A Probabilistic Specification-based Design Model: applications to search and environmental computer-aided design

by

David R. Wallace

Master of Science in Mechanical Engineering
Massachusetts Institute of Technology
August, 1991

Bachelor of Mechanical Engineering
Carleton University
June, 1989

Bachelor of Industrial Design
Carleton University
June, 1986

Submitted to the Department of Mechanical Engineering
in Partial Fulfillment of the Requirements for the Degree of

DOCTORATE OF PHILOSOPHY
in Mechanical Engineering

at the
Massachusetts Institute of Technology
October, 1994 [Feb 1995]

© Massachusetts Institute of Technology, 1994, All rights reserved.

Signature of Author ____________________________

Certified by ________________________________ Woodie Flowers
Department of Mechanical Engineering
Professor, Mechanical Engineering
October 24, 1994
Thesis Supervisor

Accepted by ________________________________ Ain A. Sonin
ARCHIVES
Chairman, Department Committee
M A S S A C H U S E T T S I N S T I T U T E
OF TECHNOLOGY
ARCHIVES
1994
A Probabilistic Specification-based Design Model: applications to search and environmental computer-aided design

David R. Wallace

Submitted to the Department of Mechanical Engineering on October 21, 1994 in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Mechanical Engineering

Abstract

The importance of considering environmental issues in the design of consumer products is increasingly recognized. The current interest in environmental design is part of a general trend toward systems-oriented product design. In the future, even non-technical consumer products will be designed as part of a larger anthropological and ecological system. This change will require that non-technical designers must comprehend and evaluate a diverse range of design problems over a complete product life-cycle (from material extraction through manufacture and assembly, to use and obsolescence). As such, the goal of this thesis is to develop a general framework that will help product designers conveniently model and solve a diverse range of design problems.

A general method to model multiple-criteria design problems using the familiar language of specifications is proposed. This new approach to design evaluation is based upon the probability of acceptance and represents the crux of the thesis. The probabilistic representation accommodates uncertainty in both design performance and design specifications. This design evaluation principle is then used to develop capabilities that will be needed in a systems-oriented design tool.

A general and robust search facility using the specification-based model and genetic algorithms is presented. Optimization or search problems are formulated directly in the language of design specifications. Fitness function (objective formulation) is automated, and no distinction is made between discrete and probabilistic optimization or between objectives and constraints. A hierarchical catalog architecture is proposed and then combined with the specification-based optimization facility to permit simultaneous catalog selection and continuous parameter optimization. An analog to combinatorial gene regulation is introduced, thereby allowing the search to simultaneously pick elements within catalogs and choose between alternative catalogs.

Finally, a preliminary representation for encoding the structure of design problems modeled using the specification-based method is described. Templates for solving categories of design problems may be defined using this representation. Once a library of templates has been developed, it will be possible to conveniently evaluate designs from a wide variety of perspectives without modeling overhead. This is important as most product designers will not have the expertise needed to model many life-cycle design problems.

Together, these three components—the specification-based design model, the general search facility, and the template representation for encoding categories of problems—will permit the future development of an integrated systems-oriented design tool. The specification-based design model provides a meaningful evaluation principle, while templates provide the models that are needed to search for design solutions that satisfy a wide variety of integrated life-cycle goals.

Thesis Advisor: Professor Woodie Flowers
Committee Members: Professor David Marks
                                 Professor Mark Jakiela
                                 Professor David Gordon Wilson
**Acknowledgments**

Many people have made important contributions to the development of this thesis. In particular, I am grateful for the support of a New Products Program fellowship provided by Carl Chang. My thesis advisor, Woodie Flowers, and my committee members have made many insightful suggestions over the past 3 years. Barb Balents (at Wavefront Technology in California) and Eyal Zussman (at the Technical University of Berlin) have patiently heard and critiqued my ideas. Greg Goldfarb and Li Shu, graduate students also working in the area of environmental design, have shared resources with me throughout the research project.

The genetic algorithm library used in this work was written by Kazu Saito (at the department of mechanical engineering CADlab). Recently, I have begun using a genetic algorithm library written by Matthew Wall (of the MIT New Products Program). Implementation of the thesis has drawn upon the computing resources of the New Products Program (directed by Woodie Flowers) and the MIT CADlab (lab director David Gossard and associate director Mark Jakiela). Johnny Chang, an MIT undergraduate, helped with data collection for one of the examples presented in this document.

Finally, I would like to thank Kendra Borgmann for her scrupulous proofreading of the thesis.
# Table of Contents

List of Figures
List of Tables

1. **Introduction**
   1.1. Motivation
   1.2. Fueling the transition to system-oriented product design
   1.3. Thesis outline

2. **Specification-based design decision model**
   2.1. Design model hypothesis
   2.2. Development of a specification-based decision model
   2.3. Setting the Specifications
   2.4. Estimating Design Performance Data
   2.5. Suitability of the decision criterion
   2.6. Properties of the Decision Metric

3. **Comparison to other decision methods**
   3.1. Comparison to other Design Evaluation/Decision Methods
   3.1.1. Axiomatic Design
       3.1.1.1. The information content criterion
       3.1.1.2. Application to product life-cycle goal setting
       3.1.1.3. Relationship to specification-based model
   3.1.2. Qualitative measures
       3.1.2.1. Application to environmental design
       3.1.2.2. Relationship to specification-based model
   3.1.3. Figure-of-Merit
       3.1.3.1. Relationship to specification-based model
   3.1.4. Utility Analysis
       3.1.4.1. Application to environmental design
       3.1.4.2. Relationship to specification-based model
   3.1.5. Fuzzy Decision-making
       3.1.5.1. Relationship to specification-based model
   3.1.6. Pareto Sets
       3.1.6.1. Relationship to specification-based model
   3.1.7. Other Frameworks

3.2. A discussion about weights

4. **Continuous search using the specification-based decision model**
   4.1. Overview of genetic algorithms
   4.2. Search facility software architecture
   4.3. The distribution classes
       4.3.1. The delta function class
           4.3.1.1. Chromosome encoding and decoding
       4.3.2. The beta function class
           4.3.2.1. Chromosome encoding and decoding
       4.3.3. The piecewise linear distribution class
           4.3.3.1. Chromosome encoding and decoding
   4.4. The optimization class
   4.5. Objective formulation—design fitness
   4.6. Search examples
5. Catalog search using the specification-based decision model
   5.1. Catalog Organization
       5.1.1. Catalog elements
       5.1.2. Catalog hierarchies
       5.1.3. Incomplete records
   5.2. Catalog implementation
       5.2.1. The field class
       5.2.2. The record class
       5.2.3. The catalog class
   5.3. Catalog search implementation
       5.3.1. The catalog index distribution class
           5.3.1.1. Chromosome encoding and decoding
       5.3.2. The catalog selector distribution class
           5.3.2.1. Chromosome encoding and decoding
       5.3.3. Hierarchical search
   5.4. Catalog search examples
       5.4.1. Catalog search
       5.4.2. Catalog selection
       5.4.3. Search with incomplete records
       5.4.4. Hierarchical catalog search
   5.5. Summary

6. Systems-oriented CAD: specification-based analysis templates
   6.1. Existing life-cycle design tools
   6.2. The template concept
   6.3. Template Organization
       6.3.1. A graph representation for building templates
       6.3.2. Solving a template
   6.4. Template implementation
       6.4.1. Template off-diagonal class
       6.4.2. Template diagonal class
       6.4.3. Template matrix class
   6.5. Template examples
       6.5.1. A two member truss template
       6.5.2. Product life-cycle template
   6.6. Summary and discussion of the integrated use of templates and design search

7. Conclusions
   7.1. Specification-based design decision model
       7.1.1. Contributions
       7.1.2. Future Work
   7.2. Specification-based design search
       7.2.1. Contributions
       7.2.2. Future Work
   7.3. Specification-based problem templates
       7.3.1. Contributions
A Probabilistic Specification-based Design Model:
applications to design search and environmental computer-aided design

7.3.2. Future Work 190
7.4. Ongoing development of a systems-oriented CAD tool 190

Appendix A: Source code and output for the camelback optimization example 192
Appendix B: Equations for the truss design example 194
Appendix C: Equations for the policy setting example 195
Appendix D: Outline of material property catalog used in examples 196
Appendix E: Output for the catalog search example 200
Appendix F: Output for the truss material selection example 201
Appendix G: Output for the hierarchical material search example 203

References 207
### List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.1</td>
<td>The metaphor of a design evaluation tool comprised of lenses.</td>
<td>3</td>
</tr>
<tr>
<td>Figure 2.1.1</td>
<td>Example of the typical way of stating design requirements</td>
<td>9</td>
</tr>
<tr>
<td>Figure 2.2.1</td>
<td>Recycling specification as a value function</td>
<td>11</td>
</tr>
<tr>
<td>Figure 2.2.2a</td>
<td>Evaluation of a candidate design giving ( p_{acceptable} = 0.883 )</td>
<td>11</td>
</tr>
<tr>
<td>Figure 2.2.2b</td>
<td>Candidate design altered to satisfy the specification giving ( p_{acceptable} = 1.0 )</td>
<td>12</td>
</tr>
<tr>
<td>Figure 2.2.3</td>
<td>Recyclability specification with uncertainty.</td>
<td>13</td>
</tr>
<tr>
<td>Figure 2.2.4</td>
<td>Recyclability based upon material volume specification</td>
<td>14</td>
</tr>
<tr>
<td>Figure 2.2.5a</td>
<td>Technology life specification for remanufacture.</td>
<td>16</td>
</tr>
<tr>
<td>Figure 2.2.5b</td>
<td>Recoverable value specification for remanufacture.</td>
<td>16</td>
</tr>
<tr>
<td>Figure 2.2.5c</td>
<td>Price specification for remanufacture.</td>
<td>16</td>
</tr>
<tr>
<td>Figure 2.2.6</td>
<td>Feasibility of disposal specification.</td>
<td>16</td>
</tr>
<tr>
<td>Figure 2.3.1.a</td>
<td>Literal conversion of a Clean Air Act regulatory threshold into a design specification based upon strict compliance.</td>
<td>18</td>
</tr>
<tr>
<td>Figure 2.3.1.b</td>
<td>Conversion of a Clean Air Act regulatory threshold into a design specification based upon the designer's preference for compliance.</td>
<td>18</td>
</tr>
<tr>
<td>Figure 2.3.2</td>
<td>Illustration of the certainty equivalence method for a point on the emission specification.</td>
<td>19</td>
</tr>
<tr>
<td>Figure 2.3.3</td>
<td>Specifications set by a risk averse, risk neutral, and risk prone designer.</td>
<td>20</td>
</tr>
<tr>
<td>Figure 2.3.4a</td>
<td>Original specification set using lottery method.</td>
<td>22</td>
</tr>
<tr>
<td>Figure 2.3.4b</td>
<td>Approximate specification using extreme point method.</td>
<td>22</td>
</tr>
<tr>
<td>Figure 2.3.5</td>
<td>An illustration of the range over which a specification is valid based upon assumptions made when setting the specifications.</td>
<td>23</td>
</tr>
<tr>
<td>Figure 2.4.1</td>
<td>Estimate for a washing-machine design's energy consumption.</td>
<td>25</td>
</tr>
<tr>
<td>Figure 2.4.2</td>
<td>Prediction of ozone depletion in 2030 performed by different research groups if no action is taken.</td>
<td>27</td>
</tr>
<tr>
<td>Figure 2.4.3</td>
<td>Probability distribution for ozone depletion in 2030 if no action is taken.</td>
<td>27</td>
</tr>
<tr>
<td>Figure 3.1.1.1.1</td>
<td>The use of information content in a simple machine selection problem</td>
<td>39</td>
</tr>
<tr>
<td>Figure 3.1.1.2.1</td>
<td>The program interface. The left column is used to set product attributes, the middle is reserved for the design goals predicted to be most suitable, and corresponding design solution strategies or guidelines are presented in the right column.</td>
<td>41</td>
</tr>
<tr>
<td>Figure 3.1.1.2.2</td>
<td>Goals and strategies recommended by the program for the example problem.</td>
<td>42</td>
</tr>
<tr>
<td>Figure 3.1.1.2.3</td>
<td>Goals and strategies recommended for the panel modules.</td>
<td>44</td>
</tr>
<tr>
<td>Figure 3.1.2.1</td>
<td>Example of a Pugh selection chart.</td>
<td>45</td>
</tr>
<tr>
<td>Figure 3.1.2.1.1</td>
<td>Environmental assessment matrix.</td>
<td>47</td>
</tr>
<tr>
<td>Figure 3.1.2.1.2</td>
<td>Summary matrix combining the results of the four primary design evaluation matrices.</td>
<td>47</td>
</tr>
<tr>
<td>Figure 3.1.5.1.1</td>
<td>A design specification.</td>
<td>62</td>
</tr>
<tr>
<td>Figure 3.1.6.1</td>
<td>Illustration of dominance.</td>
<td>65</td>
</tr>
<tr>
<td>Figure 3.1.6.2</td>
<td>Illustration of a Pareto set.</td>
<td>65</td>
</tr>
<tr>
<td>Figure 3.1.6.3</td>
<td>Illustration of a discrete Pareto set.</td>
<td>66</td>
</tr>
<tr>
<td>Figure 3.1.6.1.1</td>
<td>Illustration of iteratively shifting a cost specification boundary to locate a Pareto optimal solution.</td>
<td>67</td>
</tr>
<tr>
<td>Figure 3.1.7.1</td>
<td>Example of a tree for a single decision for reusing a part.</td>
<td>69</td>
</tr>
</tbody>
</table>
Figure 5.2.3.1 A simplified description of the catalog class........................................ 124
Figure 5.3.1.1 Illustration of the mixed catalog and continuous variable search
problem. .............................................................................................................. 126
Figure 5.3.1.2 Outline of the catalog index distribution class. .................................. 127
Figure 5.3.1.1 Chromosome encoding for the catalog index distribution. ................. 127
Figure 5.3.2.1 Illustration of combined catalog selection and catalog search. .......... 129
Figure 5.3.2.2 Outline of the SelectCatalog class.................................................. 130
Figure 5.2.2.1 Chromosome encoding for the catalog index distribution .................. 131
Figure 5.4.1.1 Diagram of the truss stock catalog search problem. .......................... 135
Figure 5.4.1.2 Optimal truss for catalog selection using commercially available
stock sizes. ....................................................................................................... 136
Figure 5.4.1.3 Example of a well-structured and badly-structured catalog. ............... 137
Figure 5.4.1.4 Average number of generations to find the 'needle in the
haystack' as a function of catalog size. ......................................................... 139
Figure 5.4.2.1a Diagram of catalog selection problem with catalog alternatives
for truss member 1. .......................................................................................... 141
Figure 5.4.2.1b Diagram of catalog selection problem with catalog alternatives
for truss member 2. .......................................................................................... 142
Figure 5.4.3.1a Member 1 stress safety factor. ....................................................... 147
Figure 5.4.3.1b Member 2 stress safety factor. ....................................................... 147
Figure 5.4.3.1c Member 1 buckling safety factor. .................................................... 147
Figure 5.4.3.1d Truss cost. .................................................................................. 147
Figure 5.4.3.1e Truss mass. .................................................................................. 148
Figure 5.4.4.1 Hierarchical search path for the truss material section problem ......... 150
Figure 5.4.4.2a Truss cost using data describing all engineering alloys ................. 152
Figure 5.4.4.2b Truss mass using (data for all engineering alloys) ......................... 152
Figure 5.4.4.3a Truss cost using data describing all aluminum alloys ................. 152
Figure 5.4.4.3b Truss mass using (data for all aluminum alloys (p = 0.7264)) ........ 152
Figure 5.4.4.4a Truss cost using data describing all 7XXX series aluminum
alloys. ............................................................................................................ 153
Figure 5.4.4.4b Truss mass using (data for all 7XXX series aluminum alloys) ...... 153
Figure 6.2.1 General problem structure for the life-cycle of materials in a
product. ............................................................................................................ 160
Figure 6.2.2 Possible life-cycle paths for a plastic pop bottle. .................................. 161
Figure 6.3.1.1 The life-cycle template encoded using a matrix graph
representation. ............................................................................................ 163
Figure 6.3.2.1 The life-cycle template solved using data for a pop bottle. ............... 164
Figure 6.3.2.1 An example illustrating the relationship among a design, a
template, and the transfer of data when the design is evaluated
using the template. ....................................................................................... 166
Figure 6.4.1.1 Outline of the template off-diagonal class ....................................... 167
Figure 6.4.2.1 Outline of the template diagonal class ............................................. 168
Figure 6.4.3.1 Outline of the template matrix class ................................................. 169
Figure 6.5.1.1 Illustration of a template for evaluating three-element trusses ......... 172
Figure 6.5.1.2 Alternative truss template using only one diagonal element ............ 175
Figure 6.5.2.1 Catalogs used by the life-cycle prediction template ......................... 176
Figure 6.5.2.2 Schematic of the life-cycle prediction template. ............................... 177
Figure 6.6.1 Integration of templates and search into a single systems-oriented CAD tool. ........................................................................................................ 180
Figure 6.6.2a Illustration of integration for the life-cycle prediction template. ..... 181
Figure 6.6.2b Illustration of integration for the life-cycle prediction template. ..... 182
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 2.5.1</td>
<td>Positions of the members on the alternative solutions for problem two in the negotiation exercise</td>
<td>31</td>
</tr>
<tr>
<td>Table 2.5.2</td>
<td>Comparison of negotiation results to predicted results</td>
<td>32</td>
</tr>
<tr>
<td>Table 3.1.3.1</td>
<td>A scale for quantifying relative importance</td>
<td>51</td>
</tr>
<tr>
<td>Table 3.1.5.1</td>
<td>Contrasting possibility and probability</td>
<td>61</td>
</tr>
<tr>
<td>Table 4.1.1</td>
<td>Parameter values chosen for the genetic algorithm</td>
<td>80</td>
</tr>
<tr>
<td>Table 4.6.1.1</td>
<td>Design variables for the camelback minimization problem</td>
<td>96</td>
</tr>
<tr>
<td>Table 4.6.1.2</td>
<td>Performance and specification for camelback minimization test problem</td>
<td>97</td>
</tr>
<tr>
<td>Table 4.6.2.1</td>
<td>Design variables or parameters to be optimized in the truss problem</td>
<td>99</td>
</tr>
<tr>
<td>Table 4.6.2.2</td>
<td>Performance variables and specifications for the truss problem</td>
<td>100</td>
</tr>
<tr>
<td>Table 4.6.3.1</td>
<td>Performance Variables and Specifications for the policy setting example</td>
<td>108</td>
</tr>
<tr>
<td>Table 5.4.1.1</td>
<td>Design distributions for the truss stock catalog search problem</td>
<td>136</td>
</tr>
<tr>
<td>Table 5.4.2.1</td>
<td>Design distributions for the truss with catalog alternatives</td>
<td>143</td>
</tr>
<tr>
<td>Table 5.4.2.2</td>
<td>Number of single catalog searches needed to an equal the options considered in the single catalog selection search.</td>
<td>145</td>
</tr>
<tr>
<td>Table 5.4.3.1a</td>
<td>Property designation legend</td>
<td>145</td>
</tr>
<tr>
<td>Table 5.4.3.1b</td>
<td>Properties from the alloy catalog relevant to the design problem.</td>
<td>146</td>
</tr>
</tbody>
</table>
1. Introduction

1.1. Motivation

The scope of designers' responsibility is growing—they must now consider much more than product geometry alone. The pressure to design a product's entire life-cycle is being driven by consumer demand for high-quality environmentally responsible goods, regulations, and the threat of criminal liability (Fiskel 1994). A growing number of industries are now interested in accounting for life-cycle impact during product design (Sullivan and Ehrenfeld 1993). Work in design for assembly, or DFA (e.g., Boothroyd, Poli et al. 1982; Boothroyd and Dewhurst 1989), has become the prototype approach to designing various phases of a product's life.

This type of concurrent design, where many life-cycle stages are considered during the product design phase, is now expanding to include environmental considerations. As such, the initial goal of this thesis was to develop a design-for-the-environment (DFE) methodology that would allow manufacturers to proactively design for the environment (in contrast with the traditional expensive end-of-pipe control philosophy) and identify new market opportunities. However, this proved to be a very difficult task.

Environmental issues involve complex systems that do not lend to resolution through simple intuition, rules of thumb, or guidelines. Every aspect of a product's life-cycle has environmental implications. Heuristics that make environmental sense in one context may be inappropriate in another. For example, a material with the lowest energy density in one application may be a poor choice under different loading, shape or processing requirements (Ashby 1992b). General rules of thumb may lead to a good solution in one situation and a poor solution in another. Additionally, guidelines (such as how to design for recyclability) do little to help a designer decide if recycling is the best choice to reduce a product's impact—they indicate only how to design for recycling.

Further, DFE efforts are usually regarded as secondary stages to primary design tasks. A designer is bound to meet a brief\(^1\), usually under severe time and cost limitations.

\(^1\)A design brief typically describes the design problem and states the requirements that the designer must satisfy.
Sidelines from this target, such as post-design environmental stewardship programs, are not likely to get much serious attention from designers. One cannot rely upon a designer's altruism or stewardship to resolve environmental problems.

Finally, low environmental impact is only one goal of developing a product—if it were the only goal, manufacturing nothing would usually be the best choice. Guidelines are isolated and do not help designers assess environmental impact within the context of other goals. In fact, it is difficult to formulate meaningful guidelines independent of material, manufacturing, assembly, performance, obsolescence, and policy alternatives or objectives. A decision tool that conveniently permits the simultaneous consideration of many issues is needed.

This work is founded upon the conviction that, to effectively address environmental issues, environmental and regulatory requirements must be integrated with other traditional design considerations. The incorporation of environmental issues is seen as the continuation towards what I describe as systems-oriented consumer product design. In this systems design model, products are considered as part of a larger anthropological and natural ecology. They are not designed as isolated geometric artifacts for which responsibility ends at point of sale, but rather as part of an environmental and economic life-cycle (e.g., material extraction, material processing, manufacture, assembly, sales, in-use performance, obsolescence, and policy or regulations are all considered). Others, such as (Glantschnig 1993), also agree that environmental gains will require a systems approach.

Therefore, instead of formalizing DFE guidelines, the goal of this thesis is to develop a method that will facilitate the rational design of mass-produced consumer products from a system perspective. A general probabilistic design decision model is proposed and used as the basis for optimization/search and problem modeling architectures. Together, these components will form a systems-oriented product design computer tool. This thesis is about helping designers create better products given multiple systems-oriented design goals.

Ultimately, the objective is to apply the method to create a design tool that can be imagined as a series of lenses through which evolving designs may be observed. Designers will switch lenses in a seamless manner to reveal designs from alternative
perspectives (e.g., assembly, manufacture, performance, or reliability). This analogy is illustrated in figure 1.1.

If a design is viewed through an assembly lens, one will obtain assembly cost or time, while the another lens provides feedback about manufacturability. A life-cycle lens might use the product's characteristics to predict where it will be 10 years after its sale.

The goal of developing such a tool is to help designers structure and understand complex life-cycle design problems.

Using such a program, one could design for the most likely life-cycle paths. It does not make sense to add extra material to a product to make it reusable if it will probably end up in a landfill. However, if designers are to receive this new systems-design-oriented approach, analysis must be convenient, indicate the quality of available data, and have relevance to the design process.

1.2. Fueling the transition to system-oriented product design

It is my belief that life-cycle design will not become widespread until environmental performance is demanded explicitly. This conviction implies that system-oriented product design will begin with industry's perception that it must articulate and fulfill specific environmental goals. Although not central to this thesis, one might wonder: what is fueling the trend towards system-oriented design and is it just the current fashion?

Case studies by (Hinnells 1993) suggest that environmental impact is voluntarily incorporated in product development only when this coincides with other business goals. Environmentally conscious design will not arise through altruism and will not always present a win-win scenario. Therefore, the widespread shift towards environmental
system design will occur only when the rules of business survival change, through market pressure, competition, or legislation. In my opinion, the present revival of interest in environmental design may be largely attributed to panic induced by German legislation intended to resolve a solid-waste crisis.

The German Packaging Ordinance was passed in 1991 and extends industrial responsibility to include the cost of disposal or recycling of packaging materials (a discussion of the legislation and its English translation may be found in Fishbein 1994). Simply stated, the manufacturers must take back the packaging they produce. As a consequence, companies now must design the package's complete life-cycle.

In addition to packaging materials, legislation is now being proposed for products such as automobiles and electronics equipment. The notion of extended producer responsibility, or the polluter pays principle, is extremely popular with the consuming public. This principle is spreading rapidly: similar legislation has been enacted in France, Austria, and Belgium. The Canadian government is also modeling its packaging legislation after Germany's approach (Fishbein 1994, p. 10).

The positive public response reflects a growing perception that resources are finite and should be conserved. From the consumer's perspective, the approach may also imply that the manufacturer is the polluter. Regardless, the legislation promotes conservation through source reduction, reuse and recycling. This is markedly different from regulations in the United States, which tend to focus on industrial waste streams (e.g., hazardous wastes are addressed by the Resource Conservation and Recovery Act, emissions are governed under the Clean Air Act). The United States federal government has been reluctant to stipulate the environmental attributes of products directly (1992 October, p. 13). In the United States, environmental issues have typically been regarded as an end-of-pipe problem—not as part of a systems-oriented life-cycle design problem. However, support for the polluter pays principle is also mounting in the United States.

In summary, my answers to the questions posed at the beginning of this discussion are as follows. I see the trend towards system-oriented design as part of a growing conservation movement. The polluter pays legislation in Germany and consumer demand has started to change the business survival rules so that a system-oriented design philosophy is required. Certainly life-cycle design is fashionable, but provided the planet remains finite
and its population continues to grow, the underlying conservation movement which has instigated this change will persist and strengthen.

1.3. Thesis outline

The thesis is divided into two main components. First, a probabilistic specification-based decision model is developed. The model hinges upon the notion of acceptability. This type of general design decision model is believed to be a prerequisite for rational system-oriented product design. The field of decision making is relatively mature, so here the goal is to use insight about the design process and knowledge of existing decision frameworks to create a model that suits the way designers work. If the decision model is to be used in design practice, it must accommodate the nature of designing mass-produced consumer products.

Having proposed a decision model, the remainder of the thesis focuses on developing the software architecture and algorithms needed to create a versatile product-design tool. This applied component of the thesis has three distinct parts.

A formulation for specification-based optimization and search is implemented and tested. The goal is to allow designers to perform optimizations in a robust and convenient manner. Most designers are not mathematical programmers, and they should not be expected to consider the differences between single/multiple objectives or probabilistic/deterministic formulations. Next, a catalog structure is developed for use with the specification-based decision model. The search formulation is generalized to accommodate this structure to solve catalogue-selection problems. Finally, a template architecture that uses the specification-based model is proposed and implemented. Templates pre-encode the structure of specific problems so that designers can conveniently analyze and understand complex systems without modeling overhead. The templates are needed to embody the lens-metaphor computer-aided design (CAD) system. It is at the template level that the specification-based decision model can become an environmental design tool.

In chapter 2, the specification-based decision model is presented. After discussing the challenges of a decision model for consumer-product design, the hypothesis of design as a specification satisfaction process is postulated. The decision model is then presented as
an extension of this hypothesis and issues related to setting design specifications and obtaining data are entertained. Last, the rationality and mathematical properties of the model are considered.

Next, in chapter 3, a comparative review is provided for: axiomatic design, qualitative methods, figure-of-merit metrics, utility theory, efficient frontiers, and fuzzy logic. Other approaches such as cost-benefit analysis, Bayesian decision analysis, decision trees, and probabilistic design are mentioned briefly. Issues related to the use of weighting factors in multiple-criteria problems are reviewed.

Chapter 4 marks the beginning of the thesis' application component. In this chapter, the probabilistic specification-based decision model is applied to design search. First, an overview of genetic algorithms is provided, and then the search software architecture is described. After describing how the search objective function is automatically formulated, results demonstrating the optimization software performance are described.

In chapter 5 the search facility is expanded to include catalog selection problems. Catalog selection is an important part of many product design tasks. An organization for specification-based catalogs is developed. After describing how the catalog search is incorporated with the continuous variable search capabilities demonstrated in chapter 4, catalog search examples are presented. The catalog search facility provides mechanisms for proceeding when data are missing, assessing where approximate design performance estimates are adequate, and communicating where design flexibility exists.

Finally, chapter 6 presents preliminary work towards developing a general representation for encoding a wide variety of system-oriented design problems. These problem 'templates' may be used to conveniently analyze design candidates from various life-cycle perspectives. The thesis concludes by summarizing the main ideas developed in this work. The software that has been implemented provides the primary kernels needed for a systems-oriented design tool. A future work section includes some of the interfaces that have been implemented in the ongoing effort to develop a working systems-oriented CAD tool.
2. **Specification-based design decision model**

In the future, mass-produced everyday goods will be designed as part of a larger product system. In addition to considering geometry and manufacture, responsibility will extend from material extraction through disposal. As previously outlined, this change is being driven by both market and legislative forces. Therefore, the challenge is to provide non-technical consumer goods designers with the means to make complex system-oriented decisions.

Achieving system-oriented design in the consumer goods market is likely to be difficult for a number of reasons. Most design work is based upon precedence, not evaluation. Designers are accustomed to working by intuition, and if something has worked in the past it will be used again. Further, most design development resources are extremely limited—in terms of both time and money. It is not possible to commit the development resources needed for in-depth design analysis (as is available for large-scale products such as aircraft). These problems are compounded by the fact that designers are not systems or environmental experts—and it is not reasonable to expect them to be experts in such areas. Therefore a multiple criteria decision model tailored for use by designers is needed.

What does tailoring a decision model for designers involve? Foremost, the model should use a familiar design language. Designers must be able to understand the model, as they will not blindly follow results from a black box. The decision model should be easy to use. If the approach is cumbersome or difficult to understand (rendering results difficult to believe) it will be rejected by practicing designers. Designers should be able to rapidly explore the implications of candidate solutions by themselves—if specialized decision analysts are required to model the problem, interactive feedback is not likely. At the same time, it is not reasonable to sacrifice rationality for the sake of simplicity.

Evaluations using the model must provide meaningful results that have some relevance to the design brief. It would be hard for designers to profit from an analysis (including an environmental impact assessment) unless the results can be related to some goal or requirement that they are trying to achieve.
Further, the decision model should readily accommodate the nature of design problems. Design often involves uncertainty in both estimating performance and in setting appropriate specifications. Incomplete and/or poorly understood data are common, particularly when addressing environmental issues. This complication has severely restricted the use of life-cycle analysis as a design tool.

Group problem solving is also the norm. It must be possible to reflect different or even conflicting opinions in the decision model. Additionally, designers may need to simultaneously consider a very wide range of issues, such as design performance, regulatory requirements and corporate policy/culture.

Finally, using the model should help the designer. When tackling a complex problem, an evaluation that simply indicates poor performance may not be sufficient. The designer wants to know how to improve designs and choosing appropriate courses of action may not be obvious. The formulation should facilitate the use of search and optimization to suggest better alternatives.

In the following sub-sections, a decision model that, in application, exhibits these properties is presented. First, an underlying assumption about the nature of design is made. Then, based upon this assumption, a specification-based decision model is introduced through example. The chapter concludes by elaborating upon: how design specifications may be set; how data for the model may be obtained; the rationality of the model; and the method's mathematical properties.

2.1. Design model hypothesis

The goal of this section is to forward a hypothesis about the assessment of candidate designs, and thus infer the objectives of any design activity.
A design task usually begins with a design brief. This brief may be extremely specific or perhaps quite vague. Regardless, it will invariably specify some target goals or thresholds. These targets are referred to as design requirements or design specifications. Figure 2.1.1 shows a cost requirement. The requirement indicates that if the design can be manufactured below a certain cost, it will be considered acceptable. Outside of this range the design will be unacceptable, since it fails to meet the requirement.

Ultimately, most design requirements—tolerances, performance specifications, and regulations—are defined in this manner. (Fiskel 1994) identifies similar specifications for environmental goals related to waste reduction, recycling, life-cycle cost reduction, ownership cost reduction, energy consumption, and recycled content.

Therefore, given that designs are evaluated by requirements or specifications, I will suggest that the objective of design activity is to meet design requirements. Further, it usually does not pay for the designer to exceed specifications. The objective is simply to satisfy the requirements. This statement does not imply that mediocre designs are adequate. It simply implies that the designer's mandate is to meet specifications—specifications may be set to require excellence and the minimally acceptable standard.

It is my hypothesis that design is a specification-satisfaction process. In the following section this hypothesis is used to develop a general specification-based decision mechanism that models design as a satisfaction process.
2.2. Development of a specification-based decision model

Let us develop the decision model through an example. Discussion of relevant literature is postponed until the following chapter. The example presented is hypothetical and simplified to convey the essence of the approach.

A multiple criteria decision has two distinct components. First one must assess the individual value of design attributes. Then, a decision rule is needed to aggregate the individual attributes values into an overall design metric.

We begin with a single-criterion problem that highlights the treatment of design specifications as *value functions* (valuing individual design attributes) Imagine that a company, as a marketing strategy, has decided that any new products it develops must be recycled when obsolete. The company has defined the criterion for recyclability as below.

\[
\frac{\text{expected material recycling cost}}{\text{expected virgin material cost}} \leq 1.0
\]

Under this policy, any product concept that meets this criterion is accepted and those that do not are rejected. Thus, the recyclability specification could also be represented as the value function shown in figure 2.2.1.
Value is defined as the probability that a product will be deemed acceptable. Acceptability is defined by design specifications. In figure 2.2.1, each point on the specification indicates the likelihood that the corresponding recycling/virgin cost ratio is satisfactory. For example, a product with a recycling/virgin cost ratio of 0.85 will be accepted with a probability of 1.0. A design with a recycling/virgin cost ratio of 1.1 has a zero probability of being acceptable.

Figure 2.2.1 Recycling specification as a value function.

Consider a candidate design with the expected performance probability density function illustrated in figure 2.2.2a (note that the vertical scale pertains to the specification only and probability density function \( p(x) \) is scaled to have a unit area). The probability that the design will be acceptable according to the recyclability specification is given by equation (2.2.1).

\[
\text{value} = p_{\text{acceptable}} = \int s(x)p(x)\,dx \quad (2.2.1)
\]

Where,

\( s(x) = \text{specification function defining probability that performance levels are acceptable} \)

\( p(x) = \text{probability density function for a design candidate's performance (all performance distributions are normalized to have a unit area under the curve when this calculation is made).} \)

Note that the vertical scale in the figure pertains to the specification only.

Figure 2.2.2a Evaluation of a candidate design giving \( p_{\text{acceptable}} = 0.883 \) (vertical axis pertains to specification only).
The acceptability of a design attribute is the product of the probability that a given performance level is acceptable and the probability that the design will perform at the given level. In this example, the designer cannot be sure that the candidate design will meet the recycling requirement \((p = 0.883)\). The designer would try to modify the attribute to increase its value (probability that the design is acceptable as defined by specifications). A perfectly acceptable design attribute is shown in figure 2.2.2b. Thus, we can see that design is modeled as a satisfaction problem.

In summary, to this point we have suggested that: 1) A design specification defines the likelihood that a certain product performance attribute will be deemed acceptable; and 2) the value of a design attribute is the probability that it will be judged as acceptable (defined by its specification).

In this case, the recycling policy was stated in a binary regulatory fashion (a design is either within the regulation and is acceptable, or it is outside and hence unacceptable). In practice, there may be considerable disagreement about what the recyclability specification should be.

Uncertainty about the level of performance that will be judged as acceptable may have significant impact upon a decision. It would be useful to capture this uncertainty so that designs marginally outside of the artificially precise specification are not worthless.
Imagine that the specification is being set by a team, and the boundary that defines an acceptable recycling/virgin material cost is unclear. An example of the specification that might result is evaluated in figure 2.2.3 for the same design candidate shown in figure 2.2.2a. A voting process could be used to establish a specification that reflects the different opinions expressed by the team members. One would begin by bounding the specification.

Figure 2.2.3 Recyclability specification with uncertainty. Uncertainty about specifications is important—$p_{\text{Acceptable}} = 0.971$ (compare to the evaluation in figure 2.2.2a).

One boundary would indicate the ratio that all team members find perfectly acceptable for recycling (probability design is acceptable = 1.0 for cost ratio = 0.8). The other boundary is the point at which there is consensus that, regardless of other design characteristics, the design must be rejected on the basis of recyclability (probability design is acceptable = 0.0 for cost ratio = 2.0).

Probabilities for intermediate cost ratios can be obtained using a voting scheme. For instance, if 8 members of the team find a particular cost ratio acceptable and 2 reject this ratio, then the probability of acceptance would be 0.8 (ten members total). The specification value functions may also be constructed using established lottery methods based upon certainty equivalence (see Keeney and Raiffa 1976, pp. 143-203 for example). The lottery method is also discussed in the following section. When a design candidate is evaluated against the specification using equation 2.2.1, the value measures the probability of obtaining an acceptable level of performance (in this example a recyclable product).

Two additional ideas have now been emphasized. First, the specification functions are measured on an absolute acceptability (value) scale. Zero acceptability implies that the
design is unsuitable regardless of other characteristics or specifications. A fundamental assumption is that at some extreme level, any specification can eliminate a candidate design (the advantages of this absolute scale will become apparent later). Second, it is often difficult to set specifications in a binary manner, and uncertainty or disagreement about what is required may play an important role in a design decision. Lottery and consensus-based (voting) techniques can be used to estimate point probabilities along a specification function.

Thus far we have addressed a single criterion only. Now, the decision rule that is used to evaluate the overall value of a multiple attribute design is considered.

A second independent criterion will be added to the assessment of recyclability. This criterion is intended to ensure that the product volume is sufficient to justify recycling efforts. The specification in figure 2.2.4 would be determined in a fashion similar to the previous example. Again, the specification is an absolute measure and is evaluated under the assumption that the design is acceptable in all other areas.

Therefore, any design with a volume of less than 500,000 monetary units per year would be considered unacceptable regardless of the material cost ratio. The probability that a design will be acceptable based upon volume would, like the previous example, be evaluated using equation (2.2.1).

The overall value of the design alternative is given by the probability that the design is acceptable according to all design specifications. This probability is given by
\[ P_{\text{acceptable by all specifications}} = \prod_{i=1}^{n} p_i \quad (2.2.2) \]

Where

\( n = \text{number of design specifications or criteria} \)

and

\( p_i = \text{probability of being acceptable as defined by the } i^{\text{th}} \text{ specification}. \)

Hence, for our simple multiple-criteria example, the overall value of the candidate design alternative is

\[ P_{\text{acceptable by all specifications}} = P_{\text{acceptable by cost ratio}} \times P_{\text{acceptable by volume}} = 0.971 \times 1.0 = 0.971 \]

To continue, we shall now modify the initial premise of the example. Instead of working under a recycling mandate, the designer wishes, at the onset of a project, to determine what type of obsolescence path best suits the product. This would allow the designer to work towards an appropriate obsolescence goal (for example, it does not make sense to design a remanufacturable product if it will probably be thrown away). Three obsolescence options are to be considered: recycling, remanufacture and disposal.

To assess recycling feasibility, the criteria and product characteristics in figures 2.2.3 and 2.2.4 will be used. Hypothetical feasibility criteria for the two new options are evaluated against expected product characteristics in figures 2.2.5 and 2.2.6, giving

\[ P_{\text{suitable for recycling}} = 0.971 \times 1.0 = 0.971 \]
\[ P_{\text{suitable for remanufacture}} = 0.533 \times 0.876 \times 1.0 = 0.467 \]
\[ P_{\text{suitable for disposal}} = 0.904 \]

Therefore, we would conclude that recycling is the most appropriate obsolescence path, and continue to design the product accordingly. The reason for presenting this example, however, is to illustrate that the absolute scale (probability of satisfying specifications) allows us to compare alternatives that are fundamentally different. Both the number and types of criteria vary for the different obsolescence paths. The specification-based decision model is not limited options that must be evaluated using identical criteria.
In summary, the main concepts introduced in this section are: design is viewed as a specification-based satisfaction problem; the value of a design is the probability that the design will be acceptable according to all specifications; specifications are value functions that indicate the probability that products exhibiting specific performance levels will be deemed acceptable; and acceptability is defined on an absolute scale. At some
A Probabilistic Specification-based Design Model: applications to design search and environmental computer-aided design

extreme level, failing to meet a single specification will invalidate the design alternative. It is also hoped that the example has shown that the method is fairly intuitive and uses familiar design language.

2.3. Setting the Specifications

The skeptical designer might raise a number of questions about using the decision model presented in the previous section. Foremost, a design evaluation is completely contingent upon the design specifications. How does one know that the "right" specifications are being used? The obvious answer to this question is that, like most design tasks, there is no way to guarantee that the correct problem is being solved. In fact, many designers will contend that the most difficult part of a design problem is to decide what the specifications should be. In other words, specification setting is a design problem in itself. Thus, it will follow the same evolutionary develop, test and modify pattern that is observed in other design activities.

This section will provide methods that can be used to estimate design requirements, and present the notion that the specifications are not static and will iteratively evolve as the designer's understanding grows. Minding the notion that specifications are not cast in stone, let us consider a number of ways that design requirements might be established.

Specifications derived from regulatory requirements are probably the most straightforward, as regulations are usually defined in terms of maximum acceptable thresholds. For example, the Clean Air Act requirement for auto emissions (Pytte 1990; Lee 1991) states that NOX must not exceed 0.4 grams/mile. Figure 2.3.1a shows a literal translation of this regulation, indicating any emission level below the limit is completely acceptable. In other words, the design is required to strictly comply only to the regulation. Figure 2.3.1b shows a design specification where the designer would like to surpass the Clean Air Act, but might also tolerate a failure to comply.
While the literal translation of the binary emission threshold is obvious, how the interpretation in figure 2.3.1b might be established is less clear.

There is a number of methods, both individual and group oriented, that can be used to set specifications that are not clearly delineated. As alluded to in the previous section, one way to establish an individual's preference or value function is known as the certainty equivalence or lottery method. The approach was proposed by (vonNeumann and Morgenstern 1947, pp. 15-31) and is explained in detail by (Keeney and Raiffa 1976, pp. 143-203).

The certainty equivalent of a lottery $L$ is defined as the certain amount $\hat{x}$ at which the designer has no preference between the uncertain lottery $L$ and $\hat{x}$. That is,

$$\text{value}(\hat{x}) = \text{expected}[\text{value}(L)]$$  \hspace{1cm} (2.3.1)

where

$$\text{expected}[\text{value}(L)] = \sum_{i=1}^{n} p_i \times \text{value}(x_i)$$  \hspace{1cm} (2.3.2)

given the lottery $L$ yields $n$ consequences $x_1, x_2, x_3, \ldots x_n$ with the probability of each outcome occurring being $p_1, p_2, p_3, \ldots p_n$. 

**Figure 2.3.1.a** Literal conversion of a Clean Air Act regulatory threshold into a design specification based upon strict compliance.

**Figure 2.3.1.b** Conversion of a Clean Air Act regulatory threshold into a design specification based upon the designer’s preference for compliance.
To clarify how certainty equivalence is applied, we will set a point on the example NOx emission specification using this approach. One would begin by delineating the level of emissions below which the design will be completely acceptable, and the level of emissions above which the design will be unacceptable. In figure 2.3.1b, these levels were set at 0.2 grams/mile and 0.6 grams/mile. Next, the designer would use a lottery to determine the points on the specification that lie between the acceptable and unacceptable extremes.

\[
\begin{array}{c|c}
\text{Certain Outcome} & \text{Uncertain Lottery} \\
\hline
\text{Emission 0.4 g/mile} & \text{Emission 0.2 g/mile} \\
\hline
p & (1-p) \\
\hline
& \text{Emission 0.6 g/mile}
\end{array}
\]

*Figure 2.3.2 Illustration of the certainty equivalence method for a point on the emission specification (figure 2.3.1b).*

The uncertain lottery alternative has a chance \( p \) of being perfectly acceptable and \( (1-p) \) of being unacceptable. The probability \( p \) (0.8 in figure 2.3.1b) at which the designer is indifferent to the two options is the likelihood that the designer will accept a 0.4 grams/mile emission level.

The certainty equivalence definition can also be used to show that the shape of the specification function reveals the designer's risk-taking attitudes. Risk taking is discussed here only as a point of interest. A more detailed development of the issue can be found in (Keeney and Raiffa 1976, pp. 143-203) or (French 1988, pp. 175-182).

From equation (2.3.2),

\[
\text{expected}[\text{value}(L)] = \sum_{i=1}^{n} p_i \times \text{value}(x_i)
\]

it is noted that the expected value of \( L \) may be different than the value of the expected outcome of \( L \), as defined in equation (2.3.3).
\[ \text{value}[\text{expected}(L)] = \text{value}\left[ \sum_{i=1}^{n} p_i \times x_i \right] \]  
(2.3.3)

A risk-averse designer is one that will define specifications such that

\[ \text{value}[\text{expected}(L)] > \text{expected}[\text{value}(L)] \]  
(2.3.4)

Risk aversion implies that the designer will find a certain or guaranteed outcome \( \hat{x} \) more acceptable than an uncertain lottery that has the same expected outcome. This indicates that the designer places value on avoiding uncertain outcomes (risk). For the lottery used to set the point 0.4 grams/mile in the emission specification example, we obtain

\[
\begin{align*}
\text{value}[\text{expected}(L)] &= \text{value}\left[ (p)x_1 + (1-p)x_2 \right] \\
&= \text{value}\left[ 0.8 \times 0.2 \frac{g}{\text{mile}} + 0.2 \times 0.6 \frac{g}{\text{mile}} \right] \\
&= \text{value}[0.28] \\
&= 0.92 \\
\text{expected}[\text{value}(L)] &= p \times \text{value}(x_1) + (1-p) \times \text{value}(x_2) \\
&= 0.8 \times \text{value}\left[ 0.2 \frac{g}{\text{mile}} \right] + 0.2 \times \text{value}\left[ 0.6 \frac{g}{\text{mile}} \right] \\
&= 0.8 \times 1.0 + 0.2 \times 0 \\
&= 0.8
\end{align*}
\]

In this case, the value of the expected lottery outcome is greater than the expected value of the lottery—the designer is risk adverse. It can be shown that, depending upon the risk-taking characteristics of the designer, the specifications will take on the forms illustrated in figure 2.3.3.

![Figure 2.3.3 Specifications set by a risk averse, risk neutral, and risk prone designer.](image)

- **Risk Averse**: Preferences are such that the designer is willing to accept a certain outcome, even if it is less than the expected value of a lottery.
- **Risk Neutral**: Preferences are such that the designer is indifferent between the certain outcome and the lottery.
- **Risk Prone**: Preferences are such that the designer is willing to accept a lottery, even if it is less than the expected value of a certain outcome.
Through personal experience, I have always felt that product design practice, above all, involved achieving required goals while trying to avoid the possibility of a design that fails late in the development cycle. Thus, known solutions tend to be favored even when novel but untried solutions offer the potential of a larger payoff. Given this conservative environment, it comes as no surprise that I cannot recall seeing (in the literature) a single design application where value functions exhibit a risk-prone form. Curiously, this apparent desire to avoid uncertainty in design is paradoxical to the popular stereotype of good designers being risk takers. Perhaps exploration and innovation require risk taking, but ultimately the design and development of successful products require risk avoidance.

Certainty equivalence provides a very elegant method for establishing the form of an individual designer's specifications. However, one might seriously question whether most practicing designers would be willing to perform (or understand) such lotteries. Fortunately, if one assumes that the evolution of specifications is an inevitable and essential part of design (Otto and Antonsson 1991, assert that priorities change throughout the designing process), determining the precise detailed form of design specifications should not be critical. This is particularly true when design is viewed as a specification satisfaction problem. In essence, no design is finished until it is completely acceptable according to the specifications—even if this means reconciling one's expectations (specifications) in order to make a problem satisfiable. Thus, specifications which simply drive the solution in the correct direction are probably adequate.

As such, a simplified procedure is recommended. The designers need to estimate only the extreme points at which they will be either completely satisfied or reject a design. This estimation technique will be referred to as the extreme-point method. The application of this method is contrasted with the lottery-based emission specification in figure 2.3.4. This approach has the advantages of being intuitive, straight-forward, and does not ask the designer to think in unfamiliar terms.
The methods recommended thus far in this section address setting an individual's design specifications. In many instances design specifications must be set by a group. In section 2.2, a voting process to capture the participatory group's views was described. Additionally, one could use a statistical approach based upon surveys to estimate the acceptability of various levels of design performance. This type of approach could be used to identify market opportunities. For example, one might survey the characteristics of products now recycled to generate specifications to identify other products that are good candidates to be recycled.

To close this section, let us question the validity of the specification/value functions. Over what range of performances are the specifications valid? Each individual specification is set assuming that all other aspects of the design are satisfactory. Therefore, the specification value function can be said to be valid only provided all other specifications are met. Consider a design problem with two performance variables. The specification for variable 1 is established assuming that variable 2 is in its acceptable range. This idea is illustrated in figure 2.3.5.
Figure 2.3.5 An illustration of the range over which a specification is valid based upon assumptions made when setting the specifications.

One cannot be sure if specification 1 will still be valid when outside the perfectly acceptable range of variable 2. One way to resolve this dilemma is to assume that the specifications still apply beyond the conditions assumed when they are defined. After all, the design should ultimately reside in the acceptable region. This approach is reasonable provided an attitude is taken that specifications are transient and designing the specifications is part of solving the problem. Using this approach, the designer would set specifications and evaluate the design alternatives. If there are no alternatives that are perfectly acceptable, the designer may want to reconsider given this additional information. This may result in design alteration or revision of the specifications.

Another approach would be to modulate the specifications as functions of the acceptability of other performance variables. This approach requires the designer to define a specification function that will always have the appropriate conditional form. In
practice, this is probably not a simple or feasible task. The iterative approach previously suggested seems more plausible.

Finally, a third way to resolve this issue might be proposed—using weighting factors to scale the specification functions over different operating points in the space. Section 3.2 will illustrate that, when specifications are defined on an absolute acceptability scale, it is not rational to use weighting factors.

2.4. Estimating Design Performance Data

Although there is a growing consensus that environmental issues are important, the consideration of environmental impacts during design is uncommon (Jovane, Alting et al. 1993). This surprising fact may be attributed to problems related to obtaining the data necessary to perform life-cycle analysis (LCA)—a methodology for understanding environmental impacts associated with all phases of a product's life, ranging from raw material extraction through disposal (Hunt, Sellers et al. 1992; Alting and Jorgensen 1993; Vigon, Tolle et al. 1993).

Gathering data for LCA is extremely time consuming and laborious (1993; Weule 1993) and in the end, the results may be arguable. For example, two independent studies comparing disposable and cloth diapers came to very different conclusions—cloth uses 300% more energy than disposables, and disposables use 70% more energy than cloth (Stipp 1991). (A third group suggests that cloth uses 30% more energy that disposables(1992)). Further, a detailed tally of impacts (such as kg CO₂) obtained from LCA have little meaning to a design team (Alting and Jorgensen 1993). In many instances, estimation may be more appropriate.

Therefore, in addition to setting specifications, one might also wonder about estimating the performance data needed to evaluate a design. It is assumed that most designers will not want to spend the majority of their time gathering data to perform an evaluation. As mentioned above, data availability and quality are especially problematic when addressing large systems such as the environment.

These data-related problems, although pronounced in environmental analysis, arise in almost any quantitative design concept analysis. In this section I will suggest how the
probabilistic representation of design variables can be used to resolve these difficulties. If we genuinely expect designers to use the specification-based model, the proposed method must avoid unwarranted exhaustiveness and produce credible results (1993).

The first issue raised was that exhaustive data gathering is extremely laborious and time consuming. Life-cycle analysis simply takes too long and is too cumbersome to be useful during the design cycle. The probabilistic representation addresses the data-completeness problem by allowing analysis to proceed using incomplete data or approximations. It is felt that, provided the data quality is automatically reflected in analysis results, the method is a good way to simplify analysis. Further, the probabilistic analysis will help to identify problem areas in which more data may be required.

The bandwidth of a distribution can be used to reflect data quality. In general, poorly quantified systems will tend to result in smooth broad distributions. Distributions for a poorly understood system would typically look like the figure 2.4.1. This uniform distribution might be a designer's initial guess to bound a design performance characteristic.

Imagine that we are interested in the energy consumption of a washing-machine design. A precise evaluation of this figure would require a fairly detailed analysis. In lieu of such an analysis, the designer would typically make a single-point energy-consumption estimate, which is deceptively precise in appearance. Using a probability density function, one can make an educated guess using an appropriately wide bandwidth. In figure 2.4.1, it is assumed that the design would perform somewhere between the best and worst machines on the market (1992). If these data were used to evaluate the design against a specification, the wide bandwidth would probably result in a low confidence that the design will be acceptable. This scenario is illustrated in figure 2.4.1.

This approach for approximating data embraces the school of subjective probability. The probability density function represents the designer's belief that the candidate alternative will perform at various levels (for a discussion about subjective probability, see French
1988). In this example, the designer has a poor understanding of the machine's energy consumption. Therefore it would be risky, based upon current knowledge, to adopt the design if the specification in figure 2.4.1 is to be met. The low probability of an acceptable design indicates a need to improve data (understanding) in this area, alter the product, or accept the risk of proceeding with a potentially unacceptable design. In contrast, if this gross approximation had yielded a high confidence of meeting specifications, the designers would know that their estimate (knowledge) is adequate.

In the limit, if the designers believes they have absolutely no understanding of a system whatsoever, their performance estimate would be a uniform probability density function with infinite range and zero height (to have an area of 1). This statement seems deceptively similar to Laplace's argument for the principle of insufficient reason (Laplace 1917)—knowing nothing at all about the true state of nature is the same as all states having equal probability (paraphrased from French 1988, p.38). As (French 1988, pp. 217-219) explains, this definition can lead to paradoxical situations under classical probability. However, if we view our estimates as subjective or personal probabilities, it is acceptable for one designer's performance estimates to differ from another's.

The second issue raised was data credibility, as exemplified by the diaper studies. Probabilistic performance estimations help resolve the credibility issue in two ways. First, one is not interested in producing a single number for design performance. Instead, the probability density function captures all possible evaluation results. Using a range of likely outcomes should diffuse disagreement by acknowledging uncertainty. In the case of the diapers, all three studies are probably within the realm of the probability density function for life-cycle energy consumption. Of course, the question is: which of the results is more probable?

A more difficult credibility issue arises when there is widespread disagreement over the outcome of an analysis. Ozone depletion levels based upon CFC emissions is a good example of this type of situation (see figure 2.4.2). In this case, lack of data is not a problem. Rather, different research groups are in wide disagreement. Typically, this type of situation would force the designer to choose one prediction over another. Unfortunately, differences in opinion about whose data to believe can lead to controversy.
A statistical or experimental approach to performance estimation can be used to generate a probability density function that includes disparate results. This approach has been used for ozone depletion in figure 2.4.3. The area of each group's predictions has been normalized to make equal contributions to the density function. In general, broad distributions with jagged shapes indicate disparate data. This often occurs for complex multivariable systems.

We assume that more frequently predicted outcomes are more likely to be correct, but that all of the predictions have validity. This approach is consistent with a frequentist school of probability, which believes that if enough identical trials (or studies) are performed the probability distribution will emerge (French 1988, pp. 220-222).

During informal discussions with colleagues, two concerns about this approach have been expressed: that the majority of the predictions may be incorrect; and that extreme predictions may be undervalued.

The first point can be addressed using the tenets of science—general acceptance is necessary for work to be recognized as correct. Peer review/consensus is the measure of research. Ostensibly this is based upon the belief that, amongst competent scientists, the candidate hypothesis with the highest popular support has the best chance of being correct. The approach suggested here is consistent with this philosophy.

(It should be mentioned that this philosophy runs into difficulty when the mainstream of science is incorrect. Initially, new paradigms of thought are rejected, but if the data are
convincing, the mainstream will indeed shift over time and accept fundamentally new hypotheses.)

The other possible objection is that extreme estimates will be undervalued. In fact, this approach helps to ensure that all predictions are valued equally. This counters a natural tendency to over-emphasize anomalous or extreme data. People notice and remember unusual occurrences—perhaps because extraordinary events garner our attention (Goldstein 1984). Therefore, it is important that all of the different predictions are equally represented—the designer does not have to choose one prediction in favor of another. This should help prevent the distortions that can occur when decisions are based upon data produced by a single research group.

The final data related issue is that the results are often not useful in a design context. Comparing design performance estimates to the specifications (as is done using the specification-based decision model) gives the designer relevant feedback. For example, an estimated level of life-cycle energy consumption has little meaning if the design brief does not provide target value. In short, designers are concerned with designing products that meet specified design requirements. This point is important, as it implies that the impetus for environmentally responsible design does not come from the designer or the ability to quantitatively compute the impact of a product. Rather, sustainable design will arise only through changing the demands made upon our products.

Therefore, it appears that the specification-based model addresses many of the stumbling blocks that can prevent the analysis of complex systems. However, two additional points should be made before closing this discussion. One point is that certainty also depends upon time. When trying to predict performance levels it becomes more and more difficult as the length of the prediction time grows (for example, there is much more uncertainty estimating recycling costs 15 years from now as opposed to one year from now. It should also be possible to define distributions as a function of time. For example, the bandwidth of a recycling cost distribution might grow over time, while the expected value of washing machine energy consumption could decrease.

The last point pertains to the nature of design and structuring design data. During early conceptual design stages, many details are undefined. Any analysis should be performed at a similar level of resolution. When the design has progressed, more detailed analysis
may also be in order. Analysis data should be structured to facilitate this behavior. This type of structure in not intrinsically part of the decision model, but rather part of a design tools architecture. This architecture is described in chapter 5.

2.5. Suitability of the decision criterion

Since designers are familiar with the notion of working to meet specifications, the specification-based decision model seems reasonable. Defining specifications as value functions that indicate the probability a given level of performance will be acceptable also seems fairly intuitive—the designer must simply consider what levels of performance they will be satisfied with. The formulation also provides a convenient way to include regulatory or policy requirements in a design evaluation.

The proposed model is based upon the hypothesis that the goal of design activity is to develop an acceptable or satisfactory product as defined by specifications. The higher the overall probability of being acceptable (as defined by all the specifications), the better the design.

This decision metric has two distinct components. First, the value of a single attribute is the probability it is acceptable according to its specification. This subjective probability is computed by first calculating the likelihood that the design will be acceptable according to each requirement.

Then, we assume that the overall value of a design with many attributes is the probability that the design will be acceptable according to all specifications. In the remainder of this section it is this overall combination metric that is the focus of the discussion.

Is the assumption that the preferred design will have the highest probability of being acceptable according to all specifications reasonable? The designer might wish to adopt an aggressive or conservative selection strategy (Otto and Antonsson 1991)—there are any number of criteria that might be used to choose the overall best design. For example, maximin (Wald 1950), minimax regret (Savage 1951), and maximum expected-value (Laplace 1917) are well known selection strategies (these criteria and others are concisely described on pp. 36-39 of French 1988).
The maximin return criterion states that, given a number of alternatives and a number of possible scenarios, the preferred alternative will have the best worst outcome under the different scenarios. The minimax regret criterion suggests that the decision maker should calculate how much worse an alternative is compared to the best possible outcome for each scenario. This quantity is known as regret. The preferred alternative will have the smallest maximum regret over the scenarios. The expected-value criterion states that the alternative with the highest expected-value over the possible scenarios should be chosen as the best option.

The point of this section is not to discredit or refute other decision criteria, but to question whether the proposed criterion is reasonable. Thus, it would be fun to ask: how do "best designs" selected using different metrics compare to actual best designs chosen by a selection committee? This question was explored using a negotiation exercise (Arroyo 1987) from the class 11.363J, Chemicals in the environment: policy and management, which was taught by Connie Ozowa in 1993.

The goal of this exercise is to choose policy solutions for five issues related to siting a hazardous waste disposal site. For each issue, there are five possible alternative solutions. In design language, the exercise is analogous to five separate design problems, where each problem has five different concept solutions to choose among. Eight decision committee members with different concerns must negotiate and select an acceptable solution for each of the five problems. Members are assigned their preferences for the solution alternatives. For example, on problem 2 committee member 3 finds solutions A and B completely unacceptable, C is a second choice, D is the first choice, and concept E is the third choice. Members are informed of their own preferences only.

Given that the preference structure for each individual is assigned, it is relatively easy for the author to estimate a subjective probability for a member's likelihood to accept a given solution alternative. A first choice is assigned a probability of 1, a second choice 1/2, a third choice 1/3, and unacceptable options are assigned a zero probability of being acceptable. An example of these data are provided for problem 2 in table 2.5.1.
Table 2.5.1 Positions of the members on the alternative solutions for problem 2 in the negotiation exercise.

<table>
<thead>
<tr>
<th>Alternative</th>
<th>Member 1</th>
<th>Member 2</th>
<th>Member 3</th>
<th>Member 4</th>
<th>Member 5</th>
<th>Member 6</th>
<th>Member 7</th>
<th>Member 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1/5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>1/2</td>
<td>0</td>
<td>0</td>
<td>1/4</td>
<td>0</td>
<td>0</td>
<td>1/2</td>
<td>1/2</td>
</tr>
<tr>
<td>C</td>
<td>1/3</td>
<td>1/3</td>
<td>1/2</td>
<td>1/3</td>
<td>1/3</td>
<td>1/2</td>
<td>1/3</td>
<td>1/3</td>
</tr>
<tr>
<td>D</td>
<td>1/4</td>
<td>1/2</td>
<td>1</td>
<td>1</td>
<td>1/2</td>
<td>1</td>
<td>0</td>
<td>1/4</td>
</tr>
<tr>
<td>E</td>
<td>1/5</td>
<td>1</td>
<td>1/3</td>
<td>1/2</td>
<td>1</td>
<td>1/3</td>
<td>0</td>
<td>1/5</td>
</tr>
</tbody>
</table>

These data were used to compare the actual negotiation results to predictions using the expected value criterion and the overall probability of acceptance criterion.

The overall probability of satisfaction criterion used in the specification-base decision model states that the best alternative is chosen by

$$\max_{i = \text{alternative A}} \left\{ \prod_{j = \text{member 1}}^{8} p_{ij} \right\}$$  \hspace{1cm} (2.5.1)

The expected value criterion suggests that the best option will be given by

$$\max_{i = \text{alternative A}} \left\{ \sum_{j = \text{member 1}}^{8} \frac{1}{8} p_{ij} \right\}$$  \hspace{1cm} (2.5.2)

From equation (2.5.2) it is apparent that the expected-value criterion is a special case of the weighted-sum metrics that are extremely popular in engineering design. This approach, known as a figure-of-merit, is discussed later in section 3.1.3, so for the present example it is sufficient to state that equation (2.5.2) assumes each member is equally important. Based upon observation of the actual negotiation, this assumption appears to be valid for problems 1, 2 and 3. Time constraints on problems 4 and 5 precipitated a power-based negotiation dynamic that requires the use of weighting factors in the formulation. Weighting issues are addressed in section 3.2.

The actual negotiation exercise results for problems 1 through 3 are compared to the results predicted using the two criteria in table 2.5.2.
Table 2.5.2 Comparison of negotiation results to predicted results

<table>
<thead>
<tr>
<th>Issue/Problem</th>
<th>Chosen Solution Alternative</th>
<th>Expected Value Prediction</th>
<th>Overall probability Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>2</td>
<td>C</td>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td>3</td>
<td>B</td>
<td>D</td>
<td>B</td>
</tr>
</tbody>
</table>

In this simple experiment, the overall probability of acceptance criterion predicted the actual alternatives chosen, while the expected-value metric had mixed results. Although this test does not prove anything conclusively, it supports the belief that the overall probability of satisfying specifications is a reasonable concept selection metric. Further experimentation is in order.

In summary, three points are made in support of the specification satisfaction metric as a decision criterion. First, the criterion requires modeling a problem in terms of a language designers are familiar with. They must simply question what levels of performance they will accept. Second, designs are already evaluated against specifications, and the representation allows the convenient incorporation of regulations and policy. Finally, the negotiation test serves as preliminary support for the criterion's ability to emulate a rational decision team.

2.6. **Properties of the Decision Metric**

To conclude the specification-based model's introduction, I will consider the properties of the decision criterion. First, mathematical characteristics are presented in a style patterned after work in (Otto and Antonsson 1991; Otto and Antonsson 1993a). Most of these properties are somewhat self-evident. Then independence related issues, which may be a cause for confusion, are discussed.

The specification-based design metric defines the value of a design as

\[
P_{(p_1, \ldots, p_n)} = \left\{ \prod_{i=1}^{n} p_i \right\}
\]

(2.6.1)

where \( P_{(p_1, \ldots, p_n)} \) is the overall probability that the design will be acceptable. \( p_i \) is the probability that the concept is acceptable according to the \( i^{th} \) of \( n \) requirements.
The boundary conditions of this metric are stated in equation (2.6.2).

\[ P_{(0,\ldots,0)} = 0, \quad P_{(1,\ldots,1)} = 1 \]  \hspace{1cm} (2.6.2)

The lower boundary condition implies that if all aspects of the design are unacceptable, then the overall design is also completely unacceptable. Similarly, the upper condition means that if a design is perfectly acceptable for each requirement, the overall design is perfectly acceptable.

The criterion also has the property of monotonicity, as defined in equation (2.6.3)

\[ P_{(p_1,\ldots,p_k,\ldots,p_n)} \leq P_{(p_1,\ldots,p_{k'},\ldots,p_n)} \iff p_k \leq p_{k'} \]  \hspace{1cm} (2.6.3)

The interpretation of this property is straightforward. If the probability of meeting a single design requirement changes, the overall acceptability of the design will change in the same direction. When the probability of meeting a specification goes down, the overall probability of acceptance also goes down. This property does not require that the individual specification functions are monotonic.

The metric is also continuous.

\[ \lim_{p_k \to p_{k'}} \left\{ P_{(p_1,\ldots,p_k,\ldots,p_n)} \right\} = P_{(p_1,\ldots,p_{k'},\ldots,p_n)} \]  \hspace{1cm} (2.6.4)

Therefore, if the acceptability of the design based upon one requirement is changed continuously, the overall acceptability will also change continuously.

Further, the metric displays the property of strictness (equation 2.6.5).

\[ P_{(p_1,\ldots,p_k,\ldots,p_n)} < P_{(p_1,\ldots,p_{k'},\ldots,p_n)} \iff p_k < p_{k'} \]  \hspace{1cm} (2.6.5)

(Otto and Antonsson 1991) argue that this property may be over-restrictive in some instances. As an example, they cite a case where the designer may not care about the \( k^{th} \) characteristic of the design. In this case, the two designs in equation (2.6.5) might be considered equivalent. I argue that strictness is reasonable provided the designer cares about satisfying all specifications equally. Each specification is measured on an absolute acceptability scale so the design characteristic's overall importance is expressed in the specification definition. The specification for a design characteristic that has no importance to the acceptability of a design will be a horizontal line at \( p=1.0 \). (One might
also question: If the designer does not care about a characteristic, why is it included in the design evaluation?)

Equation (2.6.6) expresses the characteristic of dominance.

\[ P_{(p_1, ..., p_k, ..., p_n)} \leq p_k \forall p, \ 0 \leq p \leq 1 \]  

(2.6.6)

This quality means that the acceptability of a design can never be greater than the least satisfied specification. Dominance gives the metric the property of annihilation (2.6.7).

\[ P_{(p_1, ..., 0, ..., p_n)} = 0 \]  

(2.6.7)

I feel that annihilation is a critical property. Annihilation allows a single requirement to render a design unacceptable. For example, a design that fails to meet a stress safety requirement is unacceptable regardless of other qualities it may have.

The last three traits that will be defined are the identity property (2.6.8), communitivity (2.6.9), and associativity (2.6.10).

\[ P_{(1, ..., 1, p_k, 1, ..., 1)} = p_k \]  

(2.6.8)

\[ P_{(p_1, ..., p_{(a_1, ..., a_n, ..., p_n)} = P_{(p_1, ..., p_{(a_1, ..., a_n, ..., p_n)}} \]  

(2.6.9)

\[ P_{(p_k, p_{(p_{(a_1, ..., a_n, ..., p_n)}}, p_k) = P_{p_k, p_{(p_{(a_1, ..., a_n, ..., p_n)}}) \]  

(2.6.10)

The section will close with a discussion of independence requirements when using the specification-based decision metric. This issue seems to be a point of confusion. Two examples will illustrate that the formulation requires no independence assumptions. The only requirements for using the metric are that each specification must be relevant to the design, the specification form must represent the probability the design will be acceptable based upon the particular performance characteristic, and the probability density functions and specifications used in the evaluation must represent the current design state.

Before proceeding into the examples, some definitions are in order.
Event A is independent of B iff

\[ P_{(A|B)} = P_{(A)} \]  

(2.6.11)

Events A and B are mutually independent iff
\[ P_{(A|B)} = P_{(A)} \quad \text{and} \quad P_{(B|A)} = P_{(B)} \quad (2.6.12) \]

Preferential independence means that the preference for a design performance attribute will not depend on other design attributes. Given a two attribution design problem \((x, y)\) where \(x\) is preferentially independent of \(y\), we can say

\[ \text{if } (x, \alpha) \succeq (x', \alpha) \text{ then } (x, \beta) \succeq (x', \beta) \quad \forall \beta \in y \quad (2.6.13) \]

Mutual independence implies that (2.6.13) also holds true for \(y\). In terms of our design specifications, preferential independence implies that the form of a given specification does not depend upon the design operating point or how other specifications are fulfilled. These definitions are sufficient for the forthcoming examples, but a complete treatment of independence definitions and implications may be found (Keeney and Raiffa 1976).

Let us return to discussing the decision metric. The following examples address two different cases: Case 1 illustrates the simple situation where both performance data and specifications are independent. In Case 2, the performance data are inter-dependent but the specifications are independent. Other possible cases include: situations where the specifications depend upon the performance data, or when the specifications are inter-dependent (preferential dependence).

**Case 1: mutually independent performance data, mutual preferential independence of specifications**

Imagine that a committee is reviewing a new automobile design in 1970. The committee is interested in having a design that is efficient and has market appeal. In this period of energy conservation oblivion, fuel economy has no bearing upon market appeal. The new design achieves 8mpg ± 0mpg and a market study suggests that 70% ± 0% of potential customers approve of the design. The specification indicates that a design with this fuel consumption has a probability of 0.8 of being acceptable, while the design is perfect based upon the level of market approval.

The probability that the design is acceptable based upon a single specification is given by

\[ P_{\text{acceptable}} = P_{\text{design performs at a given level}} \cap P_{\text{given level of performance is acceptable}} \quad (2.6.14) \]

Thus, the acceptability of this design under our independence assumptions is
\[ P_{\text{acceptable}} = \left( P_{\text{mileage} = 8 \cap P_8 \text{ is acceptable}} \right) \cap \left( P_{\text{market approval} = 0.7 \cap P_{0.7 \text{ is acceptable}}} \right) \] (2.6.15)

\[ P_{\text{acceptable}} = (1.0 \times 0.8) \cap (1.0 \times 1.0) \]

\[ P_{\text{acceptable}} = 0.8 \]

**Case 2:** dependent performance data, mutual preferential independence of specifications

Unfortunately, it is now 1974 and the same design is still on the drafting board. Market changes have prompted another design review. Market acceptance is no longer independent of fuel economy, but cost saving measures prohibit performing a new market study. Given the vehicle's fuel economy, the review board feels there is a probability of 0.6 the customer acceptance will be 70% (this discrete example is analogous to shifting a probability density function for market approval as a function of fuel economy). The specifications for evaluating the design are unchanged. Using the definition of conditional probability,

\[ P_{\text{(A|B)}} = \frac{P_{\text{(A} \cap \text{B)}}}{P_{\text{(B)}}} \] (2.6.16)

the probability the design will be acceptable becomes

\[ P_{\text{acceptable}} = \left( P_{\text{mileage} = 8 \cap P_8 \text{ is acceptable}} \right) \cap \left( P_{\text{market approval} = 0.7 \text{ given mileage} = 8 \cap P_{0.7 \text{ is acceptable}}} \right) \] (2.6.17)

\[ P_{\text{acceptable}} = (1.0 \times 0.8) \times (0.6 \times 1.0) \]

\[ P_{\text{acceptable}} = 0.48 \]

This example illustrates that the formulation is valid even if performance data are dependent, provided the conditional probabilities or probability density functions can be determined before making the calculation. Of course, for problems where the data are coupled, this will not be possible. These situations may be resolved by iterating until the solution converges or by selecting fixed design points to decouple the problem.

An identical approach can be used to illustrate that performance data/specification and specification/specification dependencies also do not invalidate the design metric. The only requirement is that all of the conditional probability density functions or specifications must be resolved. For quasi-coupled problems (such as the second example) this may be done directly, but coupled calculations will require the use of iterative or problem simplification techniques.
3. Comparison to other decision methods

After reading chapter 2, one is familiar with the specification-based decision model and how it can be used to evaluate design alternatives. In this chapter other decision frameworks and alternative criteria are discussed.

The development of mathematical approaches to decision-making dates at least to Laplace's time. Over the years numerous decision methods have been proposed and developed—some of which have matured and become rigorous. Decision-making, like the activity of designing, is not bound by physical laws and thus, like design methodologies, there is an abundance of decision models. Numerous books have been written on this subject and the goal of this chapter is not to duplicate such publications.

As such, the objectives of this chapter are threefold. First, a review of established decision methodologies provides context for the model proposed in this thesis. The intent of reviewing different methods is not to imply that some approaches are inferior to others, but to highlight different decision-making philosophies. Second, it is necessary to identify the specification-based model's decision and design theory roots. The decision theory field is so mature that it would be naive to believe that a completely new model has been developed. The specification-based model is a new adaptation of older ideas. Third, the focus of this thesis is decision-making in product design. Thus applications to design and particularly environmentally responsible design will be emphasized.

To my knowledge, the use of formal decision methods by product designers is uncommon. If decision models are used, they tend to be simple or qualitative. Therefore, opinions will also be offered about the suitability of different methods for product design decision-making.

Section 3.1 begins with a discussion of axiomatic design and the information content criterion it proposes. The approach is discussed first because the specification-based metric draws heavily upon the philosophy behind the information content metric. Then, qualitative decision measures and figure-of-merit based approaches are reviewed. Variants of these methods are probably the approaches most commonly used in design practice. Next, utility analysis is discussed. Utility analysis is one of the more rigorously developed methods and is the theory that led me to the idea of using specifications as
absolute scale value functions. The remaining parts of section 3.1 address Pareto sets, fuzzy logic, decision trees, cost-benefit analysis and probabilisitic design.

Section 3.2 is a philosophical discussion about using weighting factors in design decision making. This section will probably invoke counter opinions, if not some controversy. I suggest that, if the specifications are defined on an absolute acceptability scale and they are subject to evolutionary adjustment, weighting factors are not needed.

3.1. Comparison to other Design Evaluation/Decision Methods

3.1.1. Axiomatic Design

Axiomatic design is not simply a decision model—it is a design methodology. A detailed description of axiomatic design may be found in (Suh 1990). The method derives its name from the two design axioms upon which it is founded—the independence axiom and the information axiom. The information axiom is of relevance to this thesis.

The information axiom and its use as a decision criterion is first described. Then, an application related to life-cycle design goal setting (which was the starting point of this thesis) illustrates the method's application to environmental design. The section is concluded by summarizing the key concepts that were incorporated into the specification-based design decision model and the problems that were encountered when applying axiomatic design's information-based metric.

3.1.1.1. The information content criterion

The information axiom states: given a selection of concepts that satisfy the independence axiom, the alternative with the lowest information content is the best design. Information content is a measure of the probability of successfully satisfying design requirements, and is defined as

\[ I = \log_2 \left( \frac{1}{p} \right) \text{ bits} \quad (3.1.1.1.1) \]

where \( p \) is the probability that a requirement will be satisfied. When there are \( n \) design requirements, the total information is given by
\[ I_{total} = \sum_{i=1}^{n} I_i \]  

(3.1.1.2)

A simple example will clarify the use of information content as a decision tool. This example is based upon a more detailed case study presented in (Suh 1990).

We wish to select the best machine for fabricating the keyway shown in figure 3.1.1.1. The only requirement is that the width of the keyway must be within a 0.003” tolerance. The suitability of using a milling machine is under consideration. The figure illustrates a frequency range of tolerances for the machine in question (the milling machine system range). The probability that the keyway will be within tolerance is given by the area of the milling machine system distribution within the design tolerance divided by the total area of the system range. In this case, there is an 80% chance that the part will be within tolerance. The information content of 0.3 bits may be calculated from equation (3.1.1.1). This information content is a metric of additional resources that must be added to ensure that the part is within tolerance. For example, one might specify a more skilled machinist, or dimensional checks. Information content for the milling machine can be compared with other machining options. The machine that yields the lowest information content is considered the best choice.

\[ \text{Tolerance} = \pm 0.003" \]
\[ \text{Probability} = \frac{\text{Common area}}{\text{System range}} = 0.8 \]
\[ \text{Information} = \log_2 (1/p) = 0.3 \text{ bits} \]

\[ \text{Frequency of Occurrence} \]

\[ \text{Variation from Nominal Dimension (1/1000 inches)} \]

Figure 3.1.1.1 The use of information content in a simple machine selection problem
A complete axiomatic design example may be found in (Gebala and Suh 1992). For comparative purposes, (Shannon 1948) developed the foundations of information theory, while (Ayres 1987) is a good application of information theory to design in the area of manufacturing and assembly.

3.1.1.2. Application to product life-cycle goal setting

As a project for the graduate course 2.882, Principles of Axiomatic Design, a software program that suggests suitable life-cycle goals based upon a products expected characteristics was developed. The work, which uses the information content criterion as a decision metric, is detailed in (Wallace and Suh 1993). The software is briefly explained here as this was the starting point for some of the ideas presented in chapter 2.

The designer usually is aware of many characteristics a product will have—even before it is designed. For example, a designer will know what production volumes or length of life the product should have. The objective is to use this knowledge to predict what life-cycle outcomes the product is suited for. Is the product a good candidate for recycling, or is it a better candidate for reuse? Is the product suited to manual assembly or automated assembly? Given this knowledge, the designer might better design for the most suitable eventuality.

The program interface is illustrated in figure 3.1.1.2.1. The screen is divided into three columns. In the left column, the designers use push buttons and sliders to set the values of product attributes, as well as to identify the life-cycle areas in which they are interested. When a product attribute button is selected, a slider or text input field appears (note: no product attributes are selected in figure 3.1.1.2.1—see figure 3.1.1.2.2 for examples of the slider input fields). A representative set of attributes were chosen for the demonstration: product life-cycle (years), production volume (1000s/year), value added (approximate ratio of energy labor and capital inputs to raw material cost), desired payback period (years), complexity (approximate number of parts), level of styling importance (qualitative scale 0-10), and material class (wood, plastic, metal, composite). The interest area buttons are used by the designer to select the life-cycle stages for which goals will be chosen. The middle column of the interface presents the most suitable life-cycle design goals, while the right display is reserved for design strategy (or solution) suggestions and
guidelines. The strategies might be applied to achieve the suggested life-cycle design goals.

![Diagram of areas of interest, design goals, and design solutions]

*Figure 3.1.1.2.1* The program interface. The left column is used to set product attributes, the middle is reserved for the design goals predicted to be most suitable, and corresponding design solution strategies or guidelines are presented in the right column.

For the purpose of demonstration, consider the conceptual design of a product that is extremely resource intensive—the automobile. We begin by specifying just a few product attributes: the car's life cycle will be long (about 15 years), and it is hoped to make between 95 to 100 thousand vehicles each year. The product attribute values are set interactively using the slider bars. Figure 3.1.1.2.2 shows the program output, suggesting design targets for both product assembly and end-life. Our discussion will focus primarily on end-life goals. The rightmost column suggests design strategy combinations

---

2 The abbreviations FR on the buttons beside the design goals and DP beside the strategy solutions are related to axiomatic design nomenclature. Our design goals are analogous to functional requirements (FRs), and solution strategies map to design parameters (DPs).
and will not be explained here. This aspect of the program makes its recommendations based upon independence concepts rather than information content (full details may be found in Wallace and Suh 1993).

![Diagram of Areas of Interest, Design Goals, and Design Solutions]

**Figure 3.1.1.2.2 Goals and strategies recommended by the program for the example problem.**

In the central column, the program has recommended automatic assembly and reuse as suitable design goals. Associated with the goals are two constraints: the product domain should be standardized, and a collection infrastructure for used products must be established. If a constraint or goal is unacceptable to the designer, the goal can be eliminated from the solution space using the reject button. The program will then recommend the next best alternative.

The reuse goal and *why* button have been selected so that the goal dialogue window (bottom of center column) provides an information content breakdown for all possible end-life goals. In this case, the information content indicates very high confidence that the product attributes are suitable for a reusable product. The higher information
associated with recycling suggests lesser certainty about its appropriateness. (The total information value of 1.403 indicates the probability of being well suited is about 0.4.) The program is making these predictions by comparing the expected product attributes to specifications analogous to the tolerance in the previous machine selection example. For example, the reuse goal has a number of specifications or criteria that define the attribute levels that a reusable product must exhibit.

Assuming that the constraints associated with the recommended goals are acceptable, the designer can use strategies of modularity and design for feedability/insertion to guide conceptualization. This might lead to a concept utilizing a long-lived standardized space frame onto which modules are attached—much like modular homes. Items with relatively short lives—such as body panels that become aesthetically obsolete—will be separated from durable modules. Under this concept, owners can upgrade to a 'new' automobile by changing elements instead of buying a new vehicle.

Summarizing up to this point, we began with high level product attributes for an automobile. The life-cycle planning program is used to guide the development of a modular automobile concept. This concept gives rise to a number of new design problems. For example, a support structure and host of different functioning modules are required.

The program can now be used to guide the conceptualization process for these sub-problems. Figure 3.1.1.2.3 shows the program's recommendations for the exterior body panels. The exterior will have a high degree of styling, a production volume similar to the vehicle, a payback period of one year, and a short life (about 3 years). The use of plastic is considered. Using the previously described decision mechanism, the program forwarded recycling and manual assembly as design goals.
Figure 3.1.1.2.3 Goals and strategies recommended for the panel modules.

3.1.1.3. Relationship to specification-based model

This initial project reveals that the information content decision criterion has a number desirable properties. The criterion requires that expectations or requirements are defined by tolerance-like specifications. Specifications have the advantage of being widely used and understood throughout the design community. The goal setting program illustrates that a variety of problems can be quantified using the language of specifications.

Moreover, although not explicitly stated in the axiomatic design literature, the information metric implies that the goal of design activity is to satisfy specifications. This is the hypothesis underlying the specification-based model which I have proposed.

A number of difficulties were also encountered. Foremost, the information formulation makes no provision for uncertainty about specifications. The borders of the tolerance-like
specifications are absolute and clearly delineated. A design is either in specification or out of specification. The production volumes that suit manual assembly are not bounded so clearly. This issue caused difficulties throughout the project and prompted the investigation that led to viewing specifications as acceptability functions.

Finally, designers that tested the program complained that the significance of information content was opaque. If designers are to work with and trust a decision model, they must be able to understand the meaning of the evaluation criterion. Conversely, the probability of being acceptable according to specifications (which is related to information content) was understood and deemed to be an informative design metric.

3.1.2. Qualitative measures

There are numerous methods that fall into the category of qualitative assessment measures. I view any technique that distinguishes relative merit using non-numerical scales as part of this group. These approaches seem to be fairly popular for conceptual design evaluation because they are easy to apply and do not require extensive modeling/analysis or data collection. Well-known methods of this type include Pugh selection charts (Pugh 1990) and the House of Quality (Hauser and Clausing 1988; Akao 1990). The product assessment matrices used by Consumer Reports provide yet another example.

These measures usually take the form of a matrix, where each row assesses design candidates on a specific criterion. Assessment is typically performed qualitatively against some datum. Commonly, evaluation involves qualitative comparisons between options using what (Otto 1994) describes as a better/worse scale. An example Pugh chart is provided in figure 3.1.2.1.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Alternative 1</th>
<th>Alternative 2</th>
<th>Alternative 3</th>
<th>Alternative 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>D</td>
</tr>
<tr>
<td>B</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>-</td>
<td>S</td>
<td>+</td>
<td>T</td>
</tr>
<tr>
<td>D</td>
<td>S</td>
<td>S</td>
<td>-</td>
<td>U</td>
</tr>
<tr>
<td>E</td>
<td>+</td>
<td>-</td>
<td>S</td>
<td>M</td>
</tr>
</tbody>
</table>

Figure 3.1.2.1 Example of a Pugh selection chart.
This chart indicates that the designer has four alternative concepts and wishes to identify the best alternative based upon five design criteria (A-E). Alternative 4 has been chosen as the datum, or reference, to which all other concepts will be assessed relatively. A plus sign implies higher desirability: better than, less than, less prone to, easier than, etc. The minus sign infers lower desirability: worse than, more expensive than, more prone than, harder than, etc. An 'S' denotes equivalence to the reference design.

Once the designers have completed the matrix they are to look for a pattern of preferences and perhaps an alternative will emerge as the preferred concept. The synthesis of the individual symbols into a total evaluation is left to the discretion of the designers (the assignment of values to the symbols is discussed in section 3.1.3).

3.1.2.1. Application to environmental design

Such qualitative assessment techniques are very easy to understand conceptually. In this section a qualitative matrix approach for environmental assessment, proposed in (Allenby 1991; Allenby 1991), is described. Allenby argues that the environmental assessment of designs should be qualitative—environmental systems involve so much uncertainty that quantification is, to a large extent, impossible.

He proposes four primary assessment matrices for evaluating each design concept: a manufacturing matrix, a social political matrix, a toxicology/exposure matrix and an environmental matrix. The environmental matrix is shown in figure 3.1.2.1.1.

Each matrix cell will have one of three types of entries. A '—' means that the category is not applicable or inappropriate. A '+' or '++' is used to indicate positive effects and the relative degree of benefit (the use of a reference is not stated explicitly, but relative benefit implies a datum must exist). Finally, an oval may be placed in the cell indicating concern. An empty oval means minimal concern, dots indicate some concern, lines suggest high concerns, and solid black indicates the highest level of concern. The extent to which the pattern fills the oval indicates the degree of uncertainty associated with the concern. A completely filled oval suggests certainty, while a 1/4 filled oval would indicate a high level of uncertainty.
Figure 3.1.2.1.1 Environmental assessment matrix.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Alternative 1</th>
<th>Alternative 2</th>
<th>Alternative 3</th>
<th>Alternative 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toxicity</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>S</td>
</tr>
<tr>
<td>Environmental</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>S</td>
</tr>
<tr>
<td>Manufacturing</td>
<td>-</td>
<td>S</td>
<td>+</td>
<td>S</td>
</tr>
<tr>
<td>Social/Political</td>
<td>-</td>
<td>S</td>
<td>-</td>
<td>S</td>
</tr>
</tbody>
</table>

Figure 3.1.2.1.2 Summary matrix combining the results of the four primary design evaluation matrices.

These primary matrices are then summarized for inter-concept comparison by the matrix in figure 3.1.2.1.2. How the four matrices are summarized to generate this Pugh-like chart is left to the designer's discretion.
3.1.2.2. Relationship to specification-based model

Clearly, the specification-based model does not draw much inspiration from such qualitative measures. However, they are discussed in this thesis for a number of reasons. First, the specification-based model does not preclude the use of such methods—qualitative assessment techniques have their place. Further, it is worth discussing possible weaknesses and how the specification-based model addresses these shortcomings.

The qualitative measures may help the designer in a number of ways. The exercise of developing assessment matrices and criteria helps the designer gain insight into what the design requirements should be (i.e., they can help the designer set specifications—section 2.3 described specification setting as a design process in and of itself.) Allenby's method provides a number of possible criteria to aid this process. Further, a better understanding of the design problem may emerge—hopefully any modeling exercise will achieve this. Filling out the matrix will at least get the designer thinking about potential issues. Finally, the effort might stimulate the generation of new concepts.

I believe the deficiencies of such methods reside in their use as decision tools—i.e., choosing the 'best' concepts. Unless one concept is blatantly superior to the others, the designer is left to internally balance the many criteria. In such circumstances there is a strong methodological restriction tied to human mental processing capabilities. It is accepted that we can only weigh a finite number of attributes simultaneously. This finite limit, tied to short-term memory, is believed to be somewhere between 2 to 7 conceptual units or chunks (Simon 1970, pp. 31-49). The environmental matrix in figure 3.1.2.1.1 is a short-term memory nightmare.

With respect to systems-oriented design, another concern arises. Can one intuitively assess consequences when dealing with complicated non-linear systems (such as natural eco-systems)? I gave my answer to this question when describing why guidelines alone will not result in good system-oriented product design (section 1.1). Most environmental issues are too complex to evaluate on intuition alone. When I try to assess environmental problems intuitively, I usually end up concluding, "I'm not sure." For every positive, there is a possible negative.
On the other hand, Allenby argues that the only way to cope with the uncertainties associated with environmental systems is to perform vague qualitative analyses. Assigning numbers hides assumptions and can give misleading results. Further, some things are more quantifiable than others. In section 2.4, I discussed how using subjective probability density functions addresses these concerns. The vagueness of data is retained through the distribution bandwidth and subjective probabilities represent the designer's feeling.

Finally, while these methods can be used to suggest strengths and weaknesses, they do not convey the notion of 'good enough'. In the development of the specification-based model, it was hypothesized that satisfaction is the essence of product design. It would be interesting to develop qualitative methods based upon this philosophy rather than the better/worse scale currently employed.

3.1.3. Figure-of-Merit

Another way to assess multiple criteria design problems is to apply a so-called figure-of-merit method. Using this type of approach, numerous performance indices are combined to provide an overall measure of value. Thus, this strategy overcomes our limited capability to simultaneously compare a large number of issues, as explained in the previous section. The term figure-of-merit is used because this value has no physical significance.

A typical figure-of-merit formulation might appear as equation (3.1.3.1).

\[ V = \frac{\sum_{i=1}^{n} [w_i v_i]}{\sum_{i=1}^{n} w_i} \]  

(3.1.3.1)

Where

\[ V = \text{overall value of a design alternative} \]
\[ v_i = \text{the performance indicator for the } i^{th} \text{ criterion} \]
\[ n = \text{the total number of criteria} \]
\[ w_i = \text{a weighting factor associated with the } i^{th} \text{ performance indicator} \]
The weighting factors are typically described as importance ratings for the individual criterion, and the best design alternative will have the highest overall merit figure. By comparing equation (3.1.3.1) to equation (2.5.2), we can see that the figure-of-merit is, in essence, a generalized form of Laplace's expected value criterion. The weighting factors allow the expected value formulation to combine ratings for criteria that are measured on different relative scales.

This method seems to be fairly popular (perhaps because of its apparent ease of use) and is covered extensively in the literature. It is common to see the qualitative matrix approaches, such as Pugh carts, converted to a figure-of-merit by assigning numerical values to the qualitative symbols and introducing weighting factors (for example, see Ullman 1992, pp. 175-183). Under such schemes the rating symbols convey more than better or worse (as described in the previous section). They also indicate how much better or worse.

Another representative presentation of the figure-of-merit may be found in (Steuer 1986, pp. 165-200), while a design perspective is given in (Thurston 1991). The analytic hierarchy process (Saaty 1980) provides an example of a design method that uses a figure-of-merit approach (Marsh, Slocum et al. 1993, is an example of a design case study which advocates the use of the method).

The key issue when using this class of evaluation metric is setting the weights that depict the relative importance rankings. By setting the weights, the designer explicitly states how all of the different metrics must be converted so that they may be added in a meaningful way. This is not a trivial task. How does one relate, for instance, a measure of ergonomics to a stress safety factor? The analytic hierarchy suggests the relative weighting strategy in table 3.1.3.1. These weights are applied to create a reciprocal weighting matrix that is used to compute and normalize the weighting vector used in the figure-of-merit calculation. This approach is nicely demonstrated in (Marsh, Slocum et al. 1993).

I am not aware of any design research that focuses explicitly on developing figure-of-merit approaches for environmental product design.
Table 3.1.3.1 A scale for quantifying relative importance.

<table>
<thead>
<tr>
<th>Value</th>
<th>Comparative relationship</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>equal importance</td>
</tr>
<tr>
<td>3</td>
<td>weakly more important</td>
</tr>
<tr>
<td>5</td>
<td>moderately more important</td>
</tr>
<tr>
<td>7</td>
<td>strongly more important</td>
</tr>
<tr>
<td>9</td>
<td>absolutely more important</td>
</tr>
</tbody>
</table>

3.1.3.1. Relationship to specification-based model

The figure-of-merit approach and the specification-based model do not have much in common, besides the fact that they both combine multiple attribute design problems into a single overall performance metric. However, they have a significant difference regarding the use of importance rankings—the figure-of-merit requires their use, while the specification-based model does not. A mathematical justification for not using weights in the specification-based model is provided in section 3.2.

Figure-of-merit methods have been successfully applied in design and optimization, but they are also regarded by many practicing designers with some skepticism. The point of contention is the weights. It is very difficult to relate completely disparate objectives in such a direct manner. Although often completely ad-hoc, the reciprocal matrix approach demonstrates it is possible to develop a mathematically consistent weight setting procedure. However, mathematical rigor is not the only issue. It is disturbing (to the author at least) that there is no apparent basis for deciding that 'weakly more' important is 3 and 'moderately more' important is 5. The weight setting problem is even further complicated by acknowledging that a criterion regarded as very important in one performance range may be unimportant in another (Thurston 1991). In other words, the ranking weights may not be constant. Using figure-of-merit methods to evaluate environmental problems would invite controversy over justifying specific weight values.

When using the specification-based approach, designers do not make such decisions. They do not attempt to place values on the different design attributes. Instead they must decide what level of performance they will accept in each evaluation category.

There are also two other notable differences between the figure-of-merit metric and the probability of acceptance metric. The figure-of-merit does not exhibit the dominance and
annihilation properties (equations (2.6.6) and (2.6.7)). This means that it is not possible for failure in one category to completely negate the design's value. In essence, when using a figure-of-merit good aesthetics can partially compensate for dangerously high stress levels! Finally, the figure-of-merit, evaluates designs on a relative scale. This means that each candidate must be assessed using the same evaluation criterion, eliminating the possibility of performing feasibility studies as illustrated in figures 2.2.5 and 2.2.6.

3.1.4. Utility Analysis

Utility analysis is one of the most thoroughly developed decision methods. The concept of utility is rooted in economic theory and is equated to 'satisfaction'. It is assumed that the rational consumer's goal is to maximize utility (vonNeumann and Morgenstern 1947, p. 8) and, in economics, utility maximization is typically equated with profit maximization. The premise of utility theory is that, given a number of possible alternatives, the best option is the one with the highest expected utility.

The text Decisions with Multiple Objectives (Keeney and Raiffa 1976) is a classic utility analysis reference, while an alternative presentation is provided in (French 1988). Recently, Deborah Thurston has applied utility theory to engineering design problems (Thurston 1990; Thurston 1991; Thurston, Carnahan et al. 1991; Thurston and Liu 1991; Locascio and Thurston 1992; Thurston and Essington 1993). The intention of this section is to provide sufficient material only to clarify which aspects of the specification-based model incorporate ideas from utility theory.

First, utility theory developed a procedure to establish utility functions for single design attributes—the lottery certainty equivalent (see Keeney and Raiffa 1976, pp. 193-196). The lottery concept was described in section 2.3. The key difference is that the utility function is set on a relative scale (bounded by the best and worst design alternatives). The designer would begin by assigning the best attribute level $x^*$ a utility of 1, and the lowest level $x^0$ a utility of 0. The intermediate points on the utility function are set relative to these levels. (Recall that in the specification-based model, the 'worst' acceptable level is assigned an acceptance probability of 1 and the 'best' unacceptable level is assigned a probability of 0.)
Utility theory also establishes how to combine single attribute utility functions, consistent with its assumption of maximizing expected utility, to assess multiple attribute problems. In general, all utility functions will have the form

\[ u(x_1, \ldots, x_n) = \sum_{i=1}^{n} k_i u_i(x_i) + \text{POT} \quad (3.1.4.1) \]

where:

- \( (x_1, \ldots, x_n) \) are the \( n \) design attributes,
- \( (u_1, \ldots, u_i, \ldots, u_n) \) are the \( n \) single attribute utility functions, and
- \( (k_1, \ldots, k_i, \ldots, k_n) \) are weights that scale the single attribute utility values.

POT does not refer to an illegal substance, but stands for potentially other terms.

The utility functions are scaled such that

\[ u(x_1^*, \ldots, x_n^*) = 1 \text{ and } u(x_i^*, \ldots, x_n^*) = 0 \quad (3.1.4.2) \]
\[ u_i(x_i^*) = 1 \text{ and } u_i(x_i^a) = 0, \forall x_i \quad (3.1.4.3) \]

From equations (3.1.4.2) and (3.1.4.3) it can be shown that the weights are given by

\[ u(x_i^*, x_i^a) = k_i, \quad i = 1, 2, \ldots, n \quad (3.1.4.4) \]

(Keeney and Raiffa 1976, p. 303) and (Thurston 1991) both describe how the lottery method can also be used to determine the weights in a fairly straight-forward manner.

The actual form of the multiple attribute utility function that can be used to identify designs which maximize expected utility is determined by independence conditions. For example, for any design where the attributes exhibit mutual utility independence, equation (3.1.4.5) is appropriate.

\[ u(x) = \sum_{i=1}^{n} k_i u_i(x_i) + K \sum_{j=1}^{n} k_i k_j u_i(x_i) u_j(x_j) + \text{higher order terms} \quad (3.1.4.5) \]
This can be rewritten as

\[ u(x) = \frac{1}{K} \left[ \prod_{i=1}^{n} (Kk_i u_i(x_i) + 1) - 1 \right] \quad (3.1.4.6a) \]

where

\[ 1 + K = \prod_{i=1}^{n} (1 + Kk_i) \quad (3.1.4.6b) \]

This result is described as the multipllicative form (see Keeney and Raiffa 1976, pp. 288-292 for a complete derivation). The constant \( K \) is calculated to ensure that the overall utility scales between 0 and 1. The work by Thurston assumes this form. Utility independence implies that the utility function for attribute \( x_i \) does not depend on the value of the other design attributes. Mutual independence requires this relationship in the opposite direction as well.

The additive form of the multiple attribute utility function,

\[ u(x_1, \ldots, x_n) = \sum_{i=1}^{n} k_i u_i(x_i) \quad (3.1.4.7) \]

requires mutual utility independence with an additional restriction. Intuitively, additive independence requires that the actual form of the designs' multiple attribute utility equation (if it were known) must not depend on the product of individual utility functions (for a mathematical definition see Keeney and Raiffa 1976, pp. 229-230). A simple example will clarify this point.

Imagine that we know the multiple attribute utility of a design is given by \( u(x_1, x_2) = x_1 x_2 \), thereby implying that \( u_1(x_1) = x_1 \) and \( u_2(x_2) = x_2 \) (bear in mind that the explicit form of \( u(x_1, x_2) \) is usually not known). Clearly, \( u_1 \) and \( u_2 \) are mutually independent, but \( u(x_1, x_2) \) cannot be described with the additive form (equation (3.1.4.7)). However, the function \( u(x_1, x_2) \) could be modeled using the multipllicative
form (equation (3.1.4.6a)). If the additive form is appropriate, the weights will sum to unity. That is,

$$\sum_{i=1}^{n} k_i = 1 \quad (3.1.4.8)$$

which requires that $K = 0$.

It is important to understand that, unlike the figure-of-merit approaches, the weights are not importance rankings. Instead, they indicate how the designer would like to trade-off or substitute between the utility levels on different attributes. (This issue is addressed in Keeney and Raiffa 1976, pp.271-273).

This significance of the weights is best demonstrated by an example. Consider a two attribute problem with an additive utility function form.

$$u(y,z) = k_y u_y(y) + k_z u_z(z) \quad (3.1.4.9)$$

Let $y^*, z^*$ denote the best possible performance for two design characteristics, and $y^0, z^0$ represent the poorest performance levels for each attribute. The utility functions are defined such that

$$u(y^0, z^0) = 0, \quad u_y(y^0) = 0, \quad u_z(z^0) = 0 \quad (3.1.4.10)$$

and

$$u(y^*, z^*) = 1, \quad u_y(y^*) = 1, \quad u_z(z^*) = 1 \quad (3.1.4.11)$$

These relationships imply that

$$k_y + k_z = 1$$

Now, let's imagine that the designer has to choose between a design at $(y^*, z^0)$ and another at $(y^0, z^*)$. If the designer finds the two designs equivalently desirable, then $k_y$ must be the same as $k_z$ because the designer has indicated that
\[ k_y u_y(y^*) + k_z u_z(z^*) = k_y u_y(y^o) + k_z u_z(z^o) \]  \hspace{1cm} (3.1.4.12)

or

\[ k_y(1) + k_z(0) = k_y(0) + k_z(1) \]

On the other hand, if the designer prefers the design at \((y^o, z^*)\) over \((y^*, z^o)\), then \(k_z\) must be greater than \(k_y\). In general, the weights will not be equal because each attribute's individual utility function is defined over a different scale. For example, there may be a very big difference in overall satisfaction between the \(z^*\) and \(z^o\), and only a very small overall change between \(y^*\) and \(y^o\). However, the individual utility functions are defined such that \(u(z^*) = u(y^*) = 1\) and \(u(z^o)=u(y^o) = 0\). Therefore, the weighting factors must capture the fact that the scales of the two utility functions have different meanings and thus do not trade-off equally.

Consider a situation where overall satisfaction is equated to total money earned. If moving from \(y^o\) to \(y^*\) results in a 2 dollar increase in profit, while the difference between \(z^o\) and \(z^*\) is 8 dollars, then \(k_z/k_y = 4\), yielding \(k_z = 0.8\) and \(k_y = 0.2\). By the same argument, if moving from \(y^o\) to \(y^*\) results in an 8 dollar increase in profit while the difference between \(z^o\) and \(z^*\) is 8 dollars, then \(k_z = k_y = 0.5\). Thus the weights are equal when the overall significance of moving from the best to worst is the same for all attributes.

The specifications indicating acceptability used in the satisfaction model (proposed in this thesis) measure each attribute overall implications are the same. That is, the best level of any attribute means the entire design is acceptable and the worst level means that the entire design is not. Therefore, one could use the acceptability specifications in a multiple attribute utility analysis where all the weights \(k_i\) have the same value. Such an analysis would identify the design that has the highest expected acceptability assuming that low acceptability in one area can be substituted for by high acceptability somewhere else. Note that this is different than maximizing the overall probability of being acceptable according to all specifications (e.g., if a design is measured on two acceptability specifications and \(p_1 = 1\) while \(p_2 = 0.5\), an additive utility function would yield \(u = 0.75\), but the overall probability of acceptability on all specifications is \(1.0 \times 0.5 = 0.5\).
For those not content with this qualitative argument about weights, section 3.2 attempts to mathematically illustrate that the weighting factors must be equal when each utility function has the same overall implications.

3.1.4.1. Application to environmental design

As mentioned at the start of section 3.1.4, Thurston has published a number of papers which advocate the use of utility theory in engineering design. Recently, Dr. Eyal Zussman has developed a method for assessing the best recycling options based upon disassembly requirements (Zussman, Kriwet et al. 1994). He is part of a group at the University of Berlin Institute for Machine Tools and Production Technology, headed by Professor Seliger, which is conducting research in design for disassembly and recycling. Other representative work may be found in (Jovane, Alting et al. 1993; Seliger, Zussman et al. 1993; Pnueli, Zussman et al. 1994)

In Zussman's recycling assessment and planning work, utility analysis is used to select the best alternatives. Only a brief description of the work will be described here. Utility analysis is used to evaluate feasible options for disassembling a product and applying recycling processes to its components and subassemblies. The method uses a cost-benefit approach (cost-benefit analysis is discussed in section 3.1.8) to compare the future effort that will be invested in recycling to the future benefits regarnable through reduced dumping fees and the sale of recovered materials or components. A detailed model for these costs and benefits are presented, and then utility analysis is used to identify the best combination of recycling strategies, given the designers' multiple economic and environmental objectives. An additive utility form is assumed.

3.1.4.2. Relationship to specification-based model

Ideas that were developed for utility analysis have had an important impact on the development of the specification-based design decision model. The idea that individual specifications are functions that can be established using the lottery method comes directly from this body of literature. However, one must note that a specification indicates the probability the design will be deemed acceptable and is not a utility function.
However, under the premise that design is a satisfaction process, specification value functions are defined on an overall probability of acceptance scale. Therefore, the scale of each individual specification value function is the same. In effect, the specifications are utility functions that have an absolute scale.

In utility analysis, each individual utility function is also defined using values from 0 through 1. However, the overall significance of measuring '1' on a utility function for one attribute may differ from a '1' for another attribute. This difference arises because the '0' and '1's are assigned relative to the best and worst attribute levels for each category. The single attribute utility function does not reflect the individual attribute's potential impact on the overall design utility. For this reason, weights quantify how the single attribute utility levels should be traded-off.

These substitution or trade-off weights are easier to grasp than the importance rankings required by the figure-of-merit approaches. Importance rankings require the designer to explicitly decide how disparate quantities are converted to have the same meaning. In contrast, utility analysis only requires the designer to decide how much they are willing to sacrifice on one attribute's utility scale for a unit of improvement on another attribute's utility scale. Even so, the relative scales still necessitate the use of weights. I believe this will lead to a credibility problem unless designers have a sufficient understanding of utility theory. Unfortunately, utility theory is probably not accessible to most non-technical product designers.

Since utility analysis provides a relative ranking of design alternatives, all candidates must be compared using the same criteria. Therefore, like the figure-of-merit assessment methods, it is not possible to perform studies like obsolescence example presented in section 2.2.2.

There are also important philosophical differences between utility analysis and the specification-based model. Multiple attribute utility functions assume that the best designs will have the highest expected utility (analogous to Laplace's criterion as demonstrated in equation (2.5.2)). If multi-attribute utility theory were used to combine the acceptance probabilities, the design with the highest expected probability of each requirement being acceptably met would be the best design. Alternatively, the specification-based model assumes that the best design has the highest probability of
meeting all specifications (i.e., for an additive problem the expected value is given by, $p_{\text{expected}} = \frac{1}{n} \sum_{i=1}^{n} p_i$, while the overall probability is given by $p_{\text{overall}} = \prod_{i=1}^{n} p_i$).

Further, utility analysis assumes that poor performance in one area can be compensated for by good performance somewhere else. Utility theory was developed from economic theory (von Neumann and Morgenstern 1947, p. 8), so the notion of substitutability makes sense in that context. As (Otto and Antonsson 1993a) suggest, this rationale is questionable in a product design context. For instance, an excessive stress level cannot be compensated for by good aesthetics! Multiple attribute utility functions do not exhibit the annihilation property (equation (2.6.7)), which is needed to provide any performance characteristic with the potential to disqualify a design alternative.

3.1.5. **Fuzzy Decision-making**

Fuzzy set theory did not play a role in the development of this thesis. The specification-based model is based on classical set and probability concepts. However, a number of my colleagues have questioned the apparent similarities to fuzzy sets. Further, there is a large body of fuzzy decision-making literature. Therefore, some discussion of fuzzy set theory is in order.

The goal of this section is not to provide a rigorous treatment of fuzzy decision-making. The intent is to explain the concept of fuzzy sets so that they may be contrasted with the probabilistic specification-based model proposed in this thesis. Probability and fuzzy theory seem to be similar because they both address uncertainty and use a [0-1] interval for their measures. As suggested by (Zimmerman 1991, p. 190), they are difficult to compare because fuzzy theory does not have a single unique mathematical structure (much in the same way that there are many multiple-criteria decision models). The focus here will be to demonstrate their semantic differences.

I will start by describing the concept of fuzzy sets in comparison to classical crisp sets. The entire discussion is based upon (Zimmerman 1991). Then a brief discussion of fuzzy decision-making is included. In the ensuing sub-section, the acceptability specifications are interpreted from both a fuzzy set and probabilistic viewpoint. It will be suggested that
the probabilistic interpretation is more appropriate for the specification-based design model.

The classical crisp set is dichotomous—an answer is either true or false; a value is a member of set $A$ or it is not. Thus, a crisp set has a characteristic (membership) function of [0, 1]. A crisp set may be defined as a collection of elements $x \in X$ which is finite and countable. Each element may or may not belong to a set $A$, $A \subseteq X$.

Probability is based upon classical set theory. Probability addresses stochastic uncertainty due to random variation of lack or information or knowledge about a future state (e.g. what is the likelihood that the next person to walk in the room will be over 1.8 metres tall?). However, the set membership is well defined (over 1.8 meters or not).

Fuzziness also represents certainty, but of a different type. Fuzziness is a vagueness in the description of phenomena or the semantic meaning of a description. 'Tall people' is a fuzzy set—the uncertainty lies in the class definition of the concept 'tall'. Fuzzy sets capture this imprecision in the definition of class membership. A fuzzy set would indicate the degree to which a specific height belongs to the set of 'tall people'.

Therefore, the characteristic function $\mu_{\tilde{A}}$ of the fuzzy set $\tilde{A}$ has degrees of membership that may vary from [0 through 1]. A fuzzy set consists of pairs as illustrated in equation (3.1.5.1). The first value of each pair is the set element, and the second value is the element's degree of membership.

\[
\tilde{A} = \{(1, .1), (2, .9), (3, .4)\}
\]

A fuzzy set for numbers near 10 (Zimmerman 1991, p. 13) might be defined as

\[
\tilde{A} = \\left\{(x, \mu_{\tilde{A}}(x)) \mid \mu_{\tilde{A}}(x) = \left(1 + (x - 10)^2\right)^{-1}\right\}
\]

There are many types of fuzzy sets, and fuzzy set theory uses the extension principle to develop fuzzy set operators and functions (e.g., intersection, addition, subtraction). However, for this discussion understanding the concept of fuzziness is sufficient.
Possibility theory has developed from a particular interpretation of fuzzy sets (viewing the fuzzy set as a fuzzy restriction on a variable). Under this interpretation, the membership function is a possibility distribution which indicates the possibility that a member of the fuzzy set will have a particular value (i.e., given \( x \) is a member of \( A \), what is the possibility of \( x \) being a certain value?).

\[
\pi_x = \mu_A
\]  

(3.1.5.3)

An example from (Zimmerman 1991, p. 113) should further illustrate that possibility and probability are not equivalent.

Consider the postulate "Joe ate \( x \) eggs for breakfast" \( X = \{1,2,3,\ldots\} \). The possibility distribution \( \pi_x(u) \) indicates how possible it is for Joe to eat \( u \) eggs. The probability distribution \( P_x(u) \) for how many eggs Joe will eat for breakfast may have been determined by observing a large number of Joe's breakfasts (the frequentist philosophy). Table 3.1.5.1 demonstrates this difference.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \pi_x(u) )</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.8</td>
<td>0.6</td>
<td>0.4</td>
<td>0.2</td>
</tr>
<tr>
<td>( P_x(u) )</td>
<td>0.1</td>
<td>0.8</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

This brief discussion of some of the key fuzzy set theory concepts is enough to be able to understand the semantic difference between fuzzy theory and the specification model. This difference will be described shortly.

In fuzzy decision-making, objective functions and constraints are characterized by membership functions (fuzzy sets). Since the goal is to satisfy the objectives and constraints, the 'decision' is viewed as the intersection of confluence of the fuzzy objectives and fuzzy constraints (Zimmerman 1991, p. 245). Like the specification-based model, fuzzy decision-making does not make distinctions between objectives and constraints. A decision is an aggregation of fuzzy sets, and is also a fuzzy set. The aggregation operators must be chosen to appropriately reflect trade-off and importance preferences. Thus, as seen throughout the decision theory literature, many strategies for combining the objectives and constraints are proposed. Weighting schemes are employed
frequently. If a crisp decision is desired, one might choose the alternative with the highest degree of membership in the decision set.

A detailed review of fuzzy decision making may be found in (Zimmerman, Zadeh et al. 1984; Zimmermann 1986; Lai and Hwang 1994). Additionally, a fuzzy engineering design technique called the 'method of imprecision' has been developed recently (Otto and Antonsson 1991; Otto 1992; Wood, Otto et al. 1992; Otto and Antonsson 1993a; Otto and Antonsson 1994). Work by (Koning 1991) on fuzzy material selection provides an interesting contrast to the material selection example that will be presented in chapter 5.

3.1.5.1. Relationship to specification-based model

The specification functions certainly look a lot like fuzzy set membership functions. Indeed, they might be interpreted in this manner. In this section I will interpret the specifications first as probability measures and then as fuzzy sets.

I feel that the implications of the interpretations dictate that the probabilistic model is more appropriate for this work. The specification in figure 3.1.5.1.1 will be referred to throughout the discussions.

Figure 3.1.5.1.1 A design specification

A Probabilistic interpretation of the specifications.

Designers have a clear, internal understanding of acceptability. That is, they can clearly distinguish between what they personally find acceptable or unacceptable. The word internal is an important part of this statement. It implies that the designers can judge what they find acceptable—this differs considerably from trying to 'guess' what someone else will find acceptable.
A design attribute that must be improved is unacceptable, and a design attribute that is acceptable does not require improvement. Therefore, the attribute has only two states—requires improvement or does not (a $[0, 1]$ membership function).

The specification represents the designer's uncertainty about whether he will decide if an attribute is acceptable. The specification indicates the probabilities that different performance levels will be accepted. This could be the designer's subjective probability, or could be statistically determined by means such as a poll. The point marked $(0.5, 0.6)$ means that given a design with the attribute level of 0.5, there is a 0.6 probability that the designer will decide to accept the design. The probability she will reject the design is 0.4. The probability of acceptance and its complement must sum to unity.

$$P(a) + \mathcal{C}P(a) = 1.0$$ \hspace{2cm} (3.1.5.1.1)

**A Fuzzy interpretation of the specifications.**

Designers do not have a clear internal understanding of acceptability. They cannot clearly judge what they think is acceptable. They can find a design attribute neither acceptable or unacceptable.

The specification represents uncertainty about the designer's understanding of acceptability (akin to vagueness in the definition of 'tall'). The imprecision is due to uncertainty related to grasping the concept of acceptability.

The specification represents the degree of membership that different performance levels in the fuzzy set ‘acceptable’. According to possibility theory, the value of the specification indicates, given a design is a member of the acceptable set, the possibility it will have a certain performance level. The point $(0.5, 0.6)$ indicates that the possibility of an acceptable design having a performance attribute of 0.5 is 0.6. This does not mean that 0.4 is the possibility that an unacceptable design will have a performance level of 0.5.

$$\pi(a) + \mathcal{C}\pi(a) \geq 1.0$$ \hspace{2cm} (3.1.5.1.2)

**Remarks**
The idea that a designer may have a fuzzy internal concept of acceptability seems quite reasonable at first glance. However, if a design is deemed neither acceptable nor unacceptable, what action does such a conclusion prescribe?

In reality, designers have only two courses of action. They can either leave the design as it is or they can improve it (a [0-1] membership function). Either action implies that the designer has accessed acceptability. Assuming it is the designers' responsibility to act, they must form a clear internal understanding of the notion 'acceptable'.

It is my conclusion that because all designs must fall into the unambiguous categories of needing improvement (unacceptable) or not needing improvement (acceptable), the probabilistic interpretation of the specification value functions seems more appropriate. When designers decide to iteratively alter design specifications, they are not changing their concept of acceptability. Instead, they are re-evaluating the probability that they will judge specific performance levels as acceptable.

3.1.6. Pareto Sets

Decision-making using Pareto sets (also referred to as efficient sets, undominated sets, or admissible sets) is presented briefly to contrast to the approaches described thus far. This philosophy simply states that a rational decision-maker must always choose a non-dominated solution. If a solution is undominated (Pareto optimal), becoming more desirable in one respect will require the alternative to become less desirable in some other respect (see White 1982, for detailed theory). The decision-maker must "choose" a single alternative from the Pareto set which contains all feasible non-dominated solutions. Unlike the figure-of-merit and utility-based techniques, this methodology does not try to produce a single overall metric for multiple criteria problems.

Figures 3.1.6.1, 3.1.6.2 and 3.1.6.3 (Keeney and Raiffa 1976; French 1988) should clarify the concept of dominance and the Pareto set. Consider a two attribute problem \((x_1, x_2)\) where more of each attribute is better. If the design is performing at the point labeled \(A\), the upper shaded area dominates \(A\) because any point in the region has both a better \(x_1\) and \(x_2\). Similarly, the lower left area is dominated because all \(x_1\) and \(x_2\) values are worse than \(A\).
Now, consider figure 3.1.6.2, where the feasible boundaries of a hypothetical solution space are indicated. The darkened lines form the Pareto set of non-dominated solutions. Having identified this set, the designer must use some means (intuition or perhaps even utility analysis) to pick a single solution. Figure 3.1.6.3 illustrates a discrete Pareto set.
The Pareto set is very difficult to imagine beyond 3 dimensions, but the frontier can be generated and interactively explored using linear programming techniques (for examples see Zionts and Wallenius 1976; Zionts 1987). More recently, (Charny 1989) has developed an interactive method to explore trade-off options and visualize the Pareto frontier for both dynamic and static multiple criteria problems. In his thesis the technique, called the dynamic range trade-off method, is demonstrated for controlling a boat subject to a number of goals and constraints.

### 3.1.6.1. Relationship to specification-based model

Choosing a Pareto optimal design makes sense. It implies that the maximum potential is realized for a given set of resources. Use of the proposed specification-based model does not preclude choosing a Pareto optimal solution, nor does it prevent exploration of the Pareto frontier.

However, the proposed satisfaction model employs a different philosophy. The specification-based model works on the principle that the goal of designing is to meet specifications, and once they are met further "improvements" add no additional value to the design.

Even so, if the designer is confronted with a number design alternatives that meet specifications, it makes sense to choose one of the undominated options. For all of the
search problems described in the chapters 4 and 5, there is a specification on the maximum acceptable cost (I can't think of a consumer product design problem that would not have this type of restriction). If it becomes clear that there are many possible solutions that perfectly meet the all specifications, I slowly ratchet the maximum acceptable cost down until the cost requirement (or another requirement) can no longer be perfectly met. At this point the design solution is Pareto optimal. This idea is illustrated in figure 3.1.6.1.1. This technique does not have negative implications with regard to design robustness, provided appropriate probability density functions are used to model the design's performance.

![Diagram showing the iterations of shifting cost specifications](image)

*Figure 3.1.6.1.1 Illustration of iteratively shifting a cost specification boundary to locate a Pareto optimal solution.*

### 3.1.7. Other Frameworks

As mentioned at the beginning of this chapter, there are numerous decision-models described in the literature. The previous six sections emphasized categories of decision-making that are commonly used in design (qualitative methods and figure-of-merit), had a direct role in the development of the specification model (axiomatic design and utility analysis), or provide a contrasting decision philosophy (Pareto and Fuzzy sets).

These six groups do not cover the spectrum of decision-making methodologies. It is difficult to cleanly classify many of the decision models, but in this section a loose description of other notable frameworks is provided.
Benefit-cost analysis provides a framework to logically weight the positive and negative aspects of a decision (Schmid 1989). Its philosophy dictates that present value of future monetary benefits should outweigh the expected costs. The goal is to use scarce resources to their fullest potential. The method is often applied to large scale social policy, infrastructure development, and urban planning projects. An easy-to-follow reference on cost-benefit analysis is (Sassone and Schaffer 1978). (Fabrycky and Blanchehard 1991) provide cost models for the product development life-cycle. (This reference also illustrates the frequently cited claim that 80% of a product's life-cycle cost is committed during the design process while only 20% of the cost is incurred.) Recent efforts applying benefit-cost analysis to recycling include (Chen, Navin-Chandra et al. 1993; Field, Ehrenfeld et al. 1994; Zussman, Kriwet et al. 1994). A benefit-cost model can be used with a multitude of decision frameworks. In chapter 6, a life-cycle prediction example combines a benefit-cost model with the specification-based decision model.

It should also be noted that benefit-cost analysis, like any decision-making methodology, has limitations (Campen 1986). Politicization of analyses, time discounting methods, addressing uncertainty, and measurement of non-mone\-tary benefits/costs are some of the difficult issues. Studies which assign dollar figures to environmental impact are often difficult to meaningfully apply to design problems (for example, Shapiro 1993, assigns life-cycle costs to different packaging materials). The need to consider the distribution of costs and benefits (whose costs and whose benefits) further muddies the water.

Bayesian methods use Bayes' theorem to merge knowledge from previous experience with observational data. Bayes' theorem provides the mathematical framework to modify a prior probability distribution (reflecting your degree of belief based upon previous experience) using new observations to produce an improved posterior probability distribution. Thus, the best possible estimates may be used in the decision-making process. The textbooks (Iverson 1984; Press 1989) address Bayesian inference methods. Utility theory is based upon Bayesian inference, so utility analysis may be described as a Bayesian decision method. Fuzzy sets, figure-of-merit and benefit-cost analysis are not Bayesian decision methods (French 1988).

Decision trees provide a way to model/structure multiple stage decision problems when subsequent decisions will depend upon prior actions. A tree for a single decision about reusing or discarding a product is shown in figure 3.1.7.1. Squares denote decisions and
circles represent uncertain outcomes. Typically, modeling a problem will result in many decisions and very large and unwieldy trees. Influence diagrams are an equivalent but more compact alternative notation.

The problem model may then be combined with utility analysis to maximize utility or some other measure, such as expected monetary value. The book by (Clemen 1990) details the use of decision trees and influence diagrams. Decision trees could also be used in combination with the specification-based model. An alternative generic structure for encoding design analysis problems is implemented in chapter 6.

The final related method that will be discussed is probabilistic design. This research community focuses on modeling design problems with probabilistic design variables and the development of mathematical programming techniques to optimize problems represented in this manner. Probabilistic models may result in more robust solutions—deterministic optimizations may be sensitive to stochastic variations in design variables and hence they lack robustness. Notably, the mathematical formulations for deterministic and probabilistic optimization are different—the constraints and objectives are treated uniquely. James Siddall of McMaster University has been a prominent researcher in this area (Siddall 1982; Siddall 1983; Siddall 1984; Siddall 1986). The method proposed in this thesis also uses probabilistic variables, but simultaneously permits deterministic modeling (deterministic variables are modeled using delta functions). As will be seen in chapter 4, this permits the seamless integration of deterministic and probabilistic optimization.

3.2. A discussion about weights

Figure-of-merit approaches, utility analysis, and fuzzy decision-making all use attribute weighting schemes. Throughout the development of this thesis, a point of controversy has
been that the specification-based decision model does not provide for weighting factors. As weights are difficult (if not impossible) to specify meaningfully (Steuer 1986, pp. 193-200), and tend to be regarded by designers with skepticism, this feature is perceived (by myself and designers I have spoken with) as a benefit. This section will illustrate that, if the value of the different design attributes are measured on a common absolute value scale, weights for substitution preferences or importance ranking are not needed. This point will be demonstrated first mathematically and then qualitatively.

Let us begin with a mathematical demonstration of why the specification-based model does not use weighting factors. Consider a multiple attribute problem where we wish to assess the overall value \( V \) of the design. The design's value depends upon \( n \) characteristics \( x_1, \ldots, x_n \). Each characteristic \( x_i \) has a domain of possible values \( X_i \), or \( x_i \in X_i \)—this could be \( \mathbb{R} \) or a finite set, for example. A design is then represented by \( x = x_1, \ldots, x_n \in X = X_1, \ldots, X_n \).

Suppose a value function \( u_i : X_i \) has been defined by the designer such that such that the value \( v_i \) of the \( i \)th design characteristic \( x_i \) is given by

\[
v_i = u_i(x_i)
\] (3.2.1)

Each value function is established on an individual relative scale so the values \( v_1, \ldots, v_i, \ldots, v_n \) are incommensurate. If we are rational, the \( n \) incommensurate scales must be converted to commensurate scales before they can be combined to determine the design's overall value \( V \). Presume the existence of the transformation functions \( t_1, \ldots, t_n \) such that all values \( t_i(u_i(x_i)), 0 \leq i \leq n \) are on a commensurate scale.

At this point, we will also assume that the overall metric is an additive form, as used in utility analysis and figure of merit approaches under the previously stated independence assumptions. This assumption is made because it makes the mathematical development cleaner and easier to follow. However, a similar argument can be made for multiplicative metrics like the specification-based model uses.

Given that we are using an additive metric, the overall value of the design can be computed by adding the \( n \) commensurate value indicators.
\( V(x) = \sum_{i=1}^{n} t_i(u_i(x_i)) \quad (3.2.2) \)

A first order Taylor expansion of the design's overall value is given by

\[ V(x') = V_{\text{reference}} + \sum_{i=1}^{n} \frac{\partial V}{\partial x_i} \Delta x_i \quad (3.2.3) \]

where \( \Delta x_i = x'_i - x_i^{\text{reference}} \)

Using the chain rule and equation (3.2.2), equation (3.2.3) can be rewritten as

\[ V(x') = V_{\text{reference}} + \sum_{i=1}^{n} \frac{\partial t_i}{\partial u_i} \frac{\partial u_i}{\partial x_i} \Delta x_i \quad (3.2.4) \]

Noting that \( \Delta v_i = \frac{\partial u_i}{\partial x_i} \Delta x_i \) yields

\[ V(x') = V_{\text{reference}} + \sum_{i=1}^{n} \frac{\partial t_i}{\partial u_i} \Delta v_i \quad (3.2.5) \]

Equation (3.2.5) is recognized as the utility function described in section 3.1.4 under the assumption of mutual utility (preferential) independence. The \( \frac{\partial t_i}{\partial u_i} \) terms are analogous to weights that are used to capture the substitution preferences in an additive utility function. As the transformation functions \( t_i \) are not known explicitly, the designer estimates the substitution preference weights using a lottery, under the assumption of equation (3.2.6).

\[ t_i(u_i) = w_i u_i \quad (3.2.6) \]

Thus

\[ V(x') = V_{\text{reference}} + \sum_{i=1}^{n} \frac{\partial V}{\partial x_i} \Delta x_i \quad (3.2.3) \]
becomes

\[ V_{(x')} = V_{\text{reference}} + \sum_{i=1}^{n} w_i \frac{\partial u_i}{\partial x_i} \Delta x_i \]  \hspace{1cm} (3.2.7)

\[ V_{(x')} = V_{\text{reference}} + \sum_{i=1}^{n} w_i \Delta v_i \]  \hspace{1cm} (3.2.8)

Therefore,

\[ w_i = \frac{\partial t_i}{\partial u_i} \]  \hspace{1cm} (3.2.9)

Now, let us consider the case when all of the value functions use an absolute value scale. The subjective probability that the design will be acceptable as used in the specification-based model, is such a scale. By definition, the only admissible transformation for an absolute scale is the identity transformation (French 1988, p. 328). This implies that an absolute scale is unique. Thus, if each preference function is evaluated on the same absolute scale so that

\[ t_i(u_i) = u_i \]  \hspace{1cm} (3.2.10)

giving

\[ \frac{\partial t_i}{\partial u_i} = \frac{\partial u_i}{\partial u_i} = 1 \]  \hspace{1cm} (3.2.11)

then equation (3.2.5) becomes

\[ V_{(x')} = V_{\text{reference}} + \sum_{i=1}^{n} \Delta v_i \]  \hspace{1cm} (3.2.12)

This argument effectively demonstrates that, if all the attributes are measured on a common absolute scale, substitution preference weights are not required. This result implies that the value functions are defined so that a unit change in any \( u_i \) has the same overall significance as a unit change in any \( u_j, j \neq i \). This notion was detailed in section
3.1.4 (beginning after equation (3.1.4.8)) during a discussion about the meaning of the substitution preference weights. Additionally, if each attribute's value is measured on a common but non-absolute scale weighting factors will be required. However, all of the individual weighting factors will be the same (as in Laplace's expected value criterion used in equation (2.5.2)).

The figure-of-merit approaches, described in section 3.1.3, view weighting factors as relative importance rankings. If we reconsider equation (3.2.2),

\[ V(x) = \sum_{i=1}^{n} t_i(u_i(x_i)) \]

it is apparent that the figure-of-merit approach assumes the designer can directly define the transformations \( t_1, \ldots, t_n \). These transformations are needed to convert the different scales to a commensurate scale that can be added meaningfully. Further, the figure-of-merit approaches typically assume that the form of the transformation functions is

\[ t_{i(w_i)} = w_i u_i, \text{ for } 0 \leq w_i \leq 1 \]  \hspace{1cm} (3.2.13)

Thus, for the figure-of-merit method, equation (3.2.2) becomes

\[ V(x) = \sum_{i=1}^{n} w_i u_i(x_i) \]  \hspace{1cm} (3.2.14)

However, if the functions \( u_i, 1 \leq i \leq n \) are all defined on the same absolute scale, equation (3.2.10) provides

\[ V(x) = \sum_{i=1}^{n} |u_i|_{x_i} \]  \hspace{1cm} (3.2.15)

This argument suggests that, if the value of all design attributes are measured on the same absolute scale, weighting factors need not be used. The argument shows that this is true for both weights viewed as substitution preferences and weights viewed as importance rankings. The examples above assumed (for clarity) that the overall combination form was additive. However, the argument should also generalize to metrics that
multiplicatively combine the individual ratings (as the specification-based model does). At this point, it will be helpful to give a qualitative rationale for the absence of weighting factors in the specification-based model.

The qualitative discussion will begin with an example of how the notion of importance is conveyed. Figure 3.2.1 illustrates the specification functions for two criteria. One would probably be described as “important”, and the other as “less important”. The figure illustrates that importance is captured by the range of performance over which the design alternative will be considered acceptable. A designer might consider the criterion in figure 2.3.2b as less important because it will be satisfied with a wide range of performance levels. Further, the overall acceptability declines only slightly as the design deviates from the preferred range. However, the specification also conveys that, at some extreme level, the criterion may become important in determining the overall acceptability of the design. Therefore, the absolute scale specifications portray ‘importance’.

Weights that convey the marginal substitution rates may be interpreted as a willingness to pay in terms of attribute z to improve y. In commonly used design terms, the marginal substitution rates quantify how the designer would like to “trade-off” between different objectives. When using the absolute scale acceptability specifications, this notion is captured by the relative slopes of the specification distributions. Imagine that figure 3.2.1a is the specification for attribute y and figure 3.2.1b is for attribute z. Assume that
the design is operating at the point \((y, z) = (1.1, 4.0)\). At this point \(\frac{dp}{dy} = -5\) and \(\frac{dp}{dz} = -0.05\). Therefore, the designer is willing to pay 10 units of \(z\) to improve \(y\) by 0.1 units. That is, improving \(y\) by 0.1 units increases the value \(p\) by 0.5, while paying 10 units of \(z\) decreases the value of \(p\) by 0.5. Because an absolute scale is used, the relative slopes implicitly capture the designer's substitution or tradeoff preferences.

One last claim for needing weights must still be addressed. Recall the negotiation exercise described in section 2.5. The formulation worked because the viewpoints of all team members were valued equally. What if some opinions are valued less than others (i.e., we don't really care if one of the team members is satisfied)? This situation is analogous to a designer that feels one of her specifications is not very important. The designer can correct this situation by defining a specification on how well the original requirement is being satisfied. Figures 3.2.2a and 3.2.2b illustrate this notion.

---

**Figure 3.2.2a** The original design specification.

**Figure 3.2.2b** The second specification defined by the designer conveying that satisfying the specification in (a) very important. The specification indicates the design will always be found acceptable if there is only a 50% chance of the design being acceptable according to (a). The designer would use specification (b) in the design's overall evaluation.
A Probabilistic Specification-based Design Model: applications to design search and environmental computer-aided design

From the annotation of figure 3.2.2b, what is really happening should be apparent. The designer has realized that the original specification is not truly reflecting the absolute acceptability of the design. Thus, using specification 'b' transforms the incorrect specification to the absolute acceptability scale. However, one might argue that instead of introducing the confusion associated with specifications on specifications, it is more desirable to merely adjust the original requirement directly.

In summary, considerable simplicity and ease of application is realized in the specification-based decision model by eliminating the use of weights. Simplicity and clarity are of critical importance if the method is to be used by practicing designers. This section has illustrated that eliminating weighting factors is justifiable, provided the specifications portray the overall probability of the design's acceptability.
4. Continuous search using the specification-based decision model

Chapter 3 completed the development and background work for the specification-based design decision model. To reiterate, the model was developed so that product designers might address a wide variety of product life-cycle issues using the familiar language of design specifications. However, modeling and evaluating a design will bring us only part of the way toward solving complicated and counter-intuitive system problems. To be of practical use, and thereby justify the effort required to model a problem, an analysis must provide assistance in finding better alternatives. Search capabilities are important when one is simultaneously considering many interrelated factors. Without automated search capabilities, designers may spend considerable effort randomly evaluating different variable combinations to identify ways to improve their work.

In this section an architecture and software implementation for design search is presented. If the search/optimization facility is to be a key component in a system-oriented product design tool, it must respect the designer's needs. Thus, convenient formulation and optimization of multiple criteria design problems are essential. It is not reasonable, in my opinion, to expect product designers to possess expertise in mathematical programming—designers should not be required to exhibit mastery in the area of optimization. They should be allowed to focus on their primary goal—designing products. Therefore, the software described in this section exploits the specification-based model to eliminate the distinction between single criterion and multiple criteria problems, or deterministic and probabilistic analysis. Further, the formulation of the objective function is automated. The designer can specify the search problem in the language of specifications directly, eliminating the frustrating task of adjusting the objective or penalty functions for specific problems.

One caveat is that if a 'black-box' approach to optimization is adopted, the method must be extremely robust. This is particularly true for multiple criteria solution spaces which can often have undesirable properties. A 'black-box' for design search must be able to recover elegantly from infeasible regions, successfully locate feasible regions, and avoid becoming trapped in local optima.

We will begin by introducing the key concepts of genetic algorithms and justify their use through a robustness argument. The architecture of the specification-based search facility
is outlined, and the distribution and optimization classes are described at a high level. The different distributions available for use in the optimization software are described. Next, the automatic formulation of the objective (fitness) function is detailed. Finally, a number of examples using the search facility application is presented.

4.1. Overview of genetic algorithms

The goal of this section is to briefly justify the use of genetic algorithms in this application, and then to provide an overview of the genetic algorithm for the uninitiated.

The black-box optimization software needed for convenient use by non-technical designers demands a very robust implementation. This requirement is compounded by the reality that multiple objective search spaces are frequently ill-behaved. They often have disconnected or non-convex feasible regions (thus Kuhn-Tucker conditions do not apply) (Steuer 1986, pp. 154-156). Gradient-based mathematical programming techniques had difficulties in dealing with such undesirable properties. Additionally, such optimization and search methods often enforce assumptions regarding continuity, the existence of derivatives, or unimodality. So, it appears that gradient-based algorithms do not suit the particular needs of this work.

Therefore, it is fortunate that genetic algorithms are very robust and perform well across a broad spectrum of problems (Goldberg 1989, pp. 1-6). They are not limited by restrictive assumptions about the design search space, which is important when addressing multiple criteria problems. Another positive property of genetic algorithms is that they do not require the designer to specify a feasible starting point (actually, no starting point is required at all). In contrast, gradient algorithms require a starting point. Personal experience suggests that, for complicated problems, providing a suitable starting point is not a trivial task. Overall, genetic algorithms seem to best meet the stated robustness requirement.

The genetic algorithm is patterned after evolutionary processes based upon natural selection (Holland 1975). A design alternative is analogous to an organism. The instructions that define the organism are encoded by a chromosome. In genetic algorithms, the chromosome is commonly a binary string. Each character position or group of positions in the chromosome is analogous to a gene that encodes a characteristic of the organism. In figure 4.1.1, a location of an eye color gene is highlighted. The
specific value of the character or characters within the gene, known as alleles will determine the individual expression for the trait encoded by the gene.

The eye color gene highlighted in the figure contains two alleles: '0' and '1'. Imagine that '0' means blue and '1' means brown. Thus, the genotype for this organism's eye color is '01'. Now, suppose that the individual with this genotype actually has brown eyes (implying brown dominates blue). Then, '1' is the individual's phenotype (expressed form). Although not essential, familiarity with biological genetics provides further insight into genetics algorithms. The curious will find (Suzuki, Griffiths et al. 1989) is a good introductory text.

![Figure 4.1.1 Illustration of the genetic representation of a hypothetical organism.](image)

Genetic algorithms optimize through a process that mimics evolution—breeding produces new variations while selection eliminates weak solutions. A population consists of a number of chromosomes, each of which encode a possibly optimal design.

Each “generation” of search begins by selecting the most highly fit designs in the population to serve as parents that produce off-spring. Parent selection is performed on a weighted roulette wheel, where the more fit individuals will have a greater chance of being selected. The parents are paired and mate through genetic operators, such as crossover, to create two child designs that possess traits from both parents. When crossover occurs, the two parents swap parts of their chromosomes. The frequency of crossover is determined probabilistically, so in some instances crossover will not occur. After mating, the resultant child organisms are then subjected to infrequent random mutations. This process of selection, breeding and mutation is then repeated for subsequent generations. The search cycle is illustrated in figure 4.1.2.
After successive generations, the overall quality of designs should increase because better designs are more likely to propagate and exchange desirable qualities. The mathematical foundations for this statement are formalized in schemata theory, which is described in (Goldberg 1989).

When using the genetic algorithm, several parameters, all of which affect search performance and efficiency, must be specified. These parameters include the probability of crossover, the probability of mutation and the population size. Additionally, a fourth parameter, the fitness scaling coefficient, is used to prevent the search from rapidly converging onto a single sub-optimal organism. The value of the fitness scaling coefficient can be interpreted as the number of times one can expect the most fit organism to be chosen as a parent for the next generation. Tests conducted to determine the best overall values of these parameters for the formulation used in this thesis are provided in table 4.1.1. These values generally agree with those recommended by (Goldberg 1989) based upon the work of (DeJong 1975).

Table 4.1.1 Parameter values chosen for the genetic algorithm

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Probability of Crossover</td>
<td>0.80</td>
</tr>
<tr>
<td>Probability of Mutation</td>
<td>0.031</td>
</tr>
<tr>
<td>Population Size</td>
<td>30</td>
</tr>
<tr>
<td>Fitness Scaling Coefficient</td>
<td>2.0</td>
</tr>
</tbody>
</table>
For the application presented in this section, a randomly seeded initial population, binary chromosomes, single-point crossover, point mutation, stochastic selection, and an elitist model are used. The elitist model ensures that the best organism in a given population survives unaltered into the next generation. The genetic algorithm software library was written by Kazu Saito, a graduate student in the MIT CADlab, for use by Professor Mark Jakiela in topology optimization research (Chapman, Saitou et al. 1994a).

4.2. Search facility software architecture

This section will describe the software architecture developed to use specification-based model for design search and optimization. The architecture will result in an optimization tool which is consistent with the goals outlined in the beginning of chapter 4. The section provides an overview of how a search problem is formulated and how the optimization process is executed. The software is implemented in an object-oriented fashion using C++, so a description of object classes used in the implementation is provided. An understanding of the object-oriented design concepts is assumed.

To perform an optimization, the problem is defined in terms of design variables, performance variables and specifications against which performance variables are measured.

Design variables are simply the characteristics of the design that may be altered (optimized) during the search process. These independent variables are the distributions that are encoded on the chromosome. Performance variables are the design's properties that will be used to evaluate different alternatives. Performance variables may be actual design variables, or they may be calculated from a number of variables. The specifications indicate the probability that design alternatives with different levels of the performance variables will be judged as acceptable. Specifications would usually be defined by the designer and be held constant for a given problem. However, specifications could also be design variables in a policy setting context (this is demonstrated in section 4.6). When using the specification-based model, the design variables, performance variables and specifications are all distributions.
To formulate an optimization problem, the designer must define these three lists of distributions: a list of design variables and the boundaries within which the search should be contained, a list of performance variables, and a list of specifications. Once the design variables, performance variables and specifications are defined, no further problem specific input is required. The optimization module is simply passed the variable lists when it is called. Within the optimization module, an appropriate chromosome is defined to encode the parameters for the different distributions, and then the genetic algorithm is invoked.

After every generation, each organism in the population is evaluated. The chromosomes are decoded and the parameters that define design variable distributions are updated accordingly. The performance variables automatically recalculate themselves based upon the new design variables, and then fitness is evaluated based upon the likelihood of meeting specifications.

The search is terminated as soon as a solution that satisfies all specifications is found, or when enough generations have passed that further solution improvement is unlikely. When the genetic algorithm terminates, the design variables are updated to their optimal values, and the results are output to the screen.

4.3. The distribution classes

All variables and specifications used in the decision model are distributions. Three different types of distributions have been implemented for use in the system: delta functions, beta distributions and piecewise linear distributions. The three distributions are illustrated in figure 4.3.1.

Figure 4.3.1 Illustration of the three distributions available in the search implementation.
All three distributions are derived from a general distribution class, as depicted by the simplified description in figure 4.3.2. This architecture allows the optimization software to manipulate distributions without distinguishing between the delta, beta or piecewise linear classes. The data defined in the base class—the parameters that define a given distribution and its search limits for the distribution—are inherited into the derived classes. The number of data and their meaning are determined by the derived class (see sub-sections 4.3.1-4.3.3).

Class Distribution

**data:**
distribution parameter list
distribution search limits

**methods:**
virtual computeProbability(specificationDistribution)
virtual computeDistributionFrom()
virtual encodeChromosomeSegment()
virtual decodeChromosomeSegment()

<table>
<thead>
<tr>
<th>Class Delta</th>
<th>Class Beta</th>
<th>Class Linear</th>
</tr>
</thead>
<tbody>
<tr>
<td>methods:</td>
<td>methods:</td>
<td>methods:</td>
</tr>
<tr>
<td>computeProbability(spec)</td>
<td>computeProbability(spec)</td>
<td>computeProbability(spec)</td>
</tr>
<tr>
<td>computeDistributionFrom()</td>
<td>computeDistributionFrom()</td>
<td>computeDistributionFrom()</td>
</tr>
<tr>
<td>encodeChromosomeSegment()</td>
<td>encodeChromosomeSegment()</td>
<td>encodeChromosomeSegment()</td>
</tr>
<tr>
<td>decodeChromosomeSegment()</td>
<td>decodeChromosomeSegment()</td>
<td>decodeChromosomeSegment()</td>
</tr>
</tbody>
</table>

*Figure 4.3.2 The distribution and its derived classes.*

As an object-oriented approach is employed, individual distributions contain the methods needed to use the specification-based model. The method `computeProbability(specificationDistribution)` automatically normalizes the distribution to a unit area (as a probability density function must be), and then integrates the distribution using the specification argument as a weighting function. A second order Runge-Kutta integration algorithm is used (1988, pp. 567-574). This calculation gives the probability of the design being acceptable as defined by the specifications (equation (2.2.1)).

The method `computeDistributionFrom()` allows the user, if necessary, to define a function that will compute the distribution parameters from other variables (this is needed for
performance distributions, which are often dependent variables). The distribution will automatically update itself using this function as required during the optimization. Importantly, the update function could also be used to obtain values from programs that perform specialized analyses, such as FEM packages.

Finally, each class of distribution knows how to encode itself onto a chromosome, and decode and update itself from a chromosome segment. Given that the distribution parameters are represented using a binary chromosome, the design search space will grow according to equation (4.3.1).

\[
    s = 2^n \left( \sum_{j=1}^{n} b_j \right)
\]

(4.3.1)

where

\[
    s = \text{the number of points in the search space} \\
    n = \text{the number of design variables} \\
    m = \text{the number of parameters needed to encode the } i^{th} \text{ design variable} \\
    b_i = \text{the number of bits used for the } i^{th} \text{ parameter}
\]

In the following sub-section III each distribution is described and its chromosome encoding is detailed.

4.3.1. The delta function class

The delta function exists at only a point and has infinite magnitude and an area of unity, as defined in equation (4.3.1.1).

\[
    \delta(x - a) = 0 \quad \text{for } x \neq a, \\
    \delta(x - a) = \infty \quad \text{for } x = a, \\
    \int_{-\infty}^{\infty} \delta(x - a) dx = 1
\]

(4.3.1.1)

The probability that a variable represented by a delta function will have the value \( a \) is 1.0. For example, if the designer knows the cost of a part is 50.56 cost units with complete
certainty, the cost distribution would be a delta function located at 50.56. The delta function is used to represent deterministic design variables in the decision model as

\[ \int_{-\infty}^{\infty} \delta_{(x-a)} s(x) \, dx = s(a) \]  \hspace{1cm} (4.3.1.2)

where \( s(x) \) is a design specification value function.

The delta function is completely specified by the single parameter \( a \) (the "spike's" location). The search limits for the delta distribution are the upper and lower boundaries for \( b_{\text{lower}} \leq a \leq b_{\text{upper}} \).

4.3.1.1. Chromosome encoding and decoding

The delta distribution is completely described by the single parameter \( a \), so it may be represented on a binary chromosome as shown in figure 4.3.1.1.1.

![Figure 4.3.1.1.1 Binary chromosome representation of a delta distribution.](image)

The distribution is encoded by adding a segment \( n \) bits long to the chromosome. The number of bits will determine the resolution of the variable. For example, if represented using a four bit segment, there are 16 possible decimal values for \( a \). If the designer specified the search limits \( 0 \leq a \leq 32 \), then the parameter's resolution is two.

A delta function is decoded from a chromosome by first converting the appropriate binary segment to a decimal value and then scaling between the desired search limits. This yields the value of the parameter \( a \), which is the distribution's genotype. The phenotype is the actual delta function located at \( a \).
4.3.2. The beta function class

The beta function is a very flexible probability density function that can appear like a normal distribution or be continuously skewed to the left or the right. The peakiness of the distribution may also be varied from a square function to a narrow spike. This type of function is well suited to represent probabilistic design variables.

The beta function (Scheaffer and McClave 1986, pp. 177-179) is defined in equation (4.3.1.1).

\[ f(x) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{(\alpha-1)}(1-x)^{(\beta-1)} \quad 0 < x < 1 \]  

(4.3.1.1)

where the gamma function is given by

\[ \Gamma(n) = \int_{0}^{\infty} x^{n-1}e^{-x}dx \]  

(4.3.1.2)

The parameters \( \alpha, \beta \) determine the peakiness and skew of the distribution. For a unimodal distribution \( \alpha, \beta \) must be greater than or equal to one.

The beta function is defined over the range of 0 to 1, so two more parameters, the starting and ending points \( s, e \), are required. The two points are used to translate and scale the distribution over different ranges.

Thus, the beta distribution is completely defined by four parameters, \( \alpha, \beta, s, \) and \( e \). The search limits for the distribution define the range that the distribution must fall within \( (b_{\text{lower}}, b_{\text{upper}}) \) and the upper and lower limits for the parameters \( \alpha, \beta \) (typically 1-5).

4.3.2.1. Chromosome encoding and decoding

The beta distribution is completely described by four parameters, so it may be represented on a binary chromosome as shown in figure 4.3.2.1.1.
The distribution is encoded by adding a segment $n$ bits long to the chromosome. This length is determined by adding the number of bits that will be used to encode each of the four parameters.

The beta distribution is decoded from a binary chromosome segment by first converting the distribution parameters to a decimal values scaled between the desired search limits. The distribution's end point $e$ is then further scaled to lie within the range $s < e \leq b_{upper}$. This gives the distribution's genotype $\alpha\beta\sigma\epsilon$.

### 4.3.3. The piecewise linear distribution class

Piecewise linear distributions are intended primarily for defining design specifications (as has been the case for all of the examples in this thesis). However, they may also be used to approximate unusual probability density functions.

The piecewise linear distribution is defined by a finite number of $n$ bounded line segments, as suggested in equation (4.3.3.1).

\[
\begin{align*}
f_{(x)} &= a_{i}x + c_{i}, \quad b_{i} \leq x < b_{i+1} \\
f_{(x)} &= a_{2}x + c_{2}, \quad b_{2} \leq x < b_{3} \\
f_{(x)} &= a_{n}x + c_{n}, \quad b_{n} \leq x \leq b_{n+1}
\end{align*}
\tag{4.3.3.1}
\]

The search limits are the extreme lower and upper range $(b_{lower}, b_{upper})$ that the distribution must lie within and the maximum and minimum values for $f(x)$ ($y_{min}$, $y_{max}$).
For a specification indicating the probability of acceptance, $f(x)$ must be confined between 0 and 1.

### 4.3.3.1. Chromosome encoding and decoding

The piecewise linear distribution is represented on a chromosome by a series of pairs $(x_i, y_i)$ that define the start and end points of the line segments in the distribution. Thus a distribution made of $n$ segments will require $n+1$ pairs on the chromosome. The chromosome for a distribution of two line segments is depicted in figure 4.3.3.1.1.

![Figure 4.3.3.1.1 Binary chromosome representation for a two segment piecewise linear distribution.](image)

The distribution is encoded by adding a segment $n$ bits long to the chromosome. The length is determined by adding the number of bits that will be used to encode each of the pairs needed to define the distribution.

The beta distribution is decoded from a chromosome segment by first converting the binary parameters to decimal values scaled between the desired search limits. Thus, for a specification, all parameters for $y_i$ are scaled between $y_{min} = 0, y_{max} = 1$. Each value $x_i$ is scaled first to lie between the specified search range limits. The $x_i$ values are then accumulated and scaled to calculate the line segment partitions $b_i$ such that $b_{lower} \leq b_1 < b_2, \ldots, < b_{n+1} \leq b_{upper}$.

For example, the two line segment distribution boundaries would be calculated as
\[ b_1 = x_1 \]
\[ b_2 = b_1 + \left( \frac{b_{upper} - b_1}{b_{upper} - b_{lower}} \right) x_2 \]  \hfill (4.3.3.1.1)
\[ b_3 = b_2 + \left( \frac{b_{upper} - b_2}{b_{upper} - b_{lower}} \right) x_3 \]

Finally, the parameters \( b_i, b_{i+1}, y_i \) are used to calculate the line segment parameters \( a_i, c_i \) for each line segment \( i, 1 \leq i \leq n \).

### 4.4. The optimization class

The optimization class is designed to allow the transparent operation of the search facility. A simplified outline of the data structure is provided in figure 4.4.1. On the creation of an object of class optimization, the search problem is automatically formulated, executed, and final results are displayed. (The camelback example in section 4.6.1 will clarify this).

As mentioned in section 4.2, an optimization requires: a list of the design variables that are to be optimized; a list of performance variables; and a list of the specifications that will be used to evaluate the performance variables. An optimization object is passed the variable lists when it is created.

---

**Class Optimization**

**data:**
- list of design variable distributions
- list of performance variable distributions
- list of specification distributions
- probability of crossover
- probability of mutation
- population size
- fitness scaling coefficient

**methods:**
- encodeChromosome()
- performSearch()
- outputBestResult()

*Figure 4.4.1 A simplified outline of the optimization class.*
First, the `encodeChromosome` method determines the length of the chromosome and the different parameters that need to be encoded. This is accomplished by having each distribution in the design variable list define its own chromosome segment (as described in section 4.3).

The optimization object then initializes the genetic algorithm software with the information about the chromosome and the genetic algorithm parameters (such as the probability of crossover). The genetic algorithm library creates a population of appropriate chromosomes with randomly seeded initial values.

The search is executed under the control of the optimization object. The optimization object instructs the genetic algorithm to create a new generation. The genetic algorithm pairs and mates chromosomes, evaluates the fitness of the new chromosomes, and then chooses the organisms that will survive to produce the subsequent generation. How the genetic algorithm evaluates the fitness of organisms is described in the following section. After each generation, the optimization object checks to see if a perfectly acceptable solution was found in the new generation. If a perfect solution exists in the population, the design variables and performance variables are updated to their optimal values, output to the user, and the search is terminated. If a perfect solution has not been found, the genetic algorithm is instructed to produce a new generation of organisms and the cycle is repeated.

What if it is impossible to find a solution that will completely meet all specifications? The stop criterion just described addresses only the situation where an ideal solution exists. How many generations are required until it is safe to assume the best result is not going to further improve? This is not an easy question to answer. Solution quality-based convergence criteria do not work well since the genetic algorithm may proceed several generations without improvement and then sharply improve over just a few generations. Several rules of thumb based upon chromosome length were also tested with poor results. The number of generations needed to find the best solution varies widely because the initial population is randomly seeded and the pairing/mating process is probabilistic. Another possibility is use the population's genetic diversity as a convergence criterion and stop when to population is nearly uniform.
Thus, if an ideal solution is not found, it seems reasonable to let the user decide if the search has proceeded for a sufficient number of generations. The best organism's probability of being acceptable is continually displayed so that the user can observe solution improvement and terminate the search as desired.

4.5. Objective formulation—design fitness

The selection of parents in the genetic algorithm requires that, for all generations, each candidate design is evaluated by a fitness function. The fitness function is extremely important because it determines which design alternatives are the most suitable parents to spawn subsequent generations. A poor fitness function will translate into poor optimization results. Unfortunately, finding an appropriate fitness function for a given objective is difficult. Fitness functions are usually developed through a trial and error process (as seen in Chapman and Jakiela 1994b). This problem is exacerbated when constraints must be added to the formulation. Representing constraints is often difficult (see for example, Richardson, Palmer et al. 1989; Michalewicz and Janikow 1991).

Importantly, the specification-based model eliminates problem specific fitness function and constraint formulation. When using the model, fitness naturally should be a measure of the probability of meeting design specifications. Recalling equation (2.2.2), the best organisms are those with the highest probability of meeting specifications.

\[ P_{\text{perform acceptably by all specifications}} = \prod_{i=1}^{n} p_i \]

Where

\[ n = \text{number of design specifications or criteria} \]

and

\[ p_i = \text{probability of being acceptable as defined by the } i^{\text{th}} \text{ specification.} \]

However, initial tests using this fitness measure showed that the population quickly converged to sub-optimal solutions. The problem with using equation (2.2.2) as a measure of organism fitness directly is that it is too severe. Consider a problem that must satisfy ten specifications. Assume that organism A meets nine specifications perfectly (\( p = 1.0 \)) but fails to meet one requirement (\( p = 0 \)). A second organism B is unacceptable
according to all specifications. The overall form (phenotype) for the two solutions is identical \( p_{\text{acceptable}} = 0 \), but clearly the genetic material (genotype) of \( A \) is superior. Further, an organism \( C \) with \( p = 0.1 \) for all ten objectives has a phenotype superior to design \( A \) \( p_{\text{acceptable}} = 1.0 \times 10^{-10} \) even though it meets all specifications very poorly.

A genetic algorithm should select the best genotypes, which are not necessarily the best phenotypes, to produce offspring if good solutions are to be found. Logically, the fitness function should recognize the potential of individual characteristics even if, as a totality, the organism is a failure. If we cannot discriminate at the gene level, selection should at least recognize the potential of different traits (i.e., individually consider how well each specification is met). Natural selection works in this manner. Different traits possessed by an organism can improve its chance of reproducing (for example, size and aggressiveness are often important traits in species where males must compete for females).

To resolve this dilemma, a logarithmic additive fitness function based upon the idea of information content, discussed in section 3.1.1.1, is used as a measure of fitness.

\[
I = \sum_{i=1}^{n} \log_2 \left( \frac{1}{p_i} \right) 
\]

(4.5.1)

Where

\( n = \text{number of design specifications or criteria} \)
\( p_i = \text{probability of being acceptable as defined by the } i^{\text{th}} \text{ specification} \)
\( I = \text{design information content in bits} \)

The fitness function using the information content is described by the following pseudo-code.

```plaintext
for (all distributions in the design variable list) {
    update the design variable by decoding the appropriate chromosome segment
}
ZeroProbabilityEquivalent = 0.001
MaximumFitness = \log_2 \left( \frac{1}{\text{ZeroProbabilityEquivalent}} \right) \times \text{number of specifications}
fitness = MaximumFitness
for (all distributions in the performance variable list) {
    recompute performance distribution if it depends on the design variables
    recompute the specification distribution if it depends on the design variables
    compute the probability the performance variable satisfies its specification
    if \{probability of meeting the specification < ZeroProbabilityEquivalent\} {
        fitness = fitness - \log_2 \left( \frac{1}{\text{ZeroProbabilityEquivalent}} \right)
    }
    else {
        fitness = fitness - \log_2 (1/\text{probability of meeting the specification})
    }
}
if (fitness equals MaximumFitness) organism meets all specifications perfectly
```
Using this algorithm, each specification makes an additive contribution to the fitness measure. An organism is first assigned a fitness level calculated so that it will have a zero fitness if it fails to meet all specifications. The information associated with each specification is computed and subtracted from the organism's fitness. This is done to convert the information-based metric to a maximization problem (as the goal is to minimize information content and therefore maximize the probability of having an acceptable design). A probability of 1.0 has an information content of zero, so a perfectly met specification will not reduce the organism's fitness. A probability of 0 has infinite information content. Thus, the zero equivalent probability (p=0.001) sets the maximum that can be deducted from the fitness score. Testing indicates that the algorithm's performance is relatively insensitive to the value of the zero equivalent.

This algorithm allows the fitness measure to reflect that some specifications are well satisfied even though others may be missed completely. If the algorithm is used to evaluate the three design candidates discussed above, design A (p = 1.0 for nine specifications and p = 0 for one) has a fitness 89.7, while design B (p = 0 for all specifications) has a fitness of 0, and design C (p = 0.1 for all specifications) is rated at 66.45. This result reflects that even though A is not an acceptable design, it contains desirable genetic material for a large portion of the problem. In this case, design A would have a greater chance of producing offspring than design C, and design B would be eliminated from the population.

The fitness of a perfectly acceptable solution is calculated at the onset (MaximumFitness) and then penalized when specifications are not met. Therefore, it is easy to detect when a perfectly acceptable solution has been found and to automatically terminate the search. The computational requirements of computing design fitness scales according to equation (4.5.2).

\[ C \alpha \left( \sum_{i=1}^{n} c_{d_i} + \sum_{j=1}^{r} c_{s_j} \right) \]  

where
\( C = \) computation for fitness evaluation
\( n = \) the number of design variables
\( c_d = \) the computation to decode and update the \( i^{th} \) design variable
\( r = \) the number of design specifications and performance variables
\( c_s = \) the computation to update the \( i^{th} \) performance variable and evaluate its specification

In summary, using the specification-based model eliminates both problem specific adjustment of the fitness function and the separate treatment of objectives and constraints. The logarithmic penalty function allows the genetic algorithm to identify individuals with desirable genetic material. Testing demonstrates that this prevents the algorithm from rapidly converging to sub-optimal solutions.

4.6. Search examples

The previous five sections of this chapter have conceptually addressed the design, implementation and operation of the optimization software. In this section the goal is to use the software to solve a number of problems. The problems are chosen so that they are easy to follow, and thus should clearly illustrate how design problems are formulated using the specification-based model. Even though the problems are easy to follow, they display properties that present challenges to gradient-based search algorithms.

The examples presented in this section are typical engineering problems and, for the most part, do not have an environmental component (the policy setting section is an exception). However, this search facility is an important cornerstone for the system-oriented design tool. Environmental applications are emphasized in later chapters, but if the model is to be used for truly system-oriented product design it must perform equally for traditional design problems.

The first example involves maximizing a function with several local optima, while the next section illustrates choosing parameters for a simple truss design. A policy setting problem is used to illustrate specification optimization, and then a final example shows how probabilistic variables can significantly change the optimal solution.

When reading graphs with superimposed performance probability density functions and design specifications, the vertical scale pertains to the specification function only. All
performance probability density functions are automatically normalized to a unit area when equation (2.2.1) is evaluated.

4.6.1. Optimization

In this example a test function known as a camelback (obtained from Yao 1989) is minimized. The reason for the function's name is self evident upon viewing figure 4.6.1.1. This function has six minima, two of which are global ($F = -1.0316$ at $(-.0898, 0.7126)$ and $(0.0898,-0.7126)$).

This section provides the first example of how a problem is defined using the specification-based search software. Further, it illustrates that the search algorithm is not trapped in local minima.

![Diagram of the six hump camelback function](image)

*Figure 4.6.1.1 The six hump camelback function (note that the negative z axis points up).*

The first step of any problem is to develop a model. The camelback function is defined by equation (4.6.1.1).
\[ F = \left[ 4 - 2.1x_1^2 + \frac{x_1^4}{3} \right] x_1^2 + x_1x_2 + \left[ -4 + 4x_2^2 \right] x_2^2 \] (4.6.1.1)

The goal of the search is to find the variables \( x_1 \) and \( x_2 \) that minimize the function \( F \). The optimization software needs three lists of distributions: a design variable list, a performance variable list and a design specification list.

The design variables are the independent design variables—in this instance \( x_1 \) and \( x_2 \). Therefore, the design variable list will contain the two distributions \( x_1 \) and \( x_2 \). The design variables, their distribution type, and their search limits are stated in table 4.6.1.1. The design variables are deterministic (exist at a point with complete certainty) so they are represented by delta functions.

<table>
<thead>
<tr>
<th>Design Variable</th>
<th>Distribution Type</th>
<th>Search Boundaries</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>delta function</td>
<td>(-3 &lt; x_1 &lt; 3)</td>
</tr>
<tr>
<td>( x_2 )</td>
<td>delta function</td>
<td>(-2 &lt; x_2 &lt; 2)</td>
</tr>
</tbody>
</table>

The performance variable list for the problem is just a single distribution \( F \)—a delta distribution that is located at the value of \( F \). This location is computed from \( x_1 \) and \( x_2 \) using equation (4.6.1.1). Finally, the specification list will contain the distribution that will be used to evaluate the acceptability of the performance variable \( F \). To minimize \( F \), the specification in table 4.6.1.2 indicates that only performance levels below the function's known minimum value are acceptable.
Table 4.6.1.2 Performance and specification for camelback minimization test problem.

<table>
<thead>
<tr>
<th>Performance Variable</th>
<th>Distribution Type</th>
<th>Specification Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>delta function</td>
<td></td>
</tr>
</tbody>
</table>

At present, the user must define this problem by writing and compiling a small C++ file. An outline of the source code, conveying the essence of defining and performing the optimization, is included in appendix A. Code will not be provided for other example problems. However, the software has been implemented with the intention that, when a complete computer-aided design system is implemented, problem definition will occur through a graphical user interface. This interface is an essential component if practicing designers are going to work with the software.

When the search is executed, the software exhibits robustness by successfully finding one of the two global optima on every test run. An example of the optimization software's output is also provided in appendix A. Over 20 optimizations, the average deviation of x1 from the solution was 0.0018, while the average deviation of x2 was 0.0011. A search over 100 generations takes just over a second on a Silicon Graphics Indigo² extreme (an 85 MIPS machine). Each design distribution was specified with 20 bit resolution, so the search space contains \( \approx 1.1 \times 10^{12} \) points \((2^{40})\).

The convergence of the search towards the optima is shown as a function of generation number in figure 4.6.1.2.
Figure 4.6.1.2 Fitness of the best solution in the population versus the generation number.

Figure 4.6.1.2 clearly indicates that the algorithm rapidly converges to the neighborhood of the optima (about 10-15 generations or 300 function evaluations), but obtaining the exact solution takes many generations. This observation is typical for most problems solved using genetic algorithms.

However, the concept of optimality, which implies finding the single best solution, runs counter to the philosophy of the specification-based decision model. Design is viewed as a satisfaction problem. The goal is not to locate an optimal solution, but to find solutions that satisfy the specifications. Thus, the slow convergence to exact solutions should not present a problem. If exactness were required this search formulation could reliably locate feasible starting points for other mathematical programming techniques.

### 4.6.2. Deterministic Truss Parameter Search

The camelback minimization example clarified how a problem is formulated in the specification-based search software. The example also demonstrated that the algorithm does not tend to converge to local minima. In this section, a truss design problem illustrates deterministic parameter selection. A similar problem is formulated for traditional mathematical optimization in (Arora 1989. pp. 23-31). The problem also demonstrates that the algorithm behaves well even if acceptable regions of the solution space are disconnected.
Consider the symmetric truss in figure 4.6.2.1. The truss is to be made of medium strength structural steel. We will assume that the load $W$ and truss height $h$ are specified. The designer wishes to determine the truss base $s$ and the inside and outside diameters of the members 1 and 2. An appropriate list of design variables for this problem is provided in table 4.6.2.1. Like the camelfack example, probability density functions for deterministic variables are delta functions (so that the likelihood of the variable occurring at a point value is 1.0). The variable ranges or boundaries over which the search is to be conducted are also defined. For example, the outside diameter of the members must lie between 0 and 100 mm.

![Truss Configuration](image)

**Truss Configuration**

*Figure 4.6.2.1 Definition of a simple two member truss problem (gravity is assumed to be zero to simplify buckling calculations).*

**Table 4.6.2.1 Design variables or parameters to be optimized in the truss problem.**

<table>
<thead>
<tr>
<th>Design Variable</th>
<th>Distribution Type</th>
<th>Search Boundaries</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_{1\text{(outside)}}$</td>
<td>delta function</td>
<td>$0 &lt; d_{1\text{(outside)}} &lt; 100\text{mm}$</td>
</tr>
<tr>
<td>$r_{1} = d_{1\text{(inside)}}/d_{1\text{(outside)}}$</td>
<td>delta function</td>
<td>$0 &lt; r_{1} &lt; 1$</td>
</tr>
<tr>
<td>$d_{2\text{(outside)}}$</td>
<td>delta function</td>
<td>$0 &lt; d_{2\text{(outside)}} &lt; 100\text{mm}$</td>
</tr>
<tr>
<td>$r_{2} = d_{2\text{(inside)}}/d_{2\text{(outside)}}$</td>
<td>delta function</td>
<td>$0 &lt; r_{2} &lt; 1$</td>
</tr>
<tr>
<td>$s$</td>
<td>delta function</td>
<td>$1000\text{mm} &lt; s &lt; 3000\text{mm}$</td>
</tr>
</tbody>
</table>
The next step is to identify the design requirements and form specifications indicating acceptability in terms of the performance variables. The performance variables are evaluated against specifications to determine the likelihood that a given design alternative will be judged as acceptable. A possible set of performance indicators and corresponding specifications are in table 4.6.2.2.

**Table 4.6.2.2 Performance variables and specifications for the truss problem.**

<table>
<thead>
<tr>
<th>Performance Variable</th>
<th>Distribution Type</th>
<th>Specification Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n_{a1} ) (stress safety) member 1</td>
<td>delta function</td>
<td>[\text{stress safety specification}]</td>
</tr>
<tr>
<td>( n_{a2} ) (stress safety) member 2</td>
<td>delta function</td>
<td>same as stress safety specification for member 1</td>
</tr>
<tr>
<td>( n_{B1} ) (buckling safety) member 1</td>
<td>delta function</td>
<td>[\text{buckling safety specification}]</td>
</tr>
</tbody>
</table>
Table 4.6.2.2 (continued) Performance variables and specifications for the truss problem.

<table>
<thead>
<tr>
<th>Performance Variable</th>
<th>Distribution Type</th>
<th>Specification Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>r1 member 1</td>
<td>delta function</td>
<td></td>
</tr>
<tr>
<td>r2 member 2</td>
<td>delta function</td>
<td>same as diameter ratio specification for member 1</td>
</tr>
<tr>
<td>C (cost)</td>
<td>delta function</td>
<td></td>
</tr>
</tbody>
</table>

The first two performance variables $n_{a1}, n_{a2}$ are stress safety factors that depend upon the design variables (all equations are provided in appendix B). Similarly, $n_{u1}$ is a design safety factor for buckling. The cost $C$ is calculated from the volume of material used. Rod is assumed to be 1/2 the price of pipe on a unit volume basis. Finally, the specifications on $r_1$ and $r_2$ (which are design variables) indicate that the designer prefers either solid rod or pipe. This specification illustrates that the designer may indicate preferences for both specific performance levels and values of design variables. However, the above method to specify a preference for either rod or tube is not ideal—considerable time may be spent searching a wide range of unacceptable diameter ratios. As will be discussed later, this formulation creates a discontinuous acceptable solution space. A better approach is also demonstrated later.
Having defined the truss design example, it is notable that the problem has not been described in terms of an objective or multiple objectives and constraints. The distinction between objectives and constraints does not exist. Both are characterized in terms of performance/specification pairs. Traditionally, constraints define what is unacceptable, while objectives express desired results. Using our approach, the designer has specified what they want—levels of performance that are necessary for the design to be considered acceptable. Therefore, we refer to all specifications as objectives.

Results for the solved truss problem are shown in figures 4.6.2.2. Member 1 is under compression and thus is tubular (for buckling stiffness) while the tensile member 2 is made of less expensive rod stock. Convergence to a solution typically occurred in 80 generations requiring 2.5 seconds on a Silicon Graphics Indigo² Extreme. This search involved evaluation of 2400 candidate designs (30 per generation). The search space for this problem contains \( \approx 1.1 \times 10^{12} \) points (5 parameters at 8 bit resolution gives \( 2^{40} \) points).

![Truss Configuration](image)

**Figure 4.6.2.2a** Configuration of the optimized truss design.
Figure 4.6.2.2b Comparison of truss performance variables to design specifications.
However, it must be noted that the search converged to the better solution using rod stock for member 2 in only 17% of 60 consecutive optimizations. The more common and nearly as good solution is typified by figure 4.6.2.3.

![Diagram showing truss configuration and dimensions]

**Truss Configuration**

*Figure 4.6.2.3* A result that did not find the better solution using cheaper rod stock for member 2 (that does not have buckling requirements).

The cost is 56.73 units, giving $P_{\text{acceptable}} = 0.9728$ (compared to 0.9958 for the solution in using rod for member 2). $n_{\sigma_1} = 7.49$, $n_{\sigma_2} = 2.01$, $n_{B_1} = 2.50$. The cost specification cannot be completely satisfied.

The search does not always find the best possible solution because the specification for the diameter ratio in table 4.6.2.2 was devised to create a discontinuous solution space. The two-dimensional diameter ratio space is shown in figure 4.6.2.4. The solid areas in the plot are acceptable according to the diameter ratio specifications.
The solution region using pipe has a larger acceptable region (pipe diameter ratios of 0.8 to 1.0 are completely acceptable, while only a ratio of 0 is perfectly acceptable for rod) and yields only slightly poorer results. If a gradient-based programming technique were used, the search would not find the rod solution unless the designer provided this as a starting point. When using the specification-based search program, no starting points are specified by the designer.

Given this illustration of the algorithm's performance when the acceptable solution space is disconnected, let us consider how the pipe/rod preference may be expressed more intelligently.

The resolution of the diameter ratio design variables can be changed to significantly reduce the search space for the truss example (in the example just presented, 8 bit resolution was used to encode each variable). Suppose that we are only interested in using either rod stock or pipe. Pipe typically has a diameter ratio \( r \) (ratio of inside diameter to outside diameter) of 0.88, while rod stock has a diameter ratio of 0. We could specify a resolution of 1 bit and search limits of 0 and 0.88. This approach would eliminate the need to consider the diameter ratio specifications and the pointless search of unacceptable intermediate diameter ratios. When 1 bit resolution is specified for design variables \( r_1 \) and \( r_2 \), the search consistently converges to a solution using pipe for member 1 and inexpensive rod for member 2 (see figure 4.6.2.5).
Truss Configuration

Member 1

Member 2

Figure 4.6.2.5 A typical solution when the search is restricted to using either rod stock or pipe (with a diameter ratio $r = 0.88$).

The cost is 109.98 units giving $p_{\text{acceptable}} = 0.76$. $n_{\sigma_1} = 15.3$, $n_{\sigma_2} = 2.00$, $n_{\delta_1} = 2.53$.

Of course, one is probably thinking that if one was really designing a truss, material would be chosen from readily available stock sizes. Before this type of discrete search problem can be discussed, it is necessary to develop a database architecture for use with the specification-based decision model. The truss problem will be revisited in chapter 5.

4.6.3. Policy design (parameter selection)

Designers typically approach a problem by assuming that the specifications are given, and the task is to find design parameters that satisfy the requirements. In a policy setting context, we are interested in solving the opposite problem: given the desired design parameters or performance, what should the specifications be to promote such an outcome?

In this hypothetical example, we will try to establish an energy consumption criteria for labeling washing machines as environmentally friendly. This problem is inspired by a study performed in the United Kingdom (1992).

The problem statement is:

Given:
* the energy consumption distribution that we would like the washing machine market to move towards
* the energy consumption for the existing market

Find:
• an energy consumption criterion or specification that defines qualification for green labeling

While:
• encouraging the development of machines with desired energy consumption levels
• excluding a significant portion of washing machines currently on the market
• ensuring that labeled machines may be sold at an acceptable price

Figure 4.6.3.1a shows existing energy consumption levels while figure 4.6.3.1b identifies the desired levels.

The design variable is the specification that determines if a given machine will qualify for an environmentally friendly label (the policy). The specification is defined as a piecewise linear function made of one segment. The search will locate where the specification boundaries should be. The performance indicators and specifications that will be used to evaluate the criterion are provided in table 4.6.3.1.

![Figure 4.6.3.1a Energy Consumption levels for the existing washing machine market.](image1)

![Figure 4.6.3.1b Desired Energy Consumption levels for the washing machine market (to be promoted by the labeling criteria).](image2)
Table 4.6.3.1 Performance Variables and Specifications for the policy setting example.

<table>
<thead>
<tr>
<th>Performance Variable</th>
<th>Distribution Type</th>
<th>Specification Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>desired energy levels to be encouraged by the environmentally friendly label (figure 4.6.3.1b)</td>
<td>beta distribution</td>
<td>Labeling Criteria (the design variable)</td>
</tr>
<tr>
<td>Percent of existing market within the criterion</td>
<td>delta function</td>
<td>Exclusiveness Specification</td>
</tr>
<tr>
<td>Expected change in cost to qualify for labeling</td>
<td>symmetric beta distribution (computed from the design variable).</td>
<td>Cost Change Specification</td>
</tr>
</tbody>
</table>

Note that in this problem, the cost performance variable is not deterministic and thus is represented by a beta distribution. The equations used to compute derived performance variables are in appendix C. The equation for price change is written as a function of the
magnitude of change in energy efficiency and the expected feasibility of the change. That is,

\[
\text{Price Change} = f(\text{magnitude energy change, feasibility of change})
\]  

(4.6.3.1)

Feasibility is determined by evaluating the specification defined in figure 4.6.3.2. This example illustrates that it is possible to define performance distributions or specifications that use inputs from the evaluation of other specifications \((i.e., \text{the cost calculation uses feasibility information derived from evaluating the feasibility specification})\).

The results for the labeling policy problem are shown in figures 4.6.3.3a-d.

![Figure 4.6.3.2 Specification used to determine feasibility (probability acceptable) of energy consumption reduction.](image)

![Figure 4.6.3.3a The optimal energy labeling criterion (upper limit = 0.22 kWh/kg). The overall solution has \(p_{\text{acceptable}} = 0.7217\).](image)

![Figure 4.6.3.3b Evaluation of the criterion to include then desired energy consumption levels (\(p_{\text{acceptable}} = 1.0\)).](image)
4.6.4. The importance of modeling design uncertainty

The purpose of the following example is to demonstrate how uncertainty can have a dramatic effect on the evaluation of a design—thereby providing justification for the effort to represent specifications and variables probabilistically. The example is formed so that it is not possible to meet all specifications simultaneously, allowing the exploration of how uncertainty/risk affects the acceptability of a solution. First, I will compare the difference between using deterministic and probabilistic performance variables, and then I will illustrate the impact of 'fuzziness' in specifications.

Suppose that, as a product designer, we wish to determine the energy consumption level needed by a washing machine to best satisfy the environmental labeling and cost specifications from the previous example. The washing machine’s energy consumption is the design variable.

In the first case, we have assumed that both the cost and energy consumption are deterministic variables, while in the second case the energy variable is represented by a beta distribution. The dramatically different results for the two cases are shown in figures 4.6.4.1. To reveal the influence of uncertainty in specifications we will compare figure 4.6.4.2a, where the cost specification is uncertain, to the case in figure 4.6.4.2b. Figure 4.6.4.2b uses a binary cost specification.
Figure 4.6.4.1a Optimization results when energy consumption is a deterministic variable ($p_{\text{acceptable}} = 0.9412$).

Figure 4.6.4.1b Optimization results when energy consumption is a probabilistic variable ($p_{\text{acceptable}} = 0.5719$ instead of 0.9412 in the deterministic case).
These examples demonstrate that the acceptability of solutions based upon deterministic variables can be very different from those that incorporate uncertainty. Modeling uncertainty may significantly alter the expected quality of a solution (figures 4.6.4.1) or influence where compromises are made (as apparent in both figures 4.6.4.1 and 4.6.4.2).
Further, the effects are not necessarily intuitive because of interactions between uncertainty in both specifications and performance distributions. Therefore it seems reasonable to incorporate doubts about design performance or requirements into the decision process. Otherwise, artificially rigid boundaries may actually drive a design outcome!

4.7. Summary

In this section a search facility using the specification-based decision model was described and demonstrated. The implementation eliminates the need to distinguish between multiple objective or single objective problems and probabilistic or deterministic analysis. Importantly, objective function formulation is automated and the need to provide a feasible starting point is eliminated.

The examples illustrated that the fitness/objective formulation, combined with a genetic algorithm, overcomes the problems that are normally associated with local minima (camel back optimization) or disconnected search spaces (truss parameter selection problem). Also addressed were specification optimization (policy design example) and the significant impact uncertainty can have on the apparent acceptability of a design (importance of uncertainty).

It is believed that the inherent properties of the search facility match the qualities demanded by product designers. However, before the tool can be used in practice, a graphical user interface for problem definition is required.

At the end of the truss problem, it was apparent that design search involves more than freely picking design parameters. Often the designer must work with readily existing components, and choosing a particular component may have several performance and environmental implications. This observation leads to the next capability that must be available to a systems-oriented design tool.
5. **Catalog search using the specification-based decision model**

In chapter 2, the specification-based model was proposed as a design method that may be used for evaluating systems-oriented product design problems. In chapter 4 the model was applied to help designers find solutions to design problems. However, up to this point in the thesis, search is only possible for continuous (pseudo-continuous) independent design parameters. All design variables are encoded directly on the chromosome, as illustrated for the truss problem in figure 5.1.

![Chromosome encoding the design variables](image)

*Figure 5.1 Design variable represented on the chromosome directly.*

However, in many cases such freedom for choosing design variables will not exist. In the truss problem, it would be more reasonable to choose the material stock from readily available sizes and continuously vary only the truss base $s$.

In another design problem, one might wish to select the most suitable motor from a number of alternatives. Selecting this parameter would determine a number of the design's environmental impacts, performance characteristics and specifications (e.g., energy consumption, torque and maximum operating temperature).

In general, this type of discrete search may be classified as a catalog selection problem. A catalog selection problem involves choosing a pre-defined element that has defined characteristics to be part of a larger design. Examples of other work on catalog selection in design include (Bradley and Agogini 1991a; Bradley and Agogini 1991b). Catalog selection is an integral part of product design which should not be overlooked. Catalog selection problems may range from choosing standard stock sizes and components to material selection to picking the obsolescence path for a product.
The goal of this chapter is to extend the search facility to permit the simultaneous search of both catalog elements and continuous parameters. The chapter will begin by first describing a specification-based catalog structure and its hierarchical organization. A strategy for continuing analysis with incomplete catalogs is proposed and the implementation of the catalog classes is briefly described.

Then, catalog search capabilities are integrated into the general search facility. Two new types of distributions are introduced. One type acts as an index to a catalog, and the other allows the search to look for the best catalog and the best element with a catalog at the same time. The truss problem from chapter 4 is modified to include stock selection, stock catalog options and a hierarchical material catalog.

5.1. Catalog Organization

5.1.1. Catalog elements

In this section the elements in a catalog are identified. Two simple examples are used to illustrate the catalog structure before providing a general description.

Figure 5.1.1.1 depicts a catalog of wrought steel alloys. Within the catalog there are a number of different types of wrought steel. Each type of steel contains variable fields that describe the material's properties. Each variable field contains a distribution which quantifies the property. If a specific type of steel is selected, the properties of the chosen material would be incorporated into the design.

<table>
<thead>
<tr>
<th>Wrought Steel Catalog</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AISI 1020</strong></td>
</tr>
<tr>
<td>Variable Fields</td>
</tr>
<tr>
<td>Density</td>
</tr>
<tr>
<td>Modulus</td>
</tr>
<tr>
<td>etc.</td>
</tr>
<tr>
<td><strong>AISI 1045</strong></td>
</tr>
<tr>
<td>Variable Fields</td>
</tr>
<tr>
<td>Density</td>
</tr>
<tr>
<td>Modulus</td>
</tr>
<tr>
<td>etc.</td>
</tr>
<tr>
<td><strong>AISI 1090</strong></td>
</tr>
<tr>
<td>Variable Fields</td>
</tr>
<tr>
<td>Density</td>
</tr>
<tr>
<td>Modulus</td>
</tr>
<tr>
<td>etc.</td>
</tr>
</tbody>
</table>

*Figure 5.1.1.1 Example of a small wrought steel catalog.*
Figure 5.1.1.2 shows a catalog containing different product obsolescence alternatives. Each alternative has a number of requirements or specifications that must be met for the option to be feasible. These fields contain the specification distributions that define these requirements. If a particular obsolescence option is chosen from the catalog, its specifications are added to the product’s specifications. Additionally, there are properties or impacts associated with each option. For example, one impact of remanufacturing is the use of cleaning solvents. These associated impacts should also become part of the design’s properties when a particular option is selected.

<table>
<thead>
<tr>
<th><strong>Obsolescence Options Catalog</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Recycle</strong></td>
</tr>
<tr>
<td>Specification Fields</td>
</tr>
<tr>
<td>Production volume</td>
</tr>
<tr>
<td>Material recyclability etc.</td>
</tr>
<tr>
<td><strong>Variable Fields</strong></td>
</tr>
<tr>
<td>Energy used in recycling</td>
</tr>
<tr>
<td>Virgin material savings etc.</td>
</tr>
<tr>
<td><strong>Remanufacture</strong></td>
</tr>
<tr>
<td>Specification Fields</td>
</tr>
<tr>
<td>Technical life</td>
</tr>
<tr>
<td>Recoverable value-added etc.</td>
</tr>
<tr>
<td><strong>Variable Fields</strong></td>
</tr>
<tr>
<td>Cleaning emissions</td>
</tr>
<tr>
<td>Virgin material savings etc.</td>
</tr>
<tr>
<td><strong>Reuse</strong></td>
</tr>
<tr>
<td>Specification Fields</td>
</tr>
<tr>
<td>Technical life</td>
</tr>
<tr>
<td>Level of Styling etc.</td>
</tr>
<tr>
<td><strong>Variable Fields</strong></td>
</tr>
<tr>
<td>Transportation costs</td>
</tr>
<tr>
<td>Virgin material savings etc.</td>
</tr>
</tbody>
</table>

*Figure 5.1.1.2 Example of an obsolescence options catalog.*

These two examples should clarify the catalog structure. A catalog contains a list of options to choose between. These elements will be referred to as records. Thus, a catalog is a list of records. Each record contains the data needed to describe its characteristics. In the specification-based model everything is described in terms of specifications that indicate necessary performance levels and probabilistic variables that quantify performance properties. Thus, each record may contain two lists of data fields—one for specifications and another for variables. Within each field resides an appropriate linear, beta or delta distribution. This general terminology is illustrated in figure 5.1.1.3.
5.1.2. Catalog hierarchies

The previous section described the internal structure of a catalog. Now, it is suggested that records and catalogs will, in many instances, have hierarchical relationships. Again, this idea is first motivated and explained through example.

The literature abounds with reference to the hierarchical nature of problem solving. To provide a few examples, axiomatic design, the analytical hierarchy method, structured programming, and most decision analysis texts discuss hierarchical decomposition. Knowledge or data about a design also grows in a similar fashion. One begins with vague, broad estimations, and then progressively divides and refines this information as the design's level of detail grows (this statement is based on the premise that concept generation occurs in a top down manner). Therefore, it seems reasonable that the organization of catalogs should follow the same pattern.

Consider the example steel catalog hierarchy in figure 5.1.2.1. The steel catalog contains records that describe broad classes of steel types. The distributions in each record span the performance of all materials in the class. The distributions in the wrought steel records would encompass all wrought steel alloys. Each record in the steel catalog may have a child catalog, which contains more precise information. For example, the wrought steel catalog (child of the wrought steel record) contains records describing specific alloys, such as AISI 1020 or 1045. A hierarchy of material property catalogs developed for search examples in this chapter is described in appendix D.
In essence, the catalog hierarchy structures levels of abstraction or data estimation. A hierarchical catalog organization offers three important advantages. First, the decomposition provides a way to manage catalog sizes. Imagine the breadth of single flat catalog that includes all engineering materials.

Another advantage of this structure is that it formalizes levels of data abstraction or generality. As one traverses down a catalog hierarchy, the records of the elements span increasingly specific concepts. It is my belief that designers wish to obtain the broadest set of options that will ensure specifications are satisfied—further resolution may arbitrarily restrict the design. For example, if the wrought steel record's properties (in the steels catalog) meets all specifications, designers should leave their material selection options open to all wrought steel alloys. Selecting a specific alloy from the wrought steel catalog would be completely arbitrary. However, at some later point in the design process, a new specification might arise that requires that the material selection be narrowed to a specific record in the wrought steel catalog.
The significance of this point becomes apparent when team design work is considered. If one design group arbitrarily picks a specific wrought steel alloy when any wrought steel will suffice, other design teams will not realize that the material selection is arbitrary and they may constrain their work unnecessarily. Discussions with colleagues about the recycling of automobile dashboards provides an excellent example of this phenomena. Dashboards present recycling problems because they are made out of many different types of polymers. The various dashboard components are designed by several independent design teams. Each team chooses a specific polymer despite, in most instances, the availability of a wide range of polymers that would suffice. When the complete dashboard is subsequently considered by a recycling design team, it is no longer apparent what flexibility really exists in each components' material selection. If each design team had identified the broadest sufficient class of materials, a recycling design team could use this flexibility to choose materials that meet the requirements of each component and are compatible for recycling.

The final advantage of the hierarchical catalog representation is tied to performance estimation. Let us reconsider the obsolescence catalog example in figure 5.1.1.2. The recycling record may contain broad estimates of environmental implications and feasibility requirements for all types of recycling processes. A recycling catalog (child of the recycling record) might contain records and data for more specific classes of recycling processes (e.g., polymer recycling, ferrous metal recycling). These records would point to even more detailed catalogs for specific recycling processes. Once this catalog hierarchy has been constructed, it will be possible for the designer to perform approximate life-cycle analyses by choosing data from the appropriate abstraction level in the catalog (in section 2.4 this was identified as one of the main impediments to using LCA as a design tool). For example, the polymer recycling record would contain data that encompasses all polymer recycling operations. If designers did not know what polymer they were going to use, but wanted to consider possible recycling implications of choosing a polymer, this record would provide an appropriate estimate.

Figure 5.1.2.2 shows a hierarchy using the general catalog terminology. A catalog descending from a record is the record's child and the record is the catalog's parent.
Figure 5.1.2.2 General terminology for a catalog hierarchy.

In an ideal world, all records in a catalog will be complete—the necessary data will be present in each record. However, in many cases (environmental catalogs in particular) there will be records with data 'holes'. Further, one would expect the incomplete data to become more of a problem as one travels deeper into a hierarchy. Therefore, it is essential to provide a way of coping with situations when necessary data are missing from a catalog.

5.1.3. Incomplete records

How probability density functions could be used to estimate the performance of poorly understood systems was described in section 2.4. The incomplete record problem is quite different. Incompleteness is used to refer to the situation where one expects to find a certain type of data in a record but it is missing. What happens when one needs a
material's yield strength but the record for the specific material in question is missing this information? Should the evaluation stop and require that the designer input this data into the catalog? Should the option be eliminated because data are missing? Should the evaluation process continue, reflecting that some of the necessary properties are unknown?

The first option, stopping until all records in the catalog have the necessary information, is overly constricting. This strategy would prevent searching a catalog for the best record if just one record in the catalog was missing a piece of information. The second approach, eliminating the record with missing data from the search, seems more reasonable. However, what if all records are missing some information (as may often be the case for environmental impact catalogs) or a promising record is missing a single piece of information that is usually satisfactory for other records in the catalog? In such cases, elimination of the record seems too severe. Therefore, the final suggestion—continuing the evaluation making assumptions about the missing data—seems most appropriate.

Introducing the concept of an optimism level will allow evaluations to continue even if the data are missing. For each variable (or specification) needed to evaluate a design, the designer may assign an optimism value. If, during an analysis, the distribution data for a variable are missing, the optimism level is used as the subjective probability that, if available, the data would have values that would be satisfactory for the design. The value is called an optimism level because the pessimist would assign a probability of 0 (which disqualifies any design with missing data), while a pure optimist would assign a probability of 1.

Consider that we are picking a material from the steel catalog. Suppose the stainless steel record is being evaluated as a candidate for a design. The data that are needed from the record are first transferred into the design's data. If we cared about the product's weight, the data from the record's density field would be moved to the product's density distribution. Now, imagine that the density field is not present in the stainless steel record. Therefore, the product's density distribution will not exist. When the design's weight is computed, the weight distribution recognizes that the density information is missing and thus the weight cannot be determined. The product's weight distribution will also be void, but it will incorporate the density distribution's optimism level (say that it is 0.5). Finally, when the weight performance variable is evaluated against its specification,
this optimism level is returned as the probability the specification will be met (acceptable = 0.5). Therefore, the evaluation of the alternative continues, but the acceptability of the overall solution will be downgraded because of the missing data.

Further, if a performance variable is computed from a number of distributions that are missing data, the optimism level for the performance variable will be the product of the optimism levels for the missing data needed in the calculation. Hence, the more data that are missing, the more heavily the overall acceptability of the product is penalized.

The optimism level concept permits evaluation even when essential data are missing. As will be shown in section 5.4.3, this allows the designer to make selections from catalogs containing incomplete records. Importantly, the designer can then conveniently decide how sensitive the selection is to the missing data and whether gathering more information is necessary.

### 5.2. Catalog implementation

Section 5.1 discussed the structure of a catalog for use with the specification-based design model. In this section the software implementation of catalogs is briefly described. Like the implementation section in chapter 4, this summary is quite brief—only the functions essential to a general understanding of the software's operation are provided. For example, all classes provide for names and documentation regarding data sources, administrators and data revision dates. These aspects are not addressed in this section.

#### 5.2.1. The field class

The smallest element in a catalog is the data field. A field may be imagined as a place holder for the distribution that defines a property of the record. The simplified essence of the field class is provided in figure 5.2.1.1. A field contains a distribution and can assign distribution data to itself (assignFieldDistribution) and pass its data to another object (getFieldDistribution).
Two classes of fields are derived from the base class: variable fields and specification fields. This sub-classing allows specification fields to ensure that their distributions do not have a point value larger than unity (as this would imply an impossible probability of acceptance).

5.2.2. The record class

Objects of the record class contain lists of fields that describe the properties of the particular element they represent. A simple description of the record class is in figure 5.2.2.1.

A record may contain two lists of data. The variable field list contains fields that define the properties of the record. The specification list contains requirements that pertain to the use of the record. The record may also have a child catalog, as described in section 5.1.2. The record has methods that allow it to add fields (addSpecifications and addVariables) and pass its data to other objects (getSpecification and getVariable).

5.2.3. The catalog class
Finally, the catalog class is outlined in figure 5.2.3.1. An object of the catalog class contains a list of records which embody the catalog. As described in section 5.1.2, a catalog may also have a parent record. Methods are provided to add and retrieve records from the catalog.

Figure 5.2.3.1 A simplified description of the catalog class.

5.3. Catalog search implementation

Section 5.2 was devoted to defining the elements and organization of catalogs. We are now prepared to discuss how the specification-based search facility described in chapter 4 may be extended to include simultaneous catalog and continuous variable search.

Two different catalog scenarios are considered. Under the first scenario, the designer knows which catalog each component in the design will be chosen from. For example, the bearings will come from the brandX bearing catalog, the couplings will come from the brandY coupling catalog and the gears will come from the brandX gear catalog. The design problem may involve using many catalogs, but each component’s source is known.

This type of catalog search is addressed by defining a new type of distribution, called a catalog index. The index determines which element in the record will be used and transfers the record's data to the appropriate design variables. More details about the catalog index class are in section 5.3.1. The optimization problem is still formulated in terms of a design variable list, a performance variable list and a specification distribution list (like the camback example in appendix A). However, design variables that will be obtained from catalogs are not in the design variable list that is passed to the optimization module. Instead, the optimization software uses a list containing catalog index distributions and any continuous design variables to be optimized directly.

The second search scenario addresses the case where the designer is not sure which catalog is most appropriate for a given component. For example, the designer wants to
select a bearing, but at the same time is not sure whether the bearing should come from the brandX, brandY or brandZ catalog.

This situation is accommodated by introducing a catalog selection distribution class. Each alternative catalog is first encoded using catalog index distributions. A catalog selection distribution is then defined. This distribution is encoded on the chromosome to determine which catalog will be used to update the design variables. The catalog selection distribution is described in section 5.3.2. Again, the optimization is formulated in terms of design variables, performance variables and specifications. Each catalog index distribution is passed to the optimization module as a design variable. Catalog selection distributions are also passed as additional design variables.

The following sections describe the operation and implementation of these two new distribution classes. How the distributions are encoded and decoded from a chromosome is also detailed. Then, a preliminary discussion of how the optimization might be altered to search catalog hierarchies is presented.

5.3.1. The catalog index distribution class

Catalog index distributions allow the optimization software to search catalogs. Figure 5.3.1.1 shows how this type of search is accomplished. Recall that the genetic algorithm operates on binary chromosomes. In the example illustrated, the catalog index distribution will decode the leftmost chromosome segment to determine which catalog record will be used. The catalog index also incorporates data from the record into the design variables as required. Continuous design variables are encoded and decoded for chromosome segments as described in chapter 4.
As shown in figure 5.3.1.2, the catalog index class is derived from the distribution base class (described in section 4.3.1) so that the optimization software may treat the index class consistently with continuous design variable distributions (delta, beta, linear). However, the derived class contains unique data. When a catalog index is created, it is passed the catalog that it will index and the design variable data that must come from the catalog. The index then sets its search limits (setSearchLimits) so that they range between zero and the number of records in the catalog.
When a search is performed, the distribution first reserves an appropriately sized chromosome segment (encodeChromosomeSegment). When the distribution decodes itself from the chromosome segment (decodeChromosomeSegment), the genotype is converted to index a specific catalog record and the necessary record data are transferred to the design data (transferData). Computing probabilities has no meaning for this class of distribution.

5.3.1.1. Chromosome encoding and decoding

A catalog is simply a list of records, so a record may be indexed by a single number—its location in the list. Therefore, the catalog index distribution is completely described by the single parameter $i$, and it may be represented on a binary chromosome as shown in figure 5.3.1.1.1.

![Figure 5.3.1.1.1 Chromosome encoding for the catalog index distribution.](image)
The distribution is encoded by adding a segment \( n \) bits long to the chromosome. The number of bits is determined so that its decimal equivalent is at least as large as the total number of records in the catalog. A catalog containing 16 elements will require a 4 bit long chromosome segment.

An index distribution is decoded from a chromosome by first converting the appropriate binary segment to a decimal value and then rounding to the nearest integer. This yields the index \( i \) to the selected record in the catalog. The catalog index distribution then transfers the data from the record to appropriate design variables.

The addition of a catalog index design variable will increase the search space by a factor of \( 2^n \). In most cases, catalog searches will result in smaller spaces than continuous variable searches. For example, one continuous delta function variable at 20 bits resolution increases the search space by the same amount as a catalog containing over 1 million records. Most catalogs will contain considerably fewer records. Therefore, provided the catalogs are well structured (see section 5.4.1), most catalog searches will give more consistent results than the continuous variable searches.

The scaling rule for chromosome decoding and fitness evaluation (equation 4.5.2) remains unchanged, with the exception that each catalog index will have its own unique decoding time constant.

5.3.2. The catalog selector distribution class

The catalog selector distributions allow the optimization software to both determine which catalog should be used and search within catalogs for the best record in parallel. The mechanism for this type of search is inspired by combinatorial gene regulation observed in biological organisms (this controls cell differentiation in growing organisms). Figure 5.3.2.1 demonstrates the principle for this type of search.

The figure illustrates a case where variables \( x_2 \) and \( x_3 \) are to be selected from a catalog. There are two alternative catalogs the designer would like to consider while simultaneously searching for the best record. First, each catalog is encoded on the chromosome using a catalog index distribution as described in section 5.3.1. However, when the catalog index transfers the data from its chosen record it is received by the
selector distribution. The chromosome parameter for the selector distribution determines which catalog's data will be transferred to the design data—the selector distribution regulates the data that will be expressed in the design. The selector distribution must be decoded after the catalog indices distributions are decoded.

![Diagram of catalog selection and design data](image-url)

**Figure 5.3.2.1. Illustration of combined catalog selection and catalog search.**
As shown in figure 5.3.3.2, the Select Catalog class is also derived from the distribution base class (described in section 4.3.1). Like the catalog index class, there are specific data requirements for this derived class. When a catalog selector is created, it is given the list of catalogs that it will choose between. The selector sets its search limits (setSearchLimits) so that they will range between zero and the number of catalogs it must choose between.

**Class SelectCatalog**

<table>
<thead>
<tr>
<th>data:</th>
</tr>
</thead>
<tbody>
<tr>
<td>list of catalog index distributions</td>
</tr>
<tr>
<td>design data needed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>methods:</th>
</tr>
</thead>
<tbody>
<tr>
<td>setSearchLimits()</td>
</tr>
<tr>
<td>encodeChromosomeSegment()</td>
</tr>
<tr>
<td>decodeChromosomeSegment()</td>
</tr>
<tr>
<td>transferDataToDesign()</td>
</tr>
</tbody>
</table>

Figure 5.3.2.2 Outline of the SelectCatalog class.

### 5.3.2.1. Chromosome encoding and decoding

The catalog selection distribution is completely described by the single parameter \( s \) and thus it may be represented on a binary chromosome as shown in figure 5.3.2.1.1. The distribution is encoded by adding a segment \( n \) bits long to the chromosome. The number of bits is determined so that its decimal equivalent is at least as large as the total number of catalogs that the selector must choose between. The chromosome decoding process is directly analogous to the operation of the catalog index class.
Figure 5.3.2.1.1 Chromosome encoding for the catalog index distribution.

The addition of a catalog selection problem design variable will increase the search space by the factor defined by equation (5.3.2.1.1).

\[ f = 2^{n_s + \sum_{i=1}^{m} n_i} \]  

(5.3.2.1.1)

where

\( f \) = the factor the search space is increased by  
\( n_s \) = the number of bits needed for the catalog selector distribution  
\( m \) = the number of catalogs to choose between  
\( n_i \) = the number of bits required to encode the \( i^{th} \) catalog index

There is reason to suspect that search performance will degrade severely if the number of catalogs to choose between becomes too large. Justification for this suspicion will be provided in sections 5.4.1 and 5.4.2. For the time being, I will simply state that the number of catalog alternatives should be limited to no more than 20 for the population size (of 30) chosen for this work. However, this is not really a severe restriction. First, I find it difficult to imagine situations where such a large number of alternative catalogs will need to be considered. Second, the catalog selection capability is really just a convenience—it allows exploration of catalogs alternatives and the selection of the best record in a single search. The same effect can be achieved by manually performing an individual search for each alternative catalog and then choosing the one that yields the best design. Problems with many catalog options could also be broken down into smaller manageable selection problems.

Finally, the scaling rule for chromosome decoding and fitness evaluation (equation 4.5.2) remains unchanged, with the exception that each catalog selection distribution will have its own unique decoding time constant.
5.3.3. **Hierarchical search**

The last aspect of catalog search that will be considered before demonstrating the search implementation is the search of catalog hierarchies. As presented in section 5.1.2, organizing related catalogs into a hierarchy of increasing data resolution has a number of advantages. In particular, the structure will help the designer determine just how precise their catalog selection must be to ensure that specification will be met. This approach allows other design teams to understand where design flexibility exists.

Here, the goal is to consider modifications to the search module so that, during optimization problems involving catalog selection, the program will automatically traverse down hierarchies as needed to best meet design specifications. Given this capability, the designer would be able to start a search using catalogs near the top of their hierarchies (containing data for broad classes of options). Then, the search will determine just how precise each catalog selection must be, ensuring that the widest range of acceptable options is maintained.

Hierarchical catalog search will involve performing a series of successive optimizations. An optimization is performed at a given level in the catalog hierarchy and the results are considered. If the best solution does not meet all specifications, appropriate child catalogs replace their parent record's catalog, and the optimization is repeated. This process would be continued until an acceptable solution is found.

However, the problem of selecting appropriate child catalogs is non-trivial. The complications relate to selecting the branch that will lead to the best final result. Choosing only the child catalogs of the records used in the best solution will not necessarily lead to the best solution further down the hierarchy. It would seem that this problem can be resolved using a branch and bound strategy to traverse the search tree. Unfortunately, this is only the beginning of the problem. In many cases the design will involve searching several different catalog hierarchies at the same time (e.g., a material catalog and an obsolescence options catalog). Now, possible interactions between catalogs must also be considered. One solution approach (which would lead to combinatorial explosion) is to combine the catalog hierarchies to create a single hierarchy of all possible combinations. Since hierarchical search is not a central topic to this thesis, research to resolve these problems is delegated to future work. Only a superficial
A Probabilistic Specification-based Design Model: 
applications to design search and environmental computer-aided design

hierarchical search is implemented, primarily to facilitate a discussion about the impact and interpretation of data bandwidth as included in section 5.4.4.

The simplistic algorithm used for demonstration purposes is described below.

```plaintext
do {
    perform optimization
    for (each catalog used in the search) {
        if (record used in design has a child catalog)
            change the catalog index distribution so that it will select records from the child catalog.
    }
} while (Pacceptable < 1) and (solution has improved) and (at least one record used had child catalog)
```

This approach does not resolve any of the complications outlined in the previous paragraph. It follows only paths from records used in the best solution at each level and it does not account for combinatorial interactions if the search involves multiple catalog hierarchies. The while loop's continuation criterion is straight-forward. If the best solution has an acceptance probability of 1, there is no need to search catalogs containing more precisely defined record classifications. If the solution is not perfect, it may be worth looking at more detailed child catalogs. Additionally, it is only worth searching more deeply into the catalog tree along a branch if the best solution continues to improve. Last, there is obviously no point in conducting another search if none of the records has child catalogs.

This algorithm was implemented in a new optimization class, called hOptimise. Defining a hierarchical optimization is similar to a standard optimization as outlined in appendix A for the camelback example. The only difference is that the optimization is declared of type hOptimise instead of type optimise.

Again, it must be emphasized that this treatment fails to address the complexities of ensuring that the most appropriate hierarchy branch is found. However, in section 5.4.4, this implementation helps to provide insight into interpreting results with wide bandwidth performance data.
5.4. Catalog search examples

In this section examples of combined catalog and continuous search are described. To avoid the lengthy introduction of new problems, the truss example is modified and used throughout the section.

The truss problem is modified so that stock sizes are obtained from a single catalog (section 5.4.1). The effect of catalog size on search performance is discussed. Data quantifying performance for the worst case catalog search problem are presented. In section 5.4.2, the problem is reformulated as a catalog selection problem—the search must choose between catalogs for different classes of stock while simultaneously selecting the best stock size. The scaling limitations of the catalog selection method are discussed.

Next, the problem is expanded to include material selection from an engineering alloy catalog. This example illustrates how the optimism concept described in section 5.1.3 allows the search to proceed even though some material records are missing necessary data. Also demonstrated is how the designer can verify if the missing data has relevance to the search outcome. Finally, the material selection problem is performed over a large material hierarchy. This example is used to gain insight into the interpretation of search results when using very approximate data.

To avoid confusion it is again stressed that when reading graphs with superimposed performance probability density functions and design specifications, the vertical scale pertains to the specification function only. All performance probability density functions are automatically normalized to a unit area when equation (2.2.1) is evaluated.

5.4.1. Catalog search

In this section, the truss problem described in section 4.6.2 is modified to reflect a more realistic design situation where the truss members must be standard manufactured stock sizes. The diagram of the search problem is provided in figure 5.4.1.1.

Two catalog index distributions will be used to pick the best stock size for the two truss members from the stock catalog. The catalog was constructed using tables from (Bowes,
Russell et al. 1984). When the index distributions decode their chromosome segments to select a record in the catalog, they also transfer the data for the parameters (d, r) to the design's data. The truss base $s$ is still optimized as a continuous variable, so its distribution is encoded on the chromosome directly. The performance variables computed from the truss's design parameters and the specifications are unchanged from section 4.6.2.

![Figure 5.4.1.1 Diagram of the truss stock catalog search problem.](image)

The design variable list passed to the optimization software is outlined in Table 5.4.1.1. The performance and specification lists are the same as in section 4.6.2.
Table 5.4.1.1 Design distributions for the truss stock catalog search problem

<table>
<thead>
<tr>
<th>Design distribution</th>
<th>Distribution Type</th>
<th>Search Boundaries</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>delta function</td>
<td>1000mm &lt; s &lt; 3000mm</td>
</tr>
<tr>
<td>member 1 stock index</td>
<td>catalog index</td>
<td>range of stock catalog</td>
</tr>
<tr>
<td></td>
<td>(transfers $d_1$, $r_1$ to design)</td>
<td></td>
</tr>
<tr>
<td>member 2 stock index</td>
<td>catalog index</td>
<td>range of stock catalog</td>
</tr>
<tr>
<td></td>
<td>(transfers $d_2$, $r_2$ to design)</td>
<td></td>
</tr>
</tbody>
</table>

The optimization output for this example is provided in appendix E, while the optimized truss geometry is illustrated in figure 5.4.1.2. It is interesting to note that the best truss geometry using the standard stock sizes ($s_{opt} = 1133$mm) is quite different than the result obtained when the members' geometry was continuous ($s_{opt} = 2000$mm). This simple comparison effectively shows why it is not good practice to optimize a design using continuous variables and then pick the closest readily available stock sizes.

![Truss Configuration Diagram]

**Truss Configuration**

**Member 1**

**Member 2**

*Figure 5.4.1.2 Optimal truss for catalog selection using commercially available stock sizes. The cost is 147 units giving $p_{acceptable} = 0.6129$, $n_{a_1} = 14.29$, $n_{a_2} = 2.00$, $n_{B_1} = 3.04$. The cost specification cannot be completely satisfied.*

This example shows that the catalog search facility works, but an obvious question still lingers. How does the search performance scale with catalog size? The answer to this
question is an unequivocal "it depends". To explain this, it is necessary to differentiate between well and poorly structured catalogs. Figure 5.4.1.3 provides an example of each.

![Good Catalog Structure vs Poor Catalog Structure Table]

**Good Catalog Structure**

<table>
<thead>
<tr>
<th>Record</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3/8 inch pipe</td>
</tr>
<tr>
<td>1</td>
<td>1/2 inch pipe</td>
</tr>
<tr>
<td>2</td>
<td>5/8 inch pipe</td>
</tr>
<tr>
<td>3</td>
<td>3/4 inch pipe</td>
</tr>
<tr>
<td>4</td>
<td>1 inch pipe</td>
</tr>
<tr>
<td>5</td>
<td>1.25 inch pipe</td>
</tr>
<tr>
<td>6</td>
<td>1.5 inch pipe</td>
</tr>
<tr>
<td>7</td>
<td>1.75 inch pipe</td>
</tr>
<tr>
<td>8</td>
<td>2 inch pipe</td>
</tr>
<tr>
<td>9</td>
<td>2.25 inch pipe</td>
</tr>
<tr>
<td>etc.</td>
<td></td>
</tr>
</tbody>
</table>

**Poor Catalog Structure**

<table>
<thead>
<tr>
<th>Record</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2.25 inch pipe</td>
</tr>
<tr>
<td>1</td>
<td>1 inch pipe</td>
</tr>
<tr>
<td>2</td>
<td>3/8 inch pipe</td>
</tr>
<tr>
<td>3</td>
<td>1.5 inch pipe</td>
</tr>
<tr>
<td>4</td>
<td>5/8 inch pipe</td>
</tr>
<tr>
<td>5</td>
<td>1.25 inch pipe</td>
</tr>
<tr>
<td>6</td>
<td>3/4 inch pipe</td>
</tr>
<tr>
<td>7</td>
<td>1.75 inch pipe</td>
</tr>
<tr>
<td>8</td>
<td>1/2 inch pipe</td>
</tr>
<tr>
<td>9</td>
<td>2 inch pipe</td>
</tr>
<tr>
<td>etc.</td>
<td></td>
</tr>
</tbody>
</table>

*Figure 5.4.1.3 Example of a well-structured and badly-structured catalog.*

Let's begin by discussing how the search of a well structured catalog will scale. A good catalog is organized so that the records are ordered in a way that produces a continuous trend over the catalog. Conceptually, this type of organization helps guide search because a record that is close to the optimal record will be more desirable than one that is far way. A good structure for the stock catalog places the records in order of increasing size. However, it should be noted that a good catalog structure is not only limited to record orders that produce monotonic trends (e.g., strictly increasing pipe sizes). The key issue is that the neighboring records have similar properties.

Search of the well-structured pipe catalog in figure 5.4.1.3 will scale exactly like a "continuous" problem (recall that continuous variables are actually discrete—their resolution is determined by their limits and the number of bits used on the chromosome). Therefore, extremely large catalogs do not present problems provided they are well structured (for example, the camelback problem in section 4.6.1 could be solved with equivalent performance—excluding disk storage access time—using catalogs for x1 and x2 each containing over 1 million ordered records \(2^{20}\)).
It is felt that most catalogs can indeed be structured so that their records are in a logical order. Most physical components and elements can be ordered by size. The records in the wrought steel catalog can be ordered from low carbon (e.g., AISI 1020) to high carbon (e.g., AISI 1090). Even a catalog of product obsolescence alternatives could be ordered in terms of material resource conservation (dispose, incinerate, recycle, remanufacture, reuse). In the future, a facility that automatically pre-sorts the catalog records based upon the properties of interest to the designer could be implemented.

Therefore, one can conclude that large catalogs, given that they are reasonably structured, do not present a problem. In fact, most catalog searches will have smaller solution spaces than continuous searches. The truss catalog search presented here is considerably less demanding than the example presented in section 4.6.2.

Now, let us consider the worst case scenario of searching a very badly structured problem. Figure 5.4.1.3 shows a poorly structured pipe stock catalog. There is no trend in the record ordering—each record is completely unrelated to its neighbors. A simple test will be used to illustrate how performance degrades severely under such circumstances.

A catalog was constructed where all but one record contained a field with a delta function located at 0. The single exceptional record was randomly located in the catalog and had a field with a delta function at 1.0. The specification for the test simply asked for the value of 1.0. I describe this as a 'needle in the haystack' search problem. Any individual piece of straw gives no indication if the needle is nearby. In the continuous domain, this problem is equivalent to maximizing a function that is zero everywhere except for a single point. (I am not aware of any search technique that will perform well for this problem.)

This test was performed for catalog sizes ranging from ten to 20,000 records in size and the number of generations required to find the "needle" was recorded. Thirty optimizations were performed for each catalog size to obtain an average number of generations to solve the problem. The results for catalogs up to 500 records are shown in figure 5.4.1.4.
Figure 5.4.1.4 Average number of generations to find the 'needle in the haystack' as a function of catalog size.

Not surprisingly, the number of generations required is closely tied to the relationship between the genetic algorithm's population size and the catalog (search space) size. For catalog size to population sizes less than 2/3 (20 records), the solution is almost always found in the first generation, because the probability that the solution is in the random population seed is high. However, catalog size/population ratios up to three (approx. 100 generations) did not cause very severe performance degradation (average less than 4 generations). As the catalog size continues to grow, the number of generations increases rapidly and is extremely erratic (as indicated by the average deviation bars). At 500 records (giving a probability that the solution will be in the randomly seeded population of 0.06), the average number of generations is around 140—considerably worse than what is expected from a random search (4200 evaluations [140 generations x 30 organisms] compared to 500).
The result of this worst case test is somewhat obvious, but it clearly illustrates the importance of appropriate record ordering within catalogs. Fortunately, it seems that it is possible to structure most catalogs so that their size is not an issue. However, if the catalog cannot be organized, its size must be restricted to not much more than the genetic algorithm's population size (and preferably less). For this work, a population of 30 organisms is used.

Finally, there is no inherent restriction on the total number of catalogs that may be used to obtain different design variables in the same search.

5.4.2. Catalog selection

The truss problem will now be reformulated for the simultaneous consideration of different stock catalogs and the selection of the best stock within a catalog. In the previous section, the records for the pipe and rod stocks were lumped into the same catalog. From an organizational standpoint, this is not a reasonable approach. Rod stock and pipe have no logical connection. Further, what if the designer wanted to also consider W or square stock? Do we need another catalog that combines all of these options into one?

The catalog selection distribution allows the designer to specify the problem in a more natural manner. The capability allows the designer to simply state that the stocks should be chosen from a set of alternative catalogs—the pipe catalog, the rod catalog, the W catalog, the square catalog, etc. Figures 5.4.2.1a and b illustrate how this is accomplished for the truss problem when catalog alternatives are limited to rod stock and pipe stock.

A catalog index distribution is encoded on the chromosome for each catalog that may be used for member 1. A catalog selector distribution is also defined for member 2. When the catalog index distributions decode themselves from the chromosome, they pass their data to the selector distribution. When the selector distribution decodes itself, it determines which catalog's data is passed on to the design. Once again, the truss base s is a continuous design variable and is encoded on the chromosome directly.
Figure 5.4.2.1a Diagram of catalog selection problem with catalog alternatives for truss member 1.
Figure 5.4.2.1b Diagram of catalog selection problem with catalog alternatives for truss member 2.
The design variable list passed to the optimization software is outlined in table 5.4.2.1. The performance and specification lists are the same as in section 4.6.2.

**Table 5.4.2.1 Design distributions for the truss with catalog alternatives.**

<table>
<thead>
<tr>
<th>Design Distribution</th>
<th>Distribution Type</th>
<th>Search Boundaries</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>delta function</td>
<td>1000mm &lt; s &lt; 3000mm</td>
</tr>
<tr>
<td>member 1 pipe index</td>
<td>catalog index (transfers d₁, r₁ to member 1 selector)</td>
<td>range of pipe catalog</td>
</tr>
<tr>
<td>member 1 rod index</td>
<td>catalog index (transfers d₁, r₁ to member 1 selector)</td>
<td>range of rod catalog</td>
</tr>
<tr>
<td>member 1 stock selector</td>
<td>catalog selector (transfers d₁, r₁ to design)</td>
<td>number of catalog alternatives (2 in this example)</td>
</tr>
<tr>
<td>member 2 pipe index</td>
<td>catalog index (transfers d₂, r₂ to member 2 selector)</td>
<td>range of pipe catalog</td>
</tr>
<tr>
<td>member 2 rod index</td>
<td>catalog index (transfers d₂, r₂ to member 2 selector)</td>
<td>range of rod catalog</td>
</tr>
<tr>
<td>member 2 stock selector</td>
<td>catalog selector (transfers d₂, r₂ to design)</td>
<td>number of catalog alternatives (2 in this example)</td>
</tr>
</tbody>
</table>

Naturally, the results of the search are the same as in section 5.4.1, so they will not be repeated here. A partial listing of the optimization output is provided below.

```
starting optimisation!
number of generations 108

catalog selector: member 1: selected catalog is pipes
catalog index: member 1 index for catalog: pipes
chosen record is: 2.5 inch nominal
variable data used are: pipe outside diameter, pipe diameter ratio

catalog selector: member 2: selected catalog is rods
catalog index: member 2 index for catalog: rods
chosen record is: 1/2 inch nominal
variable data used are: rod outside diameter, rod diameter ratio

printing delta distribution: s
data source: none
comments: none
scale: mm
delta location at: x = 1132.942383
```
The remainder of the results is the same as the single catalog implementation in section 5.4.1. These results were listed in appendix E. The output shows the pipe catalog was selected for member 1 (buckling limited) and rod catalog was selected for member 2 (stress limited).

Again, we must question how the search performance will be affected as the number of catalog alternatives increases. In the previous section considerable time was spent explaining the difference in search performance for well structured and poorly structured catalogs. It was concluded that because it should be possible to organize most catalogs, the search can accommodate very large catalogs. If the catalogs are poorly structured, performance will poor unless the catalogs are small. These arguments also apply to the number of catalog alternatives to pick among. Imagine that the list in the selector distribution is like a catalog (see figure 5.4.2.1). If the list is well structured, a large number of catalog alternatives will not be a problem.

Unfortunately, we can count on this list being badly structured. It is simply an arbitrary list of data extracted from records in different catalogs. There is no structure to this list. Therefore, according to the logic of the previous section, unless the number of catalog alternatives is small (less than 20 for this work), erratic search behavior and poor convergence can be expected.

Fortunately, the number of alternative catalogs will usually be small. For example, if we wanted to consider the pipe, rods, W's and square tube, then the total number of catalog alternatives is only four (this means that four catalogs are being considered to set a single property set in the design—d and r). If a large number of different catalogs needs to be considered, one could always perform a number of smaller catalog selection problems. For example, the single search in this section provides the same effect as performing four individual searches shown in table 5.4.2.2 and then picking the design with the best overall result.
Table 5.4.2.2 Number of single catalog searches needed to an equal the options considered in the single catalog selection search.

<table>
<thead>
<tr>
<th>Search number</th>
<th>Member 1 stock catalog</th>
<th>Member 2 stock catalog</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>pipe catalog</td>
<td>pipe catalog</td>
</tr>
<tr>
<td>2</td>
<td>pipe catalog</td>
<td>rod catalog</td>
</tr>
<tr>
<td>3</td>
<td>rod catalog</td>
<td>pipe catalog</td>
</tr>
<tr>
<td>4</td>
<td>rod catalog</td>
<td>rod catalog</td>
</tr>
</tbody>
</table>

5.4.3. Search with incomplete records

This section will illustrate how the optimism level discussed in section 5.1.3 allows catalog searches to continue even when necessary data are missing from some of the records. Importantly, the designer can also use the optimism level to decide if the missing data is significant to the design problem.

In this case, we will start from the truss problem in section 5.4.2. (selecting the appropriate stock catalog from a number of alternatives and choosing the actual stock size), and add another catalog selection component—the truss material. Up to this point material properties for mild steel have been assumed. Now, the material will be picked from an alloy catalog. The complete catalog is in appendix D, while table 5.4.3.1b shows only the catalog properties relevant to this example. The truss mass, stiffness, strength and cost will be affected by the material selection. All material properties in the catalog are estimated using uniform probability density functions. It is now also required that the truss be light enough for a single person to transport the device without mechanical aids.

Table 5.4.3.1a Property designation legend.

<table>
<thead>
<tr>
<th>Designation</th>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>density</td>
</tr>
<tr>
<td>1</td>
<td>Young’s modulus</td>
</tr>
<tr>
<td>3</td>
<td>relative cost per volume</td>
</tr>
<tr>
<td>4</td>
<td>yield strength</td>
</tr>
</tbody>
</table>
Table 5.4.3.1b Properties from the alloy catalog relevant to the design problem.

<table>
<thead>
<tr>
<th>Material Record</th>
<th>Data in Record</th>
<th>Child Catalog</th>
</tr>
</thead>
<tbody>
<tr>
<td>aluminums</td>
<td>0, 1, 3, 4</td>
<td>aluminum catalog</td>
</tr>
<tr>
<td>coppers</td>
<td>0, 1, 3, 4</td>
<td>none</td>
</tr>
<tr>
<td>leads</td>
<td>0, 1, 4</td>
<td>none</td>
</tr>
<tr>
<td>magnesiums</td>
<td>0, 1, 3, 4</td>
<td>none</td>
</tr>
<tr>
<td>molybdenums</td>
<td>0, 1, 4</td>
<td>none</td>
</tr>
<tr>
<td>nickels</td>
<td>0, 1, 3, 4</td>
<td>none</td>
</tr>
<tr>
<td>steels</td>
<td>0, 1, 3, 4</td>
<td>none</td>
</tr>
<tr>
<td>tins</td>
<td>0, 1, 4</td>
<td>none</td>
</tr>
<tr>
<td>titanums</td>
<td>0, 1, 3, 4</td>
<td>none</td>
</tr>
<tr>
<td>tungstens</td>
<td>0, 1, 3, 4</td>
<td>none</td>
</tr>
<tr>
<td>zines</td>
<td>0, 1, 3, 4</td>
<td>none</td>
</tr>
</tbody>
</table>

The table indicates that the cost data are missing for two records in the catalog. This is noted to emphasize that some of the data needed to evaluate the design requirements are not available for all of the materials. Under normal design circumstances, the designers would probably not be aware that some of the records are incomplete. They would simply retrieve that catalog from a database and perform the optimization.

When the search is performed, the optimism levels are used in lieu of the missing data as described in section 5.1.3. The default optimism level for all of the design variable distributions is set at 0.5. The program output from the search is listed in appendix F. The best result suggests that: the design should be made from an aluminum alloy; member 1 will be a three inch pipe; member 2 is a 3/4 inch rod, and; the truss base should be 1660 mm.
Figure 5.4.3.1a Member 1 stress safety factor (p = 0.9660).

Figure 5.4.3.1b Member 2 stress safety factor (p = 0.8356).

Figure 5.4.3.1c Member 1 buckling safety factor (p = 1.0000).

Figure 5.4.3.1d Truss cost (p = 0.7234).
Many of the performance distributions are quite broad because the data in the aluminum records span the properties of pure aluminum and all aluminum alloys. Even though the data are not precise, we can already ascertain that it will be impossible to meet the cost and mass requirements using any aluminum alloy.

Do we need to reconsider our expectations of the design, or are there other options that might be satisfactory but are presently unappealing due to incomplete data? Recall that each time missing data is encountered during a calculation, the overall design quality is reduced by a factor of the variable's optimism level. For example, consider a candidate design made of a molybdenum alloy. The molybdenum record is missing its cost data. Given the optimism level is 0.5 for all variables, the acceptability based on cost will be assigned 0.5. Thus the overall solution acceptability of the design will be halved.

Designers can conveniently determine if the missing data is important by changing their optimism level to 1.0 for all variables. Now, when confronted with missing data, the assumption is that when made available, the performance will prove to be satisfactory. Thus, when evaluating the concepts using molybdenum, the cost acceptability will be 1.0.

When the search is performed using a 1.0 optimism level, the best solution is unchanged (still aluminum). Therefore, the designer can conclude that the missing data has no bearing on the problem as it is now specified. Thus, there is no need to gather the data by
the incomplete records. Designers now also know that they will need to revise their expectations since no aluminum alloy can satisfy the cost and mass requirements.

5.4.4. Hierarchical catalog search

To conclude the catalog examples, a search will be performed using a hierarchical catalog. The exercise will illustrate how the designer can infer whether approximate data or difficult requirements are the source of low design acceptability.

The example is similar to the truss material selection problem discussed in the previous section (5.4.3), but the material options are not limited to just the alloy catalog. Instead, the catalog used in the search is the engineering material catalog, which contains records for broad classes such as engineering alloys, ceramics, polymers, composites and woods. This catalog is at the top of the material catalog hierarchy outlined in appendix D. For example, the alloy catalog used in section 5.4.3, is a child of the engineering alloys record. All of the records in this top-level material catalog have child catalogs containing more detailed information.

The only other change that must be made is to declare the optimization of class hOptimize (instead of the previously described class Optimize) in the file C++ which defines the optimization problem. The approach to hierarchical optimization was discussed in section 5.3.3.

On the first run, the best design used the properties from the high-level record describing all engineering composites. The optimization module then automatically started a new optimization searching this record's child catalog, the composite catalog. This catalog contained records that describe glass reinforced polymers and carbon reinforced polymers. This second search selected glass reinforced polymers, but the solution was poor (pacceptable = 0.048). As this record had no child catalog, the search was terminated. Observing the output for the glass reinforced design indicated that the solution was dominated by missing data (three of the four necessary material properties are not available).

Therefore, a command was added to the optimization file to exclude the composite record when the top level-catalog is searched. When the optimization was performed again, the search path illustrated in figure 5.4.4.1 was followed. Using properties from the alloys
record in the top-level catalog produced the best results. Consequently, the software replaced the top-level catalog with the alloy catalog and performed another optimization. As illustrated in the figure, the alloy catalog contains records describing several different classes of alloys. At this level, the record spanning the properties of all aluminum alloys was used in the best result. Thus, the optimization software substitutes the aluminum catalog for the alloy catalog and performs yet another search. The best design from this search uses a 7XXX series high strength aluminum alloy. At this point, the search is terminated as the 7XXX series does not have a child catalog. The complete output for this search path may be found in appendix G. The results of this search will be interpreted shortly, but I will first proceed to describe the next search that was performed.

![Diagram of engineering materials catalog with alloys catalog and aluminum catalog]

Figure 5.4.4.1 Hierarchical search path for the truss material section problem.

Next, the alloys record was eliminated from the top level catalog search. The next best high-level material record had a zero acceptability. So we have determined that only the composite and alloy branches of the hierarchy have the potential to produce an acceptable solution. Finally, the aluminum record was eliminated from the alloy catalog and a search at the alloy catalog level revealed that steel was the next best solution, but it was poor due
to the weight requirement. Therefore, one can be reasonably confident that aluminum is
the best bet.

This example shows that even the simple approach to hierarchical search implemented
here can be used to facilitate the exploration of design alternatives. However, it would be
more desirable if the search algorithm could automate the entire process. This is left for
future work.

The second point to be addressed through this example relates to interpreting how data
approximation affect the apparent acceptability of a design solution. In this problem, the
cost and weight requirements are consistently the most difficult to satisfy. The cost and
weight distributions for the best design from the top-level catalog search in figure 5.4.4.1
are shown in figures 5.4.4.2a and b. The chosen material record describes the properties
of all engineering alloys. Clearly, both attributes are not very acceptable. However, even
at this very high level we can see there is the possibility that some alloys will completely
satisfy the cost requirement. The poor performance on the cost specification may be due
to the high-level data approximation. More precise data are needed. In contrast, the
weight distribution indicates that it is impossible to have a perfectly acceptable weight.
While the acceptability may improve when specific material alloys are considered, we
already know this requirement cannot have an acceptability of 1.0.

The ability to make such inferences can help the designer decide if more precise
performance estimates are needed, or whether the design is simply not going to be
satisfactory. When dealing with environmental considerations, which are difficult to
estimate precisely, this type of analysis can help to focus where resources need to be
focused for further data collection.
Figures 5.4.4.2a and b show the cost and weight results for the alloy catalog level of the hierarchy in figure 5.4.4.1. Now it is clear that even if the performance estimates are further improved, the design will not be completely acceptable on the bases of both a cost and mass.

The results for the search using the aluminum catalog (figures 5.4.4.4a and b) show that the design is about as good as it is going to get using aluminum alloys. At this point,
designers must either revise their expectations and accept this design, or consider the possibility of gathering more data on composite materials. The materials in the composite catalog were not acceptable because too many data are missing.

**Figure 5.4.4.4a** Truss cost using data describing all 7XXX series aluminum alloys \((p = 0.7201)\).

**Figure 5.4.4.4b** Truss mass using (data for all 7XXX series aluminum alloys \((p = 0.7304)\).

### 5.5. Summary

In this chapter the specification-based search facility was expanded to include catalog search. Catalog selection is an important part of many product design problems. Also, it is likely that catalog selection is the most feasible way to provide designers convenient access to environmental impact data. For example, by choosing a specific material from a catalog, data about embodied energy or processing effluent may be made available to the designer.

A structure for catalogs based upon the specification-based model was proposed and implemented. A hierarchical organization based upon data abstraction was recommended as a way to help designers communicate where design flexibility exists. The optimism concept was introduced as a way to permit analysis to continue when records in catalogs are missing essential data.

Two new distribution classes were added to the implementation described in chapter 4. These additions enable design searches that simultaneously explore continuous variables
data within catalogs, and choose between alternative catalogs. A simple approach to hierarchical search is described, but more research is required in this area.

The examples first demonstrated how the truss problem may be encoded as a catalog search. Issues relating to optimization performance when using large catalogs were considered. It was concluded that large catalogs do not present problems provided they are well structured. The ability to combine catalog search and at the same time consider the use of different catalogs was also demonstrated. The number of catalog alternatives under simultaneous consideration (e.g., should the stock be chosen from a rod catalog, pipe catalog, or W catalog, etc.) should be smaller than the population size used in the genetic algorithm.

It was also shown that the optimism level allows design search to proceed even when necessary data are missing, and that the designer can vary the optimism to determine if the missing data are having an impact on the search outcome. Finally, an example of a hierarchical material catalog search was provided.
6. **Toward systems-oriented CAD—specification-based analysis templates**

At the beginning of this work it is argued that systems-oriented product design will become a necessity in the future. Further, designers will need special integrated tools to fully comprehend the system-wide implications of their decisions. Hence, the thesis has been directed towards researching ingredients needed to develop such tools. The first step was to provide a way of modeling multiple-criteria problems in a familiar design language—product designers are not decision analysts. The development of this specification-based model was presented in chapter 2.

The second step towards a systems-oriented design tool was to help designers find better solutions. In chapters 4 and 5 I described a general parameter and catalog optimization and search tool for problems represented using the specification-based model. This facility allows simultaneous continuous parameter selection and catalog selection. Using this approach, the impacts of a catalog selection are automatically incorporated into the designs' data. For example, when a material is selected from the material catalog, the energy density impacts can be made available for evaluating an energy requirement.

The search facility is a useful design tool capable of solving a wide variety of design problems—an essential capability for systems-oriented product design. However, one cannot expect designers to have the expertise to model many of the complex system problems they will need to assess. Many product designers are non-technical, and the search facility requires that the designer must model each problem individually. Designers will want "off the shelf" tools (or models) for standard problems, such as assessing recyclability or manufacturability.

In this chapter, a preliminary investigation into how standard problems might be represented and pre-encoded is conducted. Standard problem templates might allow designers to analyze various aspects of a product's life-cycle conveniently. A general structure for representing problems is suggested. It is felt that a unified representation is needed for system-oriented design tools—each specialized analysis must be performed within the context of other life-cycle stage requirements. Let me emphasize that this chapter is preliminary work and might be most appropriately regarded as an introduction
to future work. However, this chapter is important because I demonstrate how the components developed in this thesis will unite to form a systems-oriented design tool.

I will begin the chapter by surveying life-cycle design tools that have been developed or are under development by other researchers. Many are focusing on guidelines, while others are considering specialized analytical computer tools. Research in developing design aids suggests that specialized tools are the most helpful, but that isolated approaches cannot accommodate integrated systems-oriented design goals.

Next, a concept for an integrated life-cycle design tool is presented, and a template architecture for encoding specific design evaluation tools is suggested. The template architecture is then implemented for integrated use with the specification-based architecture described in chapters 4 and 5. This preliminary implementation is then used to create some example special purpose design tools. First, the familiar truss problem is encoded as an analysis template and then a product retirement template is illustrated. The chapter closes by discussing how, in the future, the templates and the search facility may be used together to assist systems-oriented product design.

6.1. Existing life-cycle design tools

There are number of researchers and organizations working on a variety of life-cycle design tools. This section provides a brief overview of the different types of tools available and who is actively working in the area. The relationship of the approaches to system-oriented product design is also considered.

It appears that the majority of researchers and organizations is focusing on the development of design guidelines. The U.S. Environmental Protection Agency publishes many industry specific guideline documents (a catalog of these publications may be obtained from the EPA Publications and Information Center, Post Office Box 42419, Cincinnati, Ohio 45242-0419). In general, guidelines are developed for specific stages of a product's life-cycle, such as assembly (design for assembly), disassembly (design for disassembly), or recycling (design for recycling). Frameworks that attempt to integrate the individual methods into a unified framework over the entire life-cycle are often described as design for X methodologies (DFX) (for example, see Gatenby and Foo 1990).
There are thousands upon thousands of publications which present design guidelines. Some interesting examples in the area of environmental design address: waste minimization (Friedlander 1989; 1990; Kjeldgaard 1992); recycling (Henstock 1988; 1991; Navinchandra 1991; Weber 1991; Stone, Sagar et al. 1992; Seliger, Zussman et al. 1993); remanufacture/serviceability (Lund 1983; Warneke and Steinhilper 1983; Haynsworth and Lyons 1987; Gershenson and Ishii 1991; Plummer 1992; Shu and Flowers 1993), and; disassembly (Schmaus and Kahmeyer ; Simon, Fogg et al. ; Noller 1992; Wivell 1992; Jovane, Alting et al. 1993). There are also numerous guidelines published for other areas such as maintenance (e.g., Cunningham and Cox 1972), reliability, manufacturability and assembly.

Guidelines are extremely popular with designers. They are conceptually easy to grasp and convenient to apply. They are also based upon the same non-analytical normative principle that is used for a large part of product design ('if it worked in the past, do it the same way again'). As stated in section 1.1, the initial goal of this work was to develop an environmental design guideline framework. However, the problem with design guidelines is that they are case specific. Guidelines are defined under an assumed set of problem boundary conditions. Thus each time the context of a problem changes, the guidelines may also change. This is probably why there are thousands of different design guidelines in the literature. Even more disturbing, the boundary conditions and underlying assumptions that guidelines are founded upon are usually not included with the guidelines. Therefore, it is not possible to establish whether the guidelines are appropriate for a given design task. Selecting appropriate guidelines is even more difficult in the context of DFX because it is necessary to account for interactions between guidelines used for different stages of the life-cycle (Gatenby and Foo 1990, acknowledges that these interactions are critically important).

There is still one more concern about using guidelines for systems-oriented design. Typically, the guidelines do not help the designers identify the appropriate goals that will provide the largest system-wide impact. For example, recycling guidelines help the designer make a recyclable product. However, if they are designing a durable product, this effort will probably have only a minuscule impact on the overall environmental impact (e.g., 1992). The guidelines do not give any indication of what the actual benefits of their application will be. Blindly following guidelines may lead to inferior designs.
because more appropriate general solutions are overlooked (Barkan and Hinckley 1993). For example, the blind desire to eliminate unnecessary fasteners will reduce assembly time but may significantly increase product manufacturing cost.

Guidelines are very useful and convenient design aids. However, it is my opinion that guidelines alone cannot address the needs of systems-oriented product design. More sophisticated, context sensitive analytical tools are required.

Other researchers and companies are developing a variety of computer-based environmental life-cycle analysis tools. Recently, surveys of the various tools that are under development or commercially available has been published (Troy, Wong et al. 1994a; Troy, Fu et al. 1994b). These papers compare the capabilities of the different programs' and describes their principles of operation. (Goldfarb 1994) has also reviewed current efforts in this area. Like the design guidelines just discussed, these tools tend to specialize in specific aspects of the product's life-cycle. Areas addressed include disassembly (Restar, Carnegie Mellon Green Engineering Project), material processing related emissions inventory (Sima Pro, Pre Consultants, the Netherlands), and product retirement (LASer, Life-cycle Group at Ohio State University; RECREATION, The Fraunhofer Institute for Manufacturing Engineering Automation). Other programs under development not mentioned in these surveys include commercial disassembly software (Young 1992), research software for recycling analysis (Pnueli, Lussman et al. 1994) and remanufacture (Li Shu and Woodie Flowers, Massachusetts Institute of Technology, Department of Mechanical Engineering). Dr. Michael Wang (at the University of Windsor Department of Industrial Engineering in Ontario, Canada) is working on a tool for environmentally conscious design and manufacturing called E.D.I.T.

With one exception, these tools are specialized and operate in isolation of other life-cycle considerations. The effort by the group at Ohio State seems to be closest to addressing the notion of integrated system-oriented design. Their research spans reliability, maintenance, disassembly, manufacture and recycling (Bryan, Eubanks et al. 1992; Burke, Beiter et al. 1992; Marks, Eubanks et al. 1993; Yu, Krizan et al. 1993). These efforts developed separate tools, but they are all based upon a rule-based approach and a qualitative assessment technique called compatibility analysis (Ishii, Adler et al. 1988; Ishii 1991). Their recent work on the program LASer now integrates assembly, serviceability and product retirement analysis.
Given that systems-oriented design requires an integrated analysis environment, it is interesting that most of the computer-aided design tools are fragmented—just like their guideline counterparts. Research on developing computer-aided design tools suggests that designers are much more likely to use specialized tools and that general purpose tools are less popular (Ishii and Hornberger 1992). Special purpose programs are popular because they tend to be easier to use and they solve difficult, but well defined, problems. This presents an apparent paradox—designers prefer specialized single purpose tools but systems-oriented design requires a multi-purpose integrated environment.

However, I contend that generality vs. single specialized purpose is not the issue. Rather, designers simply want tools that are easy to use and help them solve specific problems quickly and effectively. It is probably easier to develop single purpose tools that provide such services.

The remainder of this chapter presents preliminary work toward developing a model for a general purpose computer tool that can become a wide variety of integrated and specialized design tools. The specification-based design model will be used as the basis for design evaluation. The survey presented in this section is suggestive of the variety of specialized analysis tools that will be required for systems-oriented design. Retirement planning, assembly, service, maintenance, manufacturing and environmental impact are all areas of keen interest.

6.2. The template concept

The metaphor for an integrated systems-oriented design tool was described in figure 1.1 as a series of lenses. Each lens is like a specialized design analysis tool. For example, there could be lenses for performing specialized assembly, manufacturing, or life-cycle analysis. When the design is observed through a lens, it is seen from a particular analytical perspective. However, the key point is that these lenses are all part of the same tool. Thus, data are common to all analyses and each lens may account for all of the design's requirements.

Each 'lens' is a structure or framework that encodes a specific analysis problem. This framework is referred to as an analysis template. A template defines the model for design
problems in a manner that will allow general solving technique to perform the analyses. Thus, once a library of templates has been defined, the designer will be able to pull a template for a specific problem "off the shelf" and immediately perform an analysis (without modeling overhead).

For example, the life-cycle diagram shown in figure 6.2.1 describes all of the life-cycle paths that materials in a product might follow.

Figure 6.2.1 General problem structure for the life-cycle of materials in a product.
This general life-cycle model would be solved for a specific product to predict its life-cycle. This would require consideration of the particular design's properties (such as material type and production volume). If the life-cycle problem were solved for a pop bottle, it would probably appear as something like figure 6.2.2.

Solving the life-cycle diagram involves predicting a specific outcome for each life-cycle stage and identifying which of the possible material flow paths will be followed. The numbers by each path indicate the percent of the total material that is expected to flow along that path.

Figure 6.2.2. Possible life-cycle paths for a plastic pop bottle.

The general life-cycle structure in figure 6.2.1 may be thought of as a problem template. The specific life-cycle for the pop bottle in figure 6.2.2 is the result of an analysis where the general life-cycle template was solved for a specific product design.

Therefore, the concept for the systems-oriented design tool described in section 1.1. requires defining a general representation that can be used to build templates which encode the structure of any design analysis problem. Once a general technique for solving problems defined by using this representation is developed, one does not need to create a new specialized program to evaluate different problems. Instead, a template for the new problem must be constructed using the general representation.

This may seem confusing, but there are many programs that work on this principle. For example, consider the decision analysis software program DAVID (for the Macintosh). The general representation used by this program is the influence diagram. See (Clemen
1990) for the treatment of influence diagrams. The software can solve any influence diagram. To solve a class of problems one uses the influence diagram components to build an influence diagram (template). When specific data are provided, the diagram will be solved to provide an analysis based upon this data.

Bond graphs may be used as yet another analog. Bond graphs provide a general representation (or language) that can be used to model dynamic systems. This language uses generalized variables (such as effort and flow), energy ports, and junctions. This language may be used to construct a model for a class of dynamic systems using symbolic parameters (e.g., a single degree of freedom spring-mass system). This becomes a template for evaluating all systems of this type. A specific system is evaluated when the symbolic parameters are assigned values and the graph is solved.

Spreadsheet programs offer a similar capability. The general representation of all problems is based upon the spreadsheet’s architecture (e.g., a relational database can solve problems defined using tables). Using macros, one can build a general problem structure (a template) within the spreadsheet, and then enter the data for different products to perform analyses. Frank Field, a professor at the MIT Center for Technology, Policy and Industrial Development, has developed Microsoft Excel spreadsheets for performing a variety of design analyses, ranging from sheet metal manufacturability to auto emissions and electric vehicle design.

So, why not just build an use an Excel spreadsheet? Here, the goal is to use a general representation (or problem definition language) that matches the structure of design problems. Encoding design problems in two-dimensional tables can become very confusing. Second, the solving mechanism will be based upon the specification-based design model so that, eventually, the template model and the search facility may be used in a seamless manner.

### 6.3. Template Organization

In this section a general representation for building analysis templates is defined. Further, the solving mechanism, which uses the specification-based design model and catalog structures developed earlier, is outlined briefly.
6.3.1. A graph representation for building templates

Graphs are made of two elements—nodes and branches. If one reconsiders the life-cycle example in figures 6.2.1 and 6.2.2, it is apparent that they have a graph structure. Each node corresponds to a life-cycle stage and the branches represent material flows. One could also represent an assembly as a graph where the nodes are components and the branches are interconnections. Manufacturing processes also have a graph structure in which nodes are operations and branches convey temporal sequencing or information transfer.

It seems that almost any process may be modeled as a graph. This statement is based primarily on intuition since this work is preliminary, but other design research seems to support this notion. Steve Eppinger, whose work relates to information flows and design process planning, presents a convenient matrix representation for graphs (Eppinger, Whitney et al. 1990; Eppinger 1991; Gebala and Eppinger 1991; Smith and Eppinger 1992; Krishnan, Eppinger et al. 1993; Krishnan, Eppinger et al. 1993). This matrix, illustrated in figure 6.3.1.1 for the life-cycle template, is also used by (Cha and Guo 1993) to model concurrent engineering problems.

<table>
<thead>
<tr>
<th></th>
<th>Material Selection</th>
<th>Manufacture</th>
<th>Assembly</th>
<th>Use</th>
<th>Obsolescence</th>
<th>Disposal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Material Selection</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manufacture</td>
<td>?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assembly</td>
<td>?</td>
<td></td>
<td>?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use</td>
<td></td>
<td></td>
<td></td>
<td>?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Obsolescence</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disposal</td>
<td></td>
<td></td>
<td></td>
<td>?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Figure 6.3.1.1 The life-cycle template encoded using a matrix graph representation.*
Each diagonal in the matrix is a node in the graph. The off-diagonal terms in a given row represent flows into the diagonal element. For example, the manufacture node may receive inputs from the material selection node and the obsolescence node. Question marks are used because the magnitude of these flows are not known since the template has yet to be solved for a specific design. The off-diagonal terms in a column indicate flows out of a node. The assembly node may have outputs to the use and obsolescence nodes.

### 6.3.2. Solving a template

Since all templates are represented using a matrix, analyzing a design involves determining the values of the diagonal and off-diagonal elements. The life-cycle solved for the PET bottle is shown in figure 6.3.2.1.

![Matrix diagram](image)

*Figure 6.3.2.1 The life-cycle template solved using data for a pop bottle.*
The diagonal elements will be considered first. The manufacture node of the template presents a number of different manufacturing options, such as injection molding, compression molding, or blow molding. Solving the node will involve considering the design's properties and the requirements of each manufacturing alternative to assess which is the most suitable choice. Once the blow molding option is selected, the requirements for blow molding (such as size limitations) would be added to the design's specifications. Other properties associated with blow molded parts, such as typical operating costs, might also be incorporated into the design's data.

This description should be reminiscent of a catalog selection as described in chapter 5. Each manufacturing process that might be used is a record in a manufacturing catalog. The specification fields for each record in the catalog are evaluated against the current state of the design data, and then the record with the highest acceptability is selected and the data fields within the record are transferred to the product design. Therefore, to solve a matrix diagonal is simply to perform a catalog selection.

The second component of solving a template is to determine the values of the off-diagonal terms. In the solved pop bottle life-cycle, the off-diagonal on the landfill row indicates that 40% of the bottles will probably end up on a landfill. This value is determined by considering the product's properties and the alternatives selected for the obsolescence and disposal diagonals (recycle and landfill). The off-diagonal is simply determined by a distribution that computes its value based upon the current data state (recall from section 4.3 that all distributions have this ability). Therefore, a template off-diagonal is the expected value of a distribution.

An example illustrating the relationship among a design that is to be evaluated, a template, and the transfer of data is depicted in figure 6.3.2.1. This figure will be used to clarify the general mechanism for solving a template.
Figure 6.3.2.1 An example illustrating the relationship among a design, a template, and the transfer of data when the design is evaluated using the template.

The designer is working on a problem represented by the list of data at the right of the figure. The distribution $x_1$ is the only information known about the design before evaluating the template on the left of the figure. The figure illustrates that when the template is invoked, the catalog in diagonal 11 uses the design data $x_1$ to evaluate the specification fields in its two records. Record 2 was more acceptable according to its specifications than record 1, so the data from record 2 (property 11 and spec. 11) are transferred to the design. Then, diagonal element 22 uses $x_1$ and property 11 to choose its most suitable record (record 1). The data from record 1 has also been transferred to the design. The off-diagonal element 21 has updated itself to the expected value of property 11. The template is now solved.

This description clearly illustrates that two distinct actions result from using a template to analyze a design. First, the template will change its form based upon the data in the
design (e.g., the general life-cycle template changes to predict the life-cycle of the pop bottle). This is akin to looking at the results of an analysis. Second, the template has added new properties and requirements to the design based upon the results of the analysis (choosing blow molding would add production volume requirements and size requirements—characteristic energy density and emissions might also be transferred to the design). Therefore, through the analysis, the design has also become more complete.

6.4. Template implementation

In section 6.2 the underlying representation for templates and a general description of the template solving mechanism was provided. This section describes how the template analysis framework is implemented. The template implementation involves three classes of objects. Each class has different behavior—template off-diagonals determine the value of graph branches, template diagonals perform catalog selection, and the template matrix encodes the structure of a problem. The template solving mechanism will also be discussed.

As in chapters 4 and 5, the class definitions are highly simplified so as to clearly illustrate the essence of their function.

6.4.1. Template off-diagonal class

A template off-diagonal object determines the value of a branch in the problem graph. An object of this class contains a list of distributions and a value, as shown in figure 6.4.1.1.

<table>
<thead>
<tr>
<th>Class Off-diagonal</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>data:</strong></td>
</tr>
<tr>
<td>distribution list</td>
</tr>
<tr>
<td>off-diagonal value</td>
</tr>
<tr>
<td><strong>methods:</strong></td>
</tr>
<tr>
<td>setValue()</td>
</tr>
<tr>
<td>getValue()</td>
</tr>
</tbody>
</table>

*Figure 6.4.1.1 Outline of the template off-diagonal class.*

One of the distributions in the list will be used to compute and set the off-diagonal object's value. The distribution that will be used from the list is determined by the diagonals in the object's row and column. For example, the off-diagonal $a_{ij}, i \neq j$ will choose the distribution corresponding to the records selected in the $i^{th}$ and $j^{th}$ template diagonals. This flexibility is provided for the sake of generality—in many cases the same distribution will be used regardless of which records are selected in the diagonals.
The method `setValue()` evaluates the off-diagonal object. First, the off-diagonal determines which records are selected in the \( i^{th} \) and \( j^{th} \) template diagonals and then chooses the corresponding distribution from its distribution list. This distribution then calls its update function (see section 4.3) so that it can modify itself according to the current data state. The off-diagonal's value is then set to this distribution's expected value.

The method `getValue()` simply allows other objects to obtain the diagonal's value.

### 6.4.2. Template diagonal class

The function of a template diagonal object is to choose the most appropriate record from a catalog. A schematic for this class is shown in figure 6.4.2.1.

![Class Diagonal](image)

<table>
<thead>
<tr>
<th>Class Diagonal</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>data:</strong></td>
</tr>
<tr>
<td>catalog index distribution list of off-diagonals in row</td>
</tr>
<tr>
<td><strong>methods:</strong></td>
</tr>
<tr>
<td>selectBestRecord()</td>
</tr>
<tr>
<td>updateOffDiagonals()</td>
</tr>
</tbody>
</table>

An object of this class contains a catalog index distribution (see section 5.3.1). The distribution indexes the catalog from which the diagonal object must select a record. When a record is selected from the catalog, the index distribution will also transfer data from the selected record to the design's data as required. The index distribution performed this same role in the catalog search problems described in chapter 5.

All diagonal objects also contain a list of the off-diagonal objects that are in the same row of the template matrix.

The method `selectBestRecord()` picks the most acceptable record from the catalog that is associated with the catalog index distribution. To determine which record in the catalog is most suitable, each records' specification fields (see section 5.2.1) are evaluated against the distributions that describe the design's properties. The record with the highest overall probability of being acceptable will be chosen. That is, selection is performed using the specification-based model. Once the most suitable record is chosen, the catalog index distribution's `transferData()` method (see section 5.3.1) moves data from the record to the design data as required.
The method `updateOffDiagonals()` instructs each of the row's off-diagonal objects to update its value (as described in section 6.4.1).

### 6.4.3. Template matrix class

Finally, objects of the template matrix class provide the structure and solving mechanism needed to encode and resolve general classes of design problems. The template matrix class is outlined in figure 6.4.3.

<table>
<thead>
<tr>
<th>Class Template Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>data:</strong> diagonal list</td>
</tr>
<tr>
<td><strong>methods:</strong> solve(), print()</td>
</tr>
</tbody>
</table>

An object of this class contains a single list of the diagonal objects that form the template. The method `solve()` instructs each diagonal to select its best record, update the product data, and set the row's off-diagonal values. The `print()` method simply lists the template matrix. Pseudo code for the solving algorithm is provided below.

```plaintext
repeat {
    for (each diagonal in the template) {
        select record with highest probability of acceptance
        update off-diagonal values
    }
} until (matrix solution has stabilized)
print the template matrix
```

The algorithm solves the template from top to bottom. Some diagonals and off-diagonal's may require data that are not available until later rows of the matrix are solved. Consider the simple template example that was provided in figure 6.3.2.1. In this instance, evaluation of diagonal 22 requires data that is not available until diagonal 11 is solved. This was not an issue in the figure, as diagonal 11 is solved first. However, what if diagonal 22 is placed before diagonal 11 (i.e., their positions in the matrix are reversed)? Now, when diagonal 22 is evaluated for the first time, property 11 will not yet be available. Even though these data are missing, the record selection process for diagonal 22 will proceed using the optimism level for property 11 (see section 5.1.3 for a discussion about optimism levels). Then, the solving process would move on to address diagonal 11. Only on a second pass through the matrix will property 11 be available for
the evaluation of diagonal 22. This illustrates that the template must be solved iteratively until the solution stabilizes. Here, changes in record acceptability ratings are used as a convergence criterion.

Like the hierarchical search algorithm implemented in section 5.3.3, this solution algorithm is very simple. Clearly, the order of the templates' diagonals is important if the number of iterations needed for convergence is to be minimized. Work by (Gebala and Eppinger 1991) suggests a way of automatically arranging the matrix so that the number of iterations will be minimized. Additionally, a study of literature in the area of graph solving is needed to see if more efficient and robust solution algorithms have been developed. However, the approach outlined here is sufficient to demonstrate the template concept and create templates for a number of design problems. Therefore, this research is left for future work.

6.5. Template examples

In this section examples of templates that perform a variety of analyses are presented. These preliminary examples are intended to illustrate how the template representation may be used to encode a range of design evaluation problems. These results also suggest that templates may be built to provide life-cycle evaluation capabilities similar to those mentioned in section 6.1.

Two templates are described. First, a template for designing two member trusses (like the one used as an example throughout chapter 4 and 5) is demonstrated. Then, another simple template is defined to emulate the functionality of the axiomatic design-based program described in section 3.1.1. This example will show that the template representation can be used to create specialized analysis programs without writing a separate CAD tool.

6.5.1. A two member truss template

Throughout chapters 4 and 5, it was assumed that the designer was able to build the model needed to evaluate the two member truss. However, what would happen if a non-engineer wanted to design a truss with this topology? An engineer could create a standard template for analyzing this type of truss. Non-technical designers could pull this template
from a library and use it to ensure that their design will not fail, even though they do not know how to model the truss problem.

Here, a template is implemented so that designers pick the stock size they wish to use (steel is assumed), set the base of the truss, and express their preferences for the cost of the design. The template will provide designers with feedback about whether the design will be safe and how much the materials needed for the design will cost. I will begin by describing the structure and operation of the template, and then provide output of the template solved for a specific truss design.

A diagram of the template's structure and how it interacts with the design's data is provided in figure 6.5.1.1.

The design's data are shown at the right of the figure. Initially, the designer will have defined the truss parameters $d_1, r_1, d_2, r_2, s$ and the cost specification. Template (at the left) has four diagonals. Let us consider the 11 diagonal element in detail. This diagonal makes a selection from the member 1 stress catalog. This catalog contains only one record, so when the diagonal is evaluated the same record will always be chosen (this record is called member 1 stress). The record contains one variable field and one specification field. The variable field holds a distribution for the normal stress safety factor for member 1 ($n_1$). The single specification field contains a stress safety factor specification. This specification is used to evaluate the stress safety factor $n_1$ during the record selection process. Notably, before the acceptability is determined, distribution $n_1$'s update function uses the design's values for $d_1, r_1,$ and $s$ to compute the stress safety factor for member 1. Thus, the record's acceptability is actually the probability that member 1 will successfully meet the stress safety requirement.
Figure 6.5.1.1 Illustration of a template for evaluating three-element trusses.

Once the record has been evaluated, the stress safety factor for member 1 and the stress requirement for member 1 are added to the design's data. In this example, the off-diagonals in each row are simply used to display the expected value of the performance characteristic computed when the diagonal's record is selected. For example, in row 1 the off-diagonals display the expected value of the safety factor n1.
The remaining three rows of the template operate in a similar manner, evaluating: the buckling performance of member 1; the stress performance of member 2; and the cost of the design. The only notable variation is that, when the specification in the cost catalog's record is evaluated, the specification first updates itself so that it matches the designer's cost preference.

So, how would this template be used to evaluate a truss design? One begins by defining the truss's parameters. Therefore, in addition to setting the truss base, the designer would choose the diameters and diameter ratios for the two truss members. In this case, the parameter values are set to correspond with the optimized truss in figure 5.4.1.2. The designers would also define their specification for the total material cost. The truss cost specification used throughout chapters 4 and 5 was also applied here. Then, the designer would select the truss template and request an evaluation of the design.

Next, the truss template will use this data to evaluate the specific truss configuration.

Output showing the results of this evaluation is provided below.

<table>
<thead>
<tr>
<th>truss template rows before evaluating</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1 stress catalog</td>
</tr>
<tr>
<td>0.000</td>
</tr>
<tr>
<td>m1 buckle catalog</td>
</tr>
<tr>
<td>0.000</td>
</tr>
<tr>
<td>m2 stress catalog</td>
</tr>
<tr>
<td>0.000</td>
</tr>
<tr>
<td>cost catalog</td>
</tr>
<tr>
<td>0.000</td>
</tr>
<tr>
<td>template has been successfully solved</td>
</tr>
<tr>
<td>m1 stress</td>
</tr>
<tr>
<td>14.238</td>
</tr>
<tr>
<td>m1 buckle</td>
</tr>
<tr>
<td>3.036</td>
</tr>
<tr>
<td>m2 stress</td>
</tr>
<tr>
<td>2.001</td>
</tr>
<tr>
<td>cost</td>
</tr>
<tr>
<td>146.737</td>
</tr>
</tbody>
</table>

Each diagonal element displays the name of the record chosen from its catalog. The suitability of the selected record is shown in the column to the right of the matrix. For this template, these values are the probability the design is acceptable according to the stress,
buckling and cost requirements. Each row of off-diagonals lists the expected value of the performance characteristic that was established during the diagonal's evaluation. (In this case the off-diagonals have no significance as graph branches). For example, the first row of the matrix evaluates the stress safety factor of member 1; its expected value is 14.238 and the probability that this is acceptable is 1.0.

The output is primitive, but it does show how the template provides designers with additional information about the truss design they are working on. (This interface simply permits testing of the template implementation.) The designer could experiment by adjusting the design's parameters and seeing how the truss's performance evaluation responds. It may seem like it trouble than it's worth to build a template for designing such a simple truss. However, once the template is defined, any designer, technical or non-technical, can use this template to analyze a design with this topology. The template separates the problem structure and solution method from design specific data.

One might also note that the truss template could also have the form illustrated in figure 6.5.1.2. This template contains only one diagonal element. The acceptability of the chosen record would indicate the overall probability of meeting all of the record's requirements.
6.5.2. Product life-cycle template

In section 3.1.1.2 and (Wallace and Suh 1993) a program that uses expected product attributes to predict suitable assembly and obsolescence life-cycle design goals is described. Now, a template providing similar functionality is presented.

To build the life-cycle goal setting template, catalogs are defined which contain records for the assembly options and the product end-life alternatives (see figure 6.5.2.1). Each record has specifications that are used to assess how well the alternative suits the product in question. For example, the dispose record will evaluate: the product's expected life against its acceptable life specification; the product's expected production volume against its acceptable production volume specification; and, the product's expected value-added ratio against its acceptable value-added specification. This evaluation determines the probability that the product will be acceptable according to the dispose record's requirements.
Figure 6.5.2.1 Catalogs used by the life-cycle prediction template.

These catalogs are then used as the diagonal elements of a life-cycle goal prediction template. The template functionality is illustrated in figure 6.5.2.2. The end-life options catalog uses the product life, production volume, value-added, styling level and material recycling potential to choose the most suitable end-life option record. Similarly, the assembly options catalog uses the product's expected production volume, desired payback period and complexity (a qualitative scale related to number of parts and assembly difficulty) to select the most suitable assembly record. When a record is selected (as depicted for the recycle and manual assembly records) its specifications are added to the design's data. The off-diagonal elements are not used in this template.
Figure 6.5.2.2 Schematic of the life-cycle prediction template.

Example output from this template is included below. The product data (attributes) are set to match the values used by the previous example in figure 3.1.1.2.3, section 3.1.1.2 (life-cycle 3 years, production volume 95,000/year, desired payback period one year, degree of
styling 8.5 on a scale from 0-10, thermoplastic material). Like the previous program, recycling is the best end-life option for the product \( p_{\text{acceptable}} = 0.750 \) and manual assembly is the best assembly method option \( p_{\text{acceptable}} = 0.370 \).

**life-cycle template rows before evaluating**

<table>
<thead>
<tr>
<th>end-life options</th>
<th>0.000</th>
<th>( p = 0.000 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>catalog</td>
<td></td>
<td></td>
</tr>
<tr>
<td>assembly options</td>
<td>0.000</td>
<td>( p = 0.000 )</td>
</tr>
<tr>
<td>catalog</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recycle</td>
<td>0.000</td>
<td>( p = 0.750 )</td>
</tr>
<tr>
<td>manual assembly</td>
<td>0.000</td>
<td>( p = 0.370 )</td>
</tr>
</tbody>
</table>

Notably, some of the product data needed to evaluate the acceptability of the different options are not defined (value-added ratio and the complexity level). Again, the optimism level is used to circumvent this problem. The program in section 3.1.1.2 ignores specifications that require unavailable data, which is equivalent to using an optimism level of 1.0 (implying that one feels that if the data were available, it would be acceptable according to specifications).

Additionally, the constraints associated with the chosen life-cycle goals shown in figure 3.1.1.2 could also be incorporated into the design's data as binary specifications (similar to figure 2.2.1). Finally, the program described in section 3.1.1.2 provides guidelines that the designer may use to achieve the suggested life-cycle goals. All though not detailed in this thesis, all record objects may contain text documentation. This documentation could be used to provide design guidelines.

### 6.6. Summary and discussion of the integrated use of templates and design search

This chapter has presented preliminary work on developing a general representation for structuring specification-based design problems. A matrix model is proposed. The diagonal elements have the behavior of a catalog-selection problem, while the off-diagonal terms are distributions that may be used to display information about the structure of the problem.
A template is a matrix which has been defined to evaluate a specific class of problems. The template captures the structure and solution technique for the problem. It is hoped that the preliminary examples in the previous section illustrate that this representation has the potential to become a powerful way of creating specialized analysis tools that share a common evaluation technique. This is critically important as systems-oriented product design requires that the analysis of different life-cycle stages should be linked. For example, when the retirement template in section 6.5.2 was solved, the specifications pertaining to the chosen retirement method were added to the design's data. One could then evaluate all of the design's specifications to see if these new requirements contradict design specification pertaining to other aspects of the product's life-cycle.

However, it is probably unclear how the templates developed in this chapter relate to the search facility developed in chapters 4 and 5. There are two important capabilities that a systems-oriented design tool must provide. First, given the model for a problem, the tool must help designers find acceptable design solutions. This capability is provided by the search facility. Second, the tool must provide designers with the models necessary to evaluate aspects of the design that are beyond their own expertise. A designer cannot be expected to understand all aspects of modeling a product's entire life-cycle! Templates are intended to provide this capacity.

Together, both facilities provide the functionality that is required for systems-oriented design. Since the template and search facilities share the same evaluation principle (the specification-based decision model) and data representation, it will be possible to combine them into a single coherent design tool. Figure 6.6.1 illustrates this idea. In this figure, the design is initially defined by the parameter $x_1$. The designer has selected a template for the class of problem he is trying to solve. Instead of simply evaluating the design using the template, the designer has manually forced the template to the outcome he wants for his design. (for example, he might select recycling from the retirement catalog). Through this action, appropriate performance variables (property 11, property 22) and specifications are added to the design's data (specification 11, specification 22). In other words, the template has transferred the problem model to the design. The designer may now use this model to optimize the design parameter ($x_1$) to best satisfy all of the design's requirements (e.g., optimize $x_1$ according to the specifications for recycling in addition to any other design requirements that may exist).
Figure 6.6.1 Integration of templates and search into a single systems-oriented CAD tool.

Figure 6.6.2 (a and b) illustrates this idea for the life-cycle prediction template described in section 6.5.2. In figure a), the desired life-cycle options for a design have been chosen by the designer. Selection causes the transfer of new specifications to the design's data.
These requirements must be met if the design is to be acceptable for the desired life-cycle. In figure b), the design variables and the new specifications are passed to the search facility. The search software may now design the product so that is acceptable for the desired life-cycle outcome.

Figure 6.6.2a Illustration of integration for the life-cycle prediction template.
Figure 6.6.2b Illustration of integration for the life-cycle prediction template.
7. Conclusions

The scope of problems that product designers must address is continually growing. Due to the increasing pressure to design high-quality environmentally responsible goods, many industries now wish to account for life-cycle impact during product design. The current interest in environmental design is part of a general trend toward systems-oriented product design. In the future, even non-technical consumer products will be designed as part of a larger anthropological and ecological system. This change will require that non-technical designers must comprehend and evaluate a diverse range of design problems over a complete product life-cycle (from material extraction through manufacture and assembly, to use and obsolescence).

Complex systems-oriented product design, which includes environmentally responsible design, does not lend itself to resolution through simple intuition, rules of thumb, or context-insensitive guidelines. Every aspect of a product's life-cycle has environmental implications. The goal of this thesis was to develop a method that will facilitate the rational design of mass-produced consumer products from a system perspective. A computer tool using such a method will help product designers, both technical and non-technical, structure and understand the subtle interaction of design decisions in life-cycle problems.

This thesis developed the three main components that will be needed to affect an integrated systems-oriented design tool capable of addressing a wide variety of problems. The first step was to develop a way of modeling multiple-criteria problems in a familiar design language—product designers are not decision analysts (chapters 2 and 3). The specification-based design evaluation principle was then used as a uniform and meaningful evaluation framework for computer-based design tools.

Second, given the model for a problem, the designer must use the specification-based evaluation technique to develop acceptable design solutions. In large multiple-specification problems it may be very difficult to assess what changes will best satisfy the design requirements. The robust search facility provides the capability to automatically find design solutions that will best meet the design specifications (chapters 4 and 5). The
general search tool allows simultaneous continuous parameter selection and catalog selection.

Third, many systems-oriented design problems are too large and time consuming to model on an individual design basis. Further, designers will not have the expertise needed to model many of the complex system problems they must address. Many product designers are non-technical and, moreover, a designer cannot be expected to understand all aspects of a product's entire life-cycle! The representation for specification-based analysis templates was proposed as a way to provide designers convenient access to the models needed for a multitude of systems-oriented design analyses (chapter 6).

Because the two software kernels share the same specification-based evaluation principle, it will be possible to combine life-cycle design templates with the general search facility (section 6.6). Together, these three components—the specification-based design model, the general search facility, and the template representation for encoding categories of problems—will permit the future development of an integrated systems-oriented design tool. The specification-based design model provides a meaningful design evaluation metric, while templates provide the models that are needed to search for design solutions that satisfy a wide variety of integrated life-cycle goals.

7.1. Specification-based design decision model

7.1.1. Contributions

It was concluded that design guidelines, in isolation, inadequately address the demands of systems-oriented product design. A method that will allow designers to rationally evaluate multiple-criteria problems is needed. If a decision method is to be used by product designers, it must use a familiar design language, be easy to understand, and provide the designer with feedback that is in a relevant form.

Therefore, a specification-based design-decision model was developed. A multiple-attribute decision has two distinct parts. The value of the individual design attributes must be assessed, and then these values must be combined to provide an overall design evaluation.
It is hypothesized that design is a specification-satisfaction process, and hence the value of a design attribute is given by the probability that its level of performance is acceptable. A design specification is an absolute scale value function which indicates the probability that various attribute levels will be deemed acceptable. Each individual specification is set assuming the design is otherwise acceptable. The probability that a design will be acceptable on the basis of a single attribute is determined by the weighted integral of the performance attribute's probability density function and the specification acceptability function. Significantly, the method accommodates uncertainty in both the design performance level and in the levels of performance which will be acceptable (the specifications).

Next, it is suggested that the overall value of a multiple-attribute design is the probability that the design will be judged as acceptable according to all of the design specifications. This probability is the product of the individual probabilities of acceptance for each design variable. Thus, the multiple-attribute decision rule states that the best design has the highest probability of acceptance.

This combination rule has two important characteristics. The formulation implies that if a design is unacceptable according to one attribute, the overall design is also unacceptable. This was described as the annihilation property. It is felt that this trait effectively emulates the design decision process. If a design has an unacceptable stress level, it is unacceptable regardless of how it performs in other areas. Secondly, it was shown that because the specification functions measure the probability the design will be deemed acceptable, weights are not needed in the overall design metric formulation. Because weighting factors are often difficult to set and many product designers are skeptical about their use, this is a positive result.

Finally, it is emphasized that design specifications are not static. It is argued that the specification setting process is as much a design problem as is creating an artifact. Therefore, design using the specification-based model is viewed as a coevolution process of both the design artifact and its specifications.
7.1.2. Future Work

The design decision model is justified primarily upon personal observations about the design process. Designers are familiar with the notion of working to meet specifications, so a specification-based decision model seems reasonable. Defining specifications as value functions that indicate the probability a given level of performance will be acceptable also seems fairly intuitive—designers must simply consider what levels of performance they will be satisfied with. In effect, traditional design specifications and regulations operate in this manner (if the design is in specification it is acceptable, if it is out of specification it is unacceptable). The overall combination metric, with its annihilation characteristic, seems to emulate how designs are presently evaluated (when one attribute of the design is unacceptable, the design is also unacceptable).

However, this combination metric is unusual in a number of ways. Most multiple attribute decision rules choose the design with the highest expected value. For example, utility analysis assumes that the goal is to maximize the expected utility. The multiplicative metric used in this thesis does not necessarily choose the design with the highest expected acceptability for each attribute. Consider two designs: design A has a probability of acceptance of 1.0 on attribute 1 and 0.0 on attribute 2; design B has a probability of acceptance of 0.3 on attribute 1 and 0.6 on attribute 2. According to the decision rule used in the thesis, the overall probability that design A is acceptable is 0.0, while the overall probability that design B is acceptable is 0.18. Design B is preferred. However, the expected acceptability of an attribute in design A is greater than for design B (0.5 vs. 0.45).

In effect, the specification-based decision rule acts on the "squeaky wheel gets the grease" principle. This approach will require that many attributes may become less than perfectly acceptable to prevent one attribute from being unacceptable. An expected value approach will ignore the "squeaky wheel" if fixing it will compromise other attributes to the extent that the overall expected value of each attribute is diminished. While the small case study presented in section 2.4 suggests that the "squeaky wheel" model is appropriate, a thorough study of the appropriateness of the acceptability metric should be conducted.
Additionally, the method assumes that the designer can define specifications on an absolute acceptability scale. Although it seems that traditional design specifications already operate in this manner, this assumption should be validated.

7.2. Specification-based design search

7.2.1. Contributions

A decision model alone brings us only part way to solving complicated and counter-intuitive system problems. To be of practical use, and thereby justify the effort to model a problem, an analysis must provide assistance in finding better alternatives.

An architecture and software implementation for design search that allows problem formulation in the language of design specifications has been implemented. It is not reasonable, in my opinion, to expect product designers to possess expertise in mathematical programming—the designer should not be required to exhibit mastery in the area of optimization.

Therefore a robust search facility which uses a genetic algorithm was developed. The implementation eliminates the need to distinguish between multiple objective or single objective problems and probabilistic or deterministic analysis. Importantly, objective function formulation is automated and the need to provide a feasible starting point is eliminated.

Further, a structure for catalogs based upon the specification model was proposed and implemented. A hierarchical organization based upon data abstraction was recommended as a way to help designers communicate where design flexibility exists.

Additionally, the optimism level concept was introduced as a way to permit analysis to continue when data needed for calculations are missing. These additions enable design searches that simultaneously explore continuous variables and data within catalogs, and choose between alternative catalogs. The optimism concept allows the search of incomplete catalogs. Finally, a preliminary approach to hierarchical search was implemented.
7.2.2. Future Work

There is a number of amendments and additions that would improve the specification-based search facility. The present implementation does not address the evaluation of specifications when the design performance variables are coupled. For all the examples presented in the thesis, it was possible to order the performance variable list so that all necessary data were up-to-date when each variable calls is update function. However, coupled performance variables can be addressed easily by iteratively updating the performance variable list until the distributions stabilize. Only then could the fitness of a design be evaluated.

It is easy to perform mathematical operations on combinations of like probability density functions (such as delta functions). However, I am not aware of a closed-form method to evaluate equations which involve arbitrary combinations of beta, piecewise linear and delta functions. This situation was not addressed by the thesis. However, this is an important capability that must be developed if the facility is to be used for general probabilistic analysis. I suspect that the best way to resolve this problem will be to perform Monte-Carlo simulations to approximate the forms of the distributions, and then fit piecewise linear or beta distributions to match the estimated distribution. More research is required.

Further, all optimization problems and the distribution update functions must be defined in a C++ file. Obviously, if one cannot expect non-technical designers to be optimization gurus, one can also assume that they are not programming experts either. An interactive method for defining optimization problems will be required before product designers can work with the tool. This is not a particularly difficult research issue, but it is a large programming endeavor.

Additionally, the genetic algorithm provides only the best answer. However, in my experience, many multiple criteria problems will have a very hilly objective function (like the camelback). Often, the designer is not only interested in the best answer, but would like to see a number of different but reasonably acceptable options. This capability could be incorporated into the search facility by using a speciating genetic algorithm. This type of genetic algorithm tends to cluster onto maxima (local or global) throughout the search space.
Finally, I believe that the most challenging research issue relates to performing searches with catalog hierarchies. This problem is complex because many different catalogs may be used in a given design problem and interactions between their hierarchies must be considered during the search. Searching among alternative catalogs further exacerbates this complexity. The issue of catalog search was only superficially addressed in this thesis.

7.3. Specification-based problem templates

7.3.1. Contributions

The research in this area is preliminary, but this work is included in the thesis because it is an essential part of achieving the longer term goal of a complete method that may be used for integrated systems-oriented product design. The work describes how standard specification-based analysis problems might be represented and pre-encoded. Problem templates will allow designers to analyze various aspects of a product's life-cycle conveniently. Once a library of templates has been developed, it will be possible to evaluate many aspects of a design's life-cycle without modeling overhead. This is important as most product designers will not have the expertise needed to model many life-cycle design problems.

It is hypothesized that many design problems exhibit graph-like properties. Therefore, a matrix representation of graph structures is suggested as an appropriate template structure. This representation has been used extensively in work by Steve Eppinger. The matrix representation contains two different type of elements—diagonal elements and off-diagonal elements. The diagonal elements are graph nodes, and off-diagonal elements display information about the relationship between nodes.

All diagonal elements have the behavior of a catalog selection problem. A template diagonal is solved when the most acceptable record is chosen from the catalog that the element indexes. All off-diagonal elements have the behavior of a distribution—they update their values based upon the records chosen in the diagonal elements and the design data. These two elements provide a language that can be used to model...
specification-based design evaluation problems (much like bond graphs provide a language for modeling dynamic systems).

A template is a matrix which has been constructed from diagonal and off-diagonal elements to evaluate a specific class of problems. The template captures the structure and solution technique for the problem. When the template is used to analyze a specific design, the template will use data from the product to compute the values of its diagonal and off-diagonal elements. It is felt that this representation has the potential to become a powerful way of creating specialized analysis tools that share a common evaluation technique.

7.3.2. **Future Work**

The template representation is in an early development stage and requires considerable additional research. A study of graph literature and the template solving algorithm is required. Additionally, the template can be used to encode assembly and disassembly graphs. I would like to study the possibility of using matrix triangularization operations for assembly and disassembly planning.

Like the search facility, problem templates are defined by writing an appropriate C++ file. It will be necessary to implement a facility to interactively define analysis templates, perform analyses, and display analysis results. Additionally, research aimed at modeling various life-cycle problems must be conducted before corresponding analysis templates can be constructed.

7.4. **Ongoing development of a systems-oriented CAD tool**

The three components addressed by this work—the specification-based design model, the general search facility, and the template representation for encoding categories of problems—will permit the future development of an integrated systems-oriented design tool. The specification-based design model provides a meaningful evaluation principle, while templates provide the models that are needed to search for design solutions that satisfy a wide variety of integrated life-cycle goals.
The lens concept described in section 1.1 illustrates how designers would use templates to view their designs from different life-cycle perspectives. There are three major issues to address before the systems-oriented computer tool becomes a reality.

An interactive graphical user interface is needed to define and analyze problems using templates and the search facility. Considerable work has already been devoted to the design and development of a graphical user interface (GUI). These interfaces are being implemented using the X-windows based Motif GUI language.

A second major area to be addressed is data management. The data needed for the examples in the thesis are encoded by writing appropriate C++ files that build the lists and catalogs used by the template and search facilities. However, the template and search implementations have been designed for eventual integration with Object Store, an object oriented database program.

Finally, there is the issue of how the data that are needed for systems-oriented analysis will be extracted from CAD models. Will the designer manually enter the necessary data, or can this process be automated? This is a difficult long-term research issue.

Even once the integrated systems-oriented design tool becomes a reality, it will be necessary to define the goals of systems-oriented design. Is it environmentally responsible or "green" design? If so, what does this mean? Is the goal to conserve resources, to reduce environmental impact and toxicity, or to increase profits?
Appendix A: Source code and output for the camelback optimization example

```cpp
#include "distribution.h" // defines the distribution base class
#include "delta.h" // defines the delta distribution class
#include "linear.h" // defines the linear distribution class
#include "optimise.h" // defines the optimisation class
#include "dynArray.h" // defines a class for handling lists
#include <math.h>

// declare the design variables (delta functions)
delta x1( "x1", // the variable's name
    -3.0, // lower search boundary
    3.0, // upper search boundary
    20); // variable resolution in bits

delta x2( "x2", // the variable's name
    -3.0, // lower search boundary
    3.0, // upper search boundary
    20); // variable resolution in bits

// define the function to compute the performance variable F
void computeCamelBack(distribution * F)
{
    F->setExpectedValue
        ((4-2.1*pow(x1,2)+pow(x1,4)/3)*pow(x1,2) + x1*x2 +
         (-4+4*pow(x2,2)))*pow(x2,2));
}

// declare the performance variable (delta functions)
delta F( "camelback function value",
    computeCamelBack); // pass the function used to set its value

// declare the specification variable (a piecewise linear distribution)
linear spec(0, 1.0, // slope and intercept of first line segment
    -0.25, 0.5, // slope and intercept of second line segment
    -4, -2, 2); // the line segment boundaries

main ()
{
    // declare the three distribution lists;
dynArray designVariableList;
dynArray performanceVariableList;
dynArray specificationList;

    // assign the design variables to the design variable list
    designVariableList.add(&x1);
designVariableList.add(&x2);

    // assign performance variable to the performance variable list
    performanceVariableList.add(&F);

    // assign the specifications to the specification list
    specificationList.add(&spec);

    // declare and perform the optimisation
    // the results are automatically printed to the screen
```
optimise camelBackExample(
    designVariableList,
    performanceVariableList,
    specificationList);
)

Program Output

starting optimisation!
number of generations 100

printing delta distribution: x1
data source: none
comments: none
scale: none
delta location at: x = 0.089501

printing delta distribution: x2
data source: none
comments: none
scale: none
delta location at: x = -0.712249

printing delta distribution: camelback function value
data source: none
comments: none
scale: none
delta location at: x = -1.031627
objective 0 value is 0.757907

overall solution value is 0.757907

done!
Appendix B: Equations for the truss design example

Constants

\[ W = 13000 \text{ N} \]
\[ h = 4000 \text{ mm} \]
\[ E = 2 \times 10^5 \text{ MPa} \]
\[ k_{\text{pipe}} = 3 \times 10^{-5} \text{ cost units/mm}^3 \]
\[ k_{\text{rod}} = 1.5 \times 10^{-5} \text{ cost units/mm}^3 \]
\[ \text{Yield Strength} = 310 \text{ MPa} \]

Stress Safety Factors \( n_{\sigma 1}, n_{\sigma 2} \)

\[ n_{\sigma} = \left| \frac{\sigma_{\text{yield}}}{\sigma_i} \right| \quad \text{and} \quad \sigma_i = \frac{-2W}{\pi s} \frac{\sqrt{h^2 + s^2}}{d_i^2 (1 - r_i^2)}, \quad \sigma_2 = \frac{2W}{\pi s} \frac{\sqrt{h^2 + s^2}}{d_2^2 (1 - r_2^2)} \]

Buckling Safety Factor \( n_{b1} \)

\[ n_{b1} = \frac{P_{cr1}}{F_1} \quad \text{and} \quad P_{cr1} = \frac{-\pi^3 E}{64} \frac{d_1^4 (1 - r_1^4)}{s^2 + h^2}, \quad F_1 = \frac{-W}{2s} \frac{\sqrt{h^2 + s^2}} \]

Cost

if rod \[ C = \frac{k_{\text{rod}} \pi}{4} d^2 (1 - r^2) \sqrt{h^2 + s^2} \]

if pipe \[ C = \frac{k_{\text{pipe}} \pi}{4} d^2 (1 - r^2) \sqrt{h^2 + s^2} \]

Total Cost \( C = C_{\text{member1}} + C_{\text{member2}} \)
Appendix C: Formulations for the policy setting example

$P_{\text{existing market within criteria}}$ is calculated from equation 2.2.1 using the existing energy consumption distribution and the design variable (energy labeling criteria).

**Cost**

Assumed to be a symmetric beta distribution with $\alpha = \beta = 3$. The mean of the distribution is calculated from

$$\Delta C_{\text{mean}} = 15 \left( \frac{1}{p_{\text{feasible}}} \right) \left( \frac{\varepsilon_E - L_U}{\varepsilon_E} / 0.05 \right)$$

Where

- $\varepsilon_E$ is the expected energy consumption of the existing market distribution
- $L_U$ is the upper limit that will meet the energy labeling criteria
- $p_{\text{feasible}}$ is the likelihood that the required change in energy consumption is possible.

As long as the energy level demanded by the criteria is deemed to be feasible, the mean cost changes 15 dollars per 5% decrease in energy consumption. The feasibility measure is determined using the criterion defined below. The mean cost rises exponentially as the probability that the criterion asks for feasible energy consumption levels goes down. The width of the cost change distribution is assumed to be 15% percent on each side of the mean.
Appendix D: Outline of material property catalog used in examples

An outline of the material property catalog developed for examples of combined parameter and catalog search problems is provided in this appendix. The outline indicates the material classes in the catalog tree and the data that are available for each category. Data were obtained from (Ashby 1989; Budinski 1989; Ashby 1992a; Ashby 1992b). All data were approximated with uniform probability density functions. Much of this catalog was encoded by Johnny Chang as part of the UROP program at MIT.

To save space, numbers are used in the tables to designate different material properties. This designation is as follows:

<table>
<thead>
<tr>
<th>Designation</th>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>density</td>
</tr>
<tr>
<td>1</td>
<td>Young's modulus</td>
</tr>
<tr>
<td>2</td>
<td>energy density</td>
</tr>
<tr>
<td>3</td>
<td>relative cost per volume</td>
</tr>
<tr>
<td>4</td>
<td>yield strength</td>
</tr>
<tr>
<td>5</td>
<td>damping coefficient</td>
</tr>
<tr>
<td>6</td>
<td>thermal conductivity</td>
</tr>
<tr>
<td>7</td>
<td>thermal diffusivity</td>
</tr>
<tr>
<td>8</td>
<td>thermal expansion</td>
</tr>
<tr>
<td>9</td>
<td>strength at temperature</td>
</tr>
<tr>
<td>10</td>
<td>normalized wear rate</td>
</tr>
</tbody>
</table>

Catalog: Engineering Materials

Parent Record: none
Parent Record in Catalog: none

<table>
<thead>
<tr>
<th>Material Record</th>
<th>Data in Record</th>
<th>Child Catalog</th>
</tr>
</thead>
<tbody>
<tr>
<td>polymer foams</td>
<td>0, 1, 2, 3, 4, 5, 6, 7, 8, 9</td>
<td>foams catalog</td>
</tr>
<tr>
<td>elastomers</td>
<td>0, 1, 2, 3, 4, 5, 6, 7, 8, 9</td>
<td>elastomer catalog</td>
</tr>
<tr>
<td>engineering polymers</td>
<td>0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10</td>
<td>polymer catalog</td>
</tr>
<tr>
<td>composites</td>
<td>0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10</td>
<td>composite catalog</td>
</tr>
<tr>
<td>woods</td>
<td>0, 1, 2, 3, 4, 5, 6, 7, 8, 9</td>
<td>none</td>
</tr>
<tr>
<td>engineering alloys</td>
<td>0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10</td>
<td>alloy catalog</td>
</tr>
<tr>
<td>porous ceramics</td>
<td>0, 1, 2, 3, 4, 5, 6, 7, 8, 9</td>
<td>porous ceramics catalog</td>
</tr>
<tr>
<td>engineering ceramics</td>
<td>0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10</td>
<td>ceramics catalog</td>
</tr>
</tbody>
</table>
**Catalog: Foams**

**Parent Record:** polymer foams  
**Parent Record in Catalog:** Engineering Materials

<table>
<thead>
<tr>
<th>Material Record</th>
<th>Data in Record</th>
<th>Child Catalog</th>
</tr>
</thead>
<tbody>
<tr>
<td>cork</td>
<td>0, 1, 2, 4, 5, 6, 7, 8</td>
<td>none</td>
</tr>
<tr>
<td>polystyrene</td>
<td>4, 5, 6, 7, 8</td>
<td>none</td>
</tr>
</tbody>
</table>

**Catalog: Elastomer**

**Parent Record:** elastomers  
**Parent Record in Catalog:** Engineering Materials

<table>
<thead>
<tr>
<th>Material Record</th>
<th>Data in Record</th>
<th>Child Catalog</th>
</tr>
</thead>
<tbody>
<tr>
<td>soft butyl</td>
<td>0, 1, 2, 4</td>
<td>none</td>
</tr>
<tr>
<td>hard butyl</td>
<td>0, 1, 2, 4</td>
<td>none</td>
</tr>
<tr>
<td>polyurethane</td>
<td>0, 1, 3</td>
<td>none</td>
</tr>
<tr>
<td>silicone</td>
<td>0, 1, 3, 4, 9</td>
<td>none</td>
</tr>
</tbody>
</table>

**Catalog: Polymer**

**Parent Record:** engineering polymers  
**Parent Record in Catalog:** Engineering Materials

<table>
<thead>
<tr>
<th>Material Record</th>
<th>Data in Record</th>
<th>Child Catalog</th>
</tr>
</thead>
<tbody>
<tr>
<td>epoxies</td>
<td>0, 1, 3, 4, 6, 7, 8, 9</td>
<td>none</td>
</tr>
<tr>
<td>melamines</td>
<td>0, 1, 4</td>
<td>none</td>
</tr>
<tr>
<td>polycarbonate</td>
<td>0, 1, 3, 9</td>
<td>none</td>
</tr>
<tr>
<td>polyester</td>
<td>0, 1, 3, 4, 5, 6, 7, 8, 9</td>
<td>none</td>
</tr>
<tr>
<td>high density polyethylene</td>
<td>0, 1, 2, 3, 4, 6, 7, 8, 9, 10</td>
<td>none</td>
</tr>
<tr>
<td>low density polyethylene</td>
<td>0, 1, 2, 3, 4, 6, 7, 8, 9, 10</td>
<td>none</td>
</tr>
<tr>
<td>polyformaldehyde</td>
<td>3, 6, 7, 8, 9</td>
<td>none</td>
</tr>
<tr>
<td>polymethylmethacrylate</td>
<td>0, 1, 3, 4, 5, 6, 7, 8, 9</td>
<td>none</td>
</tr>
<tr>
<td>polypropylene</td>
<td>0, 1, 3, 4, 5, 6, 8, 9</td>
<td>none</td>
</tr>
<tr>
<td>polytetrafluoroethylene</td>
<td>0, 1, 4, 5, 7, 8, 9, 10</td>
<td>none</td>
</tr>
<tr>
<td>polyvinylchloride</td>
<td>0, 1, 2, 3, 4, 6, 7, 8, 9</td>
<td>none</td>
</tr>
</tbody>
</table>

**Catalog: Composite**

**Parent Record:** composites  
**Parent Record in Catalog:** Engineering Materials

<table>
<thead>
<tr>
<th>Material Record</th>
<th>Data in Record</th>
<th>Child Catalog</th>
</tr>
</thead>
<tbody>
<tr>
<td>carbon fiber reinforced</td>
<td>3</td>
<td>none</td>
</tr>
<tr>
<td>glass fiber reinforced</td>
<td>3</td>
<td>none</td>
</tr>
</tbody>
</table>
### Catalog: Alloys

**Parent Record:** engineering alloys  
**Parent Record in Catalog:** Engineering Materials

<table>
<thead>
<tr>
<th>Material Record</th>
<th>Data in Record</th>
<th>Child Catalog</th>
</tr>
</thead>
<tbody>
<tr>
<td>aluminums</td>
<td>0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10</td>
<td>aluminium catalog</td>
</tr>
<tr>
<td>coppers</td>
<td>0, 1, 2, 3, 4, 5, 6, 7, 8, 10</td>
<td>none</td>
</tr>
<tr>
<td>leads</td>
<td>0, 1, 2, 4, 5, 6, 7</td>
<td>none</td>
</tr>
<tr>
<td>magnesiums</td>
<td>0, 1, 2, 3, 4, 5, 6, 7, 8, 9</td>
<td>none</td>
</tr>
<tr>
<td>molybdenums</td>
<td>0, 1, 4, 5, 6</td>
<td>none</td>
</tr>
<tr>
<td>nickels</td>
<td>0, 1, 3, 4, 8, 9</td>
<td>none</td>
</tr>
<tr>
<td>steels</td>
<td>0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10</td>
<td>none</td>
</tr>
<tr>
<td>tins</td>
<td>0, 1, 2, 4</td>
<td>none</td>
</tr>
<tr>
<td>titaniums</td>
<td>0, 1, 2, 3, 4, 5, 6, 7, 8, 9</td>
<td>none</td>
</tr>
<tr>
<td>tungstens</td>
<td>0, 1, 3, 4, 5, 6</td>
<td>none</td>
</tr>
<tr>
<td>zines</td>
<td>0, 1, 2, 3, 4, 5, 6, 7, 8</td>
<td>none</td>
</tr>
</tbody>
</table>

### Catalog: Porous Ceramic

**Parent Record:** porous ceramics  
**Parent Record in Catalog:** Engineering Materials

<table>
<thead>
<tr>
<th>Material Record</th>
<th>Data in Record</th>
<th>Child Catalog</th>
</tr>
</thead>
<tbody>
<tr>
<td>brick</td>
<td>2, 3, 4, 6, 7, 8, 9</td>
<td>none</td>
</tr>
<tr>
<td>cement</td>
<td>0, 1, 2, 3, 4, 6, 7, 8</td>
<td>none</td>
</tr>
<tr>
<td>concrete</td>
<td>0, 1, 2, 3, 4, 5, 6, 7, 8</td>
<td>none</td>
</tr>
<tr>
<td>porcelain</td>
<td>6, 7, 8</td>
<td>none</td>
</tr>
<tr>
<td>pottery</td>
<td>0, 1, 5, 7</td>
<td>none</td>
</tr>
<tr>
<td>rocks</td>
<td>0, 1, 2, 3, 4, 6, 7, 8</td>
<td>none</td>
</tr>
</tbody>
</table>

### Catalog: Ceramics

**Parent Record:** engineering ceramics  
**Parent Record in Catalog:** Engineering Materials

<table>
<thead>
<tr>
<th>Material Record</th>
<th>Data in Record</th>
<th>Child Catalog</th>
</tr>
</thead>
<tbody>
<tr>
<td>alumina</td>
<td>0, 1, 3, 4, 5, 6, 7, 8, 9, 10</td>
<td>none</td>
</tr>
<tr>
<td>diamond</td>
<td>0, 1, 3, 5, 6, 7, 8, 10</td>
<td>none</td>
</tr>
<tr>
<td>sialons</td>
<td>0, 1, 3, 6, 7, 8, 10</td>
<td>none</td>
</tr>
<tr>
<td>silicon carbide</td>
<td>0, 1, 3, 4, 5, 6, 7, 8, 9, 10</td>
<td>none</td>
</tr>
<tr>
<td>silicon nitride</td>
<td>0, 1, 3, 4, 5, 6, 7, 8, 9, 10</td>
<td>none</td>
</tr>
<tr>
<td>zirconia</td>
<td>0, 1, 3, 4, 6, 7, 8, 9</td>
<td>none</td>
</tr>
</tbody>
</table>
**Catalog: Aluminum**

**Parent Record:** aluminums

**Parent Record in Catalog:** Engineering Alloys

<table>
<thead>
<tr>
<th>Material Record</th>
<th>Data in Record</th>
<th>Child Catalog</th>
</tr>
</thead>
<tbody>
<tr>
<td>pure (1XXX)</td>
<td>0, 1, 3, 4</td>
<td>none</td>
</tr>
<tr>
<td>medium strength</td>
<td>0, 1, 3, 4</td>
<td>none</td>
</tr>
<tr>
<td>(3XXX, 5XXX)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>high strength (7XXX)</td>
<td>0, 1, 3, 4</td>
<td>none</td>
</tr>
</tbody>
</table>
Appendix E: Output for the catalog search example

Program Output

starting optimisation!
number of generations 120

catalog index: member 1 index for catalog: stocks
chosen record is: 2.5 inch nominal
variable data used are: outside diameter, diameter ratio

catalog index: member 2 index for catalog: stocks
chosen record is: 1/2 inch nominal
variable data used are: outside diameter, diameter ratio

printing delta distribution: s
data source: none
comments: none
scale: mm
delta location at: x = 1132.942383

printing delta distribution: stress safety factor1
data source: none
comments: none
scale: ratio
delta location at: x = 14.295868
objective 0 value is 1.000000

printing delta distribution: stress safety factor2
data source: none
comments: none
scale: ratio
delta location at: x = 2.000716
objective 1 value is 1.000000

printing delta distribution: buckling safety factor1
data source: none
comments: none
scale: ratio
delta location at: x = 3.047330
objective 2 value is 1.000000

printing delta distribution: cost
data source: none
comments: none
scale: ratio
delta location at: x = 146.785295
objective 3 value is 0.612859

overall solution value is 0.612859

done!
Appendix F: Output for the truss material selection example

Program Output

starting optimisation!
number of generations 126

catalog index: material selected for Catalog: engineering alloys
chosen record is: aluminum
variable data used are: density, Young's modulus, Cost/Unit volume, yield strength

catalog selector: member 1: selected catalog is pipes
catalog index: member 1 index for catalog: pipes
chosen record is: 3 inch nominal
variable data used are: pipe outside diameter, pipe diameter ratio

catalog selector: member 2: selected catalog is rods
catalog index: member 2 index for catalog: rods
chosen record is: 3/4 inch nominal
variable data used are: rod outside diameter, rod diameter ratio

printing delta distribution: s
data source: none
comments: none
scale: mm
delta location at: x = 1662.756592

printing linear distribution: stress safety factor1
data source: none
comments: none
scale: ratio
y = 0.000000x + 0.500000, 0.119777 < x < 48.118751

objective 0 value is 0.966036

printing linear distribution: stress safety factor2
data source: none
comments: none
scale: ratio
y = 0.000000x + 0.500000, 0.026158 < x < 10.50852

objective 1 value is 0.835554

printing linear distribution: buckling safety factor1
data source: none
comments: none
scale: ratio
y = 0.000000x + 0.500000, 2.550641 < x < 3.950206

objective 2 value is 1.000000

printing linear distribution: truss cost
data source: none
comments: none
scale: monetary units
y = 0.000000x + 0.500000, 64.210325 < x < 174.107436

objective 3 value is 0.723364

printing linear distribution: truss mass
data source: none
comments: none
scale: kg
y = 0.000000x + 0.500000, 20.043436 < x < 21.865566

objective 4 value is 0.726137

overall solution value is 0.423978

done!
Appendix G: Output for the hierarchical material search example

Program Output

starting optimisation!
number of generations 104

catalog index: material selected for Catalog: engineering materials
chosen record is: engineering alloys
variable data used are: density, Young's modulus, Cost/Unit volume,
                     yield strength

catalog index: member 1 index for catalog: pipes
chosen record is: 3.5 inch nominal
variable data used are: pipe outside diameter, pipe diameter ratio

catalog index: member 2 index for catalog: rods
chosen record is: 3/8 inch nominal
variable data used are: rod outside diameter, rod diameter ratio

printing delta distribution: s
data source: none
comments: none
scale: mm
delta location at: x = 2304.007812

printing linear distribution: stress safety factor1
data source: none
comments: none
scale: ratio
y = 0.000000x + 0.500000, 2.097391 < x < 303.988935

objective 0 value is 1.000000

printing linear distribution: stress safety factor2
data source: none
comments: none
scale: ratio
y = 0.000000x + 0.500000, 0.095289 < x < 13.810838

objective 1 value is 0.879355

printing linear distribution: buckling safety factor1
data source: none
comments: none
scale: ratio
y = 0.000000x + 0.500000, 0.949753 < x < 3.139856

objective 2 value is 0.406308

printing linear distribution: truss cost
data source: none
comments: none
scale: monetary units
y = 0.000000x + 0.500000, 28.564505 < x < 13547.736784
objective 3 value is 0.010832
printing linear distribution: truss mass
data source: none
comments: none
scale: kg
y = 0.000000x + 0.500000, 14.349206 < x < 171.773338
objective 4 value is 0.100920
overall solution value is 0.000391

Searching another level deeper!
number of generations 130

catalog index: material selected for Catalog: engineering alloys
chosen record is: aluminum
variable data used are: density, Young's modulus, Cost/Unit volume,
yield strength

catalog index: member 1 index for catalog: pipes
chosen record is: 3 inch nominal
variable data used are: pipe outside diameter, pipe diameter ratio

catalog index: member 2 index for catalog: rods
chosen record is: 3/4 inch nominal
variable data used are: rod outside diameter, rod diameter ratio

printing delta distribution: s
data source: none
comments: none
scale: mm
delta location at: x = 1656.891479

printing linear distribution: stress safety factor1
data source: none
comments: none
scale: ratio
y = 0.000000x + 0.500000, 0.119416 < x < 47.973914
objective 0 value is 0.965926

printing linear distribution: stress safety factor2
data source: none
comments: none
scale: ratio
y = 0.000000x + 0.500000, 0.026080 < x < 10.477221
objective 1 value is 0.835050

printing linear distribution: buckling safety factor1
data source: none
comments: none
scale: ratio
y = 0.000000x + 0.500000, 2.545605 < x < 3.952387
objective 2 value is 1.000000
printing linear distribution: truss cost
data source: none
comments: none
scale: monetary units
\[ y = 0.000000x + 0.500000, \quad 64.177004 < x < 174.017087 \]
objective 3 value is 0.723612

printing linear distribution: truss mass
data source: none
comments: none
scale: kg
\[ y = 0.000000x + 0.500000, \quad 20.033034 < x < 21.854219 \]
objective 4 value is 0.726409

overall solution value is 0.423978

Searching another level deeper!
number of generations 104

catalog index: material selected for Catalog: aluminum alloys
chosen record is: strong al (7XXX series)
variable data used are: density, Young's modulus, Cost/Unit volume, yield strength

catalog index: member 1 index for catalog: pipes
chosen record is: 3 inch nominal
variable data used are: pipe outside diameter, pipe diameter ratio

catalog index: member 2 index for catalog: rods
chosen record is: 3/4 inch nominal
variable data used are: rod outside diameter, rod diameter ratio

printing delta distribution: s
data source: none
comments: none
scale: mm
delta location at: \( x = 1568.915039 \)

printing linear distribution: stress safety factor1
data source: none
comments: none
scale: ratio
\[ y = 0.000000x + 0.500000, \quad 2.787922 < x < 45.774453 \]
objective 0 value is 1.000000

printing linear distribution: stress safety factor2
data source: none
comments: none
scale: ratio
\[ y = 0.000000x + 0.500000, \quad 0.608866 < x < 9.996871 \]
objective 1 value is 0.878448
printing linear distribution: buckling safety factor
data source: none
comments: none
scale: ratio
y = 0.000000x + 0.500000, 3.018646 < x < 3.829153

objective 2 value is 1.000000

printing linear distribution: truss cost
data source: none
comments: none
scale: monetary units
y = 0.000000x + 0.500000, 102.835328 < x < 137.113771

objective 3 value is 0.0.720102

printing linear distribution: truss mass
data source: none
comments: none
scale: kg
y = 0.000000x + 0.500000, 19.880811 < x < 21.688157

objective 4 value is 0.0.730388

overall solution value is 0.462023

done!
References


Schmaus, T. and M. Kahmeyer Design for Disassembly—challenge of the future. Fraunhofer Institute, Germany.


Young, J. (1992). Disassembly Software Demonstration. 2 Holley Road, Wakefield, RI 02879, Boothroyd and Dewhurst Inc.


