





3 1293 01029 8978

This is to certify that the

dissertation entitled

GENERATING MULTIPLE DESIGNS OF
COMPOSITE MATERIALS USING
A GENERIC TASK APPROACH

presented by

Ahmed Mohamed Kamel

has been accepted towards fulfillment
of the requirements for

Ph.D. degree in Computer Science

Major professor

Date April 7, 1994

LIBRARY
Michigan State
University

PLACE IN RETURN BOX to remove this checkout from your record.
TO AVOID FINES return on or before date due.

DATE DUE	DATE DUE	DATE DUE
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____

**GENERATING MULTIPLE DESIGN ALTERNATIVES OF
COMPOSITE MATERIALS USING A GENERIC TASK APPROACH**

By

Ahmed Mohamed Kamel

A DISSERTATION

**Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of**

DOCTOR OF PHILOSOPHY

Department of Computer Science

1994

ABSTRACT

GENERATING MULTIPLE DESIGN ALTERNATIVES OF COMPOSITE MATERIALS USING A GENERIC TASK APPROACH

By

Ahmed Mohamed Kamel

Many engineering design situations require the generation of multiple designs to meet a common set of specifications. This research introduces an effective approach for generating multiple designs. The introduced architecture both produces multiple designs, and in a second step ranks the resultant designs according to set criteria.

The method developed utilizes and builds on the generic task approach to knowledge-based systems, as well as the specific design technique, known as Routine Design.

This research has five main contributions in design knowledge-based systems and in polymer composite materials design.

In knowledge-based systems, there are three contributions:

1. The proposal of an integrated architecture for knowledge-based design problem solving. This architecture can produce designs by altering previous similar designs as well as produce designs “from scratch”. The architecture also provides a means for “testing” the generated designs.
2. The development of an effective approach for generating multiple designs to meet a common set of specifications.

3. The definition of MDSPL, a language for analyzing and implementing design systems based on the Multiple Design approach. This language is implemented in the form of a set of diagrammatic browsers for browsing and editing design problem solving systems.

In polymer composite materials, there are also 2 contributions:

1. This research introduces an integrated architecture for the material design for polymer composite materials.
2. The design and implementation of a thin film fiber reinforced polymer composite materials design system using the MDSPL language. This system is intended to be an industrial aid for composite materials designers. It is also intended to be part of the integrated design architecture intended for automating the design process.

To my wonderful wife and children: Mona, Ashraf and Ayah

ACKNOWLEDGEMENTS

Research reported here has been directly supported under the ARPA MADE Program (ARPA Order Number 8673). General additional support has come from the NSF Center for High Speed, Low Cost Polymer Composite Processing at Michigan State. Research in the Intelligent Systems Laboratory is generously supported by equipment contributions from Apple Computer.

I would like to thank my advisor, Dr. Jon Sticklen for his support and encouragement during the entire course of my doctoral studies. Without him, this simply would have not been possible.

I am deeply grateful to my wonderful family, my wife Mona and my children Ashraf and Ayah for enduring the long periods of a busy “Daddy” while doing this research and writing this dissertation.

I am indebted to my colleagues in the Intelligent Systems Laboratory for the constructive discussions we have had about the ideas behind this research. I am especially thankful to Ms. Valerie Adegbite, Mr. Mahmoud Pegah, and Dr. Eugene Wallingford.

Last, but not least, I would like to thank my parents for the love of learning that they planted in me. Without this, I would have never undertaken this endeavor.

Table of Contents

List of Tables.....	x
List of Figures.....	xi
Chapter 1: Introduction.....	1
1.1 Motivation.....	1
1.2 The Design Problem	3
1.3 The Generic-Task Approach	4
1.4 Integrated Problem Solving Architecture for the Design of Composite Materials	10
1.5 The Multiple Design Approach.....	11
1.6 Organization of the Dissertation	12
Chapter 2: Previous Research: Background.....	14
2.1 Knowledge-Based Design Systems	14
2.1.1 First Generation Design Approaches	14
2.1.1.1 Rule-Based Systems	14
2.1.1.2 Blackboard Architectures	16
2.1.2 Case-Based Design	19
2.1.3 Task-Specific Design Approaches.....	22
2.1.3.1 SALT	24
2.1.3.2 Routine Design	25
2.1.3.3 The Micon Microprocessor Configuration System	25
2.1.3.4 The HI-RISE Structural Design System.....	27
2.1.3.5 Comparison.....	27
2.1.4 Approaches Based On Integrating Multiple Problem Solvers.....	28
2.1.4.1 Goel's KRITIK Architecture	29
2.1.4.2 Punch's TIPS Architecture.....	29
2.1.4.3 SOAR	30
2.2 Other Planning Systems.....	31
2.3 Knowledge-Based Systems in Composite Materials	35
2.3.1 Knowledge-Based Systems for the Design of Composite Materials	35
2.3.2 Knowledge-Based Systems for the Fabrication of Composite Materials	35
2.3.3 Trends of Knowledge-Based Systems Applications in Composite Materials.....	37

Chapter 3: Previous Research on Routine Design.....	39
3.1 Classifications of Design.....	39
3.2 Routine Design Problem Solving.....	43
3.2.1 Problem Solving Agents in Routine Design	45
3.2.1.1 Specialists	45
3.2.1.2 Plans	46
3.2.1.3 Step.....	46
3.2.1.4 Task.....	46
3.2.1.5 Plan Sponsor.....	46
3.2.1.6 Plan Selector.....	47
3.2.1.7 Constraints	47
3.2.1.8 Failure Handlers	47
3.2.1.9 Redesigners.....	47
3.2.2 Problem Solving in Routine Design	47
3.2.3 Representative Routine Design Applications.....	50
3.2.3.1 An Engineering Design Example	50
3.2.3.2 A Planning Example	51
3.2.3.3 Discussion.....	52
3.3 Other Routine Design Research.....	52
Chapter 4: Problem Definition and Research Goals.....	55
4.1 The Nature of Knowledge-Based Systems Research.....	55
4.2 Problem Definition.....	58
4.2.1 The Composite Materials Design Problem	59
4.2.2 Knowledge-Based Extensions Needed for the Design Of Composite Materials.....	64
4.3 Research Goals.....	66
4.3.1 Specific Goals in Composite Materials.....	66
4.3.2 Specific Goals in Knowledge-Based Systems	67
4.4 Expected Outcomes	68
4.4.1 Expected Outcomes in Composite Materials	68
4.4.2 Expected Outcomes in Knowledge-Based Systems	68
4.5 Conclusion	69
Chapter 5: An Integrated Architecture for the Design of Thermoset Polymer Composite Materials.....	70
5.1 Single Design Architecture	70
5.2 Multiple Design Architecture.....	73
Chapter 6: The Multiple Design Approach and Implementation.....	76
6.1 The Relationship Between Multiple Design and Routine Design	76
6.2 The Design Database	76
6.3 Knowledge Representation Structures	78

6.4	Comparison Between Design Limiters and Constraints	79
6.5	The Control Strategy	80
6.6	Problem Solving in MDSPL	86
6.7	The MDSPL Language	87
6.7.1	The Specialist Hierarchy Browser	88
6.7.2	The Agents Hierarchy Browser	88
6.7.3	The Specialist Browser	89
6.7.4	The Plan Browser	91
6.7.5	The Task Browser	91
6.7.6	The Step Browser	92
6.7.7	The Constraint Browser	93
6.7.8	The Table Browser	93
6.7.9	The User Interface	96
6.7.9.1	The Input Browser	96
6.7.9.2	The Output Browser	97
6.8	Comparison of the Computational Expense in DSPL and MDSPL	98
6.8.1	Plan Selection	99
6.8.2	Step Evaluation	99
6.8.3	Constraint Evaluation	100
6.8.4	Selecting Multiple Plans versus Failure Handling	101
6.8.5	Discussion	101
6.9	Comparison Between MDSPL and Search Techniques	102
Chapter 7: A Polymer Composite Materials Design Example		104
7.1	Problem Description	104
7.2	Routine Design System for Material Design of Epoxy Resins	106
7.3	A Sample Single Design	107
7.4	A Sample Multiple Design	108
7.5	Discussion	115
7.6	Analysis of the Composite Materials MDSPL System	121
7.6.1	Assumptions	123
7.6.2	Definitions	125
7.6.3	Quantitative Analysis	126
7.6.4	Expectations for Future Extensions to the Composites Design System	129
7.6.5	Expected Impact of Future Extensions on System Performance	130
7.7	Generalized Performance Analysis of MDSPL	132
Chapter 8: Ranking and Selection of Composite Materials Designs		135
8.1	Selection Criteria	135
8.2	Materials Database	136

8.3	Ranking Algorithm.....	136
8.4	Output	137
8.5	Two Case Studies of the Materials Design MDSPL System	139
8.5.1	Example 1	140
8.5.2	Example 2	146
8.6	Discussion	151
Chapter 9: Contributions and Future Research.....		153
9.1	Generating Multiple Designs	154
9.2	The MDSPL Language	156
9.3	Composite Materials Design.....	156
9.4	Other Work in Progress.....	157
Bibliograghy.....		159

List of Tables

TABLE 1. Values for Current and Future Design System.....	131
TABLE 2. Estimated Running Times for the Composite Materials MDSPL System	132
TABLE 3. Expected Worst Case Running Times for Different Problem Sizes.....	133
TABLE 4. Prices of Raw Materials for Example 1	141
TABLE 5. Prices of Raw Materials for Example 2	150

List of Figures

Figure 1:	Information Processing Task for the Design of Composite Materials	2
Figure 2:	Classification of Design According to the Level of Routineness	4
Figure 3:	Generic-Task Problem Solvers	8
Figure 4:	KADS problem solvers	8
Figure 5:	An Object in SIGHTPLAN (from [Tommelein, Johnson, Hayes-Roth, & Levitt, 1987]).....	17
Figure 6:	Constraints in SIGHTPLAN (from[Tommelein, Johnson, Hayes-Roth, & Levitt, 1987]).....	18
Figure 7:	Generic Task Problem Solvers	23
Figure 8:	KADS problem solvers	23
Figure 9:	Routine Design Specialist Hierarchy	26
Figure 10:	Example Problem with Multiple Goals.....	34
Figure 11:	Decomposition of the Design Problem of a Computer System	41
Figure 12:	Classification of Design According to the Level of Routineness	43
Figure 13:	The Orthogonal Axes of Routineness and Conceptualism	44
Figure 14:	Specialist Hierarchy for AIR-CYL System.....	50
Figure 15:	Specialist Hierarchy for MPA System	51
Figure 16:	Inference Structure of MYCIN (from [Clancey, 1985])	57
Figure 17:	Inference Structure of Heuristic Classification (from [Clancey, 1985])	58
Figure 18:	Types of Matrices Used in Composite Materials	59
Figure 19:	An Information Processing View of the Composite Materials Fabrication Life Cycle.....	60
Figure 20:	Sample Input/output for the Design of Composite Materials	63
Figure 21:	Single Design System Architecture	71
Figure 22:	Multiple Design Architecture.....	74
Figure 23:	A Graphic Representation Of A Typical Multiple Design Output	77
Figure 24:	An Example Showing Plan Selection in DSPL	82
Figure 25:	An Example Showing Plan Selection in MDSPL.....	83
Figure 26:	A Subspecialist Hierarchy Browser	88
Figure 27:	An Agents Hierarchy Browser	89
Figure 28:	A Specialist Browser.....	90
Figure 29:	A Plan Browser	91
Figure 30:	A Task Browser	92

Figure 31:	A Step Browser	93
Figure 32:	A Constraint Browser.....	94
Figure 33:	A Table Matcher.....	95
Figure 34:	An Example MDSPL Input Browser	96
Figure 35:	An Example User Query	97
Figure 36:	An Example Single Design Output Browser	98
Figure 37:	The Specialist Hierarchy for The Composite Materials Design System .	106
Figure 38:	Input Specification Screen	108
Figure 39:	Single Design Output Browser	109
Figure 40:	Part 1 of the Multiple Design Output Browser	111
Figure 41:	Part 2 of the Multiple Design Output Browser	112
Figure 42:	Table Editor Showing the Pattern Matcher for Selecting an Epoxy	113
Figure 43:	Table Editor Showing the Pattern Matcher for the Glass Fiber Plan Sponsor.....	114
Figure 44:	Table Editor Showing the Pattern Matcher for the Carbon Fiber Matcher Plan Sponsor.....	115
Figure 45:	A Design Limiter.....	116
Figure 46:	Part 1 of the Output of Example 2	117
Figure 47:	Part 2 of the Output of Example 2	118
Figure 48:	Part 1 of the Output of Example 3	119
Figure 49:	Part 2 of the Output of Example 3	120
Figure 50:	Ranked Designs	138
Figure 51:	An Individual Design.....	139
Figure 52:	Input Data for Example 1	140
Figure 53:	Single Design Output for Example 1	141
Figure 54:	Part 1 of the MDSPL Output of Example 1	142
Figure 55:	Part 2 of the MDSPL Output of Example 1	143
Figure 56:	Highest Ranked Design for Example 1 (a)	144
Figure 57:	Highest Ranking Design for Example 1 (b).....	145
Figure 58:	Highest Ranking Design for Example 1 (c).....	145
Figure 59:	Ranked List of Designs for Example 1	146
Figure 60:	Input Data for Example 2.....	147
Figure 61:	Single Design Output for Example 2	147
Figure 62:	Part1 of the MDSPL Output of Example 2	148
Figure 63:	Part 2 of the MDSPL Output of Example 2	149
Figure 64:	Ranked Designs for Example 2.....	150
Figure 65:	The Highest Ranking Design for Example 2	151

Chapter 1

Introduction

The research focus of this dissertation is engineering design, more specifically the design of thermoset polymer composite materials. In this chapter, I present an overview of the research reported here.

This work is built on the foundation of the generic-task approach for knowledge-based systems [Chandrasekaran, 1983; Chandrasekaran, 1986], which is briefly described in this chapter. Additionally, a brief description of an integrated architecture for the design of composite materials based on integrating generic-task modules is also discussed, and a brief description of the MDSPL approach for generating multiple designs to a common set of specifications is described.

The organization of the rest of the dissertation is described at the end of the chapter.

1.1 Motivation

The emerging field of composite materials offers one pivotal area for the establishment of a revitalized American industrial base. There is a key enabling step for realizing this potential in which knowledge-based systems techniques may prove to be important: enabling a rapid “specifications to manufacturing” time; i.e. shortening the time between setting material specification, and successful realization of a material which meets those specifications. This key step is largely dependent on an ability to capture existing design knowledge and rapidly modify it in light of altered product specifications.

The research reported here was initiated to provide a system for the design to specifications of thermoset polymer composite materials; that is a system for generating a list of ingredients and a processing protocol to manufacture a composite material out of them

based on a set of requirements on the material (e.g. structural properties, and use profiles). These requirements are shown schematically in Figure 1.

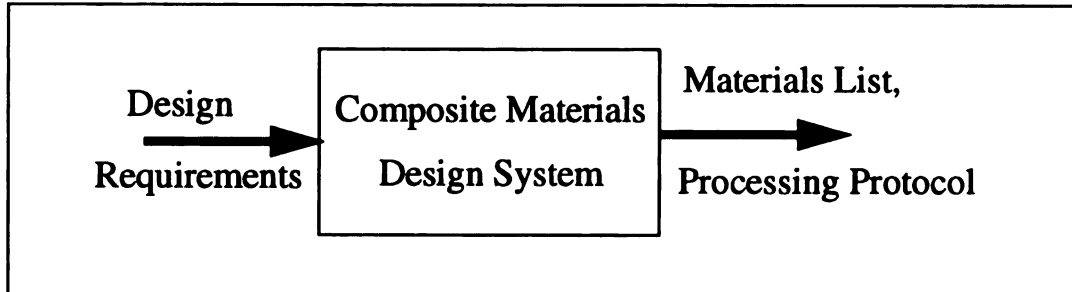


Figure 1: Information Processing Task for the Design of Composite Materials

Composite materials are relatively new, and design expertise available is limited when compared to more mature disciplines such as metallurgy. Consequently, one of the goals for this research is transferring the technology from researchers to students and from “high-tech” industries (such as the aerospace industry) to consumer-oriented industries (such as automotive industries).

Also, being “new,” the field of composite materials is in a constant state of knowledge flux. This leads to the requirement on a composites design system to be easily upgradeable in order to accommodate knowledge about new composites as they become available. This problem of acquiring new knowledge is often a “bottleneck” for knowledge-based systems requiring the continuous involvement of a “knowledge-engineer.” By structuring the information in a knowledge-based system using a vocabulary familiar to “domain experts,” this bottleneck of knowledge-acquisition is facilitated by allowing knowledge-acquisition to be accomplished by the domain experts.

1.2 The Design Problem

Over the last decades, design has been the target for a great deal of research. This follows from the perception that design is a highly structured activity, and thus requires “intelligence.” The term “design” however has been used to mean a variety of problem solving activities, from the design of computer programs to the design of high-rise buildings. A parallel line of research has concentrated on “planning.” Again, planning is perceived to require “intelligence,” and again planning is usually interpreted to include a wide variety of activities from planning a military operation to planning the processing of chemical components to produce another chemical product. Examining the type of activities included in both design and planning reveals a similarity in the type of activities involved in each. The principal difference is that the term “design” is typically used to refer to the design of an artifact, whereas “planning” is used to refer to the design of a plan of action. However, the type of information processing involved in both design and planning is identical. In this dissertation, I will use the term “design” loosely to mean both the design of a physical artifact or the design of a plan of action.

Design is typically classified into three categories: creative, innovative and routine [Brown, 1991; Gero, 1990]. Creative design deals with design situations resulting in major inventions or new products. Routine design deals with “every-day” design situations where the “how to” knowledge is readily available, such as designing an elevator system for a new building. Innovative design refers to cases where design is neither entirely creative nor entirely routine, such as using a previously known component for a new function. For example for years automatic cruise control systems have been used in automobiles to control the vehicle’s speed. This was typically done using a mechanical feed-back loop. Recently, microprocessor-controlled circuits started replacing this mechanical feedback loop. While the components of the new cruise control systems are all well known, the assembly of these components together is novel. Design activity can be viewed as a contin-

uum, with purely creative design at one end, and purely routine design at the other end as shown in Figure 2. Knowledge-based design systems are intended for routine design cases with possible involvement in innovative design by interacting with human designers during the design process. The design involved in the research reported here is of routine nature.

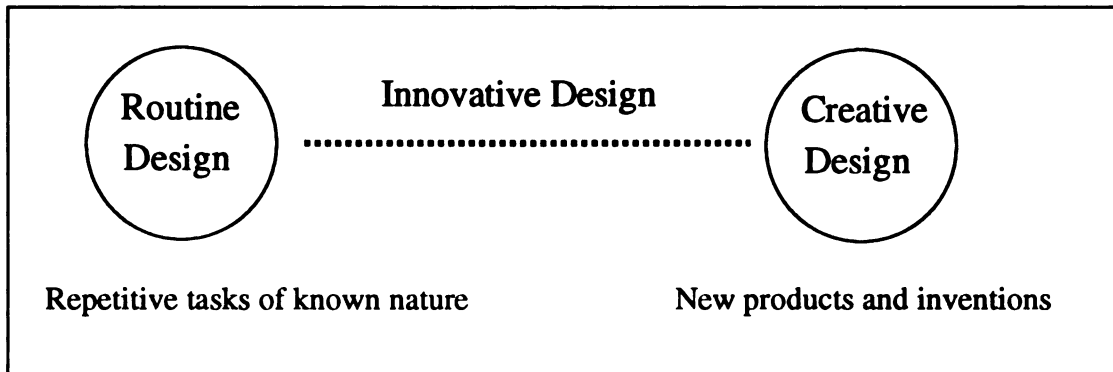


Figure 2: Classification of Design According to the Level of Routineness

1.3 The Generic-Task Approach

Early Artificial Intelligence research followed what is currently referred to as the “first generation” approach to knowledge-based systems. This approach can be characterized as the use of general purpose tools (one fits all) to model various aspects of “intelligence.” A typical example of first generation approaches is the widely known technique of production systems (or rule-based systems) [Davis & King, 1977; McDermott, 1982; Newell & Simon, 1972; Shortliffe, 1976]. In production systems, knowledge is expressed in the form of a set of independent production rules (or “if-then” rules). A rule-based system also includes a general inferencing mechanism, such as forward chaining or backward chaining.

Rule-based systems were originally perceived as general models of cognition with a rule being the basic building block of human knowledge structures. This idea was then general-

ized to conclude that rule-based systems are necessary and sufficient for intelligent problem-solving. However, this view can be shown to be neither necessary nor sufficient [Chandrasekaran, 1985]. It is not necessary, because while it might be advantageous for a knowledge-based system to model human thought at some level of abstraction, it is not obvious why this should be done at the level of knowledge formation in short-term memory, especially for capturing expert problem-solving performance [Chandrasekaran, 1985]. It is not sufficient, because even if we accept that a rule is the basic unit of human problem solving, there must exist some other constructs to account for the organization of rules into higher level units such as concepts and for their interaction with problem-solving [Chandrasekaran, 1985].

With the widespread use of production systems, several problems were recognized and led to the development of a second generation of knowledge-based systems. These problems include:

- Attempting to build large rule-based systems a 20/80 phenomenon seemed to always exist [Chandrasekaran, 1985]. That is to say, a large amount of knowledge (80%) could easily be represented with a small amount of rules (20%), while attempting to add the remaining knowledge resulted in a large increase in the rule-based system. This led to the belief that not all types of knowledge could be “naturally” represented in the form of rules.
- Problem solving in production systems followed a fixed regime, typically forward chaining or backward chaining. This did not allow for the use of control mechanisms specifically tailored for the problem on-hand. The burden was thus left on the implementor of a production system to handle domain-specific control issues by using “clever” programming techniques forcing the necessary control actions through the interaction between the different rules.

- Individual rules were originally thought of as independent, allowing the addition or deletion of rules without affecting the rest of the system. However, as large applications were developed it was found that interactions between different rules often render the task of modifying the domain knowledge difficult at best.
- As knowledge-based applications became more widespread, and as they became more sophisticated, a growing need was recognized for them to serve an additional function: transferring their expertise to novice (or student) domain experts. The unstructured nature of production systems could not serve this function. It then became apparent that methods are needed for describing knowledge-based systems at the “Knowledge Level” [Newell, 1980; Sticklen, 1988] by describing “what the system does” in implementation-independent form. In other words, describing the “information processing task” or the relationship between the outputs and the inputs of the system regardless of how the system is implemented.
- The use of a general purpose inferencing mechanisms did not provide any aid in analyzing problems. Production rules were for the most part a “programming language” and as such provided no help in acquiring the knowledge necessary to build a knowledge-based system.

The problems noted above should be viewed as difficulties with rule-based systems and not with the use of rules as a method for expressing knowledge. Higher level constructs are needed to abstractly define problem solving. However, there is nothing in principal to prevent the use of rules as the underlying knowledge representation.

Recognizing these problems, several schools emerged forming what is currently known as second-generation knowledge-based systems [Chandrasekaran, 1983; Chandrasekaran, 1985; Chandrasekaran, 1986; McDermott, 1988; Steels, 1990; Sticklen, 1988; Wielinga,

Bredeweg, & Breuker, 1988; Wielinga & Breuker, 1986]. While these schools differ in detail, they share a common philosophy, each having deviated from the general tools approach and developed task-specific taxonomies of problem-solving types or categories. The common assumption of these approaches is that human problem solving can be classified into categories of problems, each category shares a common methodology while the knowledge needed for solving individual problems would differ from one problem to another.

By grouping problems into task categories, task-specific approaches thereby avoid the generality of first generation techniques, while avoiding the pitfall of having to design a new technique for every problem. That is, by carefully analyzing a class of problems (e.g. design or classification), we can formulate a framework that can be applied for analyzing similar problems. Furthermore, by implementing the results of the analysis in the form of a problem solver building tool, the analysis and the building of another problem solver can be simplified; the tools in this manner guide the analysis of the problem [Bylander, Chandrasekaran, & Josephson, 1987; Chandrasekaran, 1985].

While the different task-specific approaches share a common philosophy, they differ in detail. For example the generic-task approach is based on the hypothesis that knowledge should be represented differently according to its intended use [Chandrasekaran, 1986] as shown in Figure 3. While this mode of knowledge representation can lead to duplication in represented knowledge (for example if some physical system is represented for the purpose of designing it and also for diagnosing it in case of a malfunction), this duplication leads to a more efficient problem solving since in each instance the knowledge is directly represented in a form suitable for its immediate use. By representing the knowledge according to its intended use, it also becomes readily usable as a vehicle for effectively transferring the system's expertise from the system to less experienced humans. The generic-task approach identifies a number of domain-independent methods (or tasks) and defines an

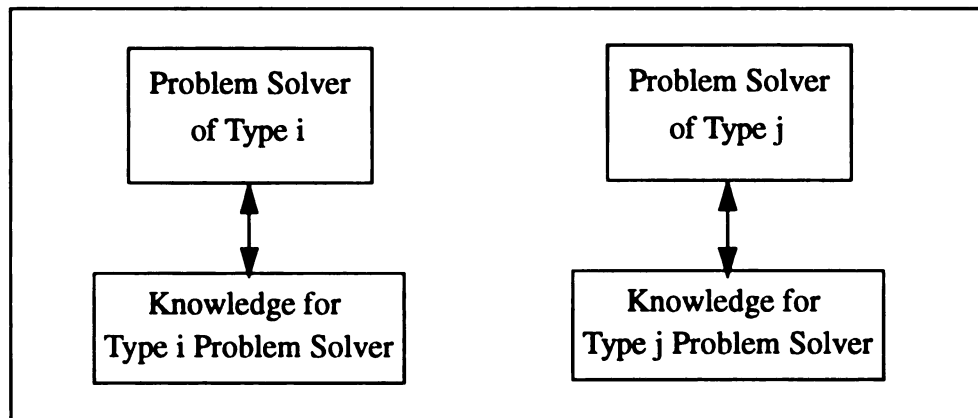


Figure 3: Generic-Task Problem Solvers

implementation tool for each of these tasks [Bylander, et al., 1987; Chandrasekaran, 1985]. Control knowledge typical of instances of these tasks is embedded in the tools. As such, these tools can be used to facilitate the acquisition of the requisite domain knowledge directly in the form in which it is going to be used. On the other hand, other task-specific approaches such as the KADS approach [Wielinga, et al., 1988; Wielinga & Breuker, 1986] views knowledge as being independent from its intended use as shown in Figure 4. Knowl-

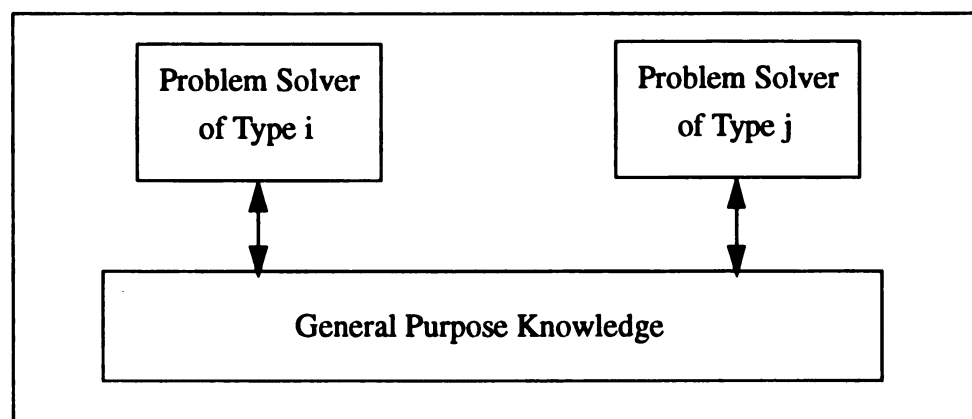


Figure 4: KADS problem solvers

edge is thus represented in an abstract neutral format that is expanded at run-time to generate the constructs needed.

An example from the generic-task approach is “Hierarchical Classification” problem solving [Bylander & Mittal, 1986; Chandrasekaran & Goel, 1988]. In hierarchical classification, knowledge is structured in the form a hierarchy of pre-specified categories. The higher level categories represent the more general hypotheses (e.g. a mechanical problem in diagnosing an automobile), while the lower level categories represent more specific hypotheses (e.g. a clogged valve or a bad spark plug). Inferencing in hierarchical classification uses an algorithm known as “establish-refine” in which each category attempts to “establish” by matching patterns of observed data against predefined matching patterns. Once a category “establishes,” it “refines” by having its sub-categories attempt to “establish.” While diagnosis of an automobile engine and medical diagnosis of the human body are very different in terms of the knowledge involved, they both can utilize the same type of classificatory problem solving. Analysis of one of the two problems thereby contributes to the analysis of the other.

Another example generic-task is “Routine Design” [Brown, 1987; Brown & Chandrasekaran, 1985; Brown & Chandrasekaran, 1986]. Routine design is a method for solving design problems that are “routine.” By routine, it is meant problems whose solution methods are completely known with no innovation involved. This does not preclude cases where a novel design might result; however, the method of reaching this design is not novel. Since routine design forms a central part of the background for this dissertation, routine designed is explained in detail in Chapter 3.

1.4 Integrated Problem Solving Architecture for the Design of Composite Materials

While the individual generic-tasks [Chandrasekaran, 1986] provide for a methodology for solving primitive problems, solution of complex problems often requires integration of multiple generic-tasks [Chandrasekaran, 1990; Goel, 1989; Punch, 1989; Sticklen, 1987]. In many cases a complex problem can be decomposed into sub-problems each of which is either further decomposable or is itself a primitive problem that can be solved by an available problem solving technique. In these cases, we often desire to decompose the complex problem into its sub-problems, employing the necessary tools to solve the individual problems, then integrating the results to solve the original problem.

In this research, I present an integrated problem solving architecture for designing composite materials. This architecture utilizes the techniques of routine design [Brown, 1987], functional modeling [Sembugamoorthy & Chandrasekaran, 1986; Sticklen, Chandrasekaran, & Josephson, 1987] and case-based reasoning [Hammond, 1989; Schank, 1982], as well as a procedural module. To effectively implement this architecture, I found it necessary to augment the routine design approach by extending it to include the capability for generating multiple design alternatives meeting a set of requirements. The extended approach is called MDSPL; multiple design specialists and plans language. I implemented the MDSPL problem solving part of the integrated architecture as well as a selection module to select among the resulting designs. The remainder of the developed architecture was not implemented in this work and is the target of future research. The integrated design architecture is presented in Chapter 5; the MDSPL approach is presented in Chapter 6 and the selection module is presented in Chapter 8.

1.5 The Multiple Design Approach

The multiple design approach (MDSPL) reported here is an approach for generating multiple designs satisfying a common set of design requirements and constraints.

The approach used closely follows and builds on the previous work on routine design reported in [Brown & Chandrasekaran, 1989; Brown, 1987; Brown & Chandrasekaran, 1986; Chandrasekaran, Josephson, Keuneke, & Herman, 1986; Chandrasekaran, Josephson, Keuneke, & Herman, 1989]. Brown's approach for routine design (DSPL) is reviewed in Chapter 3.

The research reported here originated in the construction of a knowledge-based system for designing composite materials (as well as their processing protocols) to specifications. After analyzing this problem, it became apparent that the generation of a unique "optimal" design would not suffice in many cases. There are several reasons behind this conclusion:

- The area of composite materials is relatively new. Consequently the design of composites in many cases is still art more than science. For the same set of requirements, a composites researcher would start by generating a family of design alternatives, which he would then evaluate, selecting one to adopt. The generation of multiple designs would relieve the designer from the former task (designing the alternatives), allowing him to concentrate on the later task of evaluating the alternatives and selecting among them.
- Practical industrial considerations sometimes dictate the final selection of a design. A typical scenario includes cases where, though a certain material of known lower market value can be used, a manufacturer might have a surplus of an alternative material of a higher market value. In such a case, the manufacturer might select a design that utilizes the material of which he has a surplus, thereby saving storage costs. While this type of information can be

used during the design process and contribute to the generation of the needed design, it is more practical to leave this as a post-design activity. In this fashion, the manufacturer enjoys the benefit of having the multiple designs at his disposal and the ability to compare the merits of each of the alternatives. The above scenario could in one case lead to the use of the available material being more beneficial, while in another case the merits of another design could lead the manufacturer to decide on a different direction.

- The availability of multiple designs is sometimes beneficial to accommodate varying needs. A typical scenario here would be a manufacturer in some cases choosing a design that would produce a lower cost product at the expense of a longer production time if the market conditions allow it, whereas in other cases market conditions might necessitate choosing the design with the shortest “turn around” time.

In addition, it is a common practice in engineering design to generate multiple designs as a first step [Traister, 1978]. Additional steps are typically involved for evaluating the merits of the different design alternatives and choosing the best design, depending on the current needs. These additional steps typically vary according to the domain of the design problem. From this perspective, a system for generating multiple designs can be used in a variety of engineering design situations, by overlaying it with a domain-specific tool for evaluating the candidate designs.

1.6 Organization of the Dissertation

In Chapter 2, I will discuss the different approaches for design problem solving. I conclude the discussion by comparing the different approaches to “Routine Design,” thus outlining the reasons behind the choice of routine design as the basis for this work. I will also review

some of the research efforts on knowledge-based applications in composite materials outlining the relationship between these works and the research reported here.

In Chapter 3, I will present a detailed description of routine design problem solving, detailing the original routine design approach, and its system building language; DSPL. I will also describe other research efforts that build on that approach, and outline possible interactions between these extensions to routine design and the extensions reported in this work.

In Chapter 4, I will summarize the goals of this research, both from the knowledge-based systems side and the composite materials side.

In Chapter 5, I will describe an integrated problem solving architecture for the design of composite materials based on integrating multiple problem solving approaches.

In Chapter 6, I will provide a detailed description of the multiple design approach. I will describe the additions to the original routine design approach as well. I will also describe MDSPL, the language I implemented for specifying the design knowledge. A comparison between MDSPL and DSPL (the language for Routine Design) will also be included.

In Chapter 7, I will describe an example MDSPL system for the design of polymer composite materials. A traditional single design solution is presented as well as a multiple design solution.

In Chapter 8, I will describe a module for selecting among the resulting designs. This module uses dynamic knowledge of availability and pricing of the raw materials to select among the resulting designs. I will also discuss two case studies of the materials design MDSPL system, together with the selection module.

In Chapter 9, I will summarize the contributions of this research and note directions for future research.

Chapter 2

Previous Research: Background

Design problem solving has been the focus of a great deal of knowledge-based systems research. In this chapter, I examine the major approaches for knowledge-based design, and compare them to the “Routine Design” approach which is one of the starting points for this research.

In knowledge-based design systems, knowledge is used to produce a design or a plan given some input. In this chapter, I will also examine a parallel line of research that addresses a variant of the problem addressed by knowledge-based systems in which not only the inputs are known in advance, but the output of the planning process is also known in advance. The goal then is not finding a solution, but determining “how” to reach that solution.

At the end of the chapter, I also review representative knowledge-based systems applications in composite materials. This domain review will motivate the need for the research I have undertaken.

2.1 Knowledge-Based Design Systems

In this section, I will present the major knowledge-based design approaches.

2.1.1 First Generation Design Approaches

2.1.1.1 Rule-Based Systems

In the early eighties, most of the research in knowledge-based systems centered around the use of rule-based systems, and design problem solving was no exception. Several successful design systems were implemented.

One of the most famous rule-based design systems is the R1 computer configuration system [McDermott, 1982]. R1 was originally implemented using 777 rules and had 420 different parts. It was later expanded to accommodate more components and renamed XCON [Bachant & McDermott, 1984]. The XCON system has more than 5,000 parts and 3,300 rules. The R1/XCON system configures DEC's VAX computer systems based on customer requirements. XCON checks to see if all necessary components are added to the system, if not it adds the missing components. XCON produces diagrams showing how the different components are to be associated. XCON's rules are implicitly chunked into groups each performing a sub-task with little local interaction between the rules in the different groups. Reasoning thus proceeds in a relatively direct and focused way requiring little search and no backtracking. XCON's sub-tasks include determining that the order received is a reasonable system and that it is complete, determining that all the components are compatible, selecting power supplies, selecting a backplane, assigning a backplane to a box, configuring unibus adapters, calculating unibus length, and assigning boxes to unibuses and cabinets.

R1/XCON is an example of a successful rule-based system. The main reason for its success can be attributed to its inherently modular nature resulting into rules relatively divided into chunks with little interaction between individual rules in the different chunks. However, not all systems can be naturally represented as rule-based system in this fashion. As experience with rule-based systems was amassed, it was found that they can easily represent a large amount of knowledge about a certain domain. However, as more and more knowledge was represented, the size of rule-based systems tended to increase rapidly due to the interaction between the different rules. These interactions also further complicated the systems making modification difficult at best.

2.1.1.2 Blackboard Architectures

Another AI techniques for problem solving in general and for design in particular is the blackboard architecture. The term “blackboard” is used in analogy to the use of a physical blackboard by a group of human experts. The blackboard contains a set of facts about the state of the world. Based on the facts available on the blackboard, an expert deduces and writes down more facts. Based on that expert’s action, another expert may be triggered to take action and add more facts to the blackboard. Similarly, blackboard architectures use 3 independent constructs to manage the problem solving activity:

1. A database (blackboard) that carries the state of the problem solving activity.
2. A collection of knowledge sources independently capable of manipulating the data in the “blackboard.” A knowledge source can be any entity capable of manipulating knowledge such as a rule or a procedural program, thus allowing for heterogenous forms of knowledge representation.
3. A control mechanism which monitors the blackboard and switches the control among the knowledge sources appropriately. At any point in time only one knowledge source is the “focus of attention.” This knowledge source then affects the state of the problem solving by manipulating the data available on the blackboard and posting new data. Whenever the preconditions for a knowledge source exist, that knowledge source can potentially be triggered depending on the status of other knowledge sources.

One of the most successful blackboard systems was Erman’s Hearsay-II speech understanding system [Erman, Hayes-Roth, Lesser, & Reddy, 1980]. This system was followed by Hearsay-III [Balzer, Erman, London, & Williams, 1980; Erman, London, & Fickas, 1981], a generalized version of Hearsay-II suitable for application into other domains. The

HEARSAY systems were experiments in natural language understanding in limited domains. Given a context, the system would attempt to interpret a spoken sentence. Natural languages are full of ambiguities. The HEARSAY systems worked by proposing a set of competing hypothesis and then applying knowledge of the given context to select the most likely hypothesis.

Another successful general purpose blackboard system followed later based on the Hearsay systems is Hayes-Roth's BB1 system [Hayes-Roth, 1985]. A notable characteristic of the BB1 system is the use of an additional blackboard for control. Control is not fixed but manipulated the same way as data with control decisions being dynamically taken based on the current control status. In the BB1 systems, knowledge is represented as frames representing individual objects (such as in Figure 5) as well as constraints representing rules (such as in Figure 6) representing relationships that have to be maintained between different objects. The BB1 execution cycle works as follows:

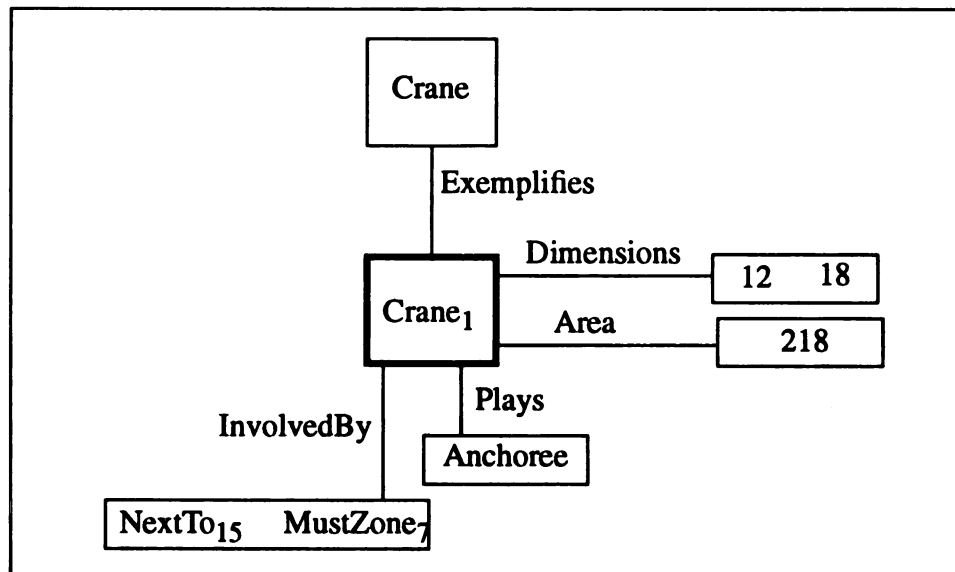


Figure 5: An Object in SIGHTPLAN (from [Tommelein, Johnson, Hayes-Roth, & Levitt, 1987])

1. A knowledge source is executed thus causing a change on the domain blackboard.

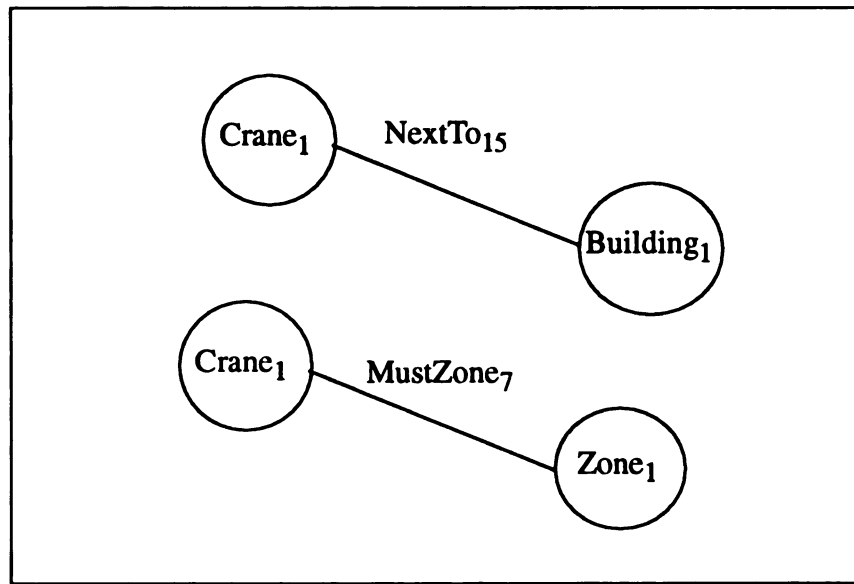


Figure 6: Constraints in SIGHTPLAN (from[Tommelein, Johnson, Hayes-Roth, & Levitt, 1987])

2. Based on the changes on the blackboard, knowledge sources are triggered to take further actions. Activated knowledge sources are placed on the control blackboard.
3. The next event to execute is selected from among the knowledge sources active on the control blackboard.
4. The cycle is repeated until a final goal is achieved.

Tommelein et al. [Tommelein, et al., 1987] successfully employed the BB1 system to implement SIGHTPLAN, a design system for generating construction-site plans; the arrangement of cranes, huts, and materials storage around a construction site. Figure 5 shows the frame representation of the object Crane₁ from the SIGHTPLAN system. The slots indicate that this object is a Crane with dimensions 12 feet. and 18 feet. and needs an area of 218 square feet. The frame also indicates that this object (Crane₁) is involved in the constraints NextTo₁₅ and MustZone₇. Crane₁ is also described as “Anchoree” meaning that its position is relative to other objects which are “Anchored” or having a fixed position.

In SIGHTPLAN, only one object has a fixed position, which is the building under construction. The constraints NextTo₁₅ and MustZone₇ are shown in Figure 6. NextTo₁₅ says that the object Crane₁ has to be next to the object Building₁ (the building under construction). The building site is divided into three zones, Zone₁ is the area immediately around the building under construction, Zone₂ is the space skirting Zone₁ and Zone₃ is the rest of the site. The constraint MustZone₇ says that the object Crane₁ has to be zoned in Zone₁.

The SIGHTPLAN system assumes a large degree of independence among the locations of the different design elements by considering proximity to the building under construction to be more important than adjacency and orientation of the design elements. The system develops its design incrementally, making opportunistic decisions about the locations of the different elements. SIGHTPLAN assumes there is a fair amount of error in the final layout and thus considers its design to be a preliminary layout, or a starting point to aid the site designer in laying out the final design.

While blackboard systems showed a fair amount of success, they eventually suffered from the same problems as rule-based systems, most importantly the problem of scalability. As with rule-based systems, as blackboard systems grew larger, it became apparent that interactions among the different knowledge sources made it difficult at best to modify an already functional system. Also, like rule-based systems, blackboard architectures did not offer any insight into the analysis of problems. By providing a fixed regime for control (opportunistic triggering of knowledge sources), blackboard systems did not allow for the use of control regimes appropriate for the domain being analyzed.

2.1.2 Case-Based Design

A more recent approach to knowledge-based systems in general and knowledge-based design in particular is case-based reasoning. This approach is being used as a model of human problem solving [Schank, 1982]. The idea behind this approach is that people do

not solve every new problem as “completely new.” Instead, they try to remember similar past experiences and reason about how this past experience is different from the current problem [Schank, 1982]. Similarly, case-based reasoning systems store a library of previous cases (problems) with the known solutions to these problems. The solution of a new problem proceeds by comparing the new problem to the cases in the library. A case is chosen whose inputs match as close as possible the inputs of the current case. The differences are then identified, and by using the available knowledge the solution of the library case is modified to fit the current case. There are 3 separate research issues involved in case-based reasoning:

1. Organization of the case memory. The case memory has to be indexed in a way to allow efficient retrieval of cases based on requirements. Typically a set of important features is defined and the case meeting the largest number of features is retrieved.
2. The techniques involved in transforming cases to fit a new situation. Some domain-dependent knowledge must be applied to alter the cases in a consistent manner to meet the requirements of the new case.
3. Learning the newly developed cases for future reference. Newly generated cases need to be assessed to determine the usefulness of storing them in the case memory. Cases that are general enough to be common in many situations need to be stored while cases that are likely to be one of a kind need not be stored. Also, cases need to be abstracted before storage to avoid storing large numbers of closely related cases. By storing newly developed cases, systems are said to “learn.”

One of the most successful case-based design systems is Hammond’s CHEF [Hammond, 1989]. CHEF’s domain is chinese cooking. It uses a store of recipes to generate new recipes to fit new requirements. CHEF starts by retrieving a recipe that meets most of the

requirements for the requested recipe. It then applies transformations (e.g. ingredient substitutions, addition or removal of ingredients). It then simulates the newly formed recipe, predicts problems and applies transformations to fix these problems. Eventually, CHEF applies some abstractions to the resulting recipe (e.g. abstracting broccoli into crisp vegetable) and stores the result in its case memory. A notable feature of CHEF is that it not only stores successes but it also stores failures indexed by the features in the world that help predict them. Learning in CHEF thus involves storing successful plans, learning new plans that avoid problems, learning the features that predict problems, as well as learning the repairs that have to be made if those problems arise again in different circumstances [Hammond, 1989].

A very similar technique to the case-based approach is what is known as “design-by-analogy.” This technique is best exemplified in Huhns’ Argo system for the design of VLSI circuits [Huhns & Acosta, 1988]. This system performs its design functions by “analogy” to previous design situations. First the system tries to find a previous “analogous” design, and then transforms that design to adapt it to the new problem situation.

Goel’s KRITIK system [Goel, 1989] described in Section 2.1.4.1 integrates the principles of case-based reasoning and those of model-based reasoning to accomplish the design of engineering artifacts.

Case-based techniques rely on the availability of an initial set of plans that fairly cover the intended domain. Consequently, case-based reasoning is only suitable for domains where such cases exist. Alternatively, case-based reasoning can be employed in conjunction with other reasoning mechanisms that can be used to accomplish the intended task in the absence of previous similar cases while storing the results for future reference. The integrated architecture for the design of composite materials presented in Chapter 5 utilizes a case-based reasoning system to efficiently reach a design in cases where the design require-

ments closely match those of a pre-existing design, and performs design from “first principals” in case of absence of previous cases.

2.1.3 Task-Specific Design Approaches

In recent years, recognizing several problems with the first-generation approaches, a second generation of knowledge-based systems developed. Several schools emerged (see for example [Chandrasekaran, 1983; Chandrasekaran, 1985; Chandrasekaran, 1986; McDermott, 1988; Steels, 1990; Wielinga, Bredeweg, & Breuker, 1988; Wielinga & Breuker, 1986]). While these schools differ in detail, they share a common philosophy, each having deviated from the general tools approach and having developed task-specific taxonomies of problem solving types or categories. Task-specific approaches consequently take a middle ground between the general tools approach of the first generation systems on one side and implementing a computer program for each new problem on the other side. Task-specific approaches classify general problem solving into categories. Problems within each category can differ in terms of the knowledge needed, but would all share the same type of problem solving activity, such as classification or design. Having classified the types of problem solving activities within each of these categories, task-specific approaches then define a general method for solving problems of that type and in most cases follow up by providing a tool for this particular class of problem solvers. Such tools provide a great deal of guidance for analyzing a new problem to be implemented.

While sharing a common basic philosophy, task-specific approaches differ in detail. For example the generic task approach [Chandrasekaran, 1983; Chandrasekaran, 1986] adopts the view that knowledge representation should be dependent on its intended use as shown in Figure 7. While this view may result in duplication of represented knowledge (such as when having a design and a diagnostic system for the same engineered artifact), this duplication often results in more efficient problem solving due to the availability of the needed knowledge in a readily usable format. On the other hand, the KADS approach [Wielinga,

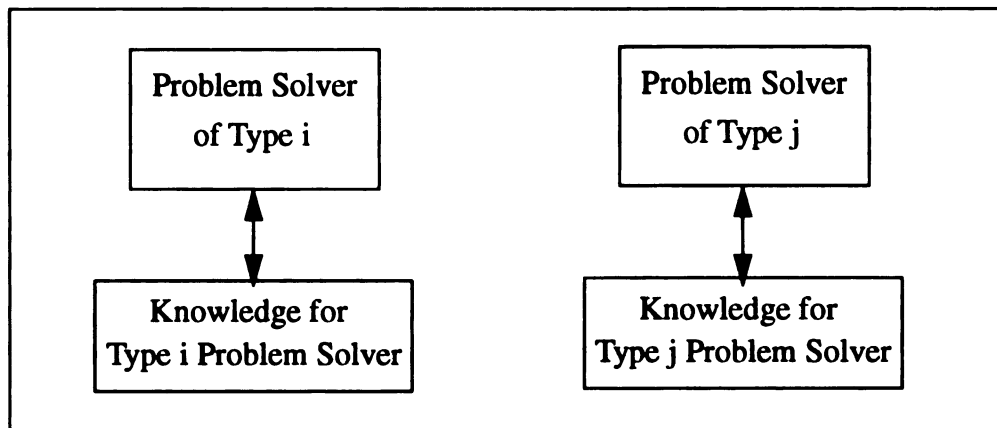


Figure 7: Generic Task Problem Solvers

et al., 1988; Wielinga & Breuker, 1986] views knowledge as being independent from its intended use as shown in Figure 8. In this approach knowledge is represented in an abstract

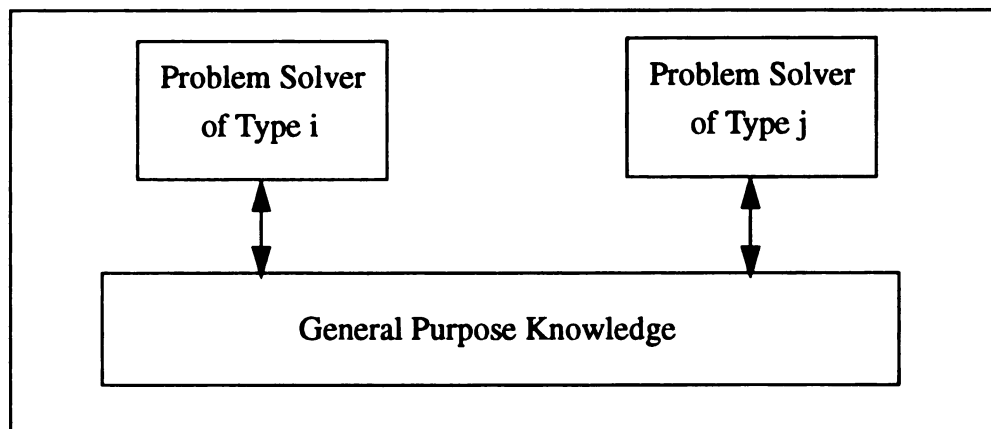


Figure 8: KADS problem solvers

neutral format that is expanded at run-time to generate the constructs needed.

In this section, I will present a review of four task-specific design problem solvers as examples. These architectures share a common characteristic, being domain-independent.

2.1.3.1 SALT

SALT [McDermott, 1988] is based on McDermott's approach for the design of knowledge-based systems known as Role-Limiting Methods. McDermott Defines role-limiting methods as methods capable of solving a particular class of problems [McDermott, 1988]. Role-limiting methods typically have domain-independent control strategies¹ that are particularly suited for the class of problems they are capable of solving. By imposing these methods on the analysis of problems, the type of problem solving (or roles) involved are limited. SALT uses a "propose-and-revise" method for design. It represents the design space as a graph where each node is a partial design and the unidirectional links between nodes represent possible design extensions. By design extensions, it is meant transformations that can be performed on the partial design to achieve a more complete design. SALT's design activity starts with an initial state that is a set of specifications and uses the following control strategy to progressively refine a design:

1. Extend a design and identify constraints on this extension.
2. Identify constraint violations if any (If none exist, repeat step 1).
3. Attempt to fix the least expensive to fix constraint violation by modifying the design.
4. Identify any constraint violations caused by the fix. If any are found repeat step 2.
5. Remove any parts of the design incompatible with the newly introduced revision.
6. If the design is not complete, repeat the entire process.

1. The term domain-independent control strategies refers to control mechanisms which depend on the type of problem (e.g. design) but are independent of the domain of the problem (e.g. the same control strategy can be used for designing electrical as well as mechanical systems).

2.1.3.2 Routine Design

Routine design problem solving is an instance of Chandrasekaran's generic task approach for building knowledge-based systems [Chandrasekaran, 1983; Chandrasekaran, 1986]. The generic task approach attempts to identify a set of "generic tasks" or problem solving methods each of which is capable of solving a particular class of problems.

The routine design approach [Brown, 1987; Brown & Chandrasekaran, 1985; Chandrasekaran, Josephson, Keuneke, & Herman, 1986] was introduced to handle a certain class of design problems, namely "routine" design problems. The reason it is called "routine" is that it deals with problems that are of a well-understood nature. In routine design problems the knowledge necessary to solve a design problem is well known, however the specific course of action necessary for solving any particular instance of the design problem is not necessarily known in advance. As such, the result of a routine design problem can be a novel design, while the method used to achieve that design is not novel.

Problem solving is represented in the form of a hierarchy of cooperating design specialists, each responsible for a particular role in the overall design process (Figure 9). A design specialist represents a particular piece of design knowledge about some part of the overall design. Higher level specialists represent the more general aspects of the design, whereas lower level specialists represent more detailed aspects of the design process.

The research reported in this dissertation uses and builds on the type of problem solving involved in Routine Design. To better understand the routine design approach, Chapter 3 gives a detailed account of the problem solving activity involved in Routine Design.

2.1.3.3 The Micon Microprocessor Configuration System

Birmingham's Micon system is a system developed for designing small computers [Birmingham, 1989b]. The complexity of Micon's resultant designs roughly equal that of a small workstation. The Micon system is implemented using a rule-based language and uses

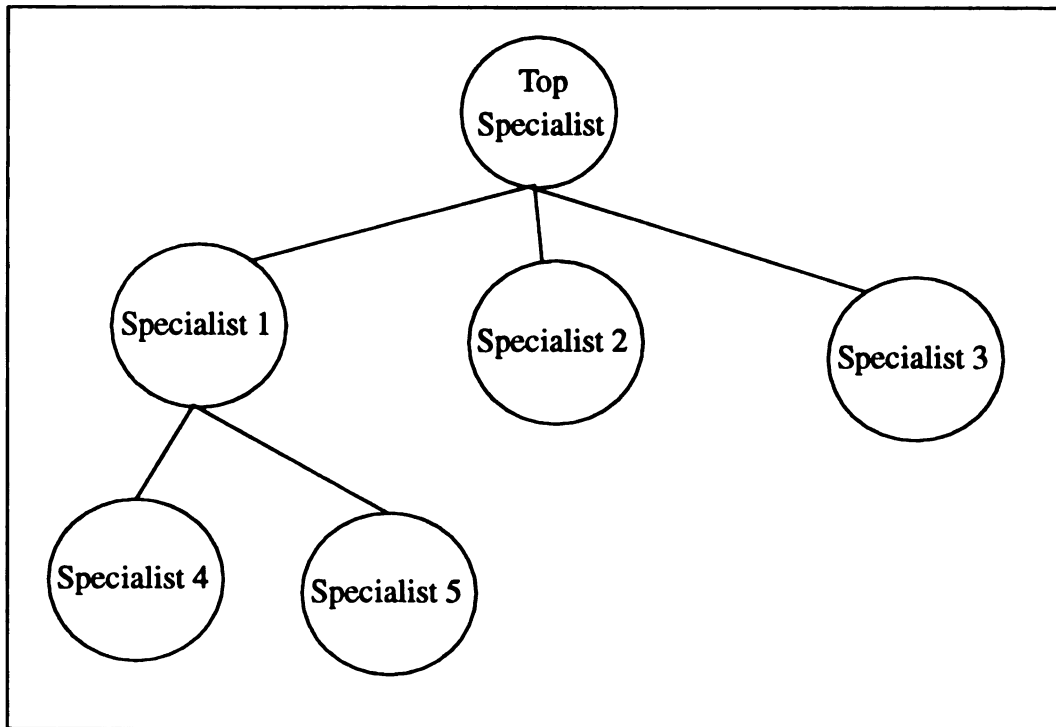


Figure 9: Routine Design Specialist Hierarchy

a synthesis mechanism to generate a complete design out of constituent design components. Even though the Micon system was originally developed for designing small computers, it was later applied to other domains such as mechanical engineering [Birmingham, 1989a].

Even though the Micon system is implemented as a rule-based system, a structure is imposed on the system rendering it characteristics typical of second-generation knowledge-based systems. Rules are grouped into different categories, namely:

- Design synthesis module.
- Reliability analysis tool.
- Physical design.

While this decomposition provides some guidance for problem analysis, yet within each of these modules, problem solving is unstructured leaving the burden on a system developer to ensure proper control through the domain knowledge embedded into the rules.

2.1.3.4 The HI-RISE Structural Design System

The HI-RISE system [Maher & Fenves, 1985] is a procedural system for the design of the structural components of hi-rise buildings. In HI-RISE, the design process is divided into two major tasks executed in a fixed order, the design of the lateral load-resisting system followed by the design of the gravity load-resisting system. Each of these two tasks is decomposed into a set of sub-tasks to be executed in a fixed order.

While the HI-RISE system shares the same intuition as Routine Design by dividing the design problem hierarchically into sub-tasks, the HI-RISE system is implemented in a strictly procedure fashion making it difficult to adapt to other domains.

2.1.3.5 Comparison

The four examples described above represent the different task-specific approaches for design.

The HI-RISE system represents one end of the spectrum where the domain knowledge is hard-wired into the design algorithm, making it difficult to adapt to other domains. While this approach provides an efficient design algorithm, every problem requires separate analysis and a separate computer program has to be written for every new problem.

On the other hand the Micon system is under-restricted. It provides only a high-level decomposition of the design problem into three main modules (design synthesis, reliability analysis, and physical design) without providing a problem solving strategy within each of these modules.

Both SALT and Routine Design provide a detailed domain-independent design algorithm. The two approaches however differ in their problem strategies. Routine Design uses a hierarchical decomposition to decompose the design problem into smaller sub-problems and then tackles each of the sub-problems individually in a fixed order. In SALT, problem solving follows a more opportunistic approach, where at any time during problem solving, any part of the design that can be accomplished next takes precedence. Each of the two approaches has its own merits. The fixed processing order of Routine Design is useful in cases where such an order can be defined. In these cases, more efficient problem solving can be achieved by following that previously defined order. However, in other cases a fixed processing order can not be defined *a priori* in which case the SALT approach is more appropriate.

2.1.4 Approaches Based On Integrating Multiple Problem Solvers

One of the most recent trends in knowledge-based systems is integrating multiple problem solving types to cooperate in solving one problem. In this section, we examine some of the techniques based on this type of integration. Integration can be done in 2 different ways:

1. Fixed integration: In this type of integration, different problem solvers have specific fixed roles to play. In other words, problem solving control is hard-wired during system implementation. For an example of this type of integration, see [Sticklen & Chandrasekaran, 1989] in which a classification system and a model-based reasoner are integrated to accomplish a medical diagnosis system.
2. Flexible integration: In this type of integration, the actual interaction between the different problem solvers is decided at run time according to the status of the problem solving at any given time. Fixed integration systems are generally viewed as being brittle. That is, if the fixed mode of integrating

the different problem solvers fails, no other problem solvers will be evaluated. Flexible integration approaches are designed to address this issue [Punch, 1989].

2.1.4.1 Goel's KRITIK Architecture

In his dissertation [Goel, 1989], Goel developed a technique for design problem solving based on combining case-based reasoning and model-based reasoning¹. This approach is an example of fixed integration approaches. The knowledge necessary for the adaptation of design cases is represented in the form of models specifying how the structure of the known design results in its output behaviors. The model explicitly specifies the expected output behaviors of the design including its functions, the elementary structural and behavioral interactions between components and the internal causal behaviors of the design that results in its output behaviors. The causal behaviors of the design are indexed by their expected outputs.

2.1.4.2 Punch's TIPS Architecture

While originally developed for diagnostic problem solving, the principles in the TIPS architecture [Punch, 1989] are directly applicable to other tasks including design. TIPS is based on the generic task approach to knowledge-based systems [Chandrasekaran, 1986]. This architecture is an example of flexible integration architectures. It employs a number of different tasks as well as special knowledge constructs for selecting among the different tasks at any given point. Each problem solving method is associated with a sponsor (in the form of a pattern matcher) to evaluate its applicability given the current problem solving status. Selectors are special knowledge constructs that use the evaluations produced by the sponsors to select which problem solving type to follow. Selectors contain knowledge for

1. Model-based reasoning is a term that refers to reasoning about a physical system based on a computer model of that system.

prioritizing the use of the problem solving types in case more than one type is applicable at any given time.

2.1.4.3 SOAR

The SOAR system [Laird, Newell, & Rosenbloom, 1987; Steier, Lewis, Lehman, & Zach-
erl, 1993] also uses a flexible integration architecture. SOAR implements a group of oper-
ators for problem solving. Operators are capable of examining the problem space in view
of a given goal and “deciding” if any operator is applicable. If no operators are applicable,
the problem solving status is considered to be in an “impasse” requiring the decomposition
of the original goal into sub-goals. The sub-goals are pursued only to the extent that the
problem space is modified enough to resolve the impasse resulting from pursuing the orig-
inal goal and then the original goal is pursued further.

Smith and Johnson [Smith & Johnson, 1993] defined what they called “Generic-Task
Problem Space Elements which is a re-examination of the standard generic-task elements
for use within the SOAR architecture by mapping the generic-task problem solving meth-
ods into SOAR operators. While this approach maintains the basic generic-task philosophy
of having different methods each capable of solving related problems, the generic-task’s
sense of control during problem solving is lost. The only sequence of operators is that
imposed by their preconditions, a system designer thus has to specify additional control
knowledge especially for cases where more than one operator can be applicable at the same
time.

Both fixed integration systems and flexible integration systems have their advantages
and their disadvantages. Fixed integration systems are brittle, once they fail, no other prob-
lem solving methods are pursued. Flexible integration systems avoid this problem by
selecting the type of problem solving to pursue at run time. However, for this same reason,
flexible integration systems offer little (if any) help for a system implementor in terms of

analyzing the design problem, whereas a fixed integration system can directly be mapped after a domain analysis of the design problem.

2.2 Other Planning Systems

In the previous sections I presented several design/planning systems. While these systems employ a wide range of approaches to achieve their goals, they all share a common goal: to generate a design or a plan to accomplish some input requirement.

Another parallel line of research in planning is concerned with a variant of this problem, not only the initial inputs are known, but the final goal is also known in advance. The goal of the planning process is then not to find a solution, but to determine “how” to find the solution. The two most classical problems within this category are theorem proving and robot planning. In theorem proving, we typically have a set of starting axioms, and a theorem and we need to prove the validity of the theorem. In robot planning, we are typically given a description of the world, an initial position for the robot as well as a goal position for the robot. The purpose of the planning process is then to find a path for the robot to navigate the world from its initial position to its final destination.

This line of research started in the early sixties when Newell, Shaw and Simon introduced their GPS (General Problem Solver) system [Newell, J.C.Shaw, & Simon, 1960; Newell & Simon, 1963]. In the GPS system, facts about the world are represented in the form of objects with characteristics. GPS also employs a set of operators to affect the status of the world. Operators are in the form of a set of preconditions, a set of facts to be added to the status of the world and a set of facts to be deleted from the status of the world. The problem solving goal is to reach a given state of the world given some initial description and a set of operators. The algorithm works by trying to minimize the difference between the goal state and the initial state. GPS accomplishes this by first locating operators that are relevant to this difference. If the preconditions of these operators are valid in the initial

state, the operators are applied and problem solving proceeds from the new modified initial state. However, the operator's preconditions might not be directly valid in the initial state in which case the preconditions are added as sub-goals for the system to accomplish and problem solving continues. This method is known as means-ends analysis.

The STRIPS system (Stanford Research Institute Problem Solver) [Fikes & Nilsson, 1971] formalized the ideas of GPS. The problem space for STRIPS consists of three entities:

1. An initial world model in the form of a set of well formed formulas (wffs) describing the present state of the world.
2. A set of operators including a description of their effects (a list of wffs to be added and another list of wffs to be deleted) and their precondition wffs.
3. A goal condition formulated as a wff.

The search strategy is the same as that of GPS; an operator is chosen that reduces the difference between the initial state and the goal state. If the preconditions of the operator are not met in the current state of the world, the preconditions are added as new sub-goals. STRIPS employs heuristics to select from among applicable operators the ones that are more likely to reduce the difference between the current state and the goal state. STRIPS main application area is the navigation of robots, and thus its wffs describe rooms, walls, doors, objects, and the status of the robot and the objects it manipulates.

To manage the complexity of the search spaces associated with the use of general purpose problem solvers such as GPS and STRIPS, Sacerdoti [Sacerdoti, 1974] introduced the ABSTRIPS system (Abstraction-based STRIPS). ABSTRIPS manages the complexity by using a hierarchy of abstraction levels. It abstracts the original problem by abstracting the preconditions of the operators. ABSTRIPS assigns criticality levels to the preconditions of the operators. At the highest abstraction levels only the preconditions with highest critical-

ity levels are checked. This results in the formation of a plan in a short period of time that covers the major steps from the initial state to the goal state while leaving a need for further planning to fill-in between the steps. For example, consider an operator that describes the effect of a robot pushing an object through a door into an adjacent room. At a high level of abstraction, the operator is applicable if the object is movable and a door into the room exists. At a lower level of abstraction, the robot and the object would be required to be in the room connected by the door to the goal room. At another lower level of abstraction, the door should be open. At a yet lower level, which in this case is the most detailed level, the robot has to be next to the box and the box has to be next to the door. Problem solving works by first assembling a plan at the highest level of abstraction and then recursively calling the algorithm at lower levels of abstraction to fill-in the details of the plan. The intuition behind this approach is that given the correct assignment of criticality levels, planning at the higher levels of abstraction can proceed faster to accomplish a “rough” plan, and that the less critical preconditions are only concerned with details that are easily achievable with small plans. An operator that would not result in a useful path would therefore be discarded faster at the higher levels of abstraction without wasting time on the details.

STRIPS and ABSTRIPS form the basis of most planning systems of the known-goal type. However, both STRIPS and ABSTRIPS were defined to handle problems with one goal. They can be applied to problems with multiple conjunctive goals by assuming the linearity of problem solving, that is, they assume that the goals can be independently accomplished sequentially. This assumption does not always work, for example consider the problem shown in Figure 10. In this example, we have a simple world consisting of three blocks A,B, and C. Initially, block C is on block A, and block B is on the table. The goal configuration is expressed as a conjunction of two goals: block A is on block B and block B is on block C. There is only one operator that can be applied: PUTON (X,Y) which puts block X on block Y only if X has a clear top and Y is the table or has a clear top. Assuming linearity, a planning systems will attempt to pursue the two goals separately. If the system

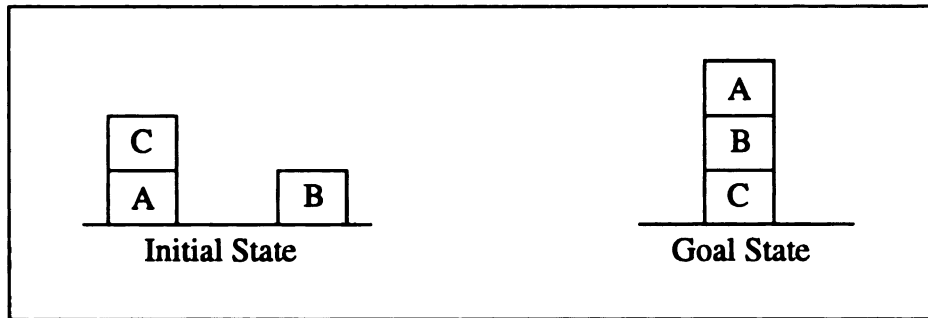


Figure 10: Example Problem with Multiple Goals

tries to put A on B first, it will first clear A by applying PUTON (C, table), then PUTON (A,B). Then, to put B on C, the planner will have to undo the first sub-goal. Suppose the system tries to put B on C first, this is directly achievable from the initial state by doing PUTON (B,C). However, this leaves us even farther from the other goal of having A on B.

Several planning systems specifically address this problem, for example Sussman's HACKER [Sussman, 1973] system works by first assuming linearity but as it fails, it compares the problem that occurred with known types of problems that can result from the linearity assumption. As bugs are encountered and solved, HACKER develops a collection of critics each of which can recognize that a certain type of bug has occurred in a plan. HACKER thus learns from experience. The three blocks problem is however reported in [Sussman, 1973] as an "anomalous problem" for which HACKER failed to achieve an optimal solution. Sacerdoti's NOAH [Sacerdoti, 1975] analyzes the conjunctive goals and applies rules to redefine the goals to avoid conflicts. Tate's INTERPLAN [Tate, 1974] maintains tables of possible interactions between conjunctive goals and uses them for initial debugging of the goals as well as backtracking to resolve the conflicts. MOLGEN [Stefik, 1981] analyzes the goals and generates constraints on the possible sequence of operator applications to avoid conflicts between the goals. MOLGEN plans gene-cloning experiments in molecular genetics by using these constraints in a hierarchical abstraction algorithm similar to the ABSTRIPS algorithm.

The above mentioned systems serve as the basis for most research efforts in planning for situations in which the final goal is known in advance. It is important to distinguish that this is a different type of planning from the design/planning activity involved in this research, in which the final outcome of the design process is not known in advance.

2.3 Knowledge-Based Systems in Composite Materials

Several researchers have applied knowledge-based systems to various aspects of composite materials. I will present some of these efforts in this section.

2.3.1 Knowledge-Based Systems for the Design of Composite Materials

The research by Nitsche, Kern and Janczak [Nitsche, Kern, & Janczak, 1990] has the same goals as our research, namely building a knowledge-Based system for the material design of composite materials. However, the domain reported by Nitsche, Kern and Janczak is that of ceramic and metal composites, while the research reported in this dissertation is in the area of polymer composites. Furthermore, Nitsche's research concentrates on the selection of materials and their ratios, while the research reported in this dissertation also includes the design of the processing protocol of the chosen materials. Nitsche, Kern, and Janczak's research follows a rule-based approach, and they are reporting problems with the fixed backward-chaining control mechanism [Nitsche, et al., 1990] and they intend to re-implement their system using another knowledge-based technique.

2.3.2 Knowledge-Based Systems for the Fabrication of Composite Materials

Hahn presents a technique [Hahn, 1991] for solving a complementary problem, that of design for manufacturability. Design for manufacturability, refers to the details of the design issues pertaining to the manufacturing process. Design for manufacturability con-

siderations include factors like the effect of part geometry and fiber layouts on the choice of manufacturing process. In assessing manufacturability, factors like technical feasibility, cost, quality of the parts and delivery time are considered. In [Hahn, 1991], an integrated architecture is used for selecting the manufacturing process. The design starts by first selecting a process based on experience with the above considerations and then simulating the process. Based on the result, the process is either accepted or modified. If the process is modified, the resulting process is simulated again, and the procedure iterates. According to [Hahn, 1991], the preliminary results are encouraging.

At the Lockheed AI Center, case-based reasoning is being used to configure autoclave loading layouts [Barletta & Hennessy, 1992; Mark, 1992]. Autoclave processing is a time consuming process, and the autoclave is a valuable resource to waste. Typically there is a large number of parts waiting to be cured. To maintain the quality of the finished parts, all parts have to be cured uniformly. Consequently, all parts in the same load must be selected such that they have the same heating rate. Moreover, parts cannot be loaded in any arbitrary configuration due to the non-uniform spatial heat distribution inside the autoclave. Furthermore, the placement of the parts in the autoclave influences the air currents inside the autoclave further changing the heat distribution. These factors combine to make the autoclave configuration a complex problem which is not fully understood. The way this problem was originally handled by autoclave operators was to look at previous successful configurations, and try to configure new configurations similarly. Case-based reasoning thus offered a natural mechanism for implementing this problem. Previous successful autoclave layouts are stored. Whenever a new batch of parts need to be processed, the system attempts to load them in patterns similar to those in the stored patterns. This is accomplished by substituting similarly shaped parts for those in the stored cases.

The research in [LeClair, Abrams, & Matejka, 1989] represents a complementary problem to that of the autoclave layout mentioned above. LeClair and Matejka present an archi-

ture for *in situ* control of autoclave processing of composite materials. By *in situ control*, it is meant the control of the autoclave during processing, by controlling the various aspects of the autoclave such as the extent and distribution of the heat sources. The method used is based on a blackboard architecture. First sensor measurements are taken and then abstracted first into symbolic data and then into more complex concepts about the state of the process. These concepts are then interpreted and necessary control signals generated for the control of the autoclave.

2.3.3 Trends of Knowledge-Based Systems Applications in Composite Materials

By inspecting the systems presented above, we notice a trend towards applying knowledge-based approaches in the different aspects of the manufacturing process of composite materials. Knowledge-based systems are being applied for designing the processing mechanism for a given composite [Hahn, 1991], for the configuration of processing equipment [Barletta & Hennessy, 1992; Mark, 1992] as well as for the control of the actual manufacturing process [LeClair, et al., 1989]. However, little effort is directed towards the design of the composite materials themselves. The research in [Nitsche, et al., 1990] is one such example in the area of ceramic and metal composites. The lack of research in the design of composite materials can be attributed in-part to the fact that composite materials are relatively new, and consequently “expert” knowledge is scarce. This does not apply to the fabrication processes applied for manufacturing composite materials since these are standard processes which have been in long use for manufacturing non-composite polymer systems.

The design and the fabrication problems are complementary problems which are together essential to cover the whole life-cycle of a composite material part from being a drawing on a “blueprint” to being realized as an actual part.

The problem addressed in this research is that of the design of a thermoset polymer composite materials from selecting the materials to be used to the design of the processing pro-

TOCOL (patterns of applications of temperature and pressure over time). This problem is described in detail in Chapter 4.

Chapter 3

Previous Research on Routine Design

The research reported here uses Routine Design [Brown, 1987] as its starting point. In this chapter, I will review the details of the Routine Design approach. I will also describe other research efforts aimed at enhancing Routine Design.

Design is a term that covers a wide-range of activities that can be classified along multiple dimensions. In order to understand where Routine Design lies with respect to design problem solving in general, I will start by examining the different classifications of design.

3.1 Classifications of Design

Design problem solving can be classified along multiple dimensions. However some of these classifications have more impact than others on choosing a design methodology for a given problem. For a given problem, more than one type of design problem solving may be applicable. It is therefore important to determine which features of the different classifications of design are important to the current problem and which ones are less important.

Design problem solving is commonly classified according to the end product of the design process being an artifact or a plan of action. It is common in the literature to refer to the former of these two types simply as design while referring to the latter as planning. In this research, I use the more general meaning of the term “design” to mean both types of activity.

Another dimension for classifying types of design is the domain of the design problem solving activity (e.g. electrical engineering, or mechanical engineering). While this type of classification seems important, it is often the case that the problem solving activity involved is similar across multiple domains. For example, a designer of a hydraulic pump and a designer of an electric motor might both undertake their problems similarly by

decomposing the design problem into smaller problems, solving the individual problems separately and then integrating the parts. The strategy is thus the same, but the details of the domain knowledge necessary to accomplish the design are different.

Along a different axis, design can be classified according to the type of problem solving involved, for example in case-based design (e.g. [Daube & Hayes-Roth, 1989]) problem solving takes place by modifying pre-existing designs of “similar” design cases. In case-based design the inputs to the design problem solver are used to index previous design cases with similar inputs. Domain knowledge is then used to modify these previous designs to match the new requirements. Another example is design by successively decomposing a complex design problem into smaller more manageable sub-problems then assembling the overall design from the constituents (e.g. [Brown & Chandrasekaran, 1989]). In this type of design, domain knowledge is applied in decomposing the original problem into meaningful sub-problems. Domain knowledge is also applied in solving the constituent sub-problems. For example, a computer system designer might decompose the problem into two subproblems; that of designing the main computer and that of designing the peripheral devices. Each of these sub-problems may be further decomposed as shown in Figure 11. Routine design uses this approach of hierarchically decomposing a design problem into its sub-problems.

A third dimension for classifying design is along the routineness axis. That is, according to how routine versus how creative the design problem is. Routineness is not an inherent property of the design problem itself but of the problem solving activity. For example, a problem that might be “routine” for an experienced engineer might be completely creative for an engineering student. Design problem solving is commonly classified as one of three

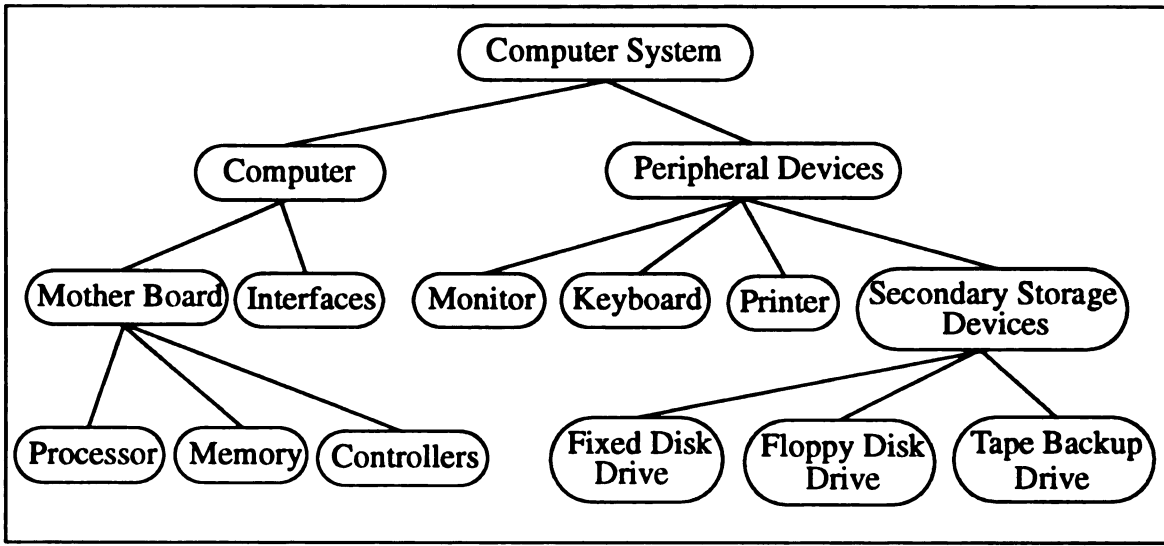


Figure 11: Decomposition of the Design Problem of a Computer System

types creative, innovative or routine [Brown, 1991; Gero, 1990]. These classes can be described as follows¹:

- **Class 1 Design or Creative Design:** This is the type of design that leads to major new products or inventions. In this type of design, neither the types of knowledge needed nor the problem solving strategies are known in advance. It is not common to encounter this type of design in typical industrial settings; it is typical of creative designs to lead to the formation of new companies or new divisions in existing companies.
- **Class 3 Design:** This is the type of design that deals with repetitive tasks that are “routinely” encountered with different but similar requirements. In this type of design both the types of knowledge needed and the problem solving strategies are known in advance. The exact solution, however, is not known in advance. Being “routine” does not make a design problem simple since

1. Class 2 design problem solving is best understood in terms of both class 1 and class 3. Consequently, I will explain both class 1 and class 3 before class 2.

the exact problem solving may still be complex and the possibilities can be numerous making it infeasible to formulate the problem as a table lookup. Routine design deals with the “everyday” type design situation where the “how-to” knowledge is readily available such as the design of the structural components of a new building. While the process of designing these components might be the same for every new building, the large number of input parameters involved makes the solution unique to every case.

- **Class 2 Design or Innovative Design:** This includes design cases which are neither entirely routine nor entirely creative. These cases include situations when a known component is used for a different functionality. This is typical of situations that are usually routine when a new requirement is introduced, driving the design problem away from routineness. Innovative design can also take place as a result of some creative design activity. The creative design might introduce a completely new component that can replace an existing component providing more favorable performance where more favorable can have a variety of definitions (e.g. more efficient, less costly, or more environmentally friendly). For example, the invention of solid-state devices is an instance of creative design that lead to a series of innovative designs replacing vacuum tubes in existing designs with the new solid state devices.

However, it is more realistic to think of design problem solving in general as being continuous along the routineness axis, and not merely at the extremes of it [Brown & Chandrasekaran, 1985], with one end of the axis being definitely routine, and the other end being definitely non-routine. Anything between the poles is considered to exhibit a different level of routineness. Figure 12 illustrates this classification.

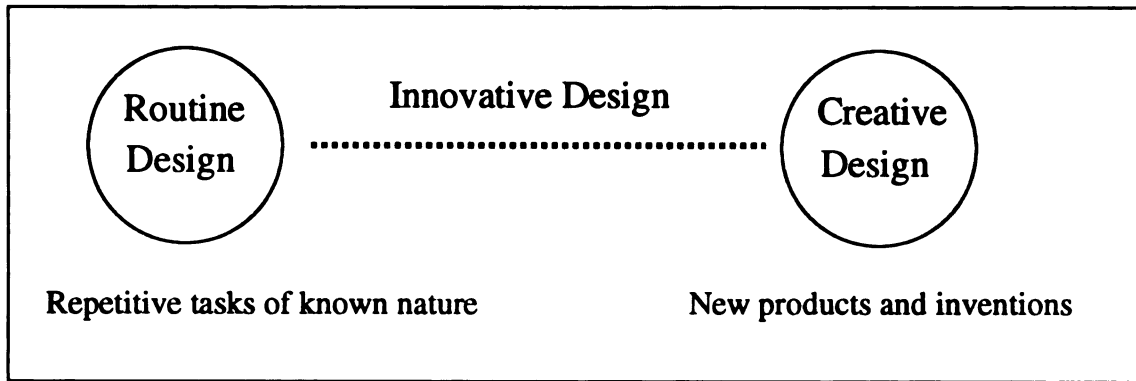


Figure 12: Classification of Design According to the Level of Routineness

Another classification of design problem solving is according to the design activity being conceptual or parametric [Brown, 1991]. Conceptual design refers to design situations where the decisions made are abstract, such as the decision to use a certain type of pump in a hydraulic system, whereas parametric design refers to situations in which the decisions are made to assign values to specific design attributes (or parameters) such as the dimensions of a component. Conceptual and parametric designs typically co-exist in the same design problem and represent the progress of the problem solving activity with time. Conceptual decisions are usually made first followed by a round of parametric design to assign values to the different design parameters. As with the routineness axis, design problems can be viewed as belonging to a continuous axis of “conceptualism”. One end of the axis is purely conceptual design and the other end is purely parametric. This axis is orthogonal to the routineness axis as illustrated in Figure 13.

3.2 Routine Design Problem Solving

Following the generic task approach to knowledge-based systems [Chandrasekaran, 1983; Chandrasekaran, 1985], Brown defined a method for Routine Design, and an accompanying representation language, DSPL (Design Specialists and Plans Language) [Brown &

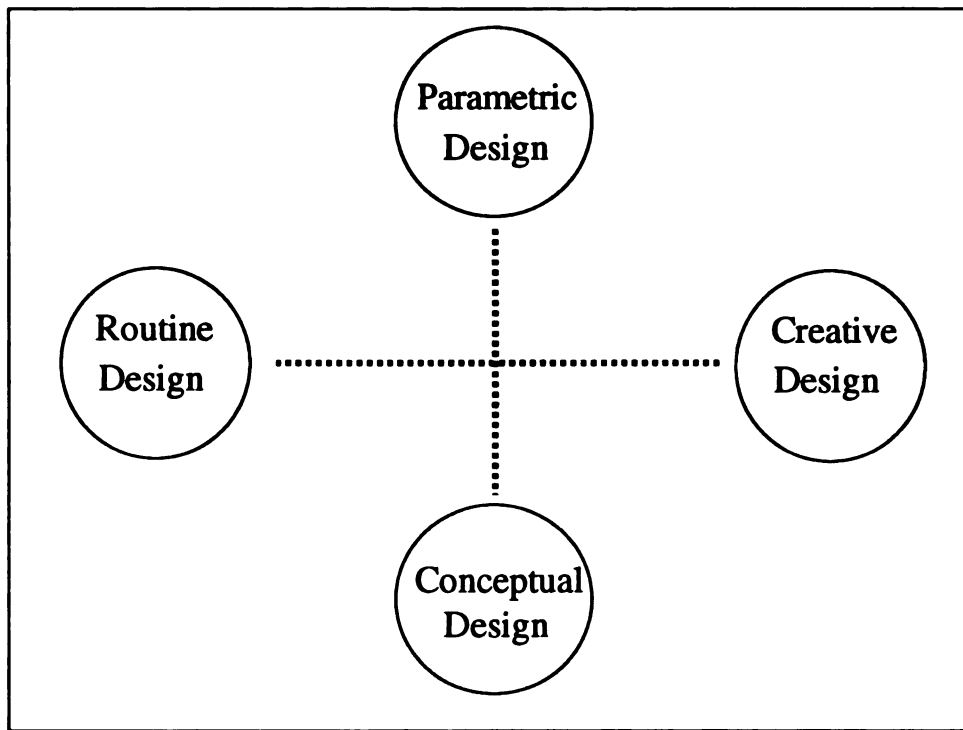


Figure 13: The Orthogonal Axes of Routineness and Conceptualism

Chandrasekaran, 1989; Brown, 1987; Brown & Chandrasekaran, 1985; Brown & Chandrasekaran, 1986].

The method is based on a hierarchy of cooperating specialists each responsible for an identified part of a complete design. The higher level specialists in the hierarchy typically represent more conceptual aspects of the design process, whereas the lower level specialists represent more parametric aspects of the design process. This hierarchy of specialists can be viewed as analogous to a group of human designers cooperating to solve a design problem. The top-level specialist can be viewed as the project leader, with the intermediate levels as team leaders. The lowest level specialists in the hierarchy are analogous to the young engineers doing the actual computations and assignment of values.

In Routine Design, each specialist follows one of a set of prespecified plans. Plans prescribe the problem solving actions to be followed and are defined at the time of building a

design system. While the pieces of the design process (the plans) are known in advance, the combination of pieces as well as their ordering is typically determined during the actual problem solving. While each specialist has a predefined set of plans to choose from, the actual plan to be followed is dynamically chosen during problem solving based on the input design requirements as well as the results of the design established so far. The choice of plans for every specialist can therefore lead to the generation of a design which is novel due to the use of plans by the different specialists that have not been used together previously. Routine Design, however, is not suitable for completely novel design problems.

3.2.1 Problem Solving Agents in Routine Design

Problem solving knowledge in Routine Design is represented in the form of a collection of agents of varying types. In the following discussion, we examine the different types of agents involved in Routine Design. For a detailed discussion of Routine Design refer to [Brown & Chandrasekaran, 1989].

3.2.1.1 Specialists

A specialist is the basic building block in a Routine Design system. It is a place holder for the design plans and the selection mechanism for selecting among the different plans. As mentioned above specialists can be thought of as analogous to a community of human designers cooperating to solve a design problem. As with the human designers, specialists in a design system may not all be involved in the solution of a given problem. A specialist might have a plan that uses some of its sub-specialists and another plan that uses other sub-specialists. The decomposition of a design problem-solver into a hierarchy of specialists mirrors the decomposition of the design problem into sub-problems. This decomposition does not necessarily reflect a structural decomposition of the artifact being designed.

3.2.1.2 Plans

A plan in Routine Design specifies the course of action to be taken to pursue the design from the current status. A specialist has one or more plans to choose from. The choice is done by a selector according to the input data and/or the current problem solving status. A plan may contain invocations of other specialists, executing design tasks, or checking design constraints, or a combination of these actions. The actions performed by a plan can be better understood using the human designers analogy. A designer (specialist) can have more than one method (plan) to solve a problem. In each of these methods, the designer may assign parts of the problems to other designers (invoke other specialists), perform a part of the design problem (execute a task) or check the consistency of the parts of the parts of the design produced by the other specialists (check constraints). The designer usually performs a combination of these duties in an order dependent upon the problem being solved.

3.2.1.3 Step

A step is the basic design action in Routine Design. A step is responsible for setting the value of one design attribute. The value assigned can be the result of performing some computation or it may be selected using a pattern matcher.

3.2.1.4 Task

A task is a collection of related steps. For example, a task may be the design of the physical dimensions of some part, and it might have a step for the length and another for the width of the part.

3.2.1.5 Plan Sponsor

Each plan is associated with a sponsor, which is typically in the form of a pattern matcher that has the necessary knowledge to examine the current status of the design and/or the

input data and assess the applicability of its plan. In other words, a sponsor determines if its plan is suitable (and how suitable). Sponsors are used by plan selectors to select from among the plans available to a specialist.

3.2.1.6 Plan Selector

Each specialist has a plan selector associated with it. Selectors consult sponsors of the different plans and decide which plan to invoke. Sponsors provide the selector with applicability ratings of their respective plans. The selector selects the plan with the highest applicability rating and applies domain knowledge to select among similarly rated plans.

3.2.1.7 Constraints

A constraint is a check on the parts of the design accomplished so far. They are typically in the form of a comparison between the values of two design attributes. A constraint can trigger a failure condition and invoke a failure handler to handle this failure. A constraint can be at any level of the design process: at the specialist level, the plan level, or the step level.

3.2.1.8 Failure Handlers

Failure handlers are invoked in situations where a constraint fails, or a design component (e.g. a plan) fails due to failure of one or more of its constituents. A failure handler examines the failure and invokes a suitable redesigner.

3.2.1.9 Redesigners

Redesigners are invoked in failure situations. They examine the failure condition and either specify corrective action (e.g. increasing the value of an attribute by some increment) or prescribe backtracking.

3.2.2 Problem Solving in Routine Design

Problem solving in Routine Design consists of three major steps in the following order:

1. Requirements checking
2. Rough design
3. Design

Requirements check is a preliminary step to the actual design problem solving where the input requirements are checked for completeness and consistency. The requirements checking phase of design problem solving involves checking that all the needed inputs to complete the design are given as well as checking for contradictory inputs (such as specifying that a material to be designed needs to have a melting temperature of at least 400°F while specifying that the same material is going to be used as a lining for an oven with an ambient temperature of 450°F).

The rough design step and the design step, both utilize the problem solving agents described in Section 3.2.1 to perform their duties. The decomposition of the problem into a hierarchy of specialists is fixed for both rough design and design. However, rough design and design follow two separate sets of plans. The plans for rough design are usually smaller and less complicated than the corresponding plans for design. In most cases, a rough design plan is a subset of a corresponding design plan. Rough design can be differentiated from design in two different ways [Brown & Chandrasekaran, 1989]:

- Rough design may be less detailed than design
- Rough design can be “rough” in terms of computing approximate values for design attributes.

The purpose of the rough design step is to quickly identify causes of failure, and resolve them or perhaps abort the design processes before expending a lot of effort on it.

In both design and rough design, problem solving works as follows:

1. First the top level specialist is invoked in the appropriate mode (design or rough design).
2. The top level specialist requests its selector to select a plan to pursue from among the plans available to that specialist.
3. The selector requests the sponsors of the different plans to assess the applicability of their respective plans.
4. The sponsors examine the current design status and the input data (design requirements) and assign applicability ratings to their plans.
5. The selector examines the applicability ratings and selects a plan to pursue.
6. The chosen plan is invoked.
7. The plan takes one or more of the following actions:
 - a. invoke other specialists in which case these specialists will follow a scenario similar to that followed by the top specialist (steps 2 through 6).
 - b. execute tasks. These tasks consist of steps each of which performs a computation or uses a pattern matcher to assign a value to a design attribute.
 - c. check constraints and if they fail, invoke a failure handler. The failure handler will examine the failure and invoke a redesigner. The redesigner further examines the failure and either prescribes a corrective action (such as incrementing the value of an attribute by a given increment), or prescribes backtracking to a specific point in the problem solver.

8. Failure of a redesigner will lead to the failure of the plan. If the plan fails, an alternate plan is selected from among those available to the specialist. If no other plans are available, the specialist fails, leading to a failure condition in the calling plan. If the top-level specialist fails, the whole design system fails indicating that no design to the specified requirements is possible given the domain knowledge that is available.

3.2.3 Representative Routine Design Applications

3.2.3.1 An Engineering Design Example

Figure 14 illustrates the hierarchy of design specialists in AIR_CYL [Brown & Chan-

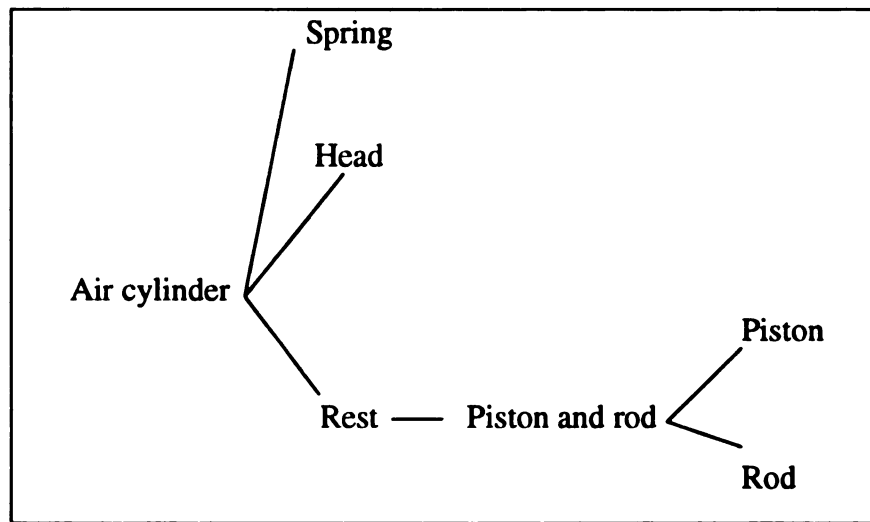


Figure 14: Specialist Hierarchy for AIR-CYL System

drasekaran, 1986], a system for the design of air cylinders. The highest level specialist (Air cylinder) decomposes its task and distributes it among its subspecialists (Spring, Head, Rest). The lowest level specialists are responsible for setting the values of the design parameters. In this example, the decomposition of the specialist hierarchy matches the decomposition of the problem solving activity, namely: design the spring, then design the

head, then design the piston, then design the rod. Every specialist in the hierarchy is used in every problem solving situation.

3.2.3.2 A Planning Example

Figure 14 shows the hierarchy of the specialists for the mission planning assistant (MPA)

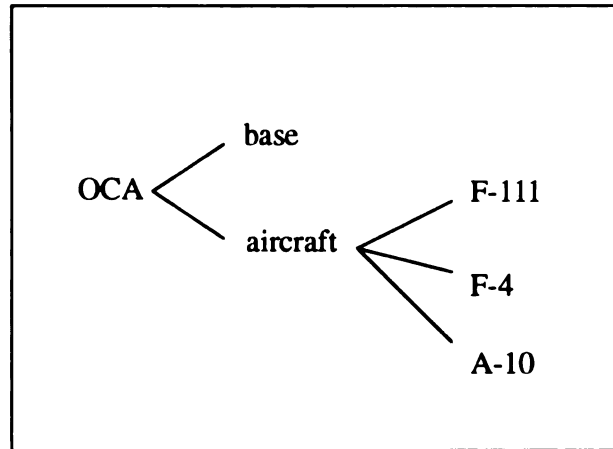


Figure 15: Specialist Hierarchy for MPA System

system [Chandrasekaran, Josephson, Keuneke, & Herman, 1986]. It is a system for planning an offensive counter-aircraft mission. The highest level specialist (OCA) decomposes the problem into two subproblems; deciding which base to use, and configuring an aircraft for the mission. These two tasks are assigned to two corresponding sub-specialists. The aircraft subspecialist assesses the situation, and decides on the type of aircraft to use (F-111, F-4, or A-10), and then assigns the task of configuring the aircraft to one of three subspecialists depending on the type of aircraft chosen. In this example, the decomposition of the hierarchy of specialists does not precisely match the decomposition of the problem into

subproblems and thus not every specialist in the hierarchy is used in every problem solving situation.

3.2.3.3 Discussion

From the above two examples we can see that the decomposition of the problem into a hierarchy of specialists to represent the major tasks involved in problem solving is not necessarily a decomposition that mandates the involvement of every specialist in every problem solving situation. At any branching point, the decomposition can typically mean one of three cases:

1. A decomposition into the subtasks at this point.
2. A decomposition into several alternatives.
3. A combination of both cases.

3.3 Other Routine Design Research

Knowledge-based systems sometimes suffer from being brittle: once they encounter a situation that was not anticipated at system building time, the system fails. Consequently, many researchers strive to add a learning component to their knowledge-based systems to add new knowledge to the system once a new situation is encountered. Motivated by the desire to add a learning capability to Routine Design systems, several lines of research emerged.

The research reported in [Brown & Chandrasekaran, 1989] attempts to reuse design knowledge by investigating how evidence from knowledge such as function, structure, and design cases, can be integrated so that an intelligent CAD system can learn to automatically decompose design problems.

The research reported in [Horner & Brown, 1990; Meehan & Brown, 1990; Sloan & Brown, 1988] is geared towards making design systems more efficient using knowledge compilation¹. This research utilizes the fact that a problem becomes more routine as experience is gained. By analyzing the history of problem solving using a design system, they include measures from that experience to enhance the performance of the system. Sloan and Brown [Sloan & Brown, 1988] use knowledge of corrective actions resulting from constraint failures to relocate constraints in the problem solving hierarchy to increase efficiency. Horner and Brown [Horner & Brown, 1990] make use of design expectations to add constraints that are originally missing to parts of the problem solving hierarchy which consistently fail to meet design expectations. Meehan and Brown [Meehan & Brown, 1990] make use of constraint failure patterns to improve the problem solver in two way; constraint absorption and constraint relaxation. Constraint absorption looks for redundant constraints that always succeed (possibly because of an earlier constraint subsuming it or because a calculation always guarantees its success). These constraints are absorbed in order not to clutter the problem solving activity. Constraint relaxation is concerned with constraints that periodically fail by small increments. This may be caused by a small error in the designer's knowledge, and if it is not in a key part of the design, the constraint is relaxed.

Task-specific architectures are designed to facilitate the process of building knowledge-based systems, by using a vocabulary that is "natural" to the type of problem addressed. However, this same feature designed to simplify system building can form an obstacle for a system builder if the needed constructs are not part of the vocabulary of the method used. To overcome this bottleneck, the recent research by David Herman [Herman, 1992] extends the ideas of the Routine Design approach by allowing the user² to extend the lan-

-
1. Knowledge compilation is a term that refers to the encapsulation of basic knowledge about some domain into a readily usable form such as a rule.
 2. by user here it is meant the developer of a knowledge-based system and not the end user of the knowledge-based system.

guage by defining additional knowledge processing methods. Herman [Herman, 1992] defined a language called DSPL++ as an extension to the original DSPL language. DSPL++ retains most of the constructs used by DSPL while allowing them to be integrated with user-defined problem solving methods.

Chapter 4

Problem Definition and Research Goals

4.1 The Nature of Knowledge-Based Systems Research

Knowledge-based systems is a multi-disciplinary field. A knowledge-based system is a medium for knowledge representation and reasoning in a given domain of expertise (e.g. the design of composite materials or the diagnosis of infectious diseases). Consequently, research in knowledge-based systems is typically composed of the solution of two simultaneous problems.

The first problem is a domain problem such as the need to automate a design process in order to shorten the specification to manufacturing time. The second problem is a knowledge-based problem; assembling and structuring the right mechanisms to accomplish a successful solution for the domain problem. Without a domain problem, research in knowledge-based systems would merely be theoretical research without an experimental mechanism for verifying the validity of the results. Hence, the first problem (the domain problem) should be the driving force for research in knowledge-based systems. Research in knowledge-based systems should thus follow the following pattern:

1. A domain problem arises and needs to be addressed (such as trouble shooting an automobile engine).
2. Using the full range of knowledge-based systems techniques, a system should be designed to solve the domain problem.
3. The resulting system should be analyzed in order to generalize it for solving other problems.

4. A problem solving method might be abstracted for solving similar¹ problems. This step should be associated with the development of a problem solving “tool” for use in future similar problems.
5. By applying the above steps to numerous problems, numerous techniques are generated for the solution of different classes of problems.
6. When a new problem arises, researchers attempt to fit it into one of the available techniques.
7. If a problem is deemed not to fit into one of the existing techniques, this opens a new line of research for analyzing the problem, designing a solution mechanism for it and later generalizing the solution into a new technique.

As an example, consider the MYCIN system [Shortliffe, 1976]. The MYCIN system was originally developed for diagnosing infectious diseases. Here, the driving force was to develop a system for the diagnosis of infectious diseases. MYCIN was implemented as a backward-chaining rule-based system. By analyzing MYCIN’s problem solving, a problem solving tool known as EMYCIN² [van Melle, 1981] was developed. This tool embodies the domain-independent parts of MYCIN, in other words it is a backward-chaining ruled-based tool. Abstracting EMYCIN from MYCIN is an example of a first-generation analysis of problem solving, i.e. describing the problem solving by describing the language used rather than the problem solving methods used. Later, the MYCIN system was re-

1. The definition of “similar” varies a great deal among researchers especially between researchers of the first generation knowledge-based systems for whom “similar” meant “can use the same language” and researchers of the second generation knowledge-based systems for whom “similar” means “uses the same control strategy.”

2. EMYCIN stands for “Essential MYCIN.”

examined and described in terms of the type of problem solving involved [Clancey, 1985].

MYCIN was thus identified to be performing the following tasks (see Figure 16):

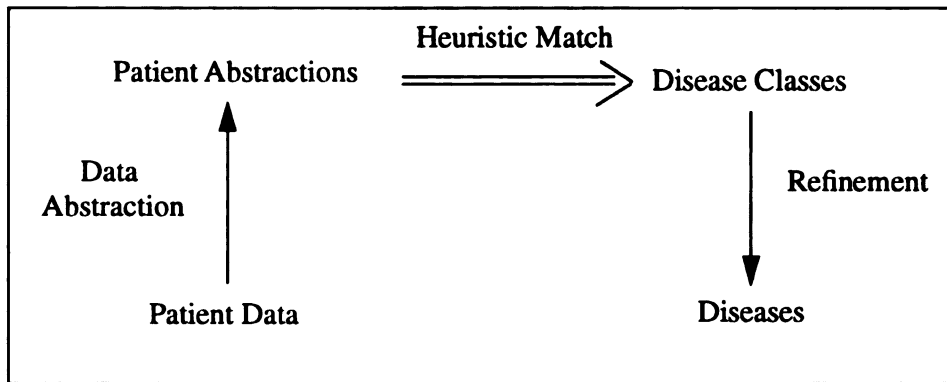


Figure 16: Inference Structure of MYCIN (from [Clancey, 1985])

1. data abstraction, for example abstracting a white blood cell count of less than 2500 into low white blood cell count.
2. heuristic Matching by matching the abstracted data into classes of diseases.
3. refinement by identifying a specific disease among the class of diseases.

Clancey labelled the type of problem solving used by MYCIN “Heuristic Classification”, and abstracted it (see Figure 16) into a problem solving tool known as HERACLES (Heuristic Classification Shell) [Clancey, 1985].

The above example illustrates the effectiveness of having domain problems as the driving force for research in knowledge-based systems. Unlike other basic sciences, it is difficult for purely theoretical research in knowledge-based systems to yield results that would be useful for practical applications. On the other hand domain-driven knowledge-based

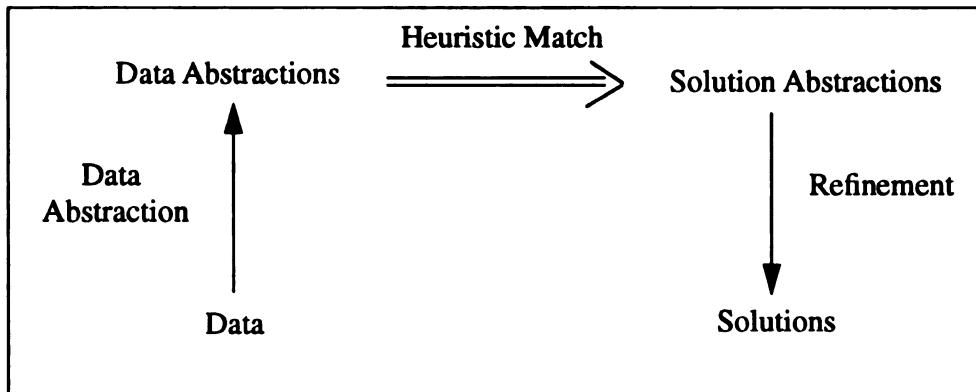


Figure 17: Inference Structure of Heuristic Classification (from [Clancey, 1985])

systems research is more likely to lead to solutions that are generalizable to other similar problems.

This research originated for the design of thermoset polymer composite materials. In the rest of this chapter, I will present the thermoset composite materials design problem as well as the knowledge-based research needed to solve this problem.

4.2 Problem Definition

The goal for this research is to analyze from a computational point of view the design of thermoset polymer composite materials, and from this analysis to develop a knowledge-based system for the design of these composite materials. In this section, I will describe the composite materials design problem and the knowledge-based systems extensions needed to solve this problem.

4.2.1 The Composite Materials Design Problem

Composite materials offer an attractive alternative to the use of metals in many applications. This is mostly attributable to their favorable properties, principally their lighter weight, and, in some cases their superior structural properties. However, composite materials have not been utilized to their full potential especially in consumer-oriented applications. Due to the high cost and considerable time involved in the initial design of composite materials, the use of composite materials have been—with few exceptions—limited to “high-tech” industries such as the aerospace industries. Knowledge-based systems can serve as a vehicle for the transfer of knowledge from these “high-tech” industries to consumer industries such as the automobile industries.

Composite materials fall into several categories according to the type of matrix¹ material used in the composite material. Figure 18 shows a classification of the different types

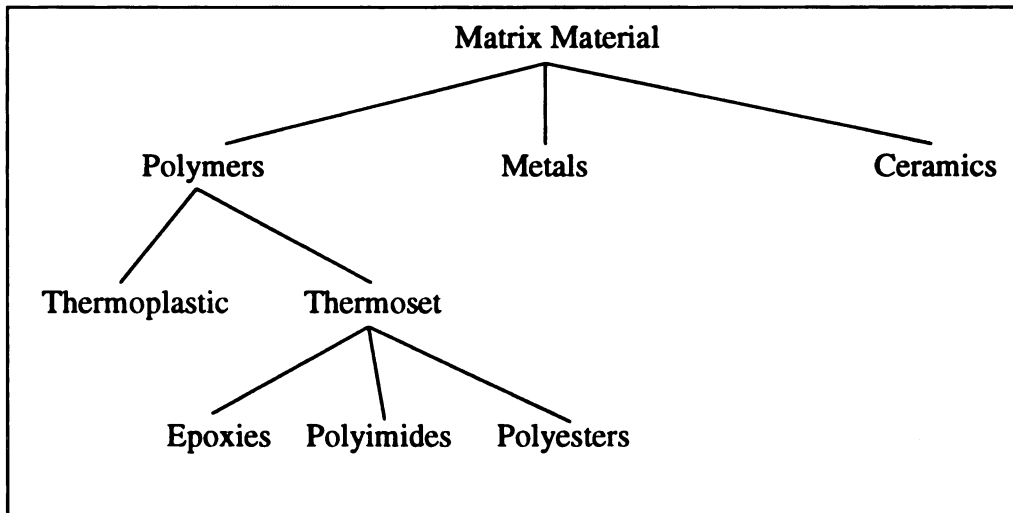


Figure 18: Types of Matrices Used in Composite Materials

1. A matrix is the material that forms the bulk of the composite material. It usually acts as a holding material for another reinforcing material (such as a fibre).

of matrices used in composite materials. This research focuses on the design of thermoset polymer composite materials, which constitute the majority of all known composites.

This research is part of a larger project whose target is automating the entire life cycle of a composite material from being an idea to being a finished product. Figure 19 illustrates

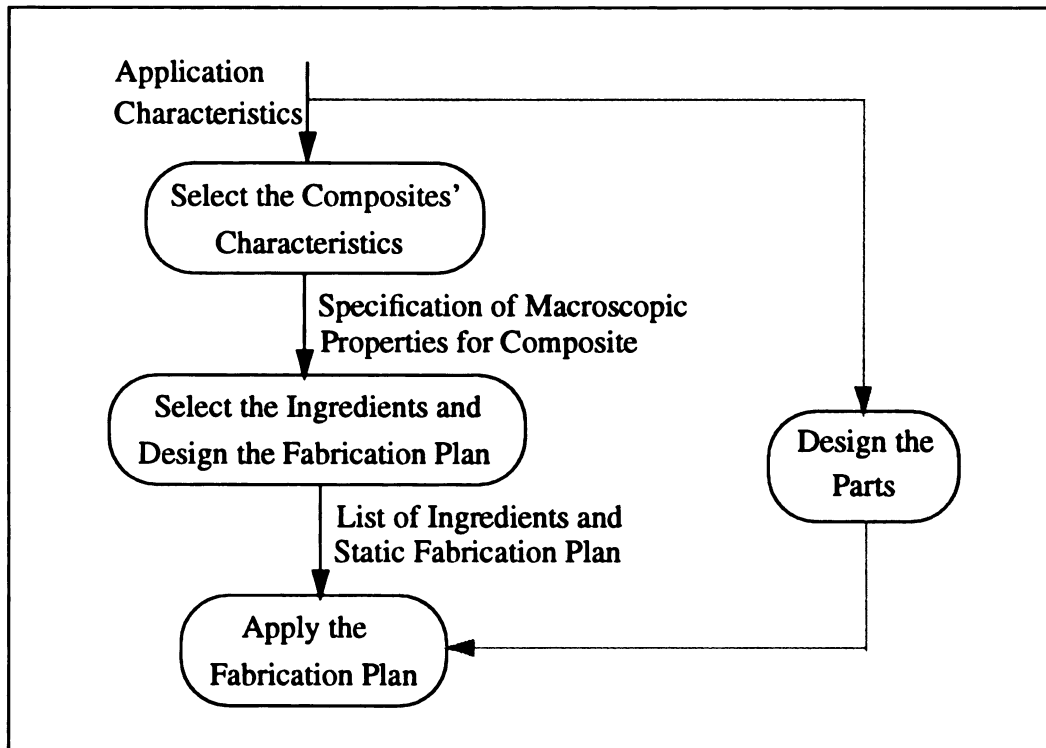


Figure 19: An Information Processing View of the Composite Materials Fabrication Life Cycle

the life cycle of a composite material until it is realized as a product. Four components are involved with the manufacturing of a composite material:

1. The first component analyzes the characteristics of the application domain and generates a set of macroscopic properties for the composite material. The inputs to this module are high-level descriptions of the application such

as load distributions if the application is a structural one, and the outputs are a set of macroscopic properties for the composite such as tensile and thermal properties.

2. The second component uses the macroscopic properties of the sought composite and applies domain knowledge to generate a design of a composite material expressed as a list of ingredients and a plan for the fabrication of a composite out of these ingredients.
3. The third component operates in parallel with the above two activities to generate a design for the parts involved.
4. The fourth component is a control system for applying the fabrication plan. This component uses the fabrication plan generated by the design system, and the information about the geometry of the parts to control the application of the fabrication plan in two ways:
 - a. It keeps the plan “on track” by sensing the environment (i.e. measures the temperature and pressure) inside the fabrication unit and generating the necessary control signals to adjust this environment as needed for the fabrication plan.
 - b. It uses the sensed parameters to generate an abstract characterization of the state of cure¹ of the composite material, and if the material is not curing as expected, this module would use domain knowledge to alter the fabrication plan to meet the expected cure pattern.

1. The term cure refers to the changes that take place in the composite material and transform it from simply a mixture of materials to a composite material.

The focus of this research is the second module or the design of a composite material given macroscopic properties for that composite. Figure 20 shows a sample input and output from a composite materials design system. A typical chronology for designing a composite material is as follows [Venkatasubramanian, Lee, & Gryte, 1987]:

1. First, macroscopic properties which are desired in the completed composite are set. Properties such as final material tensile modulus, resistance to acids and alkalis, and electrical resistance are parametrized.
2. Based on these desired properties, the composite designer proposes an initial plan for the production of the composite. This plan includes both an ingredients list for all materials to be initially present, and a preliminary protocol which states how the initial mixture is to be processed.
3. Next, the composite designer estimates how well the proposed composite design meets the initially stated, desired properties. This estimate is typically carried out by actually producing samples of the composite, then performing laboratory testing to determine properties of interest. Ultimately, a proposed composite design will result in an actual material which can be subjected to laboratory testing.
4. Following one round of design proposing, and matching to specifications, successive rounds of redesign are usually required before convergence of proposed composite properties to desired properties takes place.

One goal of composite researchers is to provide better models for proposing designs in order to limit the number of candidate materials which must actually be fabricated for testing. The ultimate goal for a knowledge-based system that would undertake the problem of accomplishing this design task is thus to minimize the number of iterations through this

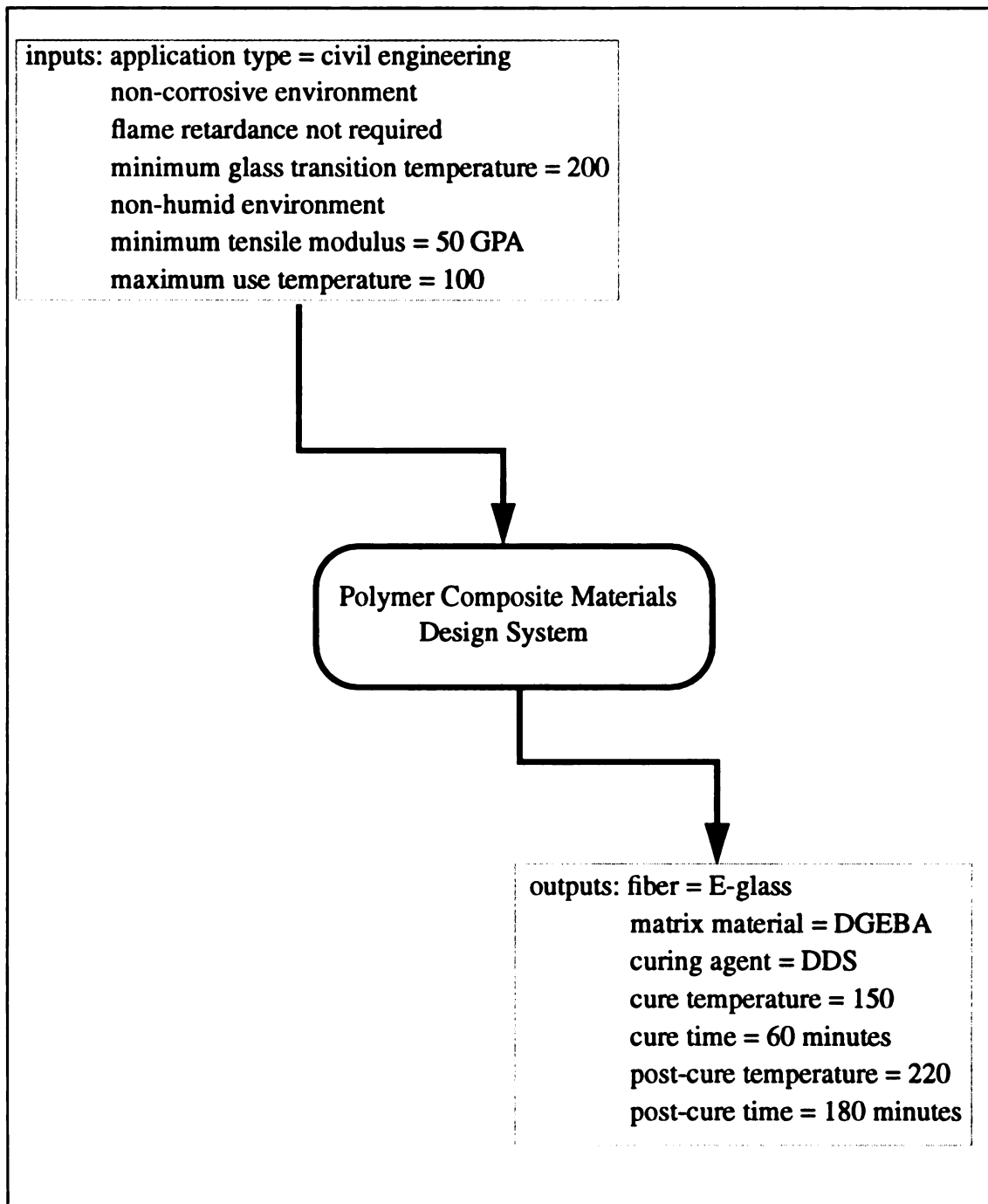


Figure 20: Sample Input/output for the Design of Composite Materials

design-fabricate-test cycle, thereby reducing the cost of designing a composite material to specifications.

4.2.2 Knowledge-Based Extensions Needed for the Design Of Composite Materials

In building a knowledge-based system for any given domain of expertise, the nature of the domain dictates the structure and the types of knowledge that should be included in the knowledge-based system. In this research, the domain problem is the design of thermoset polymer composite materials. The characteristics of this domain thus serve as the driving force for this research.

By examining the design problem of polymer composite materials we find that the design of composite materials mostly relies on previous experiences of composites designers. Designers usually attempt to generate new designs by systematically modifying existing designs to meet altered specifications. Due to the field of composite materials being relatively new, design of composite materials usually involves multiple iterations of designing a material, manufacturing it and testing it. A knowledge-based system for the design of composite materials should capture these characteristics of the design process. Hence, such a knowledge-based system should have the following capabilities in order to successfully accomplish this design task:

1. The target knowledge-based system should be able to use “expert” knowledge to design a composite material given a set of requirements.
2. The system should be able to make use of previous design experience in two ways:
 - a. If a design situation is the same as a previous one, it should be recognized as such and the previous design directly supplied.

- 100

100

In addition to the above considerations, an additional requirement for design problem solving in composite materials, and in most engineering domains, is the isolation of dynamically changing factors such as cost and market conditions from the domain knowledge about generating the designs. By isolating these factors, the main design knowledge-base can be stabilized by concentrating only on scientific design methodology. Furthermore, by isolating static factors and dynamic factors, several satisfactory designs can be generated and kept for future reference upon any change in dynamic factors. To accomplish this isolation, it becomes necessary to decompose the design process into two separate processes:

1. First, generate a set of feasible designs which all satisfy the given design requirements.
2. Use the dynamic market factors to select among the generated designs.

However, all current knowledge-based systems are geared towards generating a satisfactory design given a set of requirements (see Section 2.1 for a survey of knowledge-based design systems). As a result, it becomes necessary to develop a mechanism for generating multiple designs that meet a common set of requirements.

4.3 Research Goals

In the above discussion, we examined the problem posed for this research. In this section I will enumerate the goals for this research both for the design of composite materials and for knowledge-based systems.

4.3.1 Specific Goals in Composite Materials

The following list comprises the set of goals for this research in the composite materials area:

1. Building a knowledge-based system to automate the design of thermoset polymer composite materials to specifications.
2. Generating multiple designs for polymer composite materials and selecting among them using one of two methods:
 - a. automatically according to a set of dynamic market factors (e.g. cost and availability of raw materials).
 - b. manually according to the needs of a human designer.
3. Having the ability to easily modify the system as new materials and processes are developed.

4.3.2 Specific Goals in Knowledge-Based Systems

In knowledge-based systems the goals of this research are:

1. The development of an integrated architecture for engineering design of composite materials. As discussed in Section 4.2.2, this architecture should be capable of:
 - a. using previous design experience,
 - b. using experiential knowledge to generate design from “scratch” if no previous design experience exists,
 - c. simulate the processing part of the resulting designs, and use the simulation results to modify the designs if needed to avoid the costly manufacturing and testing of samples.
2. The development and implementation of a mechanism for efficiently generating multiple designs that meet a common set of specifications.

3. The development and implementation of a selection mechanism for selecting among the multiple designs.

4.4 Expected Outcomes

In view of the goals of the research listed above, this research is expected to have several results both in composite materials and in knowledge-based systems.

4.4.1 Expected Outcomes in Composite Materials

By developing a knowledge-based system for the design of thermoset polymer composite materials, several benefits are expected:

1. A reduced specification to manufacturing time by reducing the number of iterations of producing samples and testing them before a final design is achieved.
2. The resulting design system can be used as a vehicle for technology transfer.

4.4.2 Expected Outcomes in Knowledge-Based Systems

The expected outcomes of this research in knowledge-based systems are:

1. Proving the utility of integrating multiple knowledge-based systems techniques to provide a functionality each of the individual techniques is not individually capable of providing.
2. The development of a framework that can serve as a starting point for further research on engineering design in other domains.
3. The development of a mechanism for efficiently generating multiple designs meeting a set of specifications. This mechanism can be used in other domains where this capability is required.

4.5 Conclusion

In this chapter, I presented the problem posed for this research; namely the design of thermoset polymer composite materials. I enumerated the goals and expected outcomes of this research. While thermoset polymer composite materials constitute more than 80% of known composites, they still have not been utilized to their full potential. A design system that reduces the specification to manufacturing time and assists in the transfer of technology from “high-tech” to consumer-oriented industries is essential to achieve this potential.

In the next chapter, I will present an integrated architecture for the design of thermoset polymer composite materials. In Chapter 6, I will present the details of the technique I developed for generating a set of designs that meet common specifications. In Chapter 7, I will present the details of the implemented thermoset polymer composite materials design system, as well as the experimental results from testing this system. In Chapter 8, I will present the details of the selection mechanism for choosing among the generated designs. In Chapter 9, I will present the results of this research and point out the directions for future research.

Chapter 5

An Integrated Architecture for the Design of Thermoset Polymer Composite Materials

In this chapter, I will present an integrated problem solving architecture for the design of thermoset polymer composite materials. Two versions of the architectures are presented. The first version generates a single design that meets a given set of design specifications. The second version utilizes a novel technique which I have developed for generating multiple designs meeting a common set of specifications. This architecture also includes the component I implemented for the selection of a design from the generated set of designs.

5.1 Single Design Architecture

In the architecture shown in Figure 21, the first step is to define the design requirements (the desired properties of the finished product e.g. glass transition temperature, tensile modulus, etc.). To avoid duplicating design efforts, a library of design cases is first consulted to identify any available previous design cases that resulted in a composite whose properties are similar to the current requirements¹. The case library responds by either supplying a design case, or by announcing that no such case exists. Two routine designers are available in the system, one of these (Routine Designer 1) is for handling the cases where previous design cases exist, and the other (Routine Designer 2) for the cases where a previous design case does not exist.

In situations where previous design cases exist, experiential knowledge is utilized to modify an existing design to meet the altered specifications. Routine Designer 1 in Figure 21 serves this functionality. The input to this designer consists of the current design requirements and the design case retrieved from the library together with the specifications

1. Typically these will be similar requirements but not identical.

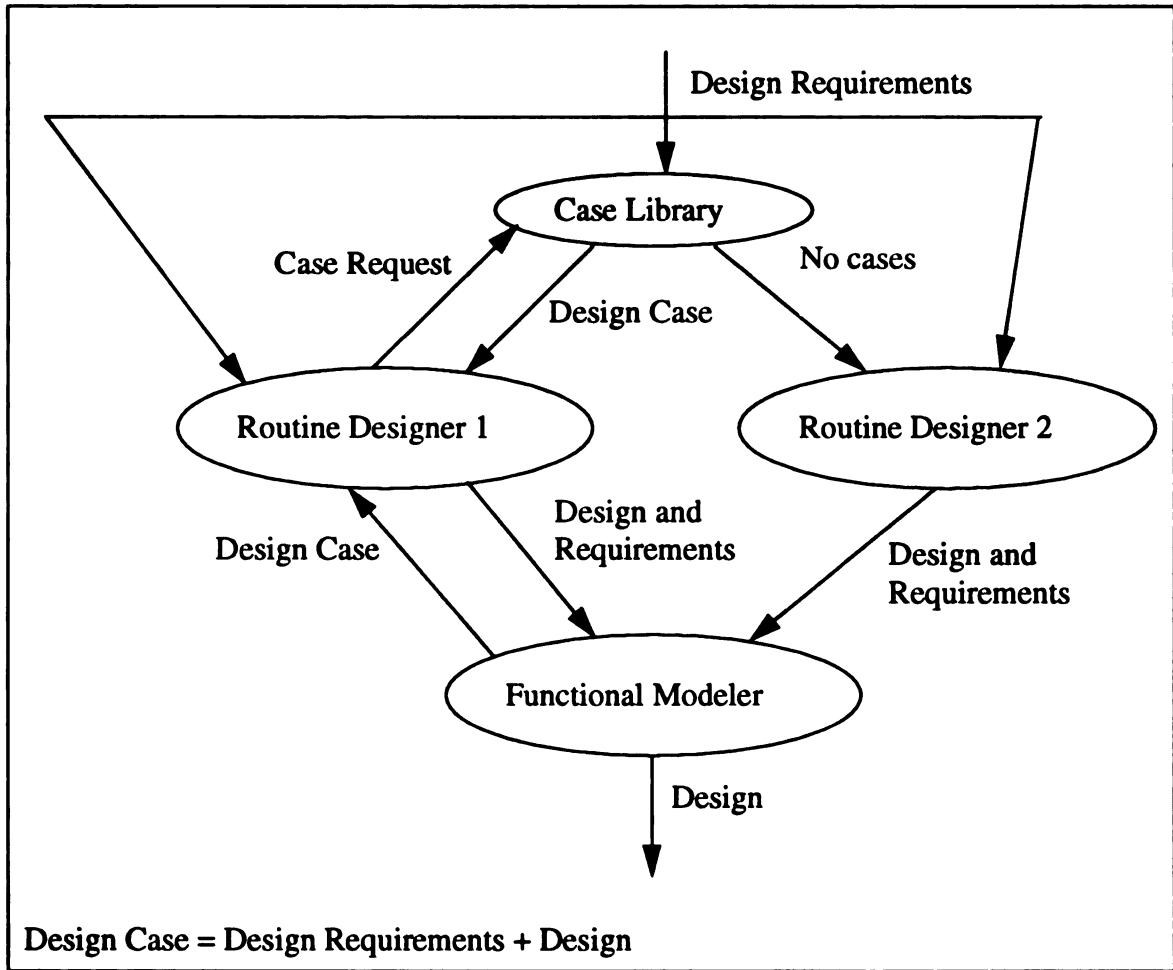


Figure 21: Single Design System Architecture

of the composite material produced using that design. The routine designer then alters the supplied design to try to make it applicable for the new requirements. During this modification process, the routine designer interacts with the functional modeler¹ in a fashion to be discussed shortly.

The situations where no previous design cases exist are handled by another routine designer (Routine Designer 2 in Figure 21). This routine designer is also based on the expe-

1. Functional modeling is a technique for model-based reasoning. Details of functional modeling can be found in [Sembugamoorthy & Chandrasekaran, 1986].

ritional knowledge of expert designers and should have the basic knowledge for designing a composite material “from scratch” based on required specifications. During the design process, this routine designer also interacts with the functional modeler in a similar fashion to the interaction exhibited by the other routine designer.

If Routine Designer 1 (the routine designer responsible for altering a design case to fit the specifications) fails (does not have the knowledge necessary to modify the design successfully), it should report to the case library requesting a different case, in which case control might be transferred to the other routine designer if no other cases exist.

The integration of routine design and functional modeling will be along the lines of the integration of classification and functional modeling described in [Sticklen & Chandrasekaran, 1989]. The functional modeler should have the basic knowledge about composite materials and about the materials involved to enable it to reason about the composite material’s properties from “first principles.” Knowledge needed here involves basic properties of the different materials (e.g. chemical properties, molecular structure, and cost). The information processing task for the functional modeling is to simulate the cure process in order to predict the properties of the finished composite given as input a design for a composite material (e.g. materials, proportions for quantities and fabrication protocol). The functional modeler thus serves as a consultant for both routine designers.

A routine designer feeds the functional modeler with a potential design. The functional modeler predicts the properties of the proposed design, if they satisfy the input requirements, the design is accepted. However, if the predicted properties do not satisfy the input requirements, the functional modeler feeds back the proposed design and the predicted properties to Routine Designer 1. Any of the two routine designers would consult the functional modeler. However, the feedback of the functional modeler is always supplied to Routine Designer 1 (the routine designer responsible for altering a design case to fit the specifications). The reason for this is that—given the correct domain knowledge—the

design accomplished so far should have properties that are close to the required properties. This design can thus be altered in a manner similar to altering a design retrieved from the case library. Routine designer 1 then modifies the suggested design, and then re-checks with the functional modeler.

If Routine Designer 2 (the routine designer responsible for designing composite materials from scratch) fails, then a complete failure should be reported to the system operator, and a decision has to be made (perhaps relaxing one or more of the design requirements).

Successful design results are reported as a materials list and a processing protocol. The processing protocol is in the form of a sequence of applications of temperature and pressure for specific periods of time. The resulting design is also added to the case library to be referenced for future designs.

5.2 Multiple Design Architecture

I originally developed the architecture presented in section Section 5.1 to produce a single design. However, as I implemented the “Routine Designer 2” module of this architecture, it became apparent that a single design will not suffice and that multiple design alternatives are required to isolate dynamic design parameters (such as cost and availability of raw materials and human designer preference) from static design knowledge (such as the technical issues of how to obtain a design). This way, static design knowledge can be applied to generate a set of designs, then dynamic parameters can be used to select among the produced designs. Section 1.5. The architecture was modified to produce a family of designs as shown in Figure 22.

The multiple design architecture functions in the same fashion as that of its single design counterpart except for the “Routine Designer 2” module which is replaced by 2 modules:

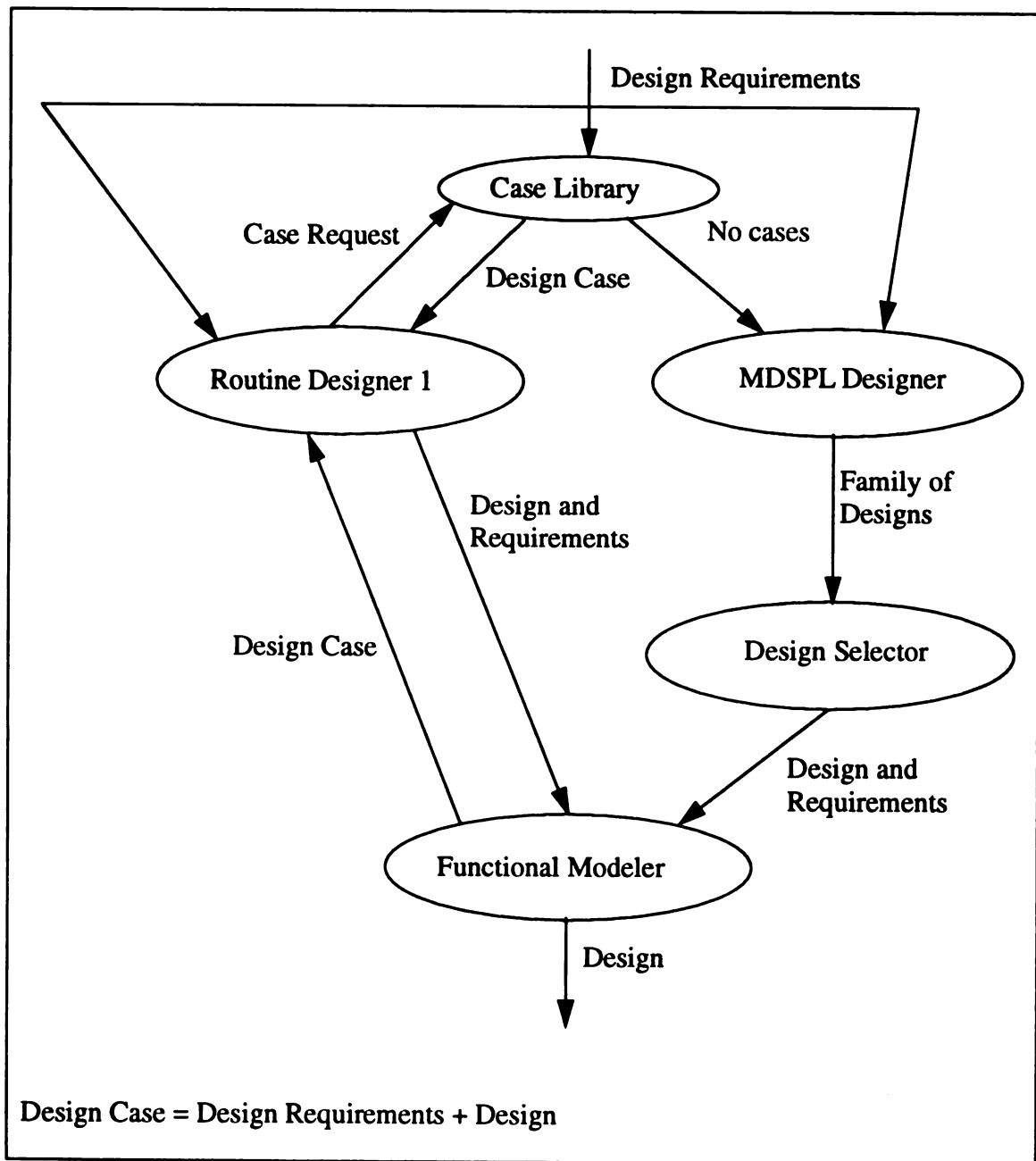


Figure 22: Multiple Design Architecture

1. **MDSPL Designer:** instead of generating a single design to meet the required specifications, this module generates a family of design alternatives to meet those specifications. This module can in many cases be used independently of the other modules to generate a family of designs and then leave the selection process to the designer's preference.
2. **A design selector:** this module examines the family of generated designs and selects a single design to pursue. The selection process can depend on many factors such as marketing factors and availability of materials. This selection process also takes into consideration the user preferences by interacting with the user during the selection process. The system ranks the design according to the estimated cost of producing them and allows the user to make the final decision of the particular design to pursue.

The division of the functionality into, first, generating a family of designs, and then selecting among these designs has the advantage of isolating the highly dynamic factors such as cost and availability of raw materials from the design knowledge. In this way, design can proceed independently of these factors generating a set of feasible designs. The selection process then involves these dynamic factors as well as the user preference in selecting the design to pursue. The other designs can then be stored for possible future changes in cost or availability of the materials.

I implemented and tested the MDSPL architecture and language and developed the MDSPL design module and the design selection module of the above architecture. The case library, Routine Designer 1 and the functional modeler are a target for future research. The MDSPL technique for generating multiple designs is described in Chapter 6 and the selection module is described in Chapter 8.

Chapter 6

The Multiple Design Approach and Implementation

A major component of this research is an approach for generating Multiple Designs satisfying a common set of design requirements and constraints.

The Multiple Design approach is based on the previous work on Routine Design reported in [Brown & Chandrasekaran, 1989; Brown, 1987; Brown & Chandrasekaran, 1986; Chandrasekaran et al., 1986; Chandrasekaran et al., 1989], and reviewed in Chapter 3. In this chapter, I will present the details of my extension—Multiple Design—and its implementation.

6.1 The Relationship Between Multiple Design and Routine Design

The Multiple Design approach uses the same types of problem solving agents used by Routine Design (see Section 3.2.1). An additional type of agents called “design limiters” is also used in Multiple Design (see Section 6.3 for a description of design limiters). However, the use of the problem solving agents in Multiple Design is different than their use in Routine Design. The control strategy for problem solving in Multiple Design is described in Section 6.5.

Since the types of problem solving agents of Multiple Design are a superset of the types of problem solving agents of Routine Design, a design system implemented in Multiple Design can readily be used in Routine Design to generate a single design.

6.2 The Design Database

To better understand the description of the problem solving strategy, it is useful to have a full understanding of the form of the design(s) generated by a Multiple Design system.

For a given set of inputs (design requirements) a Multiple Design system generates a group of trees, where each node represents a value for a design attribute. Each path from a root to a leaf node represents a complete design. Figure 23 shows this form of output graphically (In this case a “design” is a set of values for attribute1, attribute2, and attribute3).

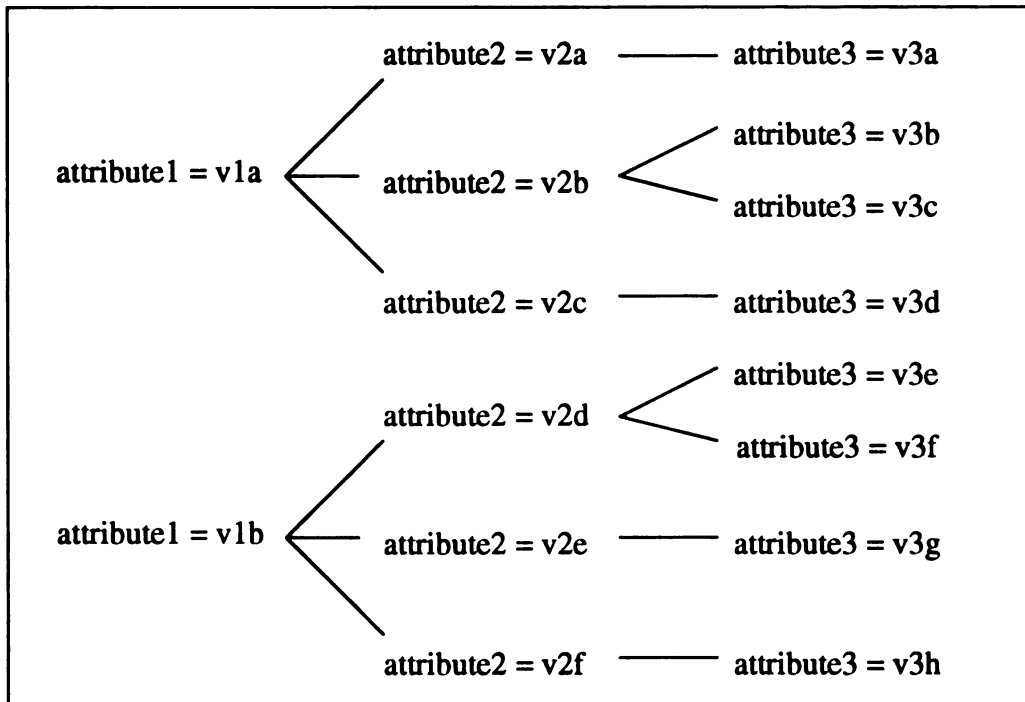


Figure 23: A Graphic Representation Of A Typical Multiple Design Output

A database accompanies each Multiple Design problem solver to track the values for the different attributes and the associations between the different values as they are generated. This database is constructed in the form of a group of trees of attributes and values as shown in Figure 23. In addition to serving as a data storage medium, the database is capable of detecting if a given pattern (a list of design attributes and values) is present as a part of the design database (i.e. is a part of a path from a root to a leaf node). This latter role of the database serves a central role in the problem solving activity. Design decisions are often dependent on earlier design decisions, and thus previous decisions should be checked

before further actions are taken. This checking is implemented by pattern matchers. The database is queried by the pattern matchers to determine if a certain pattern matches.

To simplify the search process in the tree of design values, the tree is structured such that there is a one-to-one correspondence between the design attributes and the levels of the tree, any given attribute always exists at the same level in any branch in the tree. Every attribute also indexes the nodes in the tree in which a value for the attribute is stored. This makes the search for a particular value in the tree very efficient. The tree is also doubly connected (each node points to its parent as well as to its children) to further simplify the search process.

6.3 Knowledge Representation Structures

Knowledge representation for the Multiple Design approach follows for the most part the same taxonomy of problem solving agents as the Routine Design approach described in Section 3.2.1. However, there is an additional type of problem solving agents involved in Multiple Design. These agents are called “design limiters.” The purpose of design limiters is to limit the growth in the number of design alternatives generated. Limiting the number of design alternatives serves a dual purpose:

1. To control the growth in the number of design alternatives generated by eliminating alternatives which are known, from experience, not to work.
2. To eliminate design alternatives which are “too good.” As an example for many applications, generating a composite material that is super strong and posses a high level of heat resistance would be functionally appropriate, though it would be cost prohibitive.

Design limiters are in the form of a pattern matcher and are involved at two levels in the problem solving process; the step level and the task level. At the step level, after the step is

executed and a value is chosen for its attribute, the design limiter for the step (if one exists) is then executed to check the validity of the value chosen for that attribute. Knowledge in step design limiters typically falls into one of two categories:

1. Constraints on the value to be chosen for the attribute. Typically these are to avoid the “too good” situation, usually based on empirical knowledge. For example: “If the tensile modulus of a material is more than 20% higher than the minimum required tensile modulus, then do not choose this material”.
2. Constraints on the value to be chosen based on previous decisions. These are typically empirical decisions to avoid incompatibilities between design decisions. For example: “If material A is chosen as a polymer and material B is chosen as a curing agent and there are other choices then do not use material C for the reinforcement fiber.”

At the task level, after all the steps of the particular task are all executed, the task’s design limiter (if one exists) is then invoked to check the validity of the values chosen by the different steps as a group. This is typically necessary to check the compatibility between the values chosen at the different steps.

6.4 Comparison Between Design Limiters and Constraints

Design limiters are similar in form to the constraints in Routine Design. However, knowledge limiters play a different role than that played by constraints. Constraints are primarily intended to represent knowledge about relations that have to hold true between different design attributes, such as a diameter of a mechanical part having to be smaller than the diameter of another enclosing part. Constraints are used in Multiple design to represent this same type of knowledge. Design limiters on the other hand represent the designer’s heuristic knowledge about decisions that are either:

1. known from previous experience not to result in satisfactory designs, or
2. known from previous experience to result in an overall design that is “too good”. The reason for avoiding such designs is that in many engineering applications, the price of a given design increases as the quality of the design increases, and as long as other designs (that meet the required specifications) exist, more expensive designs need to be avoided.

In Multiple Design, design limiters are used to eliminate design decisions that meet the above criteria in order to eliminate further consideration of these partial designs during the rest of the design process thus reducing the overall complexity of the design process.

In summary, constraints and design limiters are similar in structure, however, they serve a different purpose in problem solving.

6.5 The Control Strategy

The control strategy of Multiple Design is similar to that of Routine Design described in Section 3.2.2. Design proceeds in three successive steps:

1. requirements checking
2. rough design
3. design

In each of the two design modes (rough design and design), the design process begins with the top-level specialist. The top-level specialist executes one or more of its plans. The plans in turn invoke their specialists, tasks and constraints. Specialists proceed by executing their plans. Tasks proceed by invoking their steps, and constraints. Steps utilize pattern matchers to assign values to design attributes. Constraints check the design data and possi-

bly take some corrective measures. However, the control strategy for the Multiple Design approach differs from its single design counterpart in the following ways:

1. **plan selection:** In the traditional single design approach, for any specialist a single plan is executed unless it fails in which case another plan is attempted. It is useful to recall here that the selection process is completed by a plan selector in conjunction with the plan sponsors associated with the different plans. Each plan has a sponsor in the form of a pattern matcher which examines the available data and assesses the applicability of its corresponding plan. Sponsors tag their plans as belonging to one of four categories:
 - a. perfect for the current situation
 - b. suitable for the current situation
 - c. not sure
 - d. not applicable for the current situation

In the single design approach, one of the plans with the highest applicability rating category is chosen for execution. Figure 24 shows an example plan selection in DSPL.

In the Multiple Design approach, all plans in the highest applicability rating category [other than category (d)] are attempted. Only in the event that all the plans in this category fail, will the plans in the next available category be attempted. Figure 24 shows an example plan selection in MDSPL.

2. **Task execution:** An additional component is added to the execution of tasks by invoking the task's design limiter after executing all applicable steps, and thereby possibly eliminating some of the resulting subdesign alternatives.

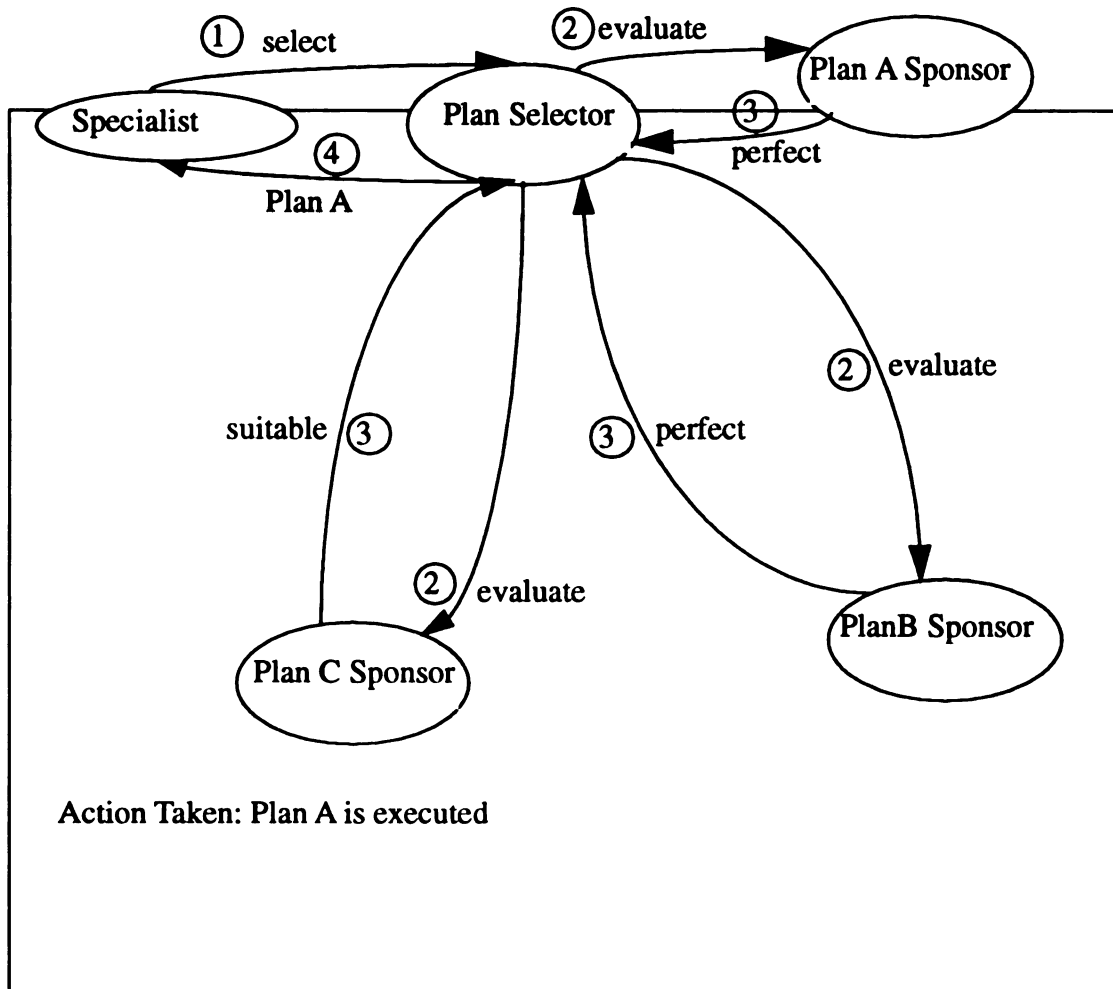


Figure 24: An Example Showing Plan Selection in DSPL

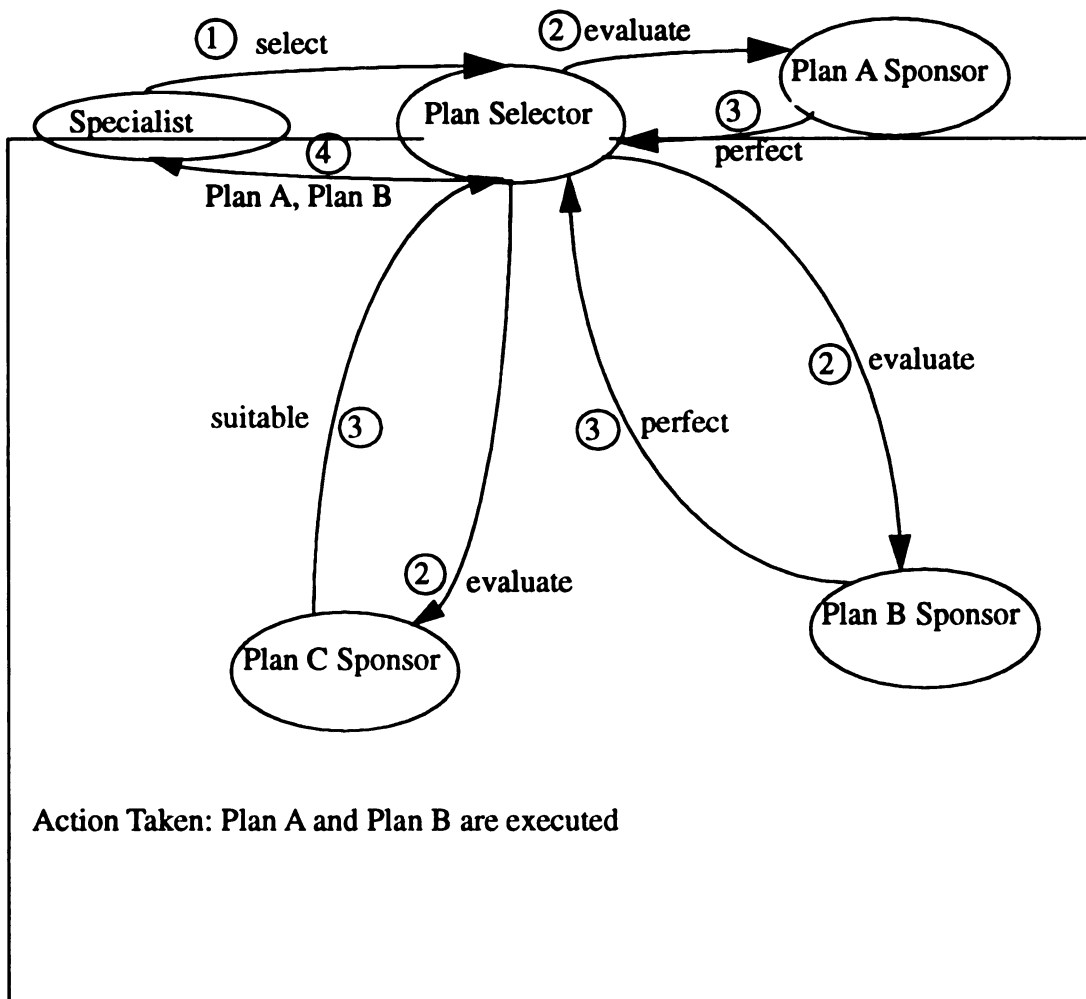


Figure 25: An Example Showing Plan Selection in MDSPL

3. Step execution: Two types of change are involved in the execution of steps:

- a. The purpose of a “step” is to assign a value to a design attribute.

A step is implemented in the form of a pattern matcher (implemented as a decision table). This pattern matcher assesses the design status and assigns a value to the attribute associated with the step. In the single design approach, the patterns are matched in order until a pattern is found (if any) and the value associated with that pattern becomes the value to assign to the attribute. In the Multiple Design approach, all the patterns are searched and the values associated with all the patterns that matched are added to the tree of design results. The patterns which match may be in different branches of the design alternatives tree (or even different trees), and the resulting values, therefore, are each added to its appropriate branch.

- b. After a step is executed, the step’s design limiter is invoked to check the values generated and to examine the possibility of eliminating some of them.

4. Redesign: In the single design approach, only one plan is attempted at a time, and only one value is assigned for each attribute. As a result, when a failure occurs (due to a failing constraint), redesign is necessary to remediate the failure (by changing some aspects of the design).

Redesign involves changing the value of the design attribute(s) that caused the failure of the constraint [Brown & Chandrasekaran, 1989]. Depending on the type of the design attribute involved, one of two cases applies [Brown, 1993]:

- a. The design attribute can take discrete values (as in the case of composite materials where discrete values are selected for the polymer material, the type of fiber, etc.). In this instance, a new value is selected for the design attribute. The new value can either be one of the “normal” values available to the designer, or it can be a “special” value to be used only in cases of failure. Knowledge of these special values is available only for redesign and not for “normal” design.
- b. The design attribute can assume continuous values (such as numeric values involved in mechanical part design). In this case the value causing the failure is “tweaked” by addition or subtraction (depending on the direction of the failure) of small increments. This type of failure results from the use of empirical numeric formulas with arbitrary constants.

In Multiple Design, the need for redesign is eliminated by including redesign knowledge as part of the “normal” design knowledge, and thus no separate redesigners are needed. The two types of design attributes discussed above (discrete-valued attributes and continuous-valued attributes) are treated differently:

- a. For discrete-valued attributes, all the possible values that the attribute can take (whether they are “normal” or “special” values) are enumerated in the decision table for the pattern matcher which is responsible for assigning a value to that attribute. In this case, since all applicable values from a pattern matcher are assigned to a step’s design attribute, should a constraint fail, the

only necessary action is to prune (from the design values tree) the branch (or sub-branch) where the failure condition occurred. No other redesign action is necessary.

- b. For continuous-valued attributes, the equations for assigning values to the design attributes are associated with tolerances. For example instead of saying “outer diameter = inner diameter + 10 mm,” the equation is rewritten in the form: “outer diameter = inner diameter + 10 mm \pm 2 mm” In this instance, the step assigns a range of values (instead of a discrete value) to the attribute. A failing constraint therefore results in a narrowing of range, splitting a range into two or more subranges, or pruning of the entire branch, without requiring further redesign action.

The composite materials design problem used as a test-bed for this work only involves discrete-valued attributes. Consequently, I implemented only the case for discrete-valued attributes.

6.6 Problem Solving in MDSPL

By incorporating the above changes into the original Routine Design algorithm described in Section 3.2, the algorithm for problem solving in MDSPL can be summarized as follows:

1. First the top level specialist is invoked in the appropriate mode (design or rough design).
2. The top level specialist requests its selector to select a set of plans to pursue from among the plans available to that specialist.

3. The selector requests the sponsors of the available plans to assess the applicability of their respective plans.
4. The sponsors examine the current design status and the input data (design requirements) and assign applicability ratings to their plans.
5. The selector examines the applicability ratings and selects a set of plans to pursue.
6. The chosen plans are invoked.
7. Each plan takes one or more of the following actions:
 - a. invoke other specialists in which case these specialists will follow a scenario similar to that followed by the top specialist (steps 2 through 6).
 - b. execute tasks. These tasks consist of steps each of which performs a computation or uses a pattern matcher to assign one or more values to a design attribute.

After the execution of every step and every task, check design limiters if available, and eliminate the current design from further consideration if it satisfies the elimination conditions specified in the design limiter.,

- c. check constraints and if they fail, remove the current design from further consideration.

6.7 The MDSPL Language

The original task-specific architecture for Routine Design has a language associated with it for analyzing and implementing design systems. This language is called DSPL (Design

Specialists and Plans Language). For the Multiple Design approach, I extended the DSPL language to include design limiters as discussed above. The extended language is called MDSPL (Multiple Design Specialists and Plans Language). MDSPL is implemented in a diagrammatic form which provides a system builder with templates to aid in the system building process. Building a problem solver is done using a series of browsers as described below.

6.7.1 The Specialist Hierarchy Browser

A specialist hierarchy browser is a browser for viewing the hierarchy of the design specialists. From the specialist hierarchy browser, the individual specialist browsers may be invoked. Figure 26 illustrates a specialist hierarchy browser with a root specialist; sp1 which has three subspecialists sp2, sp3, sp4. Specialist sp3 has two subspecialists sp5, sp6.

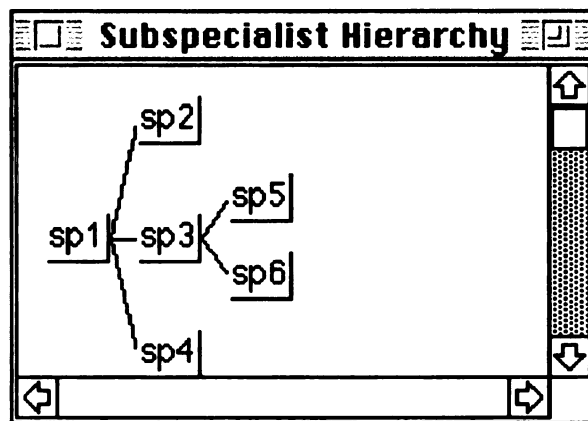


Figure 26: A Subspecialist Hierarchy Browser

6.7.2 The Agents Hierarchy Browser

An agents hierarchy is a browser for viewing the hierarchy of the design agents. This hierarchy can be viewed as a more expanded form of the specialists hierarchy showing not only

the specialists but all of the design agents. From the agents hierarchy browser, the individual agent browsers may be invoked. Figure 27 displays an agents hierarchy browser with a root specialist, sp1 which has one plan, p11. Plan p11 consists of the task tsk1 and the three subspecialists sp2, sp3, sp4. Specialist sp3 has two plans, p13 and p14. Specialist sp3 has two plans, p13 and p14.

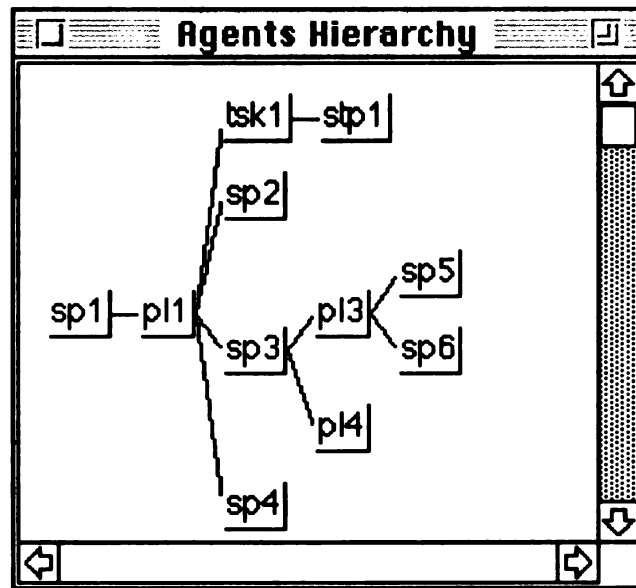


Figure 27: An Agents Hierarchy Browser

6.7.3 The Specialist Browser

The specialist browser is a browser for viewing and editing a design specialist. A specialist browser allows design plans, rough design plans, and initial and final constraints to be added, removed, or have their order changed. The browsers for the plans and the constraints belonging to a specialist can also be invoked from that specialist's browser. Figure 28 shows the browser for the specialist sp3. This specialist has two design plans, no rough design plans and no constraints.

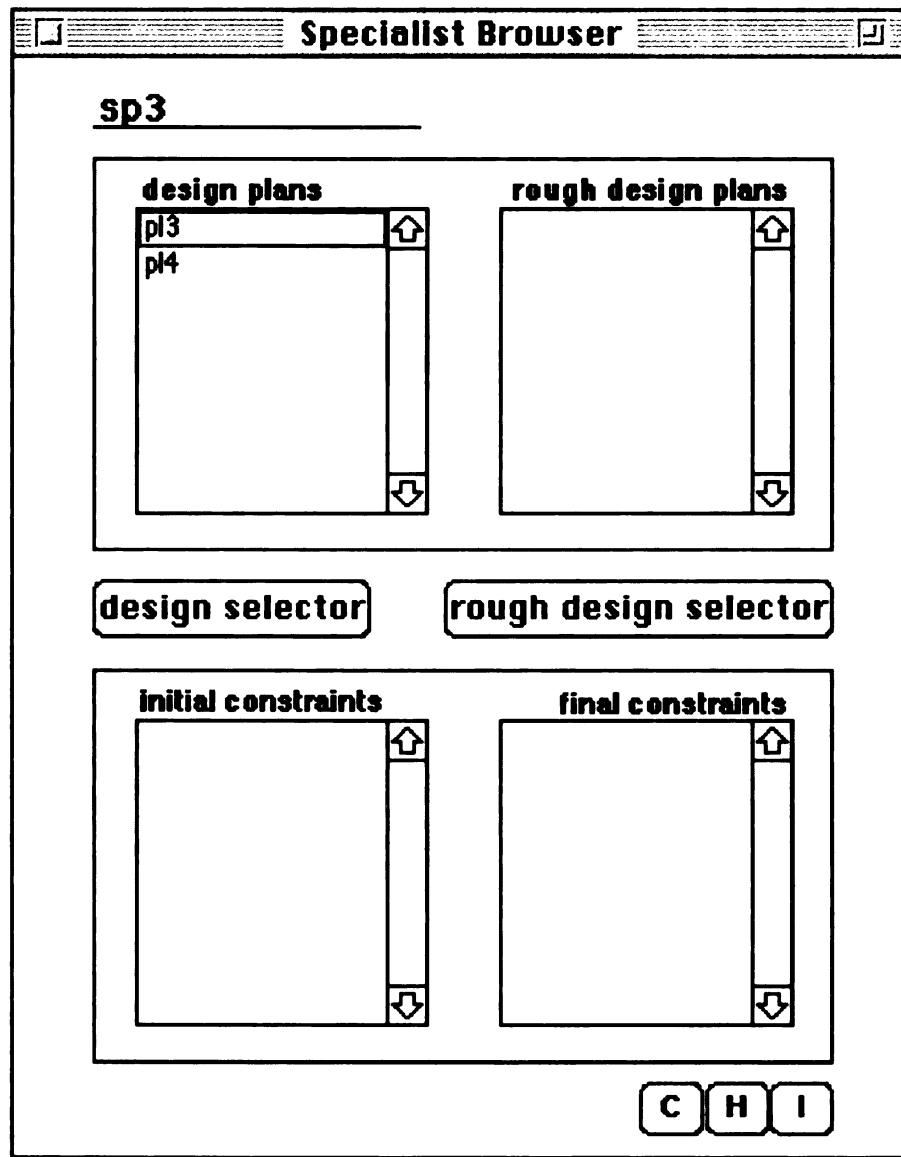


Figure 28: A Specialist Browser

6.7.4 The Plan Browser

The plan browser is used for viewing and editing a plan. A plan browser allows specialists, tasks and constraints to be added, removed, or have their order changed. From the plan browser, subsidiary browsers for specialists, tasks constraints, and the plan's sponsor can be invoked. Figure 29 shows the browser for the plan pl1. This plan consists of the task tsk1, followed by the specialists sp2, sp3, and sp4 in this order.

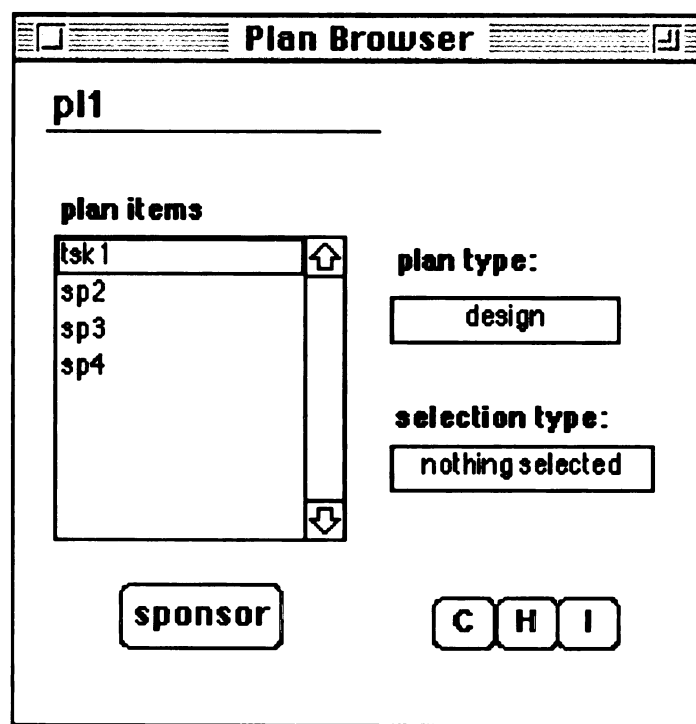


Figure 29: A Plan Browser

6.7.5 The Task Browser

The task browser is a browser for viewing and editing a design task. From a task browser, the steps constituting the task may be added, removed, or have their order changed. The browsers for the steps constituting a task, as well as the browser for the design limiter asso-

ciated with the task may be invoked from the task's browser itself. Figure 30 shows a task browser for the task `tsk1` which has a single step `stp1`.

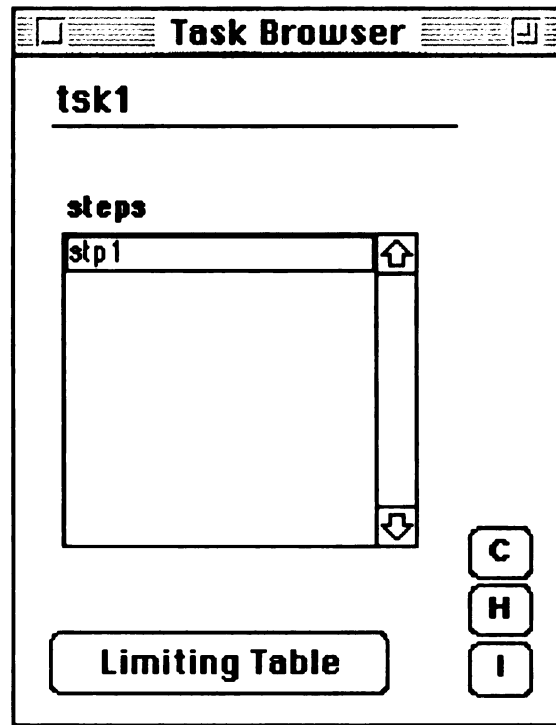


Figure 30: A Task Browser

6.7.6 The Step Browser

The step browser is a browser for viewing and editing a design step. From a step browser, the attribute for which the step is to assign a value is defined. The matching table as well as the design limiter table can be accessed. The constraints list for the step can also be edited in the step's browser. The browsers for individual constraints may also be invoked from the step browser. The browser for step `stp1` is shown in Figure 31.

Figure 31: A Step Browser

6.7.7 The Constraint Browser

A constraint browser allows viewing and editing a constraint by defining the design attributes to be tested and the required relation between them. The failure message to be returned (at execution time) in case of the failure of the constraint is also defined from the constraint browser. Figure 32 shows the browser for the constraint cons1.

6.7.8 The Table Browser

A table browser is used in a variety of places. It is used to view and edit plan sponsors, step pattern matchers, as well as the pattern matchers for the task and step design limiters. Figure 33 represents the matcher for the step stp1.

Constraint Browser

cons1

attribute 1: **define**

operator: **define**

attribute 2: **define**

failure message

Constraint "cons1" failed

C **H** **I**

Figure 32: A Constraint Browser

Table Matcher Interface

rows columns

stp1.MTR

	var1	var2	result	
1	< 5	< 5	2	
2	< 10	< 10	4	
3	?	?	8	

run clear C H I

Figure 33: A Table Matcher

6.7.9 The User Interface

Design systems built using MDSPL employ a friendly user interface to facilitate both the input and the output processes during problem solving. The user interface consists of an input browser and an output browser.

6.7.9.1 The Input Browser

The input browser is displayed to the user at the beginning of the problem solving process. It displays a list of the input parameters and allows the user to input values to as many of these parameters as the user wishes. An example is given in Figure 34. By selecting an

Application Type	'Civil Engineering'	↑
Corrosion	'No'	
Flame Retardance	'No input'	
Humidity	'No'	
Maximum Tensile Modulus	100	
Required Glass Transition Temperature	200	
Required Tensile Modulus	50	
Rigidity	'No input'	
Use Temperature	100	↓

Proceed

Figure 34: An Example MDSPL Input Browser

attribute from the input browser, the user is prompted for a value for this attribute. Figure 35 shows a query to the user about the attribute "Application Type." The system queries the user, at run time, for the values of any parameters for which the user does not specify a

Application Type

for case: case 1

What type of application is this composite going to be used for?

my selection

- Adhesives
- Casting
- Civil Engineering
- Domestic Applical
- Electrical Encapsu
- Lamination
- unknown

documentation cancel accept



Figure 35: An Example User Query

value at the beginning. The user is queried in this manner as values are needed for the problem solving process.

6.7.9.2 The Output Browser

Output browsers are displayed at the end of the design process showing the resulting design(s). Two types of output browsers are provided:

1. Single output browser: This browser is similar to the input browser. It displays a list of the design parameters and the values assigned to them as shown in Figure 36.

Cure Temperature	150	 
Cure Time	60	
Curing/Co-cure Agent	'DDS'	
Fiber Type	'E-Glass'	
Glass Transition Temperature	220	
Polymer Material	'Epoxidezed Phenolic Novolac'	
Post Cure Temperature	220	
Post Cure Time	180	
Tensile Modulus	52	

View Inputs

Proceed

Figure 36: An Example Single Design Output Browser

2. Multiple output hierarchy browser: This browser is in the form of a hierarchy browser. It shows the design parameters with their assigned values arranged in the form of a tree. Any path from a root to a leaf node represents a design. Figure 23 shows an example of a Multiple Design output browser.

In both types of output browsers a limited explanation capability is provided, by allowing the user to browse the matcher that resulted in any particular value of interest (by clicking on that value).

6.8 Comparison of the Computational Expense in DSPL and MDSPL

Knowledge-based systems are usually computationally expensive. However, while being computationally expensive in the worst case, domain knowledge is usually applied to make the “typical case” computationally feasible.

Routine Design is too complex to formulate a precise analysis of the complexity of its problem solving activity. In this section, I will informally compare the computational complexity of the problem solving activity of DSPL and MDSPL both in the worst case and in the average case. Only the components of Multiple Design which are different from the single design approach will be discussed.

6.8.1 Plan Selection

In Multiple Design, plan selectors perform the same function as in the single design approach: that of requesting all its plan sponsors to assess the applicability of their respective plans. In the single design approach, the selector chooses one of these plans, while in Multiple Design, the selector chooses all of these plans.

6.8.2 Step Evaluation

The purpose of a step is to evaluate a pattern matcher, and to assign a value (selected based on the result of the matching process) to a design attribute such as a type of a material. The value assigned depends on which pattern(s) match out of a set of patterns. The value assigned may be a direct value or the result of performing some function on other design attributes. The evaluation of a step thus involves 3 main steps:

1. matching the rows of the pattern matching table, in order, against the data.
2. (optional) computing the value to be assigned to the designed attribute.

While this step has no guarantees in terms of its potential complexity (the step can even be undecidable), in practical applications this is usually in the form of a simple calculation (e.g. fiber ratio = maximum fiber ratio * required tensile modulus / tensile modulus of fiber)

3. storing the selected value in a database.

These steps are the same in both the single design and the Multiple Design approaches. In the single design approach, however, the pattern matching step terminates as soon as a matching pattern is found, while in Multiple Design all patterns are examined for a potential match.

In the single design approach, on average half the patterns of every pattern matching table will be examined, while in the worst case all patterns will be examined. In Multiple Design, all patterns will always be examined. Consequently, the average complexity of the pattern matching process in Multiple Design is double the average complexity of the single design approach, and both are equal in the worst case.

6.8.3 Constraint Evaluation

The constraint evaluation process is very different in Multiple Design compared to the single design approach. That difference is manifest in what it means to evaluate a constraint as well as in what action is to be taken if the constraint fails.

In the single design approach, evaluating a constraint means simply comparing two design attributes and reporting success or failure. In case of a failure, redesign is initiated, which, depending on the nature of the problem, can be expensive computationally. In Multiple Design, evaluating a constraint involves comparing the two values for every path in the design tree on which values for the design attributes exist. Even though this can be computationally expensive, its complexity is reduced by the method of organizing the design data (see Section 6.2). Moreover, the failure of a constraint also has a different meaning in the Multiple Design approach: it does not necessarily imply that the design has failed. A constraint fails for a specific design only (or part of a design), in which case the design is eliminated from the designs tree, potentially saving the complexity involved in the redesign process.

6.8.4 Selecting Multiple Plans versus Failure Handling

In the single design approach, a failing constraint triggers redesign which can be expensive computationally, involving the execution of every plan in the system. In Multiple Design, however, a failing constraint simply involves the removal of the part of the design that failed. On the other hand, in Multiple Design, all the plans with the highest applicability ranking are attempted, whereas in the single design approach, only some of these plans will be attempted (all of the plans will be attempted in the worst case). Consequently, both approaches have the same complexity in the worst case.

6.8.5 Discussion

Multiple Design is more computationally expensive than single design. However, two factors in MDSPL mitigate this:

1. Only the plans which obtain the highest ranking are attempted (unless they all fail). This ensures that only the plans that are most likely to produce the best designs are attempted.
2. Design limiters are used to eliminate partial designs which are known from previous experience to be unsatisfactory (e.g. due to incompatibility with other parts of the designs). Design limiters are also used to eliminate designs which are “too good” since in most design situations significantly exceeding the design requirements considerably increases the cost of producing the resulting designs.

The above two measures combined with diligent use of domain knowledge significantly reduces the computational expense of generating Multiple Designs. While the computational expense of Multiple Design will generally be higher than that of single design, the Multiple Design approach is intended for use with industrial applications where the gener-

ation of Multiple Design alternatives is desirable and thus justifies the extra computational expense.

6.9 Comparison Between MDSPL and Search Techniques

The MDSPL approach may be compared to classic search techniques. However such a comparison compromises the knowledge-level aspects of the MDSPL approach. Expressing problem solvers using a vocabulary that matches that used in the particular domain being addressed aids in the knowledge engineering process and also in the maintenance of the system after implementing it.

As an example, the specialist hierarchy of an MDSPL system can be viewed as an AND/OR graph. From a simple viewpoint, the nodes in a specialist hierarchy (such as that of Figure 26) can be viewed as nodes where at some branching points, all nodes have to be evaluated (AND node) and at other branching points, nodes are alternatives and only one of which has to be evaluated (OR nodes). However, the vocabulary used by the MDSPL approach allows a deeper understanding of the nature of the problem being addressed through the understanding of the different specialists as specialists performing a specific function as part of the design process. This understanding allows the use of a vocabulary similar to that used in the particular domain being addressed. Furthermore, this understanding simplifies the knowledge maintenance process once an initial version has been completed. By assigning well defined functions to the different specialists, the specialists can be updated or replaced without needing to interfere with the other specialists. Characterizing the problem as an AND/OR graph also obscures the detail of the activity involved in problem solving by hiding the flexibility offered by an MDSPL problem solver. For example, in an MDSPL hierarchy, a specialist may be used for solving a particular design instance whereas the same specialist may not be used under different circumstances. More-

over, a particular node may be viewed as an AND node in one problem solving situation while the same node may be viewed as an OR node in another situation.

From a problem solving activity viewpoint, we can characterize the problem solving activity in MDSPL as best-first problem solving. However, this interpretation cannot be directly derived by a simple inspection of the specialist hierarchy since the “pieces” of any particular design are distributed over the entire hierarchy. However, at any given point during problem solving, specialists have to make choices among alternative plans. The selection among the different plans is based on a selector that utilizes sponsors associated with the different plans available to that specialist. The evaluation knowledge in the sponsors is typically in the form of heuristic knowledge—based on previous experience—that evaluates the potential of the plan to apply to the current situation (input data and previous decisions). The most promising plans are considered first. To directly represent the knowledge as a best-first hierarchy, an MDSPL hierarchy has to be restructured so that at any given “node”, branching implies different designs, and “pieces” of any design are on a single path from the root of the hierarchy to a terminal node. The resulting hierarchy would be similar to the solutions’ tree that results from using an MDSPL problem solver (such as the hierarchy of Figure 23). This type of implementation, while it captures the necessary knowledge to generate the needed designs, sacrifices the use of the domain vocabulary necessary for knowledge engineering and maintenance.

Chapter 7

A Polymer Composite Materials Design Example

In this chapter, I will discuss a prototype routine design system I implemented for automating the material design of fiber-reinforced thin film polymer composite materials. I originally implemented this design system using the single-design Routine Design approach. Later, I re-implemented this system in Multiple Design. I will illustrate by example the differences of Multiple Design compared to standard Routine Design.

7.1 Problem Description

The area of polymer composite materials can be viewed as a modern analog of the area of metallurgy. Like metallurgy, the ultimate goal in composites is to understand the properties of physical materials and the methods of material fabrication which enable the creation of materials with desired properties. Metallurgy deals with properties and fabrication of materials made from metals; polymer composites is concerned with properties and fabrication of materials created from the realms of polymer science and chemical engineering. The area is distinctly multidisciplinary. One difference between metallurgy and composite materials in general is the basis for the term “composites.” In metallurgy, metals are either reacted chemically to form a new compound (as in steel) or commingled at the molecular level as in metallic alloy formation (such as tin). In composite materials, the processed material typically will retain macroscopic areas resembling the starting materials. For example, in epoxy-resin carbon fiber composite materials, the carbon reinforcing fiber retains much of its individual identity after processing is completed.

A typical chronology for designing a composite material follows [Venkatasubramanian, Lee, & Gryte, 1987]. First, macroscopic properties which are desired in the completed composite (such as final material tensile modulus, resistance to acids and alkalis, electrical

resistance) are defined. Based on these desired properties, the composite designer proposes an initial plan for the production of the composite. This plan includes both an ingredients list for all materials to be used, and a preliminary processing protocol¹ describing how the initial mixture is to be processed. Next, the composite designer estimates how well the proposed composite design meets the initially stated, desired properties. This estimate is typically carried out by actually producing samples of the composite, then performing laboratory testing to determine the values for the properties of interest. Ultimately, a proposed composite design will result in an actual material which can be subjected to laboratory testing.

One goal of composite researchers is to furnish better models for proposed designs in order to limit the number of candidate materials which must actually be fabricated for testing. Following one round of design proposing, and matching to specifications, successive rounds of redesign are usually required before convergence of proposed composite properties to desired properties takes place.

Composite materials can be grouped into three major classes: polymer matrix², ceramic matrix, and metal matrix composites. In each of these types of materials, the matrix material is typically combined with a reinforcement such as a fiber, or particulate, in such a way as to achieve specified properties. The problem domain described here belongs to the first class, composites fabricated using polymer matrices. Polymer composites can be further classified according to the type of polymer used, either thermoset or thermoplastic, which largely determines the conditions employed in the processing of the material. The example employed in this work focuses on thermoset composites with fiber reinforcement.

-
1. A processing protocol is typically expressed as a series of applications of temperature and pressure for specific periods of time.
 2. A matrix is the material that forms the bulk of a composite material which holds the reinforcing fiber in place.

7.2 Routine Design System for Material Design of Epoxy Resins

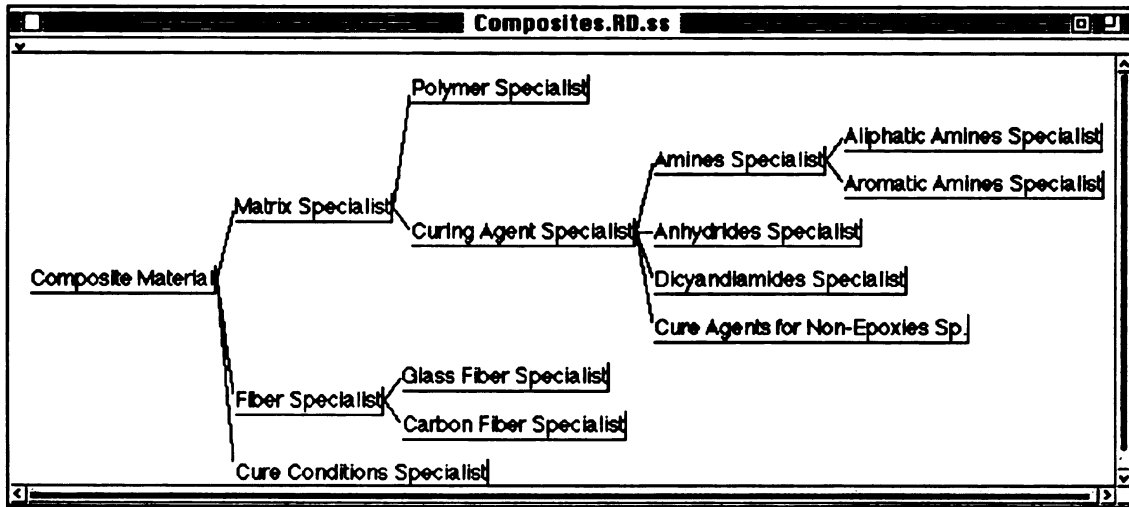


Figure 37: The Specialist Hierarchy for The Composite Materials Design System

Figure 37 includes the specialist hierarchy of the prototype design system for fiber reinforced thermoset composite materials. The top-level specialist, Composite Material, has one plan which first calls on the Matrix specialist to select a suitable matrix to achieve the required properties. It then calls on the Fiber specialist to select an appropriate fiber. Eventually, it calls on Cure Conditions to design the appropriate cure conditions given the chosen materials.

The Matrix specialist also has one plan. This plan first calls on the Polymer specialist to select an appropriate type of polymer matrix material. The Matrix specialist then calls on the Curing Agent specialist to select a suitable curing agent.

The Curing Agent specialist has four alternative plans from which to choose. Based on the type of polymer chosen and the properties required from the composite material being

designed (mainly the required usage and the thermal properties represented in the glass transition temperature) this specialist chooses one of its four plans.

The Fiber specialist similarly has two design plans to choose from. This choice is made based on the required tensile properties represented in the tensile modulus of the required product. These two plans in turn select either the Carbon Fiber specialist or the Glass Fiber specialist. These specialists are responsible for selecting the appropriate type of carbon, or glass respectively. Finally, the Cure Conditions specialist has one plan that is responsible for setting the appropriate cure conditions (application of temperature, pressure, and time) based on the selected materials.

7.3 A Sample Single Design

Consider a design in which we desire a composite material that has a glass transition temperature of at least 200°C and a tensile modulus of 50 GPa, which will be used in an outdoor structure application. The input specification screen for this example is shown in Figure 38.

The system starts by calling on the top-level specialist, CompositeMaterial. This specialist calls on the Matrix specialist, which in turn calls on the Polymer specialist which makes the decision to use the epoxies plan. This plan chooses DGEBA as the type of polymer to be used. The Matrix specialist then calls on the Curing Agent specialist. Curing Agent has embedded knowledge that amines are a preferable choice for civil engineering applications, and hence will make the appropriate selections. The Amines specialist now examines the requirements and finds that aromatic amines are more suited for civil engineering applications. Now the Aromatic Amines specialist is called upon and it uses the required glass transition temperature to select diaminodiphenylsulfone (DDS) which has a glass transition temperature of 220°C.

Application Type	'Civil Engine'
Corrosion	'No'
Flame Retardance	'No input'
Humidity	'No'
Maximum Tensile Modulus	800
Required Glass Transition Temperature	200
Required Tensile Modulus	50
Rigidity	'No input'
Use Temperature	100

Proceed



Figure 38: Input Specification Screen

The CompositeMaterial specialist then calls on the Fiber specialist to select the most appropriate fiber. This specialist examines the required tensile modulus and, because it is not a high value, it selects glass fiber as appropriate. Next, the GlassFiber specialist chooses E-glass as the type of fiber to be used because it has a tensile modulus of 52 GPa.

The CompositeMaterial specialist will then call on the CureConditions specialist which inspects the materials selected to this point, and selects a cure cycle of 1 hour at 150°C followed by 3 hours at 220°C at atmospheric pressure. The output of the system is shown in Figure 39.

7.4 A Sample Multiple Design

The above system for the design of fiber reinforced thermoset composite materials was originally developed using the traditional single design DSPL approach. Later, the same system was re-implemented under the new MDSPL Multiple Design environment.

Cure Temperature	150	 
Cure Time	60	
Curing/Co-cure Agent	'DDS'	
Fiber Type	'E-Glass'	
Glass Transition Temperature	220	
Polymer Material	'Epoxidezed Phenolic Novolac'	
Post Cure Temperature	220	
Post Cure Time	180	
Tensile Modulus	52	

View Inputs

Proceed

Figure 39: Single Design Output Browser

By running the same design example described in Section 7.3, with the same input requirements shown in Figure 38, the system performs as follows:

The system starts by calling on the top-level specialist, CompositeMaterial. This specialist calls on the Matrix specialist, which in turn calls on the Polymer specialist which makes the decision to use the epoxies plan. This plan chooses DGEBA, Epoxidized Phenolic Novolac, and Hydroxyphenyl as viable types of polymer to be used. The Matrix specialist then calls on the CuringAgent specialist. CuringAgent has embedded knowledge that amines are a preferable choice for civil engineering applications, and hence will make the appropriate selections. The Amines specialist now examines the requirements and finds that aromatic amines are more suited for civil engineering applications. Now the Aromatic Amines specialist is called upon and uses the required glass transition temperature to select diaminodiphenylsulfone (DDS) which has a glass transition temperature of 220°C.

The Composite Material specialist then calls on the Fiber specialist to select the most appropriate fiber. This specialist examines the required tensile modulus and because it is

not a high value, it selects both glass fiber and carbon fiber as appropriate. Next, Glass Fiber chooses E-glass, and S-glass as possible types of fiber to be used because they have tensile moduli of 52 GPa, and 61 GPa respectively (the input specifies a requirement of at least 50). The Carbon Fiber Specialist also chooses both AS4-Carbon and Graphite as possible fiber choices.

The Composite Material specialist will then call on the Cure Conditions specialist which inspects the materials selected to this point, and since the cure cycle depends on the type of curing agent chosen, it selects a cure cycle of 1 hour at 150°C followed by 3 hours at 220°C at atmospheric pressure. The output of the system is represented in the form of a tree with every path from the root of the tree to a leaf node representing a design alternative. The output from running the above example is shown in 2 parts in Figure 40 and Figure 41. An alternate form for viewing the output of the design process is in the form of a series of lists of design attributes and their assigned values similar to the output of the single design system shown in Figure 39.

The output from the Multiple Design system includes the same output as the single design system, in addition to eleven other designs, namely the remaining combinations of the four selected choices for fiber material and the three selected choices for the polymer material. It would have been possible to generate these other designs using the single design system. However, this would require substantial manipulation of the domain knowledge embedded in the system, and a substantial amount of time, and would not be feasible without the assistance of an expert.

As an example, the choice of polymer material is based on the pattern matcher shown in Figure 42. The first four columns of the table are patterns to match against the design input variables at the top of the columns, and the fifth column of the table represents the value to be used for the polymer material if the corresponding pattern matches. The input data matches the second, third, and fourth rows of the table, and thus the values from all three

Composites.08.mdr

Cure Temperature = 150 | Cure Time = 60 | Curing/Co-cure Agent = 'DDS'

Fiber Type = 'AS4-Carbon' | Glass Transition Temperature = 220 | Polymer Material =

Fiber Type = 'Graphite' | Glass Transition Temperature = 220 | Polymer Material =

Fiber Type = 'S-Glass' | Glass Transition Temperature = 220 | Polymer Material =

Fiber Type = 'E-Glass' | Glass Transition Temperature = 220 | Polymer Material =

Figure 40: Part 1 of the Multiple Design Output Browser

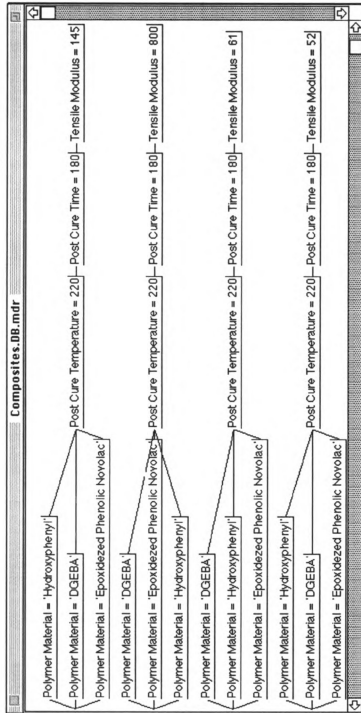


Figure 41: Part 2 of the Multiple Design Output Browser

[illegible]

Figure 42: Table Editor Showing the Pattern Matcher for Selecting an Epoxy

rows are chosen in the Multiple Design. In the single design case, only the first pattern that matches is to be considered. To force the single design system to produce a different design, the ordering of the rows of the table need to be changed.

The choice of the fiber material is even more complex when the single design approach is used. The choice is distributed at four different places, two pattern matchers for the sponsors of the glass fiber and the carbon fiber, and one pattern matcher within each of these two plans for selecting the particular type of fiber. The first two of these pattern matchers decide which of the plans to use, and the other two decide which particular type of fiber to use after a plan is already selected. Figure 43 shows the pattern matcher for the Glass Fiber

Table Matcher Interface

rows columns

Glass Fiber plan.spons

	Application Type	Application Type	required Tensile Mod	result	
1	~= Adhesives	~= Electrical Encap	<= 61	perfect	↑
2	?	?	?	rule-out	
					↓

↩ ↲

run **clear** **C** **H** **I**

Figure 43: Table Editor Showing the Pattern Matcher for the Glass Fiber Plan

plan sponsor, and Figure 44 shows the pattern matcher for the Carbon Fiber plan sponsor. Here, again, the patterns in both tables that make their plans perfect are met by the input data, so both plans are pursued for Multiple Designs. In the single design approach, the

Table Matcher Interface

rows columns

Carbon Fiber plan.spon

	Application Type	Application Type	required Tensile Mod	result
1	~= Adhesives	~= Electrical Encap	<= 800	perfect
2	?	?	?	rule-out

run clear C H I

Figure 44: Table Editor Showing the Pattern Matcher for the Carbon Fiber Matcher Plan

Glass Fiber plan gets examined first, and thus the Carbon Fiber plan never gets examined (unless the GlassFiber plan fails at a later stage). To force the system to examine the Carbon Fiber plan, the order of the two plans in the Fiber specialist has to be reversed.

7.5 Discussion

The above examples demonstrate the use of MDSPL for the generation of Multiple Design alternatives meeting the same set of specifications.

If we examine the set of outputs produced in Figure 40 and Figure 41, we find that one of the values for the fiber parameter is Graphite, resulting in a tensile modulus of 800 GPa, much higher than the required tensile modulus of 50. This typically means a correspondingly more expensive material. Design limiters are included in the MDSPL system for

these cases; namely designers' knowledge of design values that are not appropriate because they are "too good."

I added the design limiter shown in Figure 45 to the Carbon Fiber selection task to elim-

Table Matcher Interface

rows columns

Carbon Fiber Specifica

	Maximum Tensile Modulus	Fiber Type	result
1	< 145	= AS4-Carbon	exclude
2	< 800	= Graphite	exclude

run clear C H I

Figure 45: A Design Limiter

inate unneeded fiber types. Each row in this table is interpreted to mean that any design meeting the pattern in the first two columns is to be "excluded." By rerunning the same design example with the design limiter, and setting the maximum tensile modulus to 150 GPa, the resulting set of designs is the same as in the case of Figure 40 and Figure 41, except that the Graphite Fiber is excluded. The output of this design is shown in Figure 46 and Figure 47. Further restricting the maximum tensile modulus to 100 GPa (twice the required tensile modulus), causes the AS-4 Carbon to also be excluded from the set of possible designs. This case is shown in Figure 48 and Figure 49.

Composites.DB.mdr	
Cure Temperature = 150	Cure Time = 60
Curing/Co-cure Agent = 'DDS'	
Fiber Type = 'AS4-Carbon'	Glass Transition Temperature = 220
Fiber Type = 'S-Glass'	Glass Transition Temperature = 220
Fiber Type = 'E-Glass'	Glass Transition Temperature = 220
Polymer Material =	Polymer Material =
Polymer Material =	Polymer Material =
Polymer Material =	Polymer Material =
Polymer Material =	Polymer Material =
Polymer Material =	Polymer Material =
Polymer Material =	Polymer Material =
Polymer Material =	Polymer Material =
Polymer Material =	Polymer Material =

Figure 46: Part 1 of the Output of Example 2

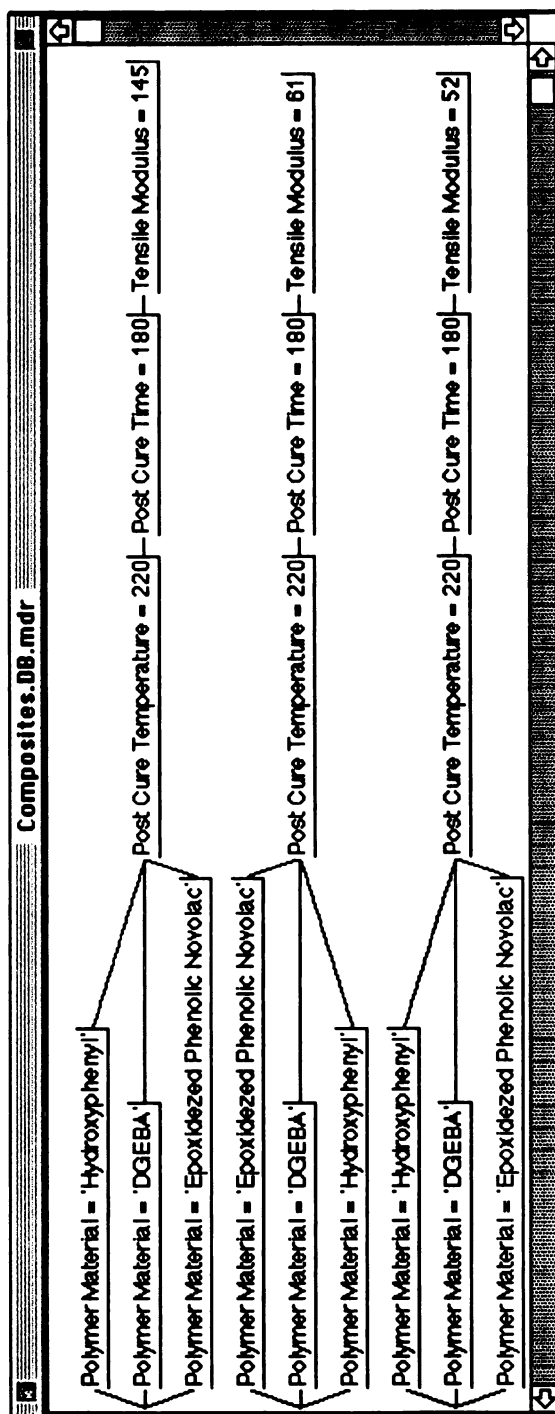


Figure 47: Part 2 of the Output of Example 2

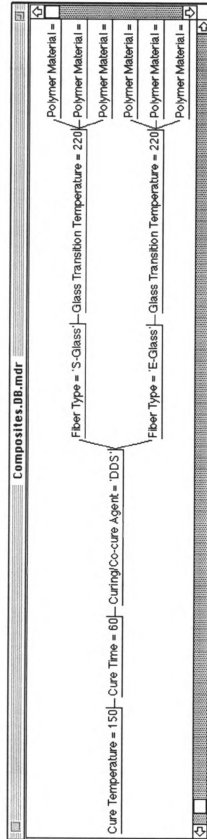


Figure 48: Part 1 of the Output of Example 3

Composites.DB.mdr				
Polymer Material = 'DGEBA'	Post Cure Temperature = 220		Post Cure Time = 180	Tensile Modulus = 61
Polymer Material = 'Hydroxyphenyl'				
Polymer Material = 'Epoxidized Phenolic Novolac'				
Polymer Material = 'Hydroxyphenyl'	Post Cure Temperature = 220		Post Cure Time = 180	Tensile Modulus = 52
Polymer Material = 'DGEBA'				
Polymer Material = 'Epoxidized Phenolic Novolac'				

Figure 49: Part 2 of the Output of Example 3

From the above discussion, we can see that the MDSPL system can also be used to examine the effect of changing constraints on the set of generated design alternatives, a feature which is desirable by industrial designers.

7.6 Analysis of the Composite Materials MDSPL System

In this section, I present an approximate analysis for the developed MDSPL system for the design of thermoset polymer composite materials. From this analysis, I will present an extrapolation for an extended system that includes other types of composite materials, as well as a wider coverage of polymer composites. The results show that the time consumed by the design process is linearly proportional to the number of polymers and the number of fibers available to the system.

Before attempting a quantitative analysis of the MDSPL system, I will first give a descriptive account of the activities involved in generating designs using this MDSPL system. The specialist hierarchy for the MDSPL polymer composite materials design system is given in Figure 37. The following discussion refers to specialists from this hierarchy:

1. The first step in the design process is for the top specialist “Composite Materials” to invoke its only design plan.
2. This design plan is composed of invocations of three sub-specialists; “Matrix Specialist”, “Fiber Specialist”, and “Cure Conditions Specialist”.
3. The matrix specialist invokes its only plan. This plan consists of invoking two sub-specialists; “Polymer Specialist”, and “Curing Agent Specialist”.
 - a. The polymer specialist has three plans to choose from; “Epoxy Design Plan”, “Polyesters Design Plan”, and “Polyimides Design Plan”. These plans correspond to selecting to use a polymer of type epoxy, polyester, or polyimide respectively.

According to the input data, the system may invoke one or more of these plans. Each of these plans consists of one task of one step. The step is in the form of a pattern matcher that examines the input data and selects one or more materials to be used as the polymer material.

- b. The curing agent specialist has four plans to choose from; “Amines plan”, “Anhydrides plan”, “Dicyandiamides plan”, and “Curing agents for non-epoxies plan”. The first three of these plans correspond to types of curing agents that can be used with epoxies, and the last type is for curing additives that are used with polyesters and polyimides.

Each of the Anhydrides, Dicyandiamides, and Curing agents for non-epoxies plans consists of an invocation of a corresponding specialist. The specialist invokes plan which consists of a task of one step. The step is in the form of a decision table that assigns one or more materials to be used as the curing agent.

The Amines plan also invokes a specialist; the “Amines specialist”, which has two plans to choose from. These two plans correspond to a type of amines; aliphatic amines and aromatic amines. Each of these two plans invokes a specialist that has one plan. Each of these plans has a task of one step that selects one or more amines to be used as the curing agent.

- 4. The fiber specialist has three plans to choose from. Two of these plans correspond to the two most commonly used types of fibers; glass fibers and carbon fibers. Each of these plans invokes a specialist that has one plan. Each

of these plans has a task with one step that selects that type of fiber to be used. The third plan is for situations where no fiber is needed. This plan has one task of one step that assigns the value “none” to the fiber to be used.

5. The cure conditions specialist has one plan of two tasks; “cure parameters task” and “post cure parameters task”. The “cure parameters task” has two steps, one for setting the cure temperature and one for setting the cure time. Similarly, the “post cure parameters task” has two steps, one for the post cure temperature and one for the post cure time.

7.6.1 Assumptions

There are two types of activities involved in MDSPL problem solving:

- Invoking an agent (specialist, plan, task, step, plan selector, plan sponsor).
- Pattern matching for decision making. Two types of decisions are made based on a pattern matcher:
 - Plan selection: a pattern matcher is used in the plan sponsors to determine the applicability of the plan to the current situation. In every specialist, there is a plan selector that invokes the sponsors for the plans of that specialist. The sponsors respond with the applicability rating of their plans. The selector then selects the plan(s) that achieved the highest applicability rating.
 - Step evaluation: A step is responsible for assigning a value for a single design attribute. For every step a pattern matcher is used to match certain conditions against the input data and/or previous design decisions and select a value to assign to the step’s design attribute.

The time for agent invocation is much smaller than the time required for pattern matching. For the purpose of this analysis, a simplifying assumption will be made and the time for agent invocation will not be considered.

Another simplifying assumption is to assume that pattern matchers for all plan selectors use the same amount of time, and that all pattern matchers for steps use the same amount of time. Pattern matchers are in the form of decision tables where the width of the table is determined by the number of factors considered in making the decision and the length of the table is determined by the number of alternatives being considered. Note that in single design the alternatives are considered in order until an applicable alternative is found, whereas in multiple design all the alternatives are considered for potential applicability. The number of alternatives thus defines the size of the problem. The constant time for processing a pattern matcher of either type can be assumed to be the average time required to process a pattern matcher of that type.

The plan sponsors assign applicability ratings from four categories:

1. Perfect
2. Suitable
3. Neutral
4. Not suitable

Plans from the highest available category are considered before the lower categories. That is “suitable” plans are not considered unless there are no “perfect” plans applicable or all of the available ones fail. Plans in the “not suitable” category are never considered. For the purpose of this analysis, I will assume all plans from the “suitable” and “neutral” categories to be “perfect”. This is equivalent (time wise) to the worst case of all “perfect” and “suitable” plans failing and thus attempting “neutral” plans.

7.6.2 Definitions

To enable the analysis of the MDSPL system, I define the following terms:

T = total time for design generation

tMatrix = time for generating a design for the matrix

tFiber = time for generating a design for the fiber

tCure = time for generating the design for the cure conditions

tPolymer = time for generating the design for the polymer material

tCuringAgent = time for generating the design for the curing agent

tEvaluation = time for a sponsor to evaluate the applicability of its plan

tStep = time to run the pattern matcher for a step.

nSponsors = the total number of non-empty sponsors (that is the sponsors that use a pattern matcher and not just a default value as in the case of sponsors for the only plan of any given specialist).

nSteps = the total number of steps in the system

avRSp = average number of rows in sponsor tables

avCSp = average number of columns in sponsor tables

avRST = average number of rows in step tables

avCSt = average number of columns in step tables

7.6.3 Quantitative Analysis

Given the above description of problem solving in MDSPL and the simplifying assumption, we derive the following formula for the time necessary to run the composite materials MDSPL designer in terms of the times for the different pattern matchers involved:

$$T = t_{\text{Matrix}} + t_{\text{Fiber}} + t_{\text{Cure}}$$

Using step 3 of the algorithm described above:

$$T = t_{\text{Polymer}} + t_{\text{CuringAgent}} + t_{\text{Fiber}} + t_{\text{Cure}}$$

The two time values “tPolymer” and “tCuring Agent” are not independent since the choice of curing agents depends on the choice of the polymer material. There are three plans for the polymer material, one for epoxies, one for polyesters and one for polyimides. These plans are similar, however, their preconditions are mutually exclusive and hence only one of them can be selected. The selection process involves the invocation of the sponsors for the three plans, and then running the plan involves one step pattern matcher. The above equation thus becomes:

$$T = 3 * t_{\text{Evaluation}} + t_{\text{Step}} + t_{\text{Curing Agent}} + t_{\text{Fiber}} + t_{\text{Cure}}$$

The curing agent selection involves selecting among four different plans. To determine the worst case, take only the case where the polymer chosen was an epoxy since this leads to three possible plan choices, whereas only one plan is possible for polyesters and polyimides. The conditions of the amines plan is mutually exclusive with the superset of the conditions of the anhydrides and the dicyandiamides plans and hence they cannot be invoked at the same time. Additionally, the conditions for the aliphatic amines plan and the aromatic amines plan are mutually exclusive and hence only one can be invoked at any time. The worst case thus becomes one of two cases:

1. the case of invoking both the anhydrides and the dicyandiamides plans at the same time. In this case the time for selecting the curing agent becomes:

$$4*tEvaluation + 2*tStep$$

2. the case of invoking the amines plan. In this case the time for selecting the curing agent becomes:

$$4*tEvaluation + 2*tEvaluation + tStep$$

Assuming that $tEvaluation$ is much larger than $tStep$, the first equation represents the worst case. The overall equation then becomes:

$$T = 7*tEvaluation + 3*tStep + tFiber + tCure$$

For the fiber selection, a choice has to be made among three plans. The conditions for no fiber are mutually exclusive with the superset of the conditions for glass fiber and carbon fiber. The worst case thus becomes choosing both glass fiber and carbon fiber. The above equation thus becomes:

$$T = 7*tEvaluation + 3*tStep + 3*tEvaluation + 2*tStep + tCure$$

Or:

$$T = 10*tEvaluation + 5*tStep + tCure$$

The setting of the cure conditions is done in one plan with two tasks, each of two steps. The equation thus becomes:

$$T = 10*tEvaluation + 5*tStep + 4*tStep$$

Or:

$$T = 10*tEvaluation + 9*tStep$$

The above analysis assumes worst case conditions. Through utilizing domain knowledge, the average case will generally exhibit a much better performance.

In the above worst case, we find that in the implemented system, 10 sponsor evaluations of plan applicability are needed and 9 step evaluation matchers are invoked. However, by examining the implemented system, we find that the system employs 14 plan sponsors and 16 steps. Using these values, and assuming the same ratios to hold for a larger system the evaluation time for the worst case then becomes:

$$T(\text{worst case}) = 0.71 * n\text{Sponsors} * t\text{Evaluation} + 0.56 * n\text{Steps} * t\text{Step}$$

In the above discussion, $t\text{Evaluation}$ and $t\text{Step}$ were assumed constant. This assumption is based on assuming the average value for each of the two time values. This time is proportional to the average number of columns of the tables and to the average number of rows of the tables. The number of columns of a table is the number of factors considered in the decision making process + one for the result in the case of steps (for steps, the result is a value stored in the database, whereas for sponsors, the result is an applicability rating returned to the plan selector). The average time for evaluating a cell in the decision table can be assumed constant for both sponsors and steps. In the case of sponsors, the rows of the table are considered in order until a match is found. The average number of rows considered is thus the average number of rows in the table. The average number of rows examined can be used in the computation of the time in the worst case due to the relatively large number of tables involved. In typical situations, the average number of rows examined will be less than half the number of rows due to the use of domain knowledge to order the rows of the table such that the more likely patterns are examined first. The worst case time thus becomes:

$$T(\text{worst case}) = t\text{Cell} * (0.35 * n\text{Sponsors} * avR\text{Sp} * avC\text{Sp} + 0.56 * n\text{Steps} * avR\text{St} * avC\text{St})$$

For the implemented system, $avRSp=2.5$, $avCSp=2$, $avRSt=4$, $avCSt=3$, $nSponsors=14$, $nSteps=16$.

7.6.4 Expectations for Future Extensions to the Composites Design System

The current thermoset polymer composite materials design system is a prototype system that serves as a basis for future extensions to include additional materials as well as other types of polymer composite materials. Current efforts are being exerted towards extending the system. Three types of extensions are being considered:

1. Extending the current design system to include a wider coverage of thermoset polymer composite materials. The current system includes nine types of thermoset matrix materials. To cover the area of thermoset composites, four more matrix materials are expected to be included.
2. Four different types of fibers are currently being used in the system, six more fibers are expected to be included in the final system.
3. Extending the current system to include thermoplastic polymers. The number of thermoplastic materials expected to be included in the system is between five and ten.

The first type of extensions are extensions involving adding more materials to the current system in its current structure. The second type of extensions involves adding an additional type of fibers (synthetic fibers) as well as adding additional fibers from the existing types (carbon and glass fibers). The third type of extensions involves adding an additional plan for thermoplastic matrices at the same level as the plan for thermoset matrices. This involves additional sponsors and steps to incorporate the knowledge of the additional materials. These additional sponsors and steps are expected to be similar in form to those for thermoset materials.

7.6.5 Expected Impact of Future Extensions on System Performance

A formula for the worst case performance of the composite materials designer was derived in Section 7.6.3. This formula is repeated here for easy reference:

$$T(\text{worst case}) = t_{\text{Cell}} * (0.35 * n_{\text{Sponsors}} * avR_{\text{Sp}} * avC_{\text{Sp}} + 0.56 * n_{\text{Steps}} * avR_{\text{St}} * avC_{\text{St}})$$

The values “avCSP” and “AvRSt” represent the average number of parameters based on which the decisions are made at the sponsors level and the steps level respectively. In the extended system, these values are not expected to change from their current values.

The first type of changes of Section 7.6.4 involves a 60% increase in the number of matrix materials. These new materials are of the same types as the current matrix materials, and thus the effect of adding them is an increase in the number of average rows of the steps' decision tables. The matrix materials affect several other decisions in the design process. By examining the current system, we find that a 60% increase in the number of matrix materials results in a 60% increase in the size of 12 step tables out of the 16 tables in the current system used in the derivation of the above equation. Thus, this increase in the number of matrix materials is expected to result in a 45% increase in the average number of rows of step decision tables (avRSt).

The second type of changes of Section 7.6.4 involves a 150% increase in the number of fibers in the system. This involves two types of changes:

1. An additional type of fibers is introduced (synthetic fibers). This results in adding an extra plan with a sponsor to evaluate the applicability of this plan, and an extra step to select among the different fibers of the additional type.
2. Additional fibers of the existing types are to be added. This results in an increase (about 50 - 100%) in the number of rows of two of the current step tables. Assuming 100% increase, this results in a 12% increase in the average number of rows of step tables.

The third type of changes of Section 7.6.4 involves a an additional type of matrix materials (thermoplastic materials). Nine sponsor tables pertain to thermoset matrices. Assume a similar number of sponsors will be used for thermoplastic matrices. In addition to the increase in the number of sponsors an increase will also be noticed in the number of steps to select among the newly added materials. Twelve steps are involved in the selection of thermoset matrices and their related parameters. Assume a similar number of steps would be involved in thermoplastic matrices. No noticeable change in the average numbers of rows of decision tables should be encountered.

Combining the above values, the values for the current system and the expected system are summarized in Table 1.

Parameter	Current System	Future System
nSponsors	14	24
nSteps	16	29
avRSp	2.5	2.5
avCSp	2	2
avRST	4	6.5
avCSt	3	3

TABLE 1. Values for Current and Future Design System

In the current design system, the average time for the generation of designs using a Macintosh Quadra 700 computer is 16 seconds, with the maximum time being 20 seconds. By substituting these values in the worst case evaluation time equation, the value t_{Cell} is thus 150 milliseconds and thus the expected average time for generating designs using the extended system is expected to be no more than 113 seconds or less than 2 minutes.

From the above analysis, if we closely examine the values, we find that an increase in the number of available alternatives (for either matrix materials or fibers) always results in

an equal increase in either the average number of rows of step tables, or in the number of plans and steps.

The above extensions cover most of the known composite materials. Other matrix materials also exist but are used in very few specialized applications. Adding these matrices to the system would at most double the number of matrix materials used in the system, thus resulting in doubling (in the worst case) the running time of the system to 4 minutes. These results are summarized in Table 2.

System Coverage	Estimated running time
Current System (9 polymers, 4 fibers)	20 seconds
Expected Extensions (23 polymers, 10 fibers)	2 minutes
(46 polymers, 10 fibers)	4 minutes

TABLE 2. Estimated Running Times for the Composite Materials MDSPL System

7.7 Generalized Performance Analysis of MDSPL

From Section 7.6.3, we concluded that the worst case performance of the MDSPL composite materials design system is determined by the following equation:

$$T(\text{worst case}) = t_{\text{Cell}} * (0.35 * n_{\text{Sponsors}} * avRSp * avCSp + 0.56 * n_{\text{Steps}} * avRSt * avCSt)$$

The parameters of the above equation are of a general nature. The constants of the equation, however, are dependent on the specific problem solver being addressed. The constants represent the number of pattern matcher cells tested in the worst case as a fraction of the total number of pattern matcher cells in the system. The two constants represent this ratio

for sponsors and steps respectively. In a more general case, the upper bound on these constants is unity. The above equation thus becomes:

$$T(\text{worst general-case}) = t_{\text{Cell}} * (n_{\text{Sponsors}} * avR_{\text{Sp}} * avC_{\text{Sp}} + n_{\text{Steps}} * avR_{\text{St}} * avC_{\text{St}})$$

The parameter “nSponsors” represents the number of sponsors in the system, which is equivalent to the number of plans in the system whose specialists have more than one plan to choose from and thus a sponsor is needed for each plan. The number of sponsors is thus less than or equal to the number of plans in the system. Since the upper bound on the number of sponsors, is the number of plans and since the number of plans is a more readily understood metric of the size of the system, the above equation can be expressed as:

$$T(\text{worst general-case}) = t_{\text{Cell}} * (n_{\text{Plans}} * avR_{\text{Sp}} * avC_{\text{Sp}} + n_{\text{Steps}} * avR_{\text{St}} * avC_{\text{St}})$$

where nPlans is the total number of plans in the system. From Section 7.6.4, $t_{\text{Cell}} = 150$ milliseconds on a Macintosh Quadra 700 computer. For an average number of rows per sponsor (avC_{Sp}) and an average number of columns per step (avC_{St}) of 5 and an average number of rows per sponsor (avR_{Sp}) of 5 and an average number of rows per step (avR_{St}) of 10, Table 3 summarizes the estimated worst case running times for different values for the number of plans and the number of steps involved in any design system.

Number of Plans	Number of Steps	Estimated Worst Case Running Time
10	15	2.2 minutes
20	30	4.5 minutes
20	50	7.5 minutes
50	50	9.3 minutes
50	100	15.6 minutes
100	200	31.2 minute.

TABLE 3. Expected Worst Case Running Times for Different Problem Sizes

In the next Chapter, I will describe a module for ranking the set of generated designs according to the availability and cost of the ingredients. This description will be followed by a case study of the designs generated by the DSPL system and the designs generated by selecting the highest ranked design from the alternatives generated by MDSPL.

Chapter 8

Ranking and Selection of Composite Materials Designs

As mentioned in Chapter 5, I implemented the composite materials design system in the form of two modules:

1. An MDSPL design system for generating multiple design alternatives.
2. A selection module to select among the generated design alternatives.

This division serves a dual purpose:

1. to provide multiple design alternatives to allow for user preferences.
2. to separate the dynamic market factors of availability and cost of raw materials from the static factors relating to the design process itself.

In the previous chapter, I described the MDSPL system for generating multiple alternatives of composite materials design. In this chapter, I discuss the selection module.

8.1 Selection Criteria

The selection process is implemented by calculating the cost per unit volume¹ of each of the resulting composite materials designs. The resulting designs are then ranked in ascending order of cost and presented to the user before the final selection is made. Before comparing the costs of the different designs, the user is first queried about the availability of the different raw materials and any design whose ingredients are not all available is excluded from the ranking process. If none of the alternative designs has all its ingredients available, the user is notified and asked whether to proceed with the ranking process anyway.

1. The cost per unit volume is a more realistic comparison than the cost per unit weight since, for a given application, and using any of the designs, parts having the same physical dimensions need to be manufactured.

8.2 Materials Database

In order to effectively execute the computation of cost for the various designs, the ranking module maintains a database for the raw materials. The database maintains information about polymer materials, curing and co-curing agents and fiber materials. For each material, the database stores the density and the latest price per unit weight¹.

To allow for changes in the design system, the materials database is configured such that any material is added to the database when it is first encountered in a design. When the ranking module encounters a new material, the system prompts the user for the density and current price for that material and adds it to the database. During routine use of the system, the user is presented with the prices stored in the database and asked to update any prices which might have changed.

8.3 Ranking Algorithm

The ranking process is based on computing the cost for producing a unit volume of each of the designs. For each design the cost is calculated as follows:

1. The fiber volume ratio is calculated (the ratio of the fiber volume compared to the total volume) using the formula:

$$\text{fiber ratio} = 0.6 * \text{required tensile modulus} / \text{tensile modulus of fiber}$$

2. The cost for the fiber is calculated using the calculated fiber ratio, the density of the fiber material and the price per unit weight of the fiber material.
3. The volume of the polymer material is computed (the remaining volume after subtracting the fiber).

1. Even though price comparisons among the different designs are based on volume, the price of raw materials is based on weight simply because these materials are sold by weight and not by volume.

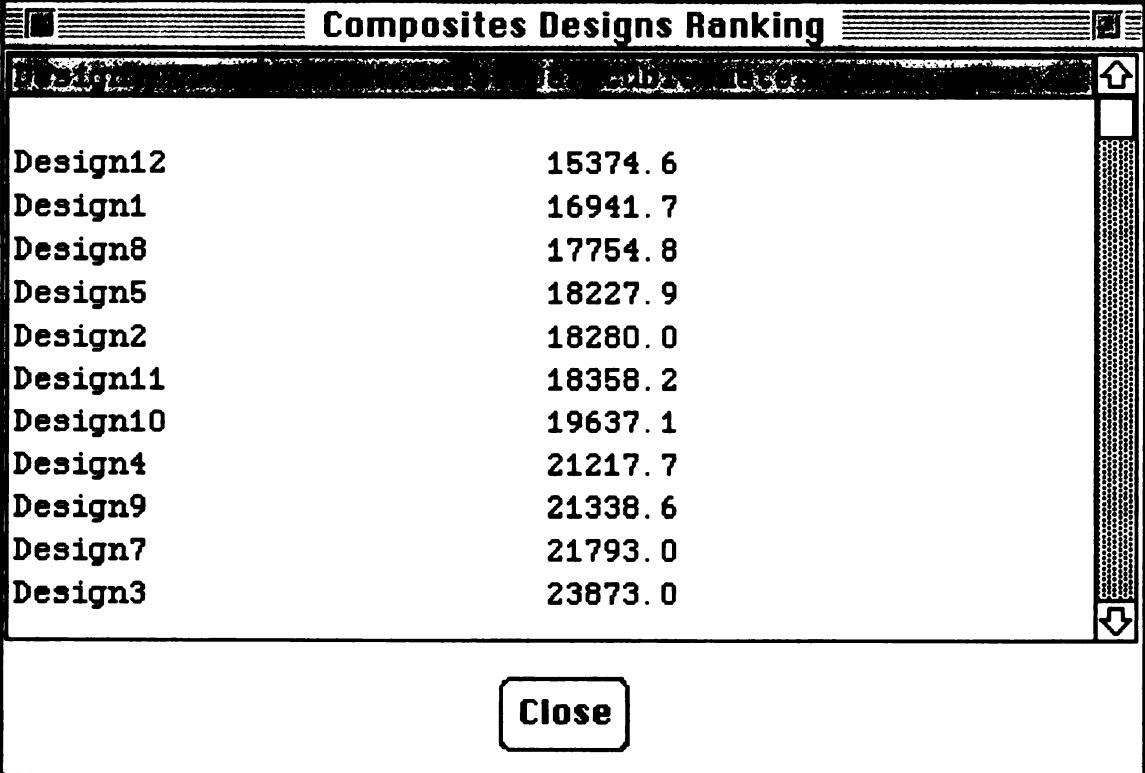
4. The cost of the polymer material is calculated based on the calculated volume, the density and the price per unit weight.
5. The weight of the curing agent (or the co-curing agent) is calculated (30% of the weight of the polymer material)¹.
6. The cost of the curing agent (or co-curing agent) is calculated.
7. The cost of operating the curing equipment is calculated based on the hourly rate of operating the equipment and the needed time for that particular design.
8. All the costs are added up to form the total cost of manufacturing for that design.

8.4 Output

The output of the system is displayed as a list of the designs and the total cost of producing a unit volume of each of the designs, arranged in ascending order as shown in Figure 50. By clicking on any of the designs, the details of that design are displayed as given in Figure 51. The example illustrated in Figure 50 and Figure 51 is the same example used in Chapter 7, the input data is shown in Figure 38, and the output from the design process is shown in Figure 40 and Figure 41. Figure 51 shows the highest ranking design (lowest cost).

In the next chapter, I will discuss two case studies of the designs generated by the DSPL system and the designs generated by selecting the highest ranked design from the alternatives generated by MDSPL.

1. Note that the curing agent (or co-curing agent) is not considered to occupy any volume in the finished product. Its purpose is to help in the cross-linking of the polymer molecules and the final volume is the same as that of the polymer material.



Design12	15374.6
Design1	16941.7
Design8	17754.8
Design5	18227.9
Design2	18280.0
Design11	18358.2
Design10	19637.1
Design4	21217.7
Design9	21338.6
Design7	21793.0
Design3	23873.0

Close

Figure 50: Ranked Designs

Composites Design	
Polymer Material	'DGEBA'
Curing/Co-cure Agent	'DDS'
Fiber Type	'E-Glass'
Cure Temperature	150
Cure Time	60
Post Cure Temperature	220
Post Cure Time	180

Close

Figure 51: An Individual Design

8.5 Two Case Studies of the Materials Design MDSPL System

In this section, I present two case studies of the materials design MDSPL system for composite materials. These examples demonstrate the added flexibility of the MDSPL design system (together with the design ranking system) compared to the single-design DSPL system. These examples also demonstrate the modularity the MDSPL approach exhibits by isolating the dynamic factors of price and availability of raw materials from the basic design knowledge.

While I present the outputs from both the DSPL system and the MDSPL system, a direct comparison between the two outputs is not meaningful. The ranked output from the MDSPL system is based on a comparison of the costs of the produced designs. However, the output from the DSPL system does not take the cost into consideration directly. In the DSPL system cost can be implicitly represented by arranging the options such that the lower cost choices are considered before the higher cost ones.

8.5.1 Example 1

Consider the example design with the input values as shown in Figure 52. The DSPL

Application Type	'Civil Engineering'
Corrosion	'No'
Flame Retardance	'No'
Humidity	'No'
Maximum Tensile Modulus	200
Required Glass Transition Temperature	200
Required Tensile Modulus	50
Rigidity	'No input'
Use Temperature	100

Proceed

Figure 52: Input Data for Example 1

output for this example is shown in Figure 53. The MDSPL output for this example is shown in two parts in Figure 54 and Figure 55. By applying the ranking mechanism discussed in Chapter 8, assuming all raw materials are available, and assuming the prices per kilogram¹ of raw materials as shown in Table 4, the highest ranked design (the design with the lowest cost) is shown in Figure 56. This design is the same as that given by the DSPL output in Figure 53. By changing the market conditions, the “best” design changes accordingly. For example, if we specify that the polymer “DGEBA” is not available, the “best” design becomes the design shown in Figure 57.

1. The prices shown do not reflect actual market prices.

Cure Temperature	150
Cure Time	60
Curing/Co-cure Agent	'DDS'
Fiber Type	'E-Glass'
Glass Transition Temperature	220
Polymer Material	'Epoxidized Phenolic Novolac'
Post Cure Temperature	220
Post Cure Time	180
Tensile Modulus	52

View Inputs
Proceed

Figure 53: Single Design Output for Example 1

Material	Price per KG.
DGEBA	10.00
Epoxidized Phenolic Novolac	12.00
Hydroxyphenyl	15.00
DDS	12.00
E-Glass	5.00
S-Glass	7.00
AS4-Carbon	10.00

TABLE 4. Prices of Raw Materials for Example 1

Composites.DB.mdr

Cure Temperature = 150	Cure Time = 60	Curing/Co-cure Agent = 'DDS'	Fiber Type = 'A S4-Carbon'	Glass Transition Temperature = 220	Polymer Material =
					Polymer Material =
					Polymer Material =
					Polymer Material =
			Fiber Type = 'S-Glass'	Glass Transition Temperature = 220	Polymer Material =
					Polymer Material =
					Polymer Material =
			Fiber Type = 'E-Glass'	Glass Transition Temperature = 220	Polymer Material =
					Polymer Material =

Figure 54: Part 1 of the MDSPL Output of Example 1

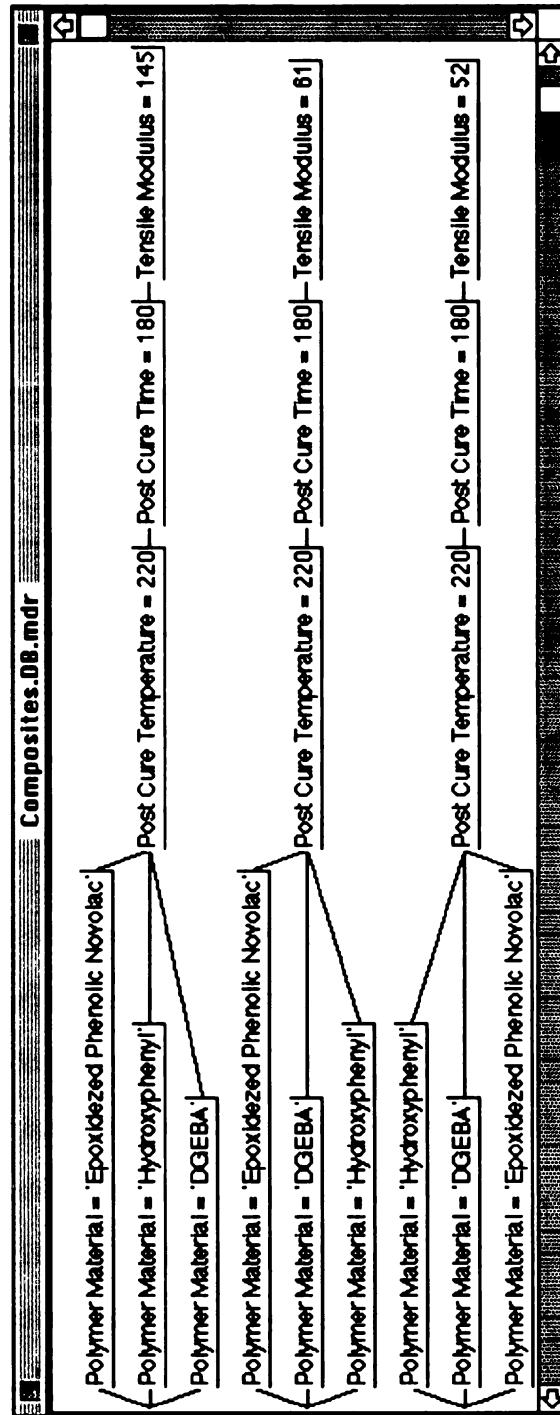


Figure 55: Part 2 of the MDSPL Output of Example 1

By changing the price per kilogram of the polymer “Hydroxyphenyl” to \$8.00/kg, the results of the ranking change—the highest ranked design—becomes the design shown in Figure 58.

Composites Design	
Polymer Material	'DGEBA'
Curing/Co-cure Agent	'DDS'
Fiber Type	'E-Glass'
Cure Temperature	150
Cure Time	60
Post Cure Temperature	220
Post Cure Time	180

Close

Figure 56: Highest Ranked Design for Example 1 (a)

By separating these issues of availability and cost from the basic design knowledge, changes in these market conditions are easily accommodated resulting in shifts in the ranking of the different designs without a need to alter the basic design knowledge. By contrast, if we had needed to accommodate these changes in the traditional single design DSPL, we would have had to edit the design knowledge to re-arrange the preferences whenever there is a change in market conditions. Additionally, in Multiple Design, the availability of design alternatives allows the user of the system to accommodate other factors which cannot be quantized, such as the preference of the user. The list of the ranked designs from the example shown above is provided in Figure 59.

Composites Design	
Polymer Material	'Epoxidezed Phenolic Novolac'
Curing/Co-cure Agent	'DDS'
Fiber Type	'E-Glass'
Cure Temperature	150
Cure Time	60
Post Cure Temperature	220
Post Cure Time	180

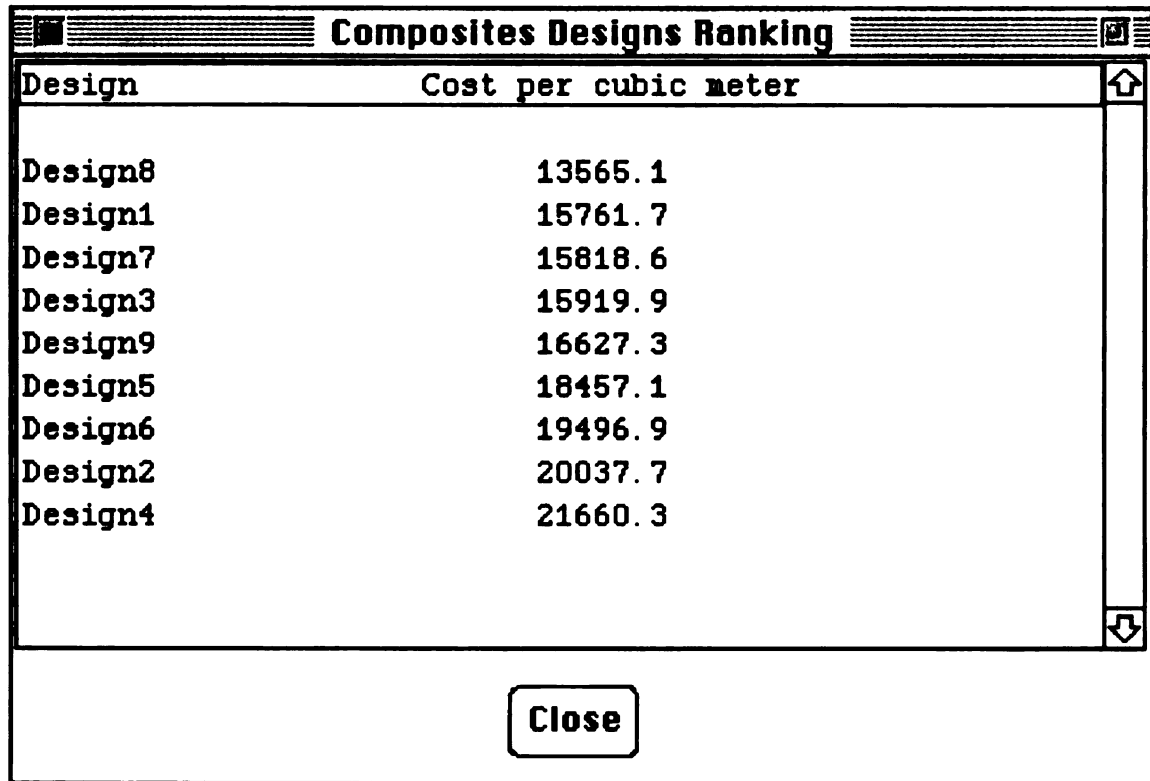
Close

Figure 57: Highest Ranking Design for Example 1 (b)

Composites Design	
Polymer Material	'Hydroxyphenyl'
Curing/Co-cure Agent	'DDS'
Fiber Type	'E-Glass'
Cure Temperature	150
Cure Time	60
Post Cure Temperature	220
Post Cure Time	180

Close

Figure 58: Highest Ranking Design for Example 1 (c)



Design	Cost per cubic meter
Design8	13565.1
Design1	15761.7
Design7	15818.6
Design3	15919.9
Design9	16627.3
Design5	18457.1
Design6	19496.9
Design2	20037.7
Design4	21660.3

Figure 59: Ranked List of Designs for Example 1

8.5.2 Example 2

To further illustrate the difference between the outputs of DSPL and MDSPL, I will present another example. The input data for this example is shown in Figure 60. The single-design output is given in Figure 61. The Multiple Design output is displayed in two parts in Figure 62 and Figure 63. The ranked designs are in Figure 64, and the highest ranking design is shown in Figure 65. The pricing data used for this example is in Table 5¹.

In this example, contrary to the previous example, the highest ranked design is not the same as the result from the single design system. In the single design system the chosen

1. The prices shown do not reflect actual market prices.

Application Type	'Domestic Applications'
Corrosion	'No'
Flame Retardance	'No'
Humidity	'No'
Maximum Tensile Modulus	180
Required Glass Transition Temperature	'No input'
Required Tensile Modulus	60
Rigidity	'No'
Use Temperature	100

Proceed

Figure 60: Input Data for Example 2

Cure Temperature	80
Cure Time	3
Curing/Co-cure Agent	'Chlorostyrene'
Fiber Type	'S-Glass'
Polymer Material	'Orthophthalic Acid'
Post Cure Temperature	0
Post Cure Time	0
Tensile Modulus	61

View Inputs **Proceed**

Figure 61: Single Design Output for Example 2

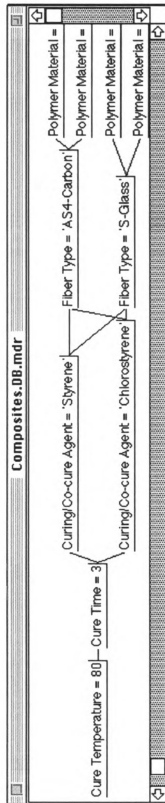


Figure 62: Part1 of the MDSPL Output of Example 2

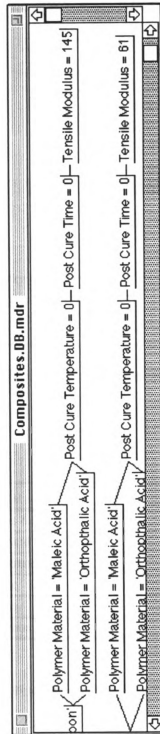


Figure 63: Part 2 of the MDSPL Output of Example 2

Composites Designs Ranking	
Design	Cost per cubic meter
Design7	12665.5
Design3	12972.2
Design1	14174.0
Design5	14341.2
Design6	14710.2
Design2	15016.9
Design8	15288.8
Design4	15456.0
Close	

Figure 64: Ranked Designs for Example 2

Material	Price per KG.
Orthophthalic Acid	6.00
Maleic Acid	8.00
Chlorostyrene	3.00
Styrene	4.00
S-Glass	7.00
AS4-Carbon	10.00

TABLE 5. Prices of Raw Materials for Example 2

fiber is “S-Glass,” whereas in the multiple design system, the fiber in the highest ranked design is “AS4-Carbon.” “AS4-Carbon” and carbon fibers in general are more expensive

fiber than “S-Glass” and glass fibers. However, carbon fibers have a higher tensile modulus than glass fibers. As a result, to achieve the same required tensile modulus, a smaller percentage (of the total material volume) of carbon fiber is required, resulting in a smaller overall cost .

Composites Design	
Polymer Material	'Orthophthalic Acid'
Curing/Co-cure Agent	'Chlorostyrene'
Fiber Type	'A54-Carbon'
Cure Temperature	80
Cure Time	3
Post Cure Temperature	0
Post Cure Time	0

Close

Figure 65: The Highest Ranking Design for Example 2

In the single design case, partial designs are ordered such that choices that are known from experience to yield lower cost are considered first. In this case the design system is formulated to select glass fibers whenever possible since glass fibers cost less than carbon fibers.

8.6 Discussion

The two examples above demonstrate the added flexibility of MDSPL design systems compared to DSPL design systems. In the MDSPL system, alternative choices are given to the user to accommodate for factors that cannot be quantized such as the user preferences.

In the MDSPL system for the material design of composite materials, the dynamic cost and availability factors of raw materials are isolated from basic design knowledge, thereby allowing easy changes in the selected designs as these factors change, as illustrated by Example 1. Attempting to include these factors in the DSPL system would involve complex modifications of the design knowledge.

In the DSPL system, cost factors are implicitly considered through the partial ordering of the design decisions to favor lower cost alternatives. However, choosing lower cost alternatives do not necessarily result in a lower cost for the entire design. In the MDSPL system, the final selection is based on computing the total costs for the different generated designs, allowing lowest cost design to be selected.

Chapter 9

Contributions and Future Research

As mentioned in Section 4.1, most research in knowledge-based systems yields contributions in two areas: knowledge-based systems and the area of the domain problem. This research provides contributions in knowledge-based systems and in composite materials design.

In knowledge-based systems, there are three central contributions:

1. I proposed an integrated architecture for design problem solving based on previous experience. This architecture can produce designs by altering previous similar designs in addition to producing designs “from scratch”. The architecture also provides a means for “testing” the generated designs.
2. I developed an effective technique for generating multiple design alternatives to meet a common set of specifications.
3. I also designed a language called MDSPL with a user-friendly interface.

In the area of composite materials design, there are two central contributions:

1. I designed an integrated architecture for material design for composite materials.
2. I implemented a prototype system for generating multiple materials designs of fiber-reinforced thin film polymer composites systems to meet a common set of specifications. I also implemented a ranking module to rank the resulting designs according to their cost.

In this chapter, I will detail each of these contributions, and present a plan for future research in each of these areas.

9.1 Generating Multiple Designs

As mentioned in Section 1.5, there are many situations in which generating a single design to meet a set of specifications is not sufficient but in which multiple designs are desirable.

These situations include:

- situations in which the design problem is in an area where sufficient design experience is not available and hence multiple alternatives need to be provided for testing. This need is especially important in relatively new areas like composite materials, where sufficient knowledge is available for generating designs but there is not sufficient knowledge available for selecting the best design. Due to the lack of experience in these areas, there may be large gaps in design knowledge. Consequently, it is difficult to guarantee that a single design would satisfy the requirements. To increase the possibility of obtaining a useful output, multiple designs are necessary.
- situations in which practical economical or market-dictated considerations have to be met; such as cases in which a more expensive raw material is more readily available than another lower cost material, or cases where market considerations dictate the use of a process with a faster turnaround time even if at a higher cost. The availability of multiple designs helps accommodate such requirements, by enabling switching between different designs at different times to meet current conditions.
- situations which require conforming with common engineering practices in order to make the knowledge-based design system acceptable as an aid for human designers. Human designers are usually reluctant to use any new technology especially if it is substantially different from normal practice. By having the design system generate multiple designs, the human user can

thus concentrate on the task of evaluating the designs and selecting among them.

In this research, I developed a technique for generating multiple designs based on the generic-task approach to knowledge-based systems (Chandrasekaran, 1985; Chandrasekaran, 1986), and in particular on the generic task “Routine Design” (Brown & Chandrasekaran, 1989; Brown, 1987).

The technique developed here provides a set of designs based on a common set of specifications. Several enhancements to this technique will be pursued in future research. These enhancements include:

- One of the motivations for generating multiple designs is to accommodate practical industrial requirements, e.g. the need to use a certain raw material. This is accommodated in the current system by generating multiple designs regardless of this requirement, and leaving it to the human designers to select the design that suits their needs.

An interesting point for future investigation is to modify the technique in a way that allows the user to specify a subset of the system outputs; in other words allows specification of a partial design and then lets the system generate only the designs (if any) that have this partial design as a subset.

- As mentioned in Section 6.7.9.2, limited explanation capability is provided by allowing the user to browse the particular matchers that resulted in interesting design values. Further research is needed for providing a more comprehensive explanation capability.
- It has been shown in Section 7.6 that the time required by the design process is linearly proportional to the number of different choices available to the system for the different design parameters. Further investigation is needed

to assess the role of knowledge in reducing the higher level complexity that would be expected in these situations into this linear form.

9.2 The MDSPL Language

In this research, I developed a language MDSPL as a superset of DSPL (Brown & Chandrasekaran, 1989), the language for Routine Design.

MDSPL is implemented in a diagrammatic form as a series of browsers as described in Section 6.7. These browsers provide the capability to view and edit the various components involved in the system in a user friendly fashion. Templates are provided for building the components for a design system. This feature provides for an enhanced knowledge acquisition capability.

While the MDSPL language provides templates for system building, it does not provide a checking mechanism to guarantee the correctness of the resulting system. For example, a system builder can define a specialist to have some plan while forgetting to define the plan. These inconsistencies would only be discovered at runtime.

In future research an overlay should be added to the system to check design systems for inconsistencies and draw the attention of the system developer to such inconsistencies.

9.3 Composite Materials Design

In this research, I developed a system for the material design of polymer composite materials using the MDSPL language (see Chapter 7). Experts in composite materials agree that this system produces results similar to those produced by human designers given the same inputs. I also developed a ranking module for ranking the generated designs according to their cost (see Chapter 8). Several extensions to this system are planned for future research. These extensions include:

- expanding the coverage of the system to include more types of materials as well as more types of composites.
- the implementation of the other modules in the integrated architecture of Chapter 5, Figure 22:
 - Implementing the case library would make subsequent design cases faster by making use of similar previous cases, if such cases exist.
 - Implementing the functional modeling component would reduce the number of iterations for manufacturing and testing samples of proposed designs. The functional modeling component would model the fabrication process of the proposed composites design and predict the properties of the finished composite eliminating the need for actual manufacturing and testing in most cases.

9.4 Other Work in Progress

As mentioned in Section 4.2.1, this research is part of a larger project to automate the entire life cycle of composite materials fabrication. There is other work in progress for overlaying the system with a module for selecting a processing method for the designed composite. The current system generates multiple designs. There are several methods for processing composite materials. Not all processing methods are compatible with all materials. This should be accounted for in the processing method selection module by interacting with the design system. An additional layer that evaluates the combined outputs of the design system and the processing method selection system and selects a combined design/processing method is also planned.

A design for a composite material, in addition to the design of the material itself, includes the design of the parts to be manufactured. Other research is in progress for the design of composite materials parts [McDowell, 1993b; McDowell, Kamel, Sticklen, & Hawley, 1993].

Other work is also in progress for controlling the processing of the composite materials [Adegbite, Hawley, Kamel, Pegah, & Sticklen, 1992; Adegbite, Hawley, Sticklen, & Kamel, 1991]. The overall objective is to combine the results from this work with the design system (both for the design of materials and for the design of parts) to accomplish an integrated industrial system for the design and manufacturing of composite materials [McDowell, 1993a; Sticklen, Kamel, Hawley, & Adegbite, 1992].

BIBLIOGRAPHY

Bibliography

- Adegbite, V., Hawley, M., Kamel, A., Pegah, M., & Sticklen, J. (1992). Monitoring and Control of Microwave Processing Utilizing Functional Modeling. In ANTEC-92, . Detroit, Michigan: Society of Plastics Engineers.
- Adegbite, V., Hawley, M., Sticklen, J., & Kamel, A. (1991). Implementation of an Artificial Intelligence Technique for Real-Time Control of Microwave Processing of Composite Materials. In Seventh Annual ASM/ESD Advanced Composites Conference, (pp. 409-413). Detroit, Michigan:
- Bachant, J., & McDermott, J. (1984). R1 revisited: four years in the trenches. AI Magazine, 5(3), 21-32.
- Balzer, R., Erman, L. D., London, P. E., & Williams, C. (1980). Hearsay-III: A domain-independent framework for expert systems. In The First Annual National Conference on AI (pp. 108-110). Stanford, California:
- Barletta, R., & Hennessy, D. (1992). Case Adaptation in Autoclave Layout Design. In 37th International SAMPE Symposium, (pp. 203-207).
- Birmingham, W. P. (1989a). The Design of an Integrated Environment for the Automated Synthesis of Small Computer Systems. In 22nd Hawaii International Conference on System Sciences, 1 (pp. 49-58). IEEE CS Press.
- Birmingham, W. P. (1989b). The Micon System for Computer Design. IEEE Micro, 1989(October), 61-67.
- Brown, D. (1993). Personal Communication.
- Brown, D., & Chandrasekaran, B. (1989). Design Problem Solving: Knowledge Structures and Control Strategies. London: Pitman.
- Brown, D. C. (1987). Routine Design Problem Solving. In J. Gero (Ed.), Knowledge Based Systems in Engineering and Architecture Addison-Wesley.
- Brown, D. C. (1991a). Design. In S. C. Shapiro (Ed.), The Encyclopedia of Artificial Intelligence Wiley-Interscience.
- Brown, D. C. (1991b). Routineness Revisited. In M. Waldron & K. Waldron (Eds.), Mechanical Design: Theory and Methodology Springer-Verlag.
- Brown, D. C., & Chandrasekaran, B. (1985). Expert Systems for a Class of Mechanical Design Activity. In J. S. Gero (Ed.), Knowledge Engineering in Computer-Aided Design (pp. 259-282). North-Holland.
- Brown, D. C., & Chandrasekaran, B. (1986). Knowledge and Control for a Mechanical Design Expert System. IEEE Computer, 19(7), 92-100.

- Bylander, T., Chandrasekaran, B., & Josephson, J. (1987). The Generic Task Toolset. In Second International Conference on Human Computer Interaction, . Honolulu, Hawaii:
- Bylander, T., & Mittal, S. (1986). CSRL: A Language for Classificatory Problem Solving and Uncertainty Handling. AI Magazine, 7(2), 66-77.
- Chandrasekaran, B. (1983). Towards a Taxonomy of Problem Solving Types. The AI Magazine, 4(winter/spring), 9-17.
- Chandrasekaran, B. (1985). Expert Systems: Matching Techniques To Tasks. In W. Reitman (Ed.), Artificial Intelligence Applications for Business (pp. 41-64). Ablex Corp., publishers.
- Chandrasekaran, B. (1986). Generic Tasks in Knowledge-Based Reasoning: High-Level Building Blocks for Expert System Design. IEEE Expert, 1(fall), 23-30.
- Chandrasekaran, B. (1990). Design Problem Solving: A Task Analysis. AI magazine, 11(4), 59-71.
- Chandrasekaran, B., & Goel, A. (1988). From Numbers to Symbols to Knowledge Structures: Artificial Intelligence Perspectives on the Classification Task. IEEE Transactions on Systems, Man, and Cybernetics, 18(3), 415-424.
- Chandrasekaran, B., Josephson, J., Keuneke, A., & Herman, D. (1986). An Approach To Routine Planning. The International Journal of Man-Machine Studies(special issue on Human and Machine Planning).
- Chandrasekaran, B., Josephson, J., Keuneke, A., & Herman, D. (1989). Building routine planning systems and explaining their behaviour. International Journal of Man-Machine Studies, 30, 377-398.
- Clancey, W. J. (1985). Heuristic Classification. Artificial Intelligence, 27, 289-350.
- Daube, F., & Hayes-Roth, B. (1989). A Case-Based Mechanical Redesign System. In The Eleventh IJCAI, 2 (pp. 1402-1407). Detroit, MI: Morgan-Kaufman.
- Davis, R., & King, J. J. (1977). A Overview of Production Systems. In E. Elcock & D. Michie (Eds.), Machine Intelligence Chichester, England: Horwood.
- Erman, L. D., Hayes-Roth, F., Lesser, V. R., & Reddy, D. (1980). The Hearsay-II speech understanding system: integrating knowledge to resolve uncertainty. Computing Surveys, 12(2), 213-253.
- Erman, L. D., London, P. E., & Fickas, S. F. (1981). The design and an example use of Hearsay-III. In Seventh International Joint Conference on Artificial Intelligence, (pp. 409-415). Vancouver, BC:
- Fikes, R. E., & Nilsson, N. J. (1971). STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. Artificial Intelligence, 2(3/4), 189-208.
- Gero, J. S. (1990). Design Prototypes: A Knowledge Representation Schema for Design. AI magazine, 11(4), 26-36.
- Goel, A. K. (1989). Integration of Case-Based Reasoning And Model-Based Reasoning For Adaptive Design Problem Solving. Ph.d., The Ohio State University.
- Hahn, H. T. (1991). Design For Manufacturability (DFM) For Composites. In ICCM-91, .

- Hammond, K. J. (1989). Case-Based Planning: Viewing Planning as a Memory Task. San Diego: Academic Press, Inc.
- Hayes-Roth, B. (1985). A Blackboard Architecture for Control. Artificial Intelligence, 26(3), 251-321.
- Herman, D. J. (1992). An Extensible, Task-Specific Shell For Routine Design Problem Solving. Doctoral, The Ohio State University.
- Horner, R., & Brown, D. C. (1990). Knowledge Compilation Using Constraint Inheritance. In The Artificial Intelligence in Engineering Conference, (pp. 161-176).
- Huhns, M. N., & Acosta, R. D. (1988). Argo: A System for Design by Analogy. In IEEE Fourth Conference on Artificial Intelligence Applications, (pp. 146-151). IEEE.
- Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). SOAR: An Architecture for General Intelligence. Artificial Intelligence, 33, 1-64.
- LeClair, S. R., Abrams, F. L., & Matejka, R. F. (1989). Qualitative Process Automation: Self-Directed Manufacture of Composite Materials. AIEDAM, 3(2), 125-136.
- Maher, M. L., & Fenves, S. J. (1985). HI-RISE: an expert system for the preliminary structural design of high rise buildings. In J. S. Gero (Ed.), Knowledge Engineering in Computer-Aided Design (pp. 125-164). Amsterdam: North Holland.
- Mark, W. S. (1992). Case-Based Reasoning for Autoclave Management. In 37th International SAMPE Symposium, (pp. 176-180).
- McDermott, J. (1982). R1: A Rule-Based Configurer of Computer Systems. Artificial Intelligence, 19(2), 39-88.
- McDermott, J. (1988). Preliminary Steps Toward a Taxonomy of Problem-Solving Methods. In S. Marcus (Ed.), Automating Knowledge Acquisition for Expert Systems (pp. 225-258). Boston: kluwer Academic Publishers.
- McDowell, J. K., Jon Sticklen and Martin C. Hawley (1993a). Integrated Design and Manufacturing for Composite Materials: Knowledge-based Systems and Product Data Exchange Standards. In D. G. Goldsein (Ed.), AAAI/SIGMAN Workshop on Intelligent Manufacturing Technology, (pp. submitted). Washington, D.C.: American Association of Artificial Intelligence/Special Interest Group on Manufacturing.
- McDowell, J. K., Ahmed Kamel, Jon Sticklen and Martin C. Hawley (1993b). Integrating Material/Part/Process Design for Polymer Composites. A Knowledge-based Problem Solving Approach. In D. G. Newaz (Ed.), ASC 8Th Technical Conference on Composite Materials, (pp. accepted for presentation). Cleveland, OH: American Society of Composites.
- McDowell, J. K., Kamel, A., Sticklen, J., & Hawley, M. C. (1993). A Knowledge-Based Architecture for Integrated Material/Part/Process Design in Polymer Composites. In AIChE 1993 Summer National Meeting, . Seattle, Washington, USA: American Institute of Chemical Engineers.
- Meehan, E., & Brown, D. C. (1990). Constraint Absorption and Relaxation Using A design History. In ASME Design Theory and Methodology Conference, . Chicago, IL:
- Newell, A. (1980). The Knowledge Level. In AAAI-80, (pp. Presidential Address). Stanford, CA: American Association for Artificial Intelligence.

- Newell, A., J.C. Shaw, & Simon, H. A. (1960). Report on a General Problem Solving Program. In International Conference on Information Processing, (pp. 256-264). Paris: UNESCO.
- Newell, A., & Simon, H. A. (1963). GPS: A Program that Simulates Human Thought. In E. A. Feigenbaum & J. Feldman (Eds.), Computers and Thought (pp. 279-293). R. Oldenbourg KG.
- Newell, A., & Simon, H. A. (1972). Human Problem Solving. Englewood Cliffs, N.J.: Prentice-Hall.
- Nitsche, A., Kern, H., & Janczak, J. (1990). Composites Design Based on Expert Knowledge. In The Fifth Technical Conference of The American Society for Composites, (pp. 382-390). East Lansing, MI:
- Punch, W. F. (1989). A Diagnostic System Using A Task Integrated Problem Solver Architecture (TIPS). Including Causal Reasoning. Ph.d., the Ohio State University.
- Sacerdoti, E. D. (1974). Planning in Hierarchy of Abstraction Spaces. Artificial Intelligence, 5(1), 115-135.
- Sacerdoti, E. D. (1975). The Nonlinear Nature of Plans. In IJCAI-75, (pp. 206-214).
- Schank, R. C. (1982). Dynamic Memory: A theory of learning in computers and people. Cambridge University Press.
- Sembugamoorthy, V., & Chandrasekaran, B. (1986). Representation of Devices and Compilation of Diagnostic Problem-Solving Systems. In J. Kolodner & C. Reisbeck (Eds.), Experience, Memory, and Learning Lawrence Erlbaum Associates.
- Shortliffe, E. H. (1976). Computer-based Medical Consultations: MYCIN. New York: Elsevier.
- Sloan, W. N., & Brown, D. C. (1988). Adjusting Constraints in Routine Design. In Seventh National Conference on Artificial Intelligence. Workshop on AI in Design, . St. Paul, MN:
- Smith, J. W., & Johnson, T. R. (1993). A Stratified Approach to Specifying, Designing, and Building Knowledge Systems. IEEE Expert, 8(3), 15-25.
- Steels, L. (1990). Components of Expertise. AI Magazine, 11(2), 28-49.
- Stefik, M. (1981). Planning with Constraints. Artificial Intelligence, 16(1), 111-140.
- Steier, D. M., Lewis, R. L., Lehman, J. F., & Zacherl, A. L. (1993). Combining Multiple Knowledge Sources in an Integrated Intelligent System. IEEE Expert, 8(3), 35-44.
- Sticklen, J. (1987). MDX2: An Integrated Medical Diagnostic System. ph.d., The Ohio State University.
- Sticklen, J. (1988). Problem Solving Architecture at the Knowledge Level. Journal of Theoretical and Applied Intelligence.
- Sticklen, J., & Chandrasekaran, B. (1989). Integrating Classification-Based Compiled Level Reasoning with Function-Based Deep Level Reasoning. In W. Horn (Ed.), Causal AI Models: Steps Towards Applications (pp. 191-220). Hemisphere Publishing.
- Sticklen, J., Chandrasekaran, B., & Josephson, J. (1987). Modularity of Domain Knowledge. Expert Systems: Research and Applications, 1.

- Sticklen, J., Kamel, A., Hawley, M., & Adegbite, V. (1992). Fabricating Composite Materials: A Comprehensive Problem Solving Architecture Based on a Generic Task Viewpoint. IEEE Expert, 7(2), 43-53.
- Sussman, G. J. (1973). A Computational Model of Skill Acquisition (Tech. Note No. AI-TR-297). Artificial Intelligence Laboratory, MIT.
- Tate, A. (1974). INTERPLAN: A Plan Generation System which can deal with Interactions between Goals (Memorandum No. MIP-R-109). Machine Intelligence Research Unit, University of Edinburgh.
- Tommelein, I. D., Johnson, M. V., Hayes-Roth, B., & Levitt, R. E. (1987). SIGHTPLAN: A blackboard expert system for construction site layout. In J. S. Gero (Ed.), Expert Systems in Computer-Aided Design (pp. 153-167). Amsterdam: North Holland.
- Traister, J. E. (1978). Handbook of Electrical Systems Design Practices. Reston, VA: Reston Publishing Co.
- van Melle (1981). The EMYCIN Manual (Technical Report No. Heuristic Programming Project, Stanford University).
- Venkatasubramanian, V., Lee, Y., & Gryte, C. (1987). Design of Polymer Composites: A Knowledge-Based Framework. In Annual meeting of The American Institute of Chemical Engineers, . New York: American Institute of Chemical Engineers.
- Wielinga, B. J., Bredeweg, B., & Breuker, J. (1988). Knowledge acquisition for expert systems. In R. Nossur (Ed.), ACAI-87, . Berlin: Springer.
- Wielinga, B. J., & Breuker, J. A. (1986). Models of Expertise. In B. d. boulay, D. Hoggs, & L. Steels (Eds.), Advances in Artificial Intelligence Amsterdam: North-Holland.