reich, Y (1991a). The acquisition and utilization of basic esthetic criteria in design. Technical Report EDRC 12-39-91, Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, PA.

Reich, Y (1991b). *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 (1991c). 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 (1991d). The value of design knowledge. Unpublished manuscript.

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.

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.

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

Yoshikawa, H. (1981). General Design Theory and a CAD system. In Sata, T. and Warman, E., editors, *Man-Machine Communication In CAD/CAM$_f$ Proceedings of the IFIP WG5.2-53 Working Conference,* pages 35-57. North-Holland, Amsterdam.