INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Supplying Concurrent Engineering Information to the Designer: The Conceptual Design Information Server

by

William Holmes Wood III

B.S. (Duke University) 1985
M.S. (Duke University) 1989

A dissertation submitted in partial satisfaction of the
requirements for the degree of

Doctor of Philosophy

in

Engineering - Mechanical Engineering

in the

GRADUATE DIVISION

of the

UNIVERSITY of CALIFORNIA, BERKELEY

Committee in charge:

Professor Alice M. Agogino, Chair

Professor Homayoon Kazerooni

Professor Randy H. Katz

1996

This dissertation of William Holmes Wood III is approved:

Chair — Date 4 March 96

Date 28. Feb, 96

Date 26 Feb 96

University of California, Berkeley

1996

Supplying Concurrent Engineering Information to the Designer:
The Conceptual Design Information Server

by

William Holmes Wood III

to Marianne

Table of Contents

List of Figures

List of Tables

List of Acronyms

Acronym	Expansion
ADD	Augmenting Design Documentation
AI	Artificial Intelligence
ARPA	Advanced Research Projects Agency
CAD	Computer Aided Design
CADET	Case-based Design Tool
CADSYN	Case-Based Design Synthesizer
CBR	Case-Based Reasoning
CBDA	Case-Based Design Aid
CDIS	Conceptual Design Infomration Server
DDIS	Design Desision Information System
DFA	Design for Assembly
DFAA	Design for Automated Assembly
DFX	Design for *X*
DPF	Design Pattern Formulator
DME	Device Modeling Environment
EFD	Empirical Formula Discovery
EVPI	Expected Value of Perfoect Information
FABEL	(translated from german)
HoQ	House of Quality
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IBIS	Issue-Based Information System
IDA	Institute for Defense Analysis
IDeAL	Integrated Design by Analogy and Learning
IRTD	Intelligent Real-Time Design

Acronym	Expansion
KIF	Knowledge Interchange Format
MFK	Multimedia Forum Kiosk
MIME	Multipurpose Internet Mail Extension
NLP	Natural Language Processing
PDES	Product Data Exchange Specification
PENS	Personal Electronic Notes System
QFD	Quality Functional Deployment
SBF	Structure-Behavior-Function
SOPNN	Self Organizing Probabilistic Neurla Network
SQL	Structured Query Language
STRUPLE	Structural Planning Expert
TREC	Text Retrieval Cnference
URL	Uniform Resource Locator
WebCAMILE	Web Collaborative and Multimedia Interactive Environment
WAIS	Wide Area Information Service
WWW	World Wide Web

Acknowledgements

I want to thank the following people, who have made this work possible:

Alice Agogino, chair of my dissertation committee and advisor, for her support and encouragement throughout my tenure at Berkeley.

Professors Randy Katz and H. Kazerooni, dissertation committee members whose interest in my work and direction in its evolution is greatly appreciated.

Members of the Case Studies Team: Sherry, Jay, Chuck, Ian, Anil, Armineh for their efforts in developing individual cases and for providing inspiration that a greater whole could be synthesiszed from them.

Members of the Concept Database Team: Steve, Anil, Bala, and Andy for valuable discussion and support for ideas which continued to evolve into the CDIS.

Fellow Kiosk Developers: Sherry, Chris, and Ben whose encouragement, enthusiasm, and knowledge helped shape the design discussion server.

Members of the BEST Lab: In addition to those mentioned above, I want to thank Kai, Satnam, and Robert for continual moral and intellectual support.

My Family: Foremost I want to thank my wife whose support and love have made this possible. I also want to thank the rest of my family for thier support throughout the years.

Chapter 1

Introduction

The Conceptual Design Information Server (CDIS) is a computer aided design tool targeted at providing the kind of information that can improve decisions made early on in the design process. Conceptual design is differentiated from routine or parametric design by both the nature of design activity and the impact this activity has on the quality of the final design. In conceptual design a loosely structured set of design requirements is iteratively transformed into functional requirements, design constraints, and performance criteria. Often this is done through a process of proposing and evaluating candidate solutions, reducing design abstraction in response to the decision making needs of the process. Because of this interplay of problem definition and solution, decisions made during conceptual design have a large impact on the subset of solutions considered during the later design phases where search is ordinarily associated with design optimization over mathematically formalized problems. This state of affairs is reflected in studies of large scale projects by National Research Council [1991] and the Institute for Defense Analysis [1988] which estimate that 80-90% of the life cycle design costs (including fabrication, construction, energy, maintenance and disposal) are determined in the first 10-20% of the design phase – conceptual design. Ulrich and Pearson [1993], among others, question the scale of this figure; clearly one cannot reduce the cost of a design by 80% simply through clever conceptualization. Nonetheless, there is general support for the notion that opportunities for overall cost savings are predominantly found within conceptual design.

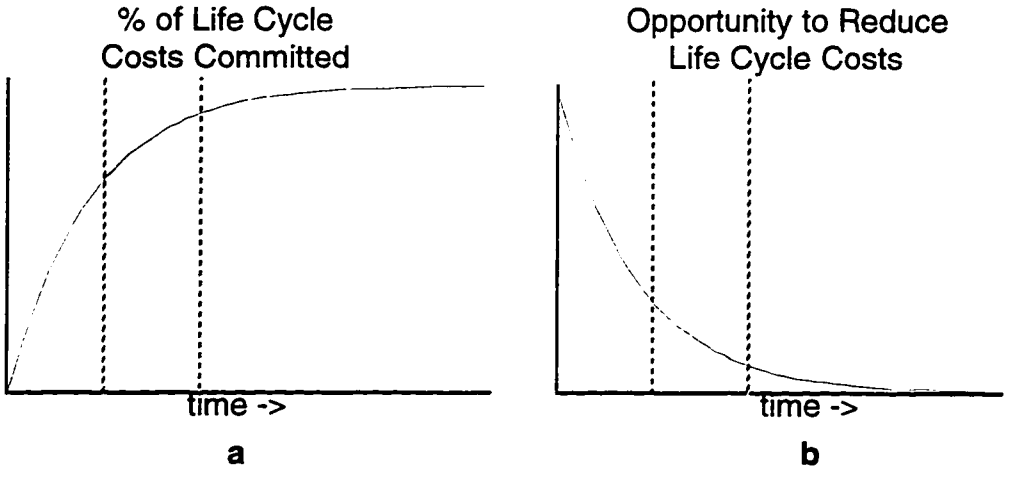


Figure 1.1 The importance of conceptual design is demonstrated by its impact on overall life cycle costs. Graph **a** depicts the 'percentage of life cycle costs determined' by [IDA, 1988], graph **b** reflects the 'opportunity for reducing life cycle costs' during the design time line.

Figure 1.1 demonstrates the transformation from 'determining life cycle cost' to 'opportunities to reduce life cycle cost' that unify the argument that conceptual design tools are key components enhancing overall engineering effectiveness.

Concurrent engineering [Rosenblatt and Watson, 1991] – a popular strategy which redefines the 'customer' of a design to include not only the end user, but the marketing, manufacturing, distribution, and retirement (update/reuse/recycle/disposal) organizations – is the preeminent strategy for improving the quality of conceptual design. In a study of a major automobile manufacturer, Salzberg and Watkins [1991] conclude that a lack of information about the life cycle impact of conceptual design decision making is the primary factor reducing final design effectiveness. Substantial roadblocks exist in implementing concurrent engineering ideas. By introducing so many simultaneous concerns into the conceptual design process, the concurrent engineering philosophy severely taxes the ability of a designer to gather and manage all of the information necessary for making good conceptual design decisions. The CDIS targets this issue to aid the designer in obtaining, managing, and communicating these life cycle design concerns. To provide the maximum impact on the overall design process, the CDIS is focused specifically on how this

information can be integrated into the earliest possible stages of design.

1.1 Information Demands of Concurrent Engineering

Concurrent engineering concerns are typically integrated into the design decision making process by combining representations of the constraints and desires of each "customer" along the life cycle of the design[1] [Dieter, 1991] into a single objective for the design. Multiobjective optimization techniques within the concurrent engineering research community can operate on a single abstraction level for representing life cycle issues – mathematical equations set in the form of a nonlinear programming problem. Neither the demands of 'internal' customers like design, manufacturing, distribution, and disposal nor those of the traditional customer – the consumer – are based on mathematical representations. 'House of Quality' techniques [Hauser and Clausing, 1988] like quality functional deployment (QFD), a mechanism for representing the informal demands of customers and mapping them into a (non-numerical) weighted set of (non mathematic) objectives. This is done so that the designer might be able to more fully understand the objectives and tradeoffs embodied in the design at hand. However, the mappings derived using this technique are informal, highly context dependent ones intended to be a source of guidelines for design, not as shorthand for an optimization problem.

It is necessary to give a definition of 'informality' in terms of design representations for conceptual design. Perhaps the most appropriate graphical description of the changing abstraction level in the engineering design process can be derived from that offered by Hubka [1988], shown in Figure 1.2. Hubka's design process model shows a trajectory in cylindrical coordinates which proceeds vertically along an axis from a base point of abstract conceptual design downward toward concrete final design. A slice at each stage of design provides a snapshot of the degree to which the concerns of the various stakeholders has been addressed: the angle of rotation about the vertical describes the current design focus

[1] In this discussion, concurrent engineering and life cycle design are treated synonymously. From the standpoint of the current argument of increasing the number of participants in the formation of the design problem, they are the same. In more formal presentations, concurrent engineering refers to design process whereas life cycle design refers to design issues.

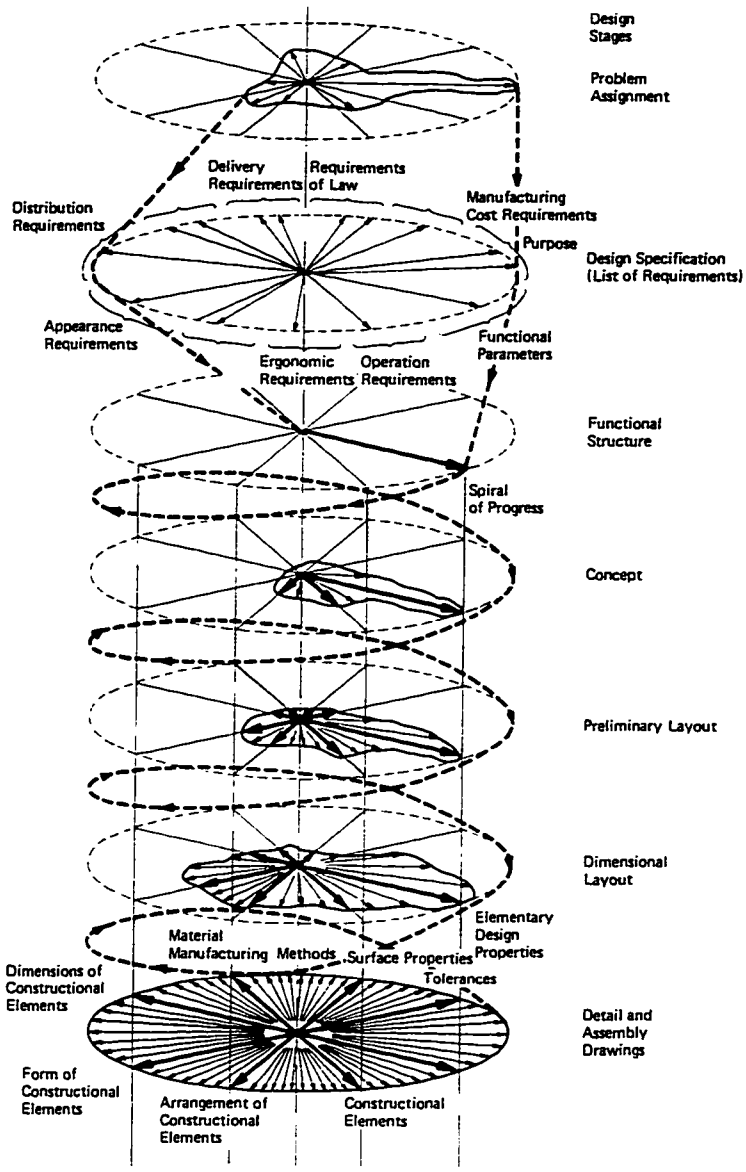


Figure 1.2 Representation from Hubka [1988] of the progression of design abstraction throughout the design process. The spiral trajectory shows the design path through various activities. Note that early design stages are projected from surfaces to vectors representing narrow concerns later in the process. (Reprinted with permission)

(one of the life cycle "customers"), the length of the vector is the level of detail in which that customer's concerns are represented in the current design decision context. In early phases of design, manufacturing concerns are difficult to integrate because it is difficult to evaluate the manufacturability of an abstract functional block, so the trajectory passes close

to the origin. It is more appropriate at this stage to explore the customer's design preferences in order to begin the embodiment process which can then be guided by other concerns. Thus, early on the trajectory loops outward toward the user. The key to effective concurrent engineering is to integrate the concerns of each life cycle customer as early as is appropriate in the design process. Because of the variation in design ambiguity throughout the design process, this requires operating over uncertain problem statements and evaluation metrics toward defining the design problem as well as possible solutions to it. With increased pressure on time-to-market, Hubka's representation must be compressed in the vertical axis and many concerns considered in parallel. Concurrent engineering produces an expansion of scope at every stage in the design process, particularly conceptual design where previously strong customer emphasis is augmented by other life cycle concerns.

The Intelligent Real-Time Design (IRTD) method [Bradley, 1993; Bradley and Agogino, 1991a-c] helps designers focus their information gathering efforts toward reducing uncertainty in areas of the design that have the greatest impact on the current design decision. For our present discussion we will treat uncertainty and ambiguity as synonyms although in later stages of design, ambiguity is eliminated while some sources of uncertainty remain. IRTD does not distinguish among the sources of uncertainty – design parameters, evaluation models, and customer preferences are all analyzed to determine where design attention should be focused and at what level of ambiguity. Rather, IRTD represents a normative methodology for actively shaping the transformation of the design space over the course of conceptual design. Because it is an extension of typical stochastic nonlinear programming, IRTD is limited in application to mathematical representations of design evaluation functions. The results of IRTD are a set of expected costs that each uncertain value is imposing on the current design decision. The designer is charged with determining whether the impact of updating preference functions, design models, or design parameters offsets the cost of information gathering. In this way, the design process becomes one of expressing a design model, evaluating design options using this model, and determining how well the design space is parsed, and which new information might

produce a greater separation among viable candidate designs.

1.2 Information Sources for Concurrent Engineering

A normative design method has thus been proposed for concurrent engineering conceptual design which matches the nature of information within the design process. IRTD recognizes the need to embrace ambiguity in all aspects of the design problem formulation. It embodies an information value strategy for operating on an ambiguous problem which lends focus to the information gathering process of conceptual design. The question for the user remains how to act on the recommendations of a system which assigns value to information but does not identify potential sources for it. Even to provide this limited service the methodology depends on a mathematical formulation of the design problem – it does not support the important tasks of proposing a relevant design model or communicating with customers. These are aspects of conceptual design which are perhaps most greatly influenced by the experience of the designer. Many have cast design primarily

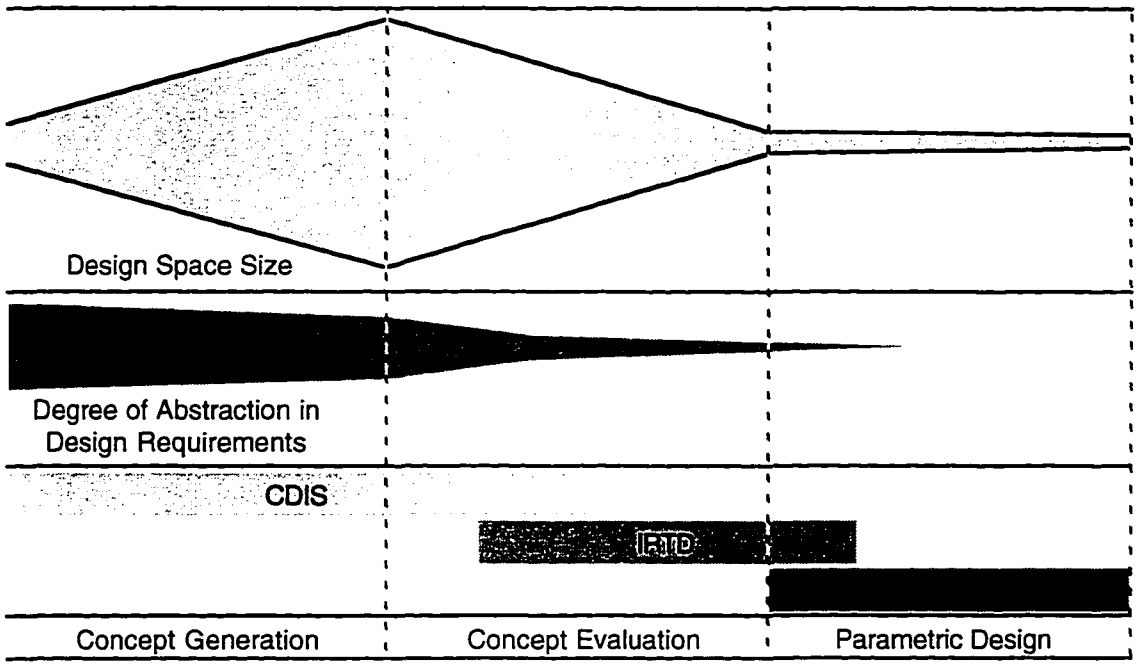


Figure 1.3 Abstraction level of design requirements plotted within a representation of the size of the design space from early conceptual design through parametric design. Labeled by potential design tool.

as redesign; it makes sense to try to understand how cases of previous design instances might be used as a source for both initial design direction and for information gathering once the current problem is defined well enough for IRTD to become an effective tool. Figure 1.3 demonstrates the progression of design from the abstract stages of conceptual design, expansion of the design space during concept generation, reduction of the design space through concept evaluation, and incremental design space reduction during parametric design. It is the latter two stages where IRTD methods contribute to concept evaluation, helping the designer to manipulate the state of design information until final mathematical models can be applied to optimization methods. The goal of the CDIS is to expand the use of IRTD's decision theoretic methods to earlier stages of the design process, including the whole of concept evaluation and the latter stages of concept generation. Still earlier in the design process, the abstraction of information is such that it is implausible to apply IRTD. Here, the experience gained from past designs enables the designer to operate over abstract stakeholder requirements toward producing evaluation metrics used to direct search in the design space.

The idea of operating over previous design instances is not original to this work. Case-based reasoning has been proposed by many researchers [Slade, 1991; Kolodner, 1995] as general semantic model for the design process. Cases can influence conceptual design in varied ways: some are used as frameworks for accomplishing the design process, pointing out high level concerns like the formation of design teams and definition of responsibilities among team members; some are used to help create plausible constraint and evaluation formulations by illustrating those used in successful projects; other cases provide more detailed information directly applicable to the current design, suggesting design alternatives and raising detailed design issues. The information base made available through the CDIS contains design cases in three distinct formats: case studies of industry 'best practices' of design, design discussion threads, and a database of design application cases which include analytical design models. The combination of these three elements with an effective method for indexing them greatly extends the scope of computerized case based reasoning systems.

1.3 The Conceptual Design Information Server

We have defined three distinct sets of information, all of which must be accessible through the CDIS. The first, case studies of concurrent engineering 'best practices' provides general information about how concurrent engineering is successfully integrated into the engineering process. This is done through a set of media-rich case studies of design projects from varied industrial settings which illustrate the improvement in design effectiveness through the implementation of concurrent engineering practices. Implicit in these case studies is a set of models for implementing concurrent engineering into the design process reflecting the organizational constraints of each company. Explicit are concrete examples of how designs were improved by focusing on aspects of the life cycle appropriate to the goals of the organization. These case studies are implemented as highly structured, hand crafted hypermedia documents, each indexed in a variety of ways on a local basis. They are not complete descriptions of designs, but contextualized presentations of the lessons exemplified by the application of concurrent engineering techniques.

At a less abstract level of representation, design discussions threads provide a more complete trace of a design process. Implemented again as a set of locally indexed hypermedia documents, the CDIS follows the IBIS [Kunz and Rittel, 1977] model for design discussion. Argumentation is supported among multiple users through a mechanism much like computer bulletin boards. The CDIS implementation adds value to these discussion models through a set of indexing and communication mechanisms which both enhance discussion and access to it. The goal is to capture both strategy and solution from these design discussions as highly contextualized cases of conceptual design in an expanded version of the typical design notebook. Because concurrent engineering demands group efforts, a collective discussion becomes a natural way of capturing and representing the design process.

The CDIS supports more concrete reasoning through the implementation of a conceptual design database system. Again implemented within a hypermedia environment, the concept

database [Bradley et al., 1994] provides access to mechatronic components, mathematical relations germane to these components, and component modeling/selection templates for evaluating design alternatives. Indexing within the concept database takes advantage of the structure of components, relations, and templates as well as abstract descriptions of their functions.

The choice of knowledge representation within the CDIS is based on expression and indexing efficiency. The knowledge representation must be expressive enough to capture all of the pertinent design information while limited enough to provide a means of applying the knowledge in a context-free framework for design automation. The CDIS recognizes two languages: free text from hypermedia documents and mathematical relationships within the concept database. The third main form of engineering expression – images – can be expressed within CDIS documents (free text descriptions and keywords associated with images and movies are indexed in some experiments but the method is not automated at this point). Because interfaces to indexing within the system are accomplished in the media of all documents within the CDIS, these can easily be added to the system once reliable indexing methods are developed. For now, we have augmented the mathematical-symbolic representations typical of design reasoning systems with the least biased language that is practical – free text. The ability to index this language to provide effective access to concurrent design information is the major research focus within the CDIS.

1.4 CDIS Implementation

The CDIS provides links among the three main applications along with abstract and structured indexes for searching within them through an implementation focused on network standard technology. All hypermedia documents and applications are distributed through a World-Wide-Web (WWW) server. The document server relies on a relational database for storage and structured indexing which provides a hardened Structure Query Language (SQL) interface for information retrieval. In addition, application specific views on this database can provide interfaces to other case-based design applications. For

indexing hypermedia documents, a Wide Area Information Service (WAIS) server is used. The integration of these along with example design tools is shown in Figure 1.3.

1.5 Road Map

Having defined the goal of the CDIS as a computer tool to support concurrent engineering principles in the conceptual phase of design the CDIS, a road map for detailed discussion is now provided:

Chapter 2 introduces computerized case-based reasoning. Case-based design systems are reviewed to gain insight into the representation models used, the abstraction level of this representation, and how cases are used to automate design. Representation of knowledge used to operate on the cases is also discussed and some conclusions drawn for extending case-based design toward concurrent engineering conceptual design.

Chapter 3 focuses on the representational demands of conceptual design through a brief review of findings from design protocol analyses. In particular, the manipulation of abstraction level throughout conceptual design is discussed within the framework of IRTD and other multiple abstraction methodologies. Case-based reasoning within conceptual design is cast as a problem separate from the typical case-based design systems of Chapter 2; a review of results in systems with goals similar to that of the CDIS is presented.

Chapter 4 defines the CDIS architecture and describes the technology in which it is implemented: the World-Wide-Web (WWW), Wide Area Information Service (WAIS), and a relational database. The specific functionality of each component of the CDIS is described and related to the type of design information that it represents or indexes. A general flow of operation within the system is presented.

Chapter 5 introduces the three demonstration applications constructed within the CDIS architecture. Implementation of the Concurrent Engineering Case Studies and the Design Discussion Server are presented along with typical information content. In addition,

navigation, use, and connectivity among them are presented.

Chapter 6 describes structured indexing within the CDIS, introducing the implementation of the Concept Database for storing cases of component selection methods in combination with a component catalog. A formal method for representing and manipulating design abstraction through the use of IRTD is presented. Using abstraction level indexing and the component database, fluid design case base is created. Decision making using IRTD is demonstrated as an effective tool for the conscious manipulation of abstraction level during conceptual design.

Chapter 7 extends the abstraction taxonomy concept to include concurrent engineering issue and functional abstraction. In combination, the three taxonomies are used to enhance the search effectiveness of WAIS toward providing an automatic index for design information and documentation that greatly simplifies access to design experience. Background for the standard WAIS searching system is given along with a discussion of applications of similar techniques in information retrieval and natural language contextualization. The design of the CDIS global indexing system is presented along with results from experiments to determine effective indexing strategies among various design resources. This overall discussion of the success of the indexing techniques is supplemented with an analysis of specific tuning aspects toward improving or at least manipulating system performance.

Finally, Chapter 8 provides a discussion of the CDIS, the conclusions garnered from the project thus far, and the impetus for extending the research program. As a final note, due to the numerous acronyms used throughout this work, a brief translation table can be found on page ix.

Chapter 2

Case-Based Design Systems

Direct case experience plays a major role during the process of design. A designer highly experienced in one industry or technology brings this experience to bear on new problems, gradually extending the state of his art in response to ever increasing performance standards and constantly changing technology. Such a method is most effective when the designer need only extend his design activities incrementally; by definition this is the case in routine or parametric design. For this type of design the decisions and evaluation procedures are well defined, as is the informational foundation underlying them. In order to meet more stringent requirements, some extrapolation beyond the limits of experience is allowed. Case-based design exploits this general reasoning method. This chapter defines the standard case-based reasoning (CBR) paradigm and illustrates successful applications of CBR to design.

2.1 Case-Based Reasoning: An Artificial Intelligence Paradigm

Case-based reasoning has been identified as a generic knowledge processing paradigm within the artificial intelligence (AI) community [Slade, 1991]. Schematized in Figure 2.1, the general process is one of retrieving solutions to similar problems, choosing the most promising, patching it to fit the current context, and iterating or backtracking to resolve open issues. It is apparent from this formulation that CBR is closely related to machine

learning, specifically transformational analogy as defined by Carbonell [1981]: direct experience is applied to new instances by mapping solutions from prior 'similar' problems to the new instance. The way in which knowledge is represented within the system along with the state of knowledge about the domain determines how the CBR paradigm is applied. Three general categories of CBR have developed:

Analogy by Similarity: In an environment where modeling the relationships within the design representation is difficult (many times the motivation for using CBR is a weak or nonexistent theoretical model of the domain), CBR applications follow a form defined by Russell [1988] as analogy by similarity: no attempt is made to understand the attribute space for a case, retrieved similar cases are merged through simple domain independent rules (e.g. interpolation over a numerical representation of attributes).

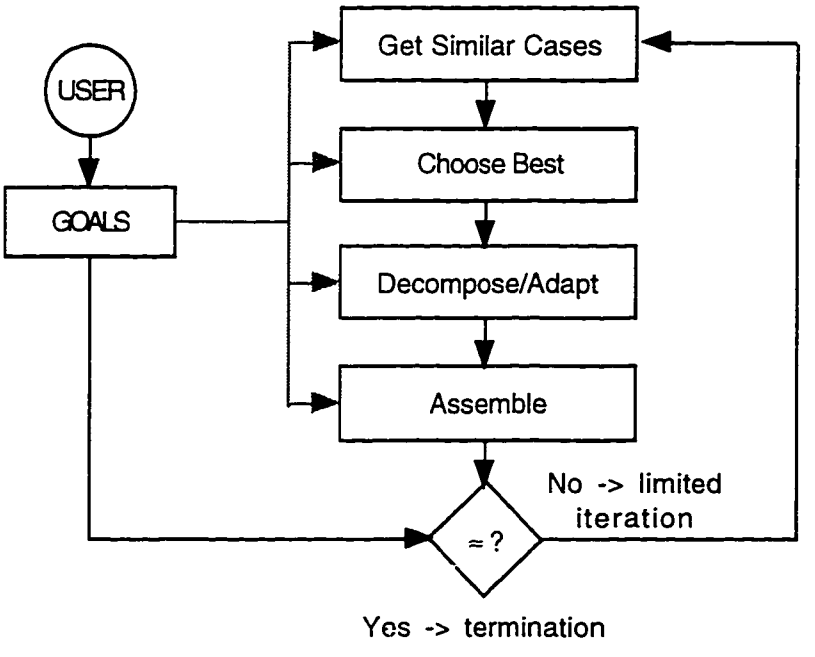


Figure 2.1 Typical flow of information and control in a case-based reasoning system. The user specifies problem aspects – goals, partial solutions; cases are retrieved, the closest candidate(s) selected; these candidates are decomposed and adapted to current goals or partial solutions; solutions are reassembled. Iteration allows the further application of case information.

Transformational Analogy: When the representation selected for the domain encodes the bulk of the modeling knowledge, the system appears to operate in this way as well although the representation itself might ensure some basic level of feasibility or compatibility of results. However, because the representation implicitly expresses adaptation rules, the reasoning is more along the lines of transformational analogy defined by Carbonell. Similarities among case goals and constraints suggest similar cases which are operated upon by adaptation rules that guarantee that the result of reasoning is feasible.

Model-Based Transformational Analogy: When a general theory for the domain is available and expressed within the CBR system, the derivation of a case can be used to help drive reasoning in new cases. Typically, domain knowledge is expressed in a logical formalism, derivation encoded as the chain of logical progressions that achieve various subgoals of the system. In general, cases are decomposed into knowledge 'subsystems' – partial solutions – which can then be combined, in response to new design requirements, into an overall solution.

2.2 Case-Based Applications in Design

Case-based design systems have been built around the three paradigms described above. The following is a survey of these systems which focuses on the availability of domain knowledge, the representation chosen for the system and the type of design being done. Organization for the survey is based on the general level of domain knowledge encoded within the system – from no background knowledge to representation encoded knowledge to representation expressed knowledge.

2.2.1 CBR in the Absence of Models

The absence of domain models within a CBR system deprives it of specific decomposition or adaptation rules. The focus of the system changes from retrieve / decompose / adapt to retrieve / adapt with very simple adaptation rules. The following examples illustrate this

method:

Jamalabad and Langrana [1993] describe a system which induces process parameter designs for complicated manufacturing equipment like extrusion ovens and metal extruders. The case library consists of numerical representations of operating parameters and performance measures for the system. This library is probed in response to new design specifications. Cases whose specification values are numerically close (in Euclidian space) are retrieved; interpolation of their process parameters results in parameters assigned to the new design. Testing this new design on the actual equipment provides more experience to add to the case base. Iteration is done when the design does not match the specifications. The use of direct case experience in this way has dramatically decreased the number of iterations needed to specify the operating parameters of the system. When compared to general system models derived from the same case library, local interpolation proved to have better performance. Thus the adaptation rule, while not domain specific, was optimized for this domain. A similar representation scheme has been employed for weld process design [O'Conner et al., 1992].

In the domain of structural engineering, STRUPLE [Zhao and Maher, 1988] operates over a database of building features and corresponding structural designs stored with a mixed continuous and discrete encoding of data. In this case a modification of the nearest neighbor approach is used to retrieve relevant cases; distance along a representational dimension is weighted according to expert defined heuristics. This non-Euclidian similarity function allows seamless mixture of discrete and continuous variables. Instead of adapting case, STRUPLE presents similar design instances directly to the designer.

A later application called CONCEPTOR follows in the mold of STRUPLE but is applied to conceptual design of structural elements for bridges, Maher and Li [1994] offer some adaptation rules. Separate methods are employed for continuous and discrete representations: empirical formula discovery (EFD) for continuous valued case information and design pattern formulation (DPF) for discrete values. CONCEPTOR, performing

functions similar to STRUPLE, automatically forms prototypes from the design cases using distance metrics. From there, EFD applies regression analysis within each of these prototype classes to learn linear relationships (and quantify their strength) among continuous attributes. DPF creates probability distributions among qualitative (discrete) attributes. Case-based reasoning in the system proceeds by identifying prototypes from design specifications (retrieval), operating over these specifications to identify parameters to be inferred by the system (decomposition), and then using the relations learned by DPF and EFD to propose design solutions (adaptation and recombination).

These projects exemplify the need of a case based design system to support the use of case knowledge in a form which does not require domain knowledge. Due to the abstract nature of conceptual design formulations, many design activities must be carried out in the absence of strong technical models. Case based reasoning using induction in the absence of domain knowledge is used in the concept generation and evaluation stages to help reduce the design search space. In each case, the system uses experience to help drive the design process but does not attempt to adapt final solutions. Any solutions derived from these systems are thoroughly processed using mathematical engineering analysis models.

2.2.2 Implicit Theory Representation in CBR

CBR techniques are not limited to cases where domain knowledge is poor. Considerable work has been done in systems where case generalization is made through the use of 'deep knowledge', or general domain theories. This brand of CBR is distinguished by a set of domain independent models which map to domain specific cases. Indexing is based on the instantiations of this general theory in a specific domain. Adaptation involves decomposing cases into subsets that match retrieval criteria and synthesizing them into a whole according to the requirements of the user and with the direction of the domain theory. One subset of CBR which applies domain knowledge is that in which the knowledge is implicitly encoded in the design representation.

The first example of such a system is CADET [Navinchandra, 1988] where the general

domain theory is a synthesis of qualitative physics and the prescriptive morphological design methodology of Pahl and Beitz [1984]. At the shallow representational level, domains are separated into fluid, electronic, and translational or rotational mechanical systems where force or flow variables are defined. CADET's cases are stored as a set of qualitative relations between input and output variables from this predefined set – influence graphs. An influence graph is a directed arc from input to output, labeled as positive (negative) if increasing the input variable increases (decreases) the output variable. Design cases are stored as complete input / output influence graphs but are also decomposed into its subcomponents, basic engineering building blocks. Because case decomposition is precompiled and the domain is circumscribed by the representational language, decomposition and adaptation are eliminated from the CBR method. An example of a design case and decomposition within it is shown in Figure 2.2. Retrieval remains the lone residue of CBR and is performed through means-ends analysis strategy; influence graphs are used to transform both input and output states 'toward' each other using a small set (2) of case combination rules. Michelena and Sycara [1994a] discuss the limitations of using the simple (first order) influence graphs for a knowledge representation scheme and introduce representational and transformation rule changes to accommodate more specific input output relationship specification. Additional engineering models are applied to the results of behavioral synthesis to produce more complete geometric instantiations of design candidates [Michelena and Sycara, 1994b], reducing design abstraction to a level at which more concrete evaluation metrics can be applied.

In Schemebuilder, Chaplin et al.[1994] employ a well known representational scheme for their case based mechatronic design synthesis system: bond graphs [Paynter, 1961]. While the representation changes, the resulting system operates in much the same way as CADET: input/output relations, classified by domain (fluid, electrical, translational, or rotational) and type (flow or effort), are used to index into a set of predefined components. Both input and output are transformed in a means ends analysis strategy which naturally favors the use of preexisting collections of basic components(cases). Bond graphs are a natural representation language for energetic systems, encoding effort and flow conservation along

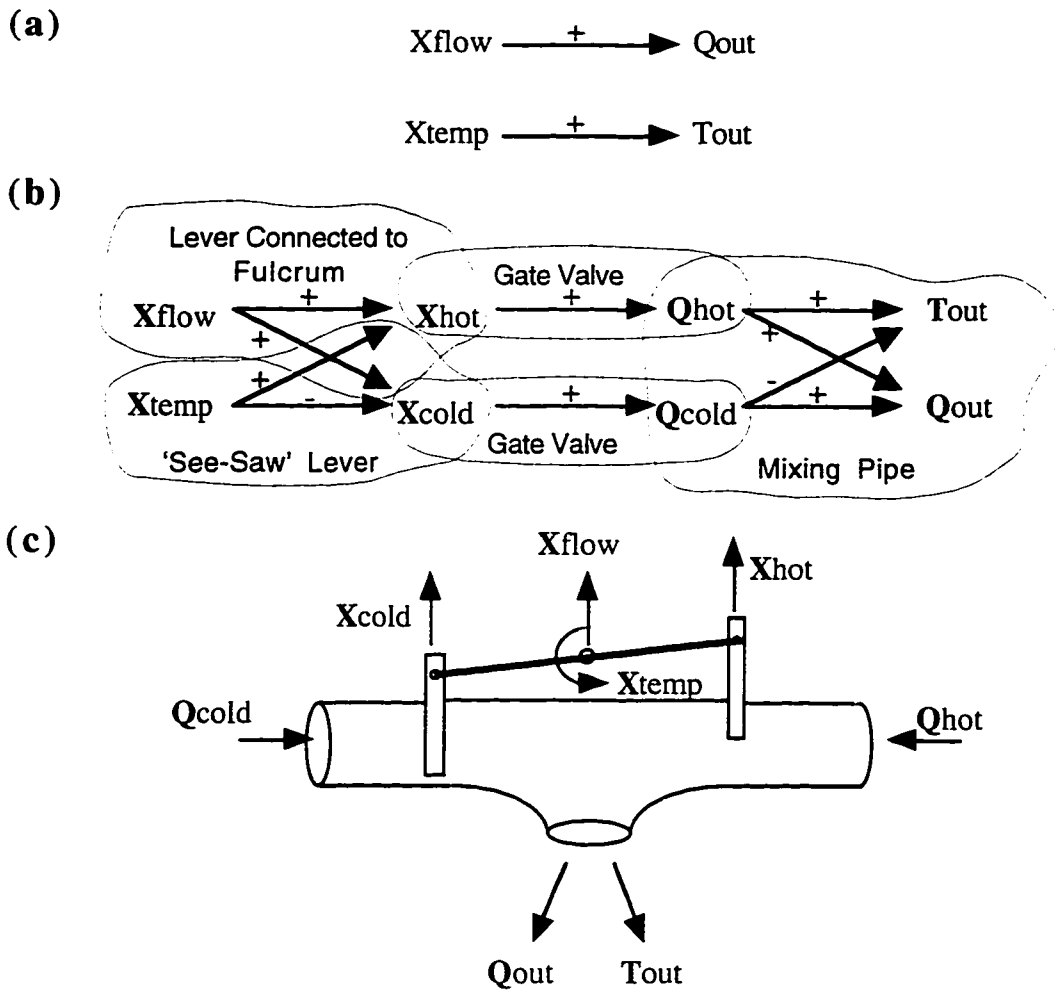


Figure 2.2 Behavioral representations in CADET [1988]: (a) a behavioral specification given for the system as influence graphs, (b) CADET system synthesis using known components (c) shows an embodiment of the system.

with conservation of energy – the ability to relate not only influence, but influence among time derivatives of components. In addition to the basic input /output relations described by bond graph class, components are stored at a second level of abstraction – mathematical behavioral models (specifically Matlab SIMULINK modules). Case-based reasoning thus provides adaptation of stored cases to general input/output relations. The user is responsible for adaptation based on more specific input/output relations.

These representations are not unique to the CBR systems examined here. Michelena and

Agogino [1993] have proposed monotonic influence diagrams as an abstract representation of a design model which can be easily manipulated to reveal promising strategies for reducing design search. Ulrich and Seering [1990] employed bond graphs in a design synthesis system which shares a great many ideas with Schemebuilder. Dixon et al. [1993] propose behavioral graphs as augmentation of bond graphs, providing representation of vector relationships in flow and effort variables (i.e. knowing the difference between pinion gears and bevel gears as transformation elements). Among these Schemebuilder is notable for representing design cases at multiple levels of abstraction, although the case-based module operates only on the more abstract of the two representational schemes.

2.2.3 Explicit Theory Representation in CBR

A third class of case-based design systems integrates heuristic with case-based reasoning for case retrieval and adaptation. This combination relaxes to a large extent the circumscription problem that limits the performance of deductive reasoning systems while providing a means of 'coaching' the learning process within case-based systems.

In CLAVIER, an application supporting the design of loading configurations for autoclave processing, Hennesy and Hinckle [1992] use a representation that distinguishes between local and global contexts for case adaptation. This is in direct response to the nature of autoclave (a controlled temperature and pressure curing environment for composite material processing) processing where global temperature and pressure are regulated but local part to part interactions must also be modeled. Rough models for the global context parse the overall loading parameters by composite type, general curing requirements and mold types. Local contexts provide loose models of local air flow and thermal capacity interactions. Thus retrieval is done on both local and global contexts to suggest overall autoclave processing parameters and also the arrangement of parts within the autoclave.

Maher and Zhang augment a structural design synthesis expert system with a case-based component in CADSYN [1993]. An expert system (EDESYN [Maher and Zhang, 1991]) contains design decomposition knowledge along with limited synthesis knowledge in

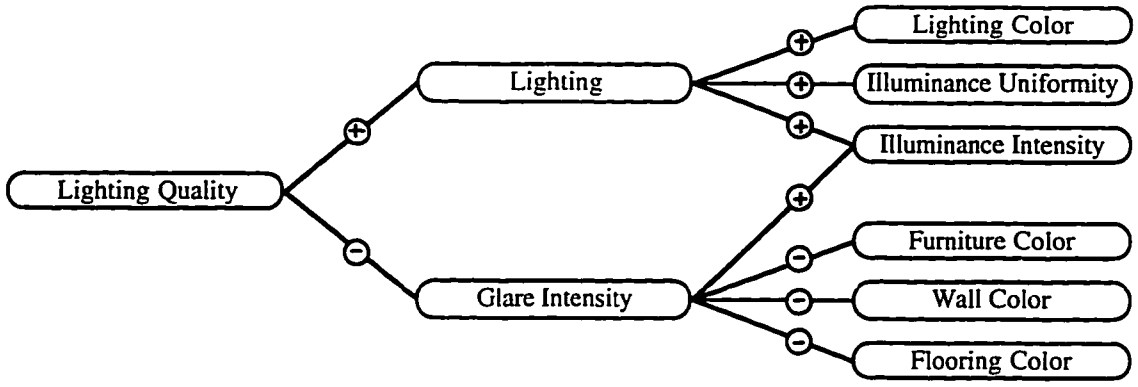


Figure 2.3 Lighting model from Archie, showing taxonomy of lighting issues along with simple directional influences [Pearce et al. 1992].

structural design for buildings. This background knowledge is used to develop a basic description of the current design context. Once this context is established, design cases are retrieved and adapted in response to the current state of knowledge using typical case similarity metrics. More heuristic knowledge is then applied to determine the feasibility of the resulting case by looking for conflicts with simple design rules. Adaptations that violate these design rules are retracted and the retrieval and adaptation process repeated until all case violations are resolved or the case base is exhausted.

Archie, a case-based system for architectural design [Pearce et al., 1992], adds general performance evaluations and generalizations about their importance within several design classes. An instance of a design model is shown in Figure 2.3. Note that it is similar in flavor to the representation shown in Figure 2.2 in that quantities are related through a directional sign. Here, however, the model does not represent a set of design components to be synthesized but a set of design parameters that are themselves derived from a much larger scale synthesis of architectural elements. Thus, Archie's models are just a part of the story, each representing the generalities of a design issue. The collection of models is simplified by restricting the scope of each model to a subset of relevant design parameters and purposefully not modeling interactions among design evaluations. The design models provide evaluations of specific aspects of an architectural design, and the designs themselves offer representations of successful sets of context-specific tradeoffs made among these evaluations. In the end, these models are used primarily to cluster design

cases by issue; each case is contextualized by design models. Adaptation here is not a major issue – Archie is not used to synthesize designs but to show general tendencies among design cases. The design is in charge of synthesizing these tendencies, putting them into the current design context.

Garcia and Howard take a slightly different approach in the framework of a model enhanced design documentation and support system, ADD [1992]. The domain of ADD (Augmenting Design Documentation) is the layout of structural and system elements in ceiling cross-sections. Instead of using cases to define preferences among various heuristic models, ADD assumes a general preference function based on application type. When the user 'violates' the prescribed preference set (a weighting of design evaluations), the system prompts him for justification. A new preference set is then assumed based on the user's final decisions and a new context generated. This context can then be applied whenever its preference set models those of the user better than the prescribed set. Eventually the new context is added to the context classification mechanism within the system at which point the system operation essentially parallels that of Archie.

Goel and Bhatta [1995] embed cases into a design reasoning system based on the structure-behavior-function (SBF) representation model proffered by Gero [1990]. The structure of artifacts is neglected for this application, so two representations for a design coexist: a functional level description in which an abstract description of the process is encoded (e.g. reduce acid temperature), and a behavioral level which instantiates a model for accomplishing the function (e.g. send the acid through an acid-to-water heat exchanger). Cases are indexed on both levels of abstraction in IDeAL (Integrated Design by Analogy and Learning). Abstractions of functional classes (e.g. cases using the zeroth law of thermodynamics) are used to retrieve functionally similar cases. This set is then examined for similarity on the behavioral level along appropriate indices like material, component set, etc. Background knowledge declared in IDeAL includes behavioral models for components and substances 'known' to the system expressed as an ontology. This background knowledge is used to verify the behavior of designs while supplying a basis

for functional abstraction.

2.2.4 Derivational Analogy

In the discussion thus far, design cases are represented as abstractions of artifacts and in some instances multiple abstractions are used. CBR has been confined, by the declarative nature of the case representation, to a process of transformational analogy. Even in the latter discussion where substantial design knowledge is provided to the CBR process, most of this background knowledge is used to partition the representation to decompose cases into viable sub-cases which can then be used by applying transformational analogy. Carbonell [1983] describes derivational analogy – the application of problem solving strategies or domain knowledge in contextually similar reasoning processes – as meta-level CBR. This is a step beyond the CBR described in the above design systems. Clearly, it would be advantageous to apply the procedural information of successful design cases to new situations. In fact, some design cases are undistinguished from an artifact standpoint but are extremely interesting from the standpoint of the design process applied. Thus far, each case-based design system discussed has used a generic, machine understandable representation for the reasoning control process.

Implemented as an expert system that uses competing heuristic and case-based knowledge for structural design (specifically electrical transmission tower foundation design), DDIS [Howard et al., 1989] stores not only the typical artifact specifications but also the design actions used to arrive at them. CBR is used to retrieve similar cases based on both of these design parameters and to suggest design activities like goals to satisfy, parameters to assume, constraints to activate, and subsystem design tasks. This is all done in a blackboard system where control knowledge is declared within the knowledge base.

2.3 Summary and Discussion

This survey of case-based design systems illustrates the many ways in which CBR can be applied to design problems. In each case, the general reasoning flow of retrieve / adapt

over design cases is retained (although adaptation in the systems discussed above is by far the weakest component). It is the nature of what is known (or at least known to be important) about the domain that suggests a case representational scheme which, in turn defines the nature of reasoning about the cases.

For systems where knowledge about the underlying mechanisms of the design problem is scarce, CBR operates over raw data to provide solutions to new design problems. This is most effective when the solution space can be partitioned by similarity and when the adaptation of design cases is done by some form of interpolation rather than extrapolation. This is the general problem of inductive reasoning and suggests more representation schemes than those hit upon in the survey above; decision trees [Quinlan, 1986], probability density functions [Cheeseman et al. 1988; Specht, 1988; Tseng 1991], classification rules [Mitchell, 1975; Winston 1977], exemplars [Salzberg, 1991], neural network weights [Tesauro and Sejnowski, 1989] have all proven effective schemes for inductive reasoning. Perhaps the most important consideration in selecting the representation is the understanding of the bias that it encodes [Mitchell, 1980] and the impact that that bias has on the types of concepts that can be learned.

For the functional embodiment problems described above, design knowledge is simple enough to be encoded directly into the representational bias. In CADET components are defined by influence graphs; in Schemebuilder bond graphs are the representational mechanism. Neither representation is able to select components by weight, efficiency or even cost, much less more complex considerations like packaging or assembly difficulty. Only the fulfillment of rather simplistic functional goals is considered in these systems, rendering them useful to neophyte designers but perhaps less useful in the real-world concurrent engineering context. The CBR framework is implemented as straightforward retrieval (generally queries into deterministic databases) by design goal satisfaction framework. Decomposition is accomplished by caching non redundant snapshots of a design as it is being constructed. When means ends analysis is coupled to a limited set of subgoal generation rules, the retrieval / adaptation cycles in CBR are indistinguishable from

more model-based planning systems [Fikes et al. 1972]. The use of cases provides an efficiency enhancing mechanism, but does not extend the reasoning capability of the system beyond that attainable through the direct synthesis of solutions from sets of primitive elements – model based reasoning. In fact, applying CBR to such problems can actually hinder efficiency unless careful analysis of the utility of caching each new component is performed [Segre, 1987; Minton, 1988].

CBR techniques seem most fruitful where it is not possible or desirable to express design knowledge solely through a representational bias. In this case, the knowledge that exists can be declared in a richer representation and incorporated into CBR systems. CLAVIER and CADSYN use cases to 'fill the gaps' in declared background knowledge by adapting cases from subsets defined by reasoning within the context of explicitly declared background knowledge. The simplified design knowledge encoded in Archie is used to identify cases from which the user could learn design 'morals' – textual analysis of the overall case and the lessons to be learned from it. ADD contains an expert system that evaluates solutions along various design 'dimensions'. Cases stored as sets of general goals and weight vectors anticipate the decision made by a designer; when a designer deviates from the anticipated path, justifications for violating the case-based assumptions become the basis of a new case added to the systems case base. IDeAL gains clustering information that helps it define useful contexts in which to apply its function-behavior case models.

It is difficult to pigeon hole case-based design systems as a consistent application of a well defined paradigm. Perhaps that is the lure of CBR; it provides an attractive framework for exploiting the case knowledge whose importance is intuitively obvious to those working in the field of design. As reasoning paradigms go, CBR is a very inclusive one; we have seen it interpreted in different ways in the systems described above, each instance successfully applying some form of case knowledge toward solving design problems. These varied interpretations are driven largely by the representational issues each application suggests. The goal of the CDIS is to exploit generic case knowledge for supporting conceptual design

and concurrent engineering. Thus, the nature of conceptual design information and how it can be represented in the various contexts of concurrent engineering is the next obvious step toward applying CBR to the CDIS. Because all domain knowledge must be encoded within this representation, the primary concern is to match its expressiveness (bias) with the complexity of the domain concepts it must encode. Too limited a representation and one has problems differentiating among concepts; too rich a representation and generalization becomes difficult – each case becomes and isolated 'island'. The next chapter takes up the issue of the representing conceptual design information and how that representation drives the realization of the CDIS as a case-based design system.

Chapter 3

Case-Based Concurrent Engineering Conceptual Design

The previous chapter addressed the current state of case-based reasoning applications for design. Each system discussed is limited to a well-constrained subset of design activity. Almost all of them deal with the mechanism for transforming information stored about an object into slightly different contexts. In systems like Schemebuilder and IDeAL, multiple levels of representational abstraction are used to extend capabilities by allowing this transformation to occur on separate knowledge 'levels'. While the systems described in Chapter 2 can claim to aid significant (albeit small) parts of the overall design process, none adequately addresses conceptual design.

In *Design Paradigms: Case Histories of Error and Judgment in Engineering,* Petroski presents the viewpoint that treating design artifacts themselves as successful design cases has resulted in some of histories most infamous failures [1994]. He has even identified a time constant, about 30 years, for catastrophic bridge failures that he claims is due to the lack of communication of design strategy from one generation of designers to the next. In each case, the new generation incorporates improvements in materials or construction techniques in new designs that they believe accurately extrapolate the reasoning embodied in the projects of their forbears. Factors that were not primary constraints on previous designs gradually increase in importance, generally due to the scaling effects of bigger and better projects. Safety factors included in the historical prototypes mask slight violations of

these secondary constraints. As the design program progresses farther and farther away from the prototype, successes breed the false impression that all is well. Eventually, catastrophic or unexpected failures result when the safety factor is consumed by design constraints previously of secondary importance, constraints ignored because their influence was not apparent in the original design artifact.

In *To Engineer is Human: The Role of Failure in Successful Design* [1985], a separate treatise on the nature of engineering design failure, Petroski reinforces this view by pointing out instances of miscommunication in shorter time periods (i.e. project life cycles) which led to the catastrophic Kansas City Hyatt skyway failure and the explosion of the booster rocket in the space shuttle Challenger. In the former case, a design that was difficult to build was changed by the fabrication company to a design which was thought to be of equivalent strength (i.e., the new design addressed what was thought to be the primary mode of failure for the system) but much easier to build. Because it was not communicated back to the original designer, the new design was not modeled according to the original rationale (which would have prevented the accident). Similar poor communication of design rationale contributed to the Challenger explosion. While operating conditions were only marginally outside of the design range, this small perturbation was enough to cause significant changes in the performance of a key gasket in the booster rocket. Unaware of this extreme sensitivity to operating conditions, mission operations approved the liftoff which resulted in tragedy.

In both of these cases, communication among project participants was carried out largely on the artifact level. Had more of the design decision process been open and available to all life cycle participants, the final failures might have been averted. Had there been better communication among life cycle customers, more informed design decision might have precluded failure. The general lesson from Petroski's analysis of design failure is that artifacts do not explicitly embody the decision process from which they arise. To improve case-based design methods, we must focus on communicating to the designer (computer or human) aspects of prior design decisions and processes that are relevant to the current

context.

Case-based design systems, as they currently exist, generally violate these tenets. Even in applications involving routine design where old data points are used to synthesize new designs, the representational bias used to express a design instance might hide whether a dangerous design extrapolation is taking place. Suppose that in a building design system, cases in which the primary failure criterion for the design was flexural stress are used to generate a new beam design with a long span but relatively low strength requirements. Parameter-based design systems like STRUPLE might generate a design in which the failure mode of localized panel buckling becomes the overriding design constraint. This change in critical design constraint is not effectively communicated through the artifacts. Part of the motivation for the CONCEPTOR/EFD/DPF evolution of STRUPLE is to provide a clustering preprocessor to separate the design space so that assumptions like 'same failure mode' are more likely to hold over the subdomains in which the transformational subsystems synthesize new designs. Unfortunately, failure mode criterion is not expressed directly in the representation of the bridge design database for the system. Petroski's findings indicate a greater need to move from the transformational analogy embodied in most CBR systems toward derivational analogy. Case-based design systems implementing transformational analogy are limited in scope to domains where a very strict 'sameness' assumption is valid. Perhaps it is the validity of this assumption that delineates routine design from creative, conceptual design and provides the impetus toward employing a more derivational approach for case-based conceptual design. The CDIS approach to solving this problem is: determine the nature of conceptual design information, identify a means for effectively representing this information, and provide methods for manipulating this representation for use in new design instances.

3.1 Conceptual Design Case Content

Design cases contain three major classes of information: contextual information, design process information, and product information. Contextual information represents the

relationships among all of the product life cycle customers. Most closely associated with design intent, it describes the societal bases for undertaking the design activity in the first place. Design process information describes the evolution of the design: the generation of alternatives (and potential evaluation procedures), the evaluation of these alternatives, and the design decisions based on these evaluations. Product information concerns a description of the artifact and the processes with which it is involved throughout its life cycle.

3.1.1 Design Context

In defining the information storage requirements for design history capture and use, Ullman [1994] emphasizes design process and product as the major sources for reusable information. Design context strongly influences concurrent engineering processes because of relationships that are embodied. Clearly design is not a context-free process – toy companies undertake projects in ways far different from aerospace companies. This is due to several factors – relationships between company and customer, relationships among organizations within the company, even the skill level within each organization. For derivational analogy to be effective, the system must be able to distinguish such contextual differences so that design strategies are 'replayed' in appropriate settings.

Context is fundamentally difficult to encode because it is highly abstract and often pervasive to the point of being unnoticeable. It becomes a noticeable aspect of a design project when participants attempt to explain design decisions to 'outsiders' who do not share the contextual base of the project. Design capture studies have traditionally taken an 'insiders' view to analyzing the design process, focusing on small sets of designers participating in projects which often do not realistically model organizational or societal impact on design decisions. The design process is not generic but shaped by context; when that context is matched, derivational analogy can make stronger assumptions about successful design strategies and make more appropriate mappings.

In all of the case-based design support tools discussed in the previous chapter, context is

not represented explicitly. Even in DDIS which employs derivational rather than transformational analogy, context is prescribed by the expert system knowledge base; the analysis attempts to specify foundation supports for single pillar steel transmission towers based on successful past strategies. The user has to have already decided that such a tower best fits the needs of the current design and the system then applies appropriate strategies toward specifying the parameters associated with that design class. Garcia's ADD system directly acquires new design contexts through the preference systems stated by designers who contradict the system's predictions about a desirable solution. For conceptual design, finding the class of a particular design is much of the problem. This calls for explicit representation, or at the very least acknowledgement of context for design cases.

3.1.2 Design Process

Storing and representing design process has been the focus of more extensive study. Modeled as a series of decisions, the design process representation generally falls into one of two philosophies – issue-centered and information-centered. Issue-centered design is concerned with gathering consensus among life cycle stakeholders and is derived from the ideas of Rittel [Kunz and Rittel, 1977; Rittel]. Information-centered design representations model the process of operating on abstract objectives and constraints to produce concrete solutions.

Rittel's Issue Based Information System (IBIS) is a good model for conceptual design because the main focus is determining and discussing issues critical to the design problem itself. Design solutions are proposed only to resolve or discover conflicts among participants. Argumentation over these conflicts repeatedly hone the statement of the design problem until this statement becomes the conceptual design. From that point onward, routine design methods referring to this statement of the design goals and constraints are used to flesh out the actual artifact. Because it supports agents which hold differing personal objectives and constraints which must be resolved into one design (or design specification at least), IBIS is a compelling model for design within concurrent

engineering. As such, it has been extensively studied as the basis for encoding design intent and design rationale [Conklin and Begeman, 1988; McCall, 1989&91; Nagy et al., 1992; Hirose et al. 1994].

Information-centered design process representations embody the philosophy that conceptual design is primarily an information gathering exercise. An initial set of objectives and constraints stated in relatively abstract terms are the basis for a search to help determine objectives and constraints that are concrete and to find suitable solutions or solution classes. Such a philosophy is conceptually in line with the iteration that is at the heart of almost every description of the design process. Initial specifications provide only loose constraint on the design search space, so sets of abstract solutions are proposed. These solutions are considered and selection among them often brings to light new or more concrete design specification information. This process continues until the design space is sufficiently narrowed for routine design to take place.

Lakin et al. [1989] chose to capture the design process by recording design steps. Retrospective analysis of these steps are then 'tagged' for rationale or intent, however the reliability of such a tagging scheme calls into question the accuracy of retrospective studies in general [Ullman, 1994]. Hirose et al. [1994] describe an information elaboration scheme in which a design is represented as navigation between design decisions, each step providing design focus for the next. Others have suggested that information gathering is the result of the need to perform iterative decomposition to define sub-problems whose design space is of tractable size [e.g., Bascaran et al., 1989]. As mentioned in Chapter 1, IRTD explicitly operates on the state of information within a design, suggesting the information gathering efforts which will provide the greatest benefit to the current design decision. In each of these cases, the progression of gathering and grouping of information defines the design process.

Semantically, issue-based and information-centered representations for design process are two sides of the same coin. IBIS representations focus on agents, each trying to get the

'best deal' out of a design solution from its own perspective. Expression of this perspective is a prerequisite for bargaining power; since consensus through argumentation is the design process model, positions can only be made tenable through the expression of one's viewpoint. Information-centered representations focus more on the design artifact and the information necessary to decide among design prototypes. To the extent that IBIS focuses on design alternatives for argumentation and that information gathering is directed toward refining specifications, the two bases converge. They diverge when IBIS methods degenerate into bickering over specifications without regard to their impact on the design and when information gathering is directed toward subsystem decomposition and modeling.

3.1.3 Product

Product level representation has received the greatest amount of research focus. From PDES/STEP [NIST, 1995] to the ARPA knowledge sharing efforts [Neches, 1993], information about artifacts takes on a broad range of abstractions, each tailored to a specific application. The following is a brief description showing the range of representations researchers have found useful:

Parametric: This is the representation of choice for case-based design systems discussed in Chapter 2. Each system uses a custom symbolic representation, tailored toward the reasoning that is done within the system. This means that case information is generally not portable or meaningful outside of the context of the system operating on it – a highly undesirable trait.

PDES/STEP: This effort has grown as an abstraction built on the representational foundations of IGES [NIST, 1995] and extending toward representing parametric information within product classes in a standard way. Currently, much of the standard focuses on tolerances and other geometrically related operating parameters for a very restricted set of products. Information is captured in a schema-based representation which supports product classes and inheritance.

Graph Representations: Several graph representations are used for modeling design information at the product level. Bond graphs are a direct encoding of the relationships among components in a linear dynamic system. Used by Ulrich and Seering [1989] and in Schemebuilder [Chaplin et al. 1994], bond graphs provide an abstract language that is portable across domains. Electrical, mechanical, fluid, and fluid systems can all be encoded by bond graph abstractions which model a finite set of energy generative, storage, transforming, and dissipative elements.

Influence graphs are used by CADET to encode mathematical abstractions of product performance. A set of directed arcs among a defined set of physical parameters within various domains (perhaps not coincidentally the domain and parameter set are quite similar to those represented by bond graphs) are used to model the behavior of assemblies of components. These behaviors are not dissimilar from graphical expressions of the qualitative representations used to model general physical systems [Forbus et al., 1991; deKleer, 1993]. In addition, having found the simplistic bias too limited in expression for all but the most abstract representations of physical systems, Michelana and Sycara [1994a] present options for relaxing the bias (while still maintaining the influence graph basis) along with the more complex reasoning methods necessary to make up for this increase in expressiveness.

Mathematical Models: Perhaps the most ubiquitous of design product models in practice, mathematical representations are often de-emphasized within knowledge based systems. In a representation less abstract than the bond graph representations it uses for case-based synthesis, Schemebuilder expresses mathematical models for components as Matlab SIMULINK block diagrams.These block diagrams are linked together through the graph representation results of abstract design synthesis to provide a behavioral system model. In the EDF subsystem for case-based bridge design, Maher and Li induce mathematical expressions as abstractions of case information using a fixed, linear regression model.

A more typical mathematical representation for a product is the generic nonlinear programming problem statement. In this, functions describing the objectives for product performance are stated along with those constraining product behavior. Additional mathematical modeling information is included so that a set of design parameters can be evaluated within these objectives and constraints.

Ontologies: The ARPA Knowledge Sharing Effort [Neches, 1993] is rooted in the need to translate and transport knowledge into various machine and human readable forms. The Knowledge Interchange Format (KIF)[Genesereth and Fikes, 1992] has been defined as the language for encoding sharable knowledge, it provides for declaratively stating semantics (representations do not rely on an interpreter for meaning) and is able to express arbitrary logical constructs. Within KIF, products are described by ontologies [Gruber 1993] which encode all that is known on the product level through the definition of classes, relations, and functions.

The expressiveness of KIF has a great impact on the type and amount of information to be represented about any product. Since the goal is to create 'portable knowledge', everything about a product that might be relevant to any reasoning process must be declared to the system. KIF does provide a building block ontology for engineering mathematics; portable mathematical models must be encoded within the constructs of these standards. Because everything about a product must be declared KIF carries a lot of extra baggage[1]. In return, KIF promises reusability of ontologies and portability among domains and systems.

Structure-Behavior-Function (SBF): Proposed by Gero to model three typical layers of abstraction necessary for representing design product information, SBF representations connect three separate logical expressions of a design. In decreasing order of abstraction: the functional level is concerned with the intended purpose of

[1] An example elevator design domain requires over 90 pages of declarations for modeling in KIF. Not many designers are willing to create such a large model, even if it were practical to do so in light of uncertainty in the problem statement.

the artifact embodiment, the behavioral level is used to derive the physical means by which this function is accomplished, and the structural level describes the embodiment of the actual artifact. The implementation of the SBF model is left to the particular application. In the case of IDeAL only behavior and function are modeled, represented as sets of logical constructs.

The functional aspect of design representation is accomplished through influence graphs in CADET; Michelena and Sycara [1994b] propose a structural abstraction for the kinematic embodiment of functionally complete designs. A post processor operates on physical connections within the functionally complete influence graphs to determine valid sets of kinematic connections for the mechanical aspects of the design.

In Schemebuilder, function is represented as bond graphs, behavior (discussed below) represented through mathematical models, and structure again represented using more complete forms of bond graphs which provide input to an embodiment module. This module then produces CAD solid models of an overall design embodiment including structural components.

In the Device Modeling Environment (DME) under development by Iwasaki et al. [Iwasaki and Low, 1991, Iwasaki and Levy, 1993], the structure, behavior, and function of engineering products will be explicitly encoded within product ontologies. These ontologies must provide the mapping among the SBF conceptual layers for a component or set of components along with complete descriptions of them on each layer. To use the system, the engineer would select components by function, define structural relationships among them, and then generate behavioral simulations to test the arrangement. DME currently supports specification of components and structural connections with limited simulation capabilities. A natural language explanation of the model and the system operation it encodes is

also supported[2].

Of course, there is yet another model for representing product information. It is also the current industry standard: Sets of engineering drawings, calculations, memos, design reports, operation manuals, etc. collectively define a design on the product (and process) level. The diversity of media suggests that no single representation is suitable for representing the final design. It also suggests that even such theoretically unbiased representation systems as ontologies must be augmented to capture fully the information relevant to designing product.

3.1.4 Information Mapping

In the above discussion, the separate mappings of function, structure, and behavior within the single 'Product' representational layer were discussed. A similar mapping must take place among the context, process, and product representational layers as well. Clearly the rationale and intent encoded in the design process representation is at least partially based on information that comes from the design context. Likewise, the design product (including the analysis by which it was specified) is heavily influenced by the design rationale and intent. Perhaps the main failing of the current case-based design systems is that their representational schemes are heavily biased toward automated case-based reasoning (transformational analogy) and so only express the 'what' of a design. They rely on that 'what' to fully express the 'why' and the 'how' of the design process. Their success can largely be attributed to their focus on routine design problems which can be resolved in a relatively context-free way.

When the scope of the system is broadened to include conceptual design in a concurrent engineering environment, design representation using only the product level no longer suffices. Concurrent engineering design is often modeled as a multiobjective optimization problem. Conceptual design requires defining these objectives and constraints from rather

[2] This is natural language *generation* and is mentioned here to lend support to the need to bridge the gap between logical / mathematical constructs and more descriptive interpretations of them.

abstract beginnings, using iterative search to collect promising elements of a successful design – including objective, constraints and solution candidates. In addition, to make this multiobjective problem easier to solve, it is desirable to synthesize the viewpoints of the many customers into a single, concrete objective. This requires mapping from representations of the 'why' and 'how' aspects that are the focus of the context and process levels of representation onto the 'what' that has proven tractable for case-based reasoning.

Hauser and Clausing [1988] have proposed a mapping scheme to aid engineers in defining the critical performance aspects of a product and mapping these onto mathematical representations – the 'House of Quality' (HoQ). The foundation for the HoQ is an information-centered process representation. The mathematical performance measures are rooted in the product level. Competitive benchmarking, also part of the HoQ model, brings in some of the design context through a description of the current or projected performance parameters within the market for the product. Thus it is difficult to ascribe the HoQ to a single representational layer.

The original intent of the HoQ is to promote communication between engineers, marketers, and customers. Others have taken steps to further enhance or formalize the results of this communication. Ulrich [1993] adds engineering models to the HoQ to provide more than just heuristic feedback on performance parameter interactions. Mathematical models are used to check consistency among performance requirements. They use the available design space to augment the HoQ toward a realistic set of customer preferences. Locascio and Thurston [1994] take another tack, mapping the HoQ directly onto multiattribute utility function used for optimization. This provides a direct link between the process and product representational layer, although this mapping is at best a rough estimate of the true design specification. These two approaches recognize the need for mapping among the representational layers laid out above.

3.1.5 Information Quality

Conceptual design requires decision making under uncertainty. Often customers don't

know (or can't articulate) precisely what their 'need' is. Expanding the scope of the term customer to include all life cycle members brings about even more uncertainty. Preliminary designs must be evaluated on the basis of partial information using imprecise evaluation and constraint models, and by nature conceptual design precludes consideration of complete manufacturing concerns. The reality is that during much of the design process, information is 'informal'.

Ullman [1993] describes product level design representations, based on SBF classifications, as evolving concurrently through the design process. Design begins with informal representations of function which suggest behaviors which in turn suggest structure. Conceptual design is thus the co-evolution of a product representation on all three levels as information about structure is fed back through behavior which in turn suggests less abstract definitions of function. This design process is much different from the prescriptive models of design that force more or less complete development of function, then behavior, then structure, with slight adjustments allowed in the periods of transition from one focus to the next. If such a prescriptive model were the case, then design structures could be derived using CBR just from behavioral specifications. Likewise, behaviors follow from completely specified function (this is the model for IDeAL). Perhaps, again, the distinction must be drawn between conceptual and routine design.

Other studies of design have revealed similar findings in terms of the abstraction level of information during the design process. Under the prescriptive design model, abstract information would be present only in the very early stages when the functional specifications are being derived. Baya and Leifer [1994] suggest that information during conceptual design is largely informal, quantitative information remains relatively constant at low levels throughout the development of design concepts. McGinnis and Ullman [1992] performed a study in which they determined that 55% of the information involved in conceptual design is informal (in addition, at least 17% of detail design information was also found to be informal). Such studies point toward representations much different from the extremely formal models currently in use by case-based design systems. The following

discusses the evolution of one such case-based design system.

3.2 CBR for Case-Based Conceptual Design Aids

As discussed in Chapter 2, Archie is a case-based system developed to support architectural design. Architectural design is a much more conceptual and less routine design domain than any of the other case-based systems discussed. For this reason it is of particular interest toward the development of the specifications for the CDIS. In its first incarnation, Archie starts off within the general case-based paradigm of retrieving cases similar to the current design instance. Similarity is measured by clustering cases using a set of causal models for performance (along several axes of interest). After the retrieval, Archie diverges from the case-based paradigm; it does not attempt to adapt or combine cases. What Archie does is present an overall parametric description of the case which is augmented by the key lessons embodied by this description. CBR is transformed into directed browsing of design cases. The lessons learned from Archie (paraphrased from [Pearce et al., 1992]):

1. Real-world design cases are incomplete: Retrospectively gathered cases do not contain complete (or sometimes even partial) descriptions of design goals and constraints. They do not contain justifications for the designer's decisions.

2. Real-world cases are large: For a case to be useful in a variety of contexts it must contain a great deal of information on context, process, and product levels. This information must be made available to the designer at varying levels of abstraction, posing the difficulty of providing navigational support for the user within the design case.

3. Systems must offer multiple knowledge types: Although Archie used domain models to cluster design cases (see Figure 2.3), these were not available to the end user. Because these models often represent design prototypes, it is useful to present them to the user as well.

4. Systems must present relevant information: Archie presented all of the available information about a case instead of just that which was relevant to the current design activity. Thus important parts of design cases are obscured by information overload.

The development of Archie II [Domeshek and Kolodner, 1992] was based on these lessons and led to a much different type of case based design system: a Case-Based Design Assistant (CBDA). Although design cases are still the main source of knowledge within the system, Archie II deviates even further from standard CBR than in the earlier implementation. Most of the reasoning is eliminated in favor of a system that contains small sub-cases indexed by design issues and relevant stakeholders. These sub-cases, or lessons, are presented to the user in response to queries along issue/stakeholder axes and represent the lessons learned about a design taken largely from post-occupancy evaluations of buildings. Design intent is estimated from rough building usage goals due to the difficulty of acquiring such knowledge retrospectively. The framework for Archie II has been stripped out as a CBDA shell, Design MUSE [Domeshek and Kolodner, 1994]. Design MUSE supports the creation of specific stories about a design using free text and design drawings.

A third iteration of Archie [Kolodner, 1995] is currently under development and moves even further away from the traditional CBR paradigm. Archie III is directed more toward student architects. The same lessons from Archie II are more closely tied to standard architectural contexts. The main indexing scheme is based around floor plans and general arrangement drawings which provide a substantial backbone for navigation. Objects within these visual elements are highlighted by color corresponding to post occupancy evaluation of their performance. They are also linked to multimedia representations of lessons to be derived from the case.

In the floor plan, architects have a product level representation from which they can move to more abstract discussions concerning the effectiveness of the product in relation to

customer needs. This then gives them insight into customer needs for future projects. Currently operating under a simple hypermedia navigation model, the indexing scheme within Archie III is being extended in attempts to have it operate in a critiquing or context sensitive help mode. This involves pattern matching within drawings and recognition of a standard set of architectural graphic gestures to define intent and context [Gross et al., 1994]. This work is similar to that being done in another architectural case-based design system, FABEL [Voss et al., 1994]. In this system, a suite of similarity metrics are being derived to cover the many abstraction levels necessary to identify case information from the knowledge base relevant to the current design instance. These range from strict numerical evaluations of building performance parameters to gestalts [Schaaf, 1994] which attempt to capture relevance in a holistic fashion by recalling clusters of design object groupings.

3.5 CDIS Representation & Operation

As a case-based conceptual design system for concurrent engineering, the CDIS is more closely related to Archie and FABEL than to any of the previously described engineering case-based design systems. The abstraction level appropriate to architectural design is similar to that demanded by engineering conceptual design. Such abstraction must be supported through a representational scheme that can encode the broad spectrum of design information types in such a way that they are useful both to designer and to case-user. To support loosely structured design information, the CDIS employs a hypermedia representation where indexing among elements encodes the original design process. In this way, the CDIS supports the use of derivational analogy for conceptual design[3] .

There are key differences between the application domain of the CDIS and that of its architectural brethren. Compared to the relatively finite set of architectural design contexts, engineering design is an extremely broad field. Architectural context is often highly localized, buildings are naturally fixed within socio-geographic context that is relatively

[3] This is done by representing the process explicitly within the system. The varying abstraction level within the CDIS promotes variance in the degree to which a designer can re-derive or replay a design in new instances. Through the reuse of IRTD models under altered constraints, the CDIS can directly support derivational analogy.

well defined. Concurrent engineering design must account for a more global context among an expanded set of customers throughout the product life cycle.

Information support for engineering models must also be a part of the CDIS. Models of component behavior are an effective tool in the negotiation process that is at the center of conceptual design. They help shape the dialog among stakeholders in a design by grounding abstract concepts in engineering reality. Because they must be used to support argumentation at different levels of abstraction, multiple models for engineering components making use of differing amounts of information must be provided. Knowledge about the accuracy of such models, gained through experience, is also a key component toward realizing IRTD as a normative method for design decision support. Because mathematical representations (in our case stochastic nonlinaer programming problems) are the 'industry standard' for design models, the CDIS will implement a method for encoding and relating them.

Another goal of the CDIS is to promote the use of concurrent engineering in conceptual design. Prototypes of the design process using concurrent engineering principles as interpreted in specific product and manufacturing sectors will provide a background for application of concurrent engineering in new contexts. This information source is similar to that used within Archie III in that retrospective studies of industrial application of concurrent engineering are presented. Floor plans are replaced by the product life cycle as the main navigational backbone for these case studies. This local index is augmented by a more global index which helps link modular case studies into a more useful corpus of concurrent engineering applications.

We have chosen the general parameters for the CDIS architecture in the context of current case-based design systems and the informational aspects of the conceptual design process. Chapter 4 provides a discussion of the technologies that foment an effective implementation based on these general parameters. Chapter 5 then focuses on the core CDIS applications implemented using this architecture.

Chapter 4

The CDIS Architecture

In the previous chapter, the type of information that must be captured for the effective application of case experience in design was outlined. This chapter is concerned with the representation of that information and the definition of an architecture for serving it. The three general levels of information representation presented in Chapter 3 are repeated:

Context: Project information relating to the overall engineering project of which a case is a part. This information includes the objectives of the organizations involved in the project including intra- and inter-organizational operations. Such information is vital toward encoding overall design intent for a large scale project. Use of this information is to establish design context and, having done so, present successful applications of concurrent engineering principles within this context.

Process: Representation of the decision and information gathering aspects of the design case is done on this level. Primarily associated with design rationale, this information provides the primary means by which a design can be 'replayed' using derivational analogy. For this reason, the decision process encoded in this representation must reflect arguments and constraints on decisions that were made. While the artifact that results from these decisions bears a direct relation to their content, the decisions and the information basis for them must be represented

Product: Product information is that which is typically stored in case-based design systems. Specifically models for structure, behavior, and function are stored and indexed for potential reuse in similar design instances. This limits the understanding of the case on the product level to the abstraction chosen. It is more useful to archive, within the context of decision being made, the various abstractions useful to each member of the concurrent engineering team. This includes the usual analysis models but must also include manufacturing process plans, part drawings (including tolerances), etc.

Representation methods within the CDIS have been divided for accomplishing the above objectives. For information that is loosely structured, mainly on the context and process level, hypermedia has been chosen. For engineering analysis on the product level, mathematical relations, as sets or as individual entities to be combined, model design function and behavioral. Additional loosely structured product information like CAD drawings, sketches, or catalog specification sheets will be stored as hypermedia. Thus the CDIS architecture must be designed to accomplish this mixture of representational formalism effectively while not communicating to the user the distinction in representation.

Hypermedia is an oft used buzzword, but the richness of multimedia elements combined with local indexing is a powerful tool for information representation. As a design evolves, no single medium of expression stands out as the representation formalism for concepts. Designers annotate sketches to explain general relationships among components, they verbalize arguments within teams of designers, they use CAD to draw and analyze concepts, they use mathematical expressions to evaluate and refine concepts. Only in combination do these media provide a formalism for the 'object world' [Goel and Pirolli, 1992] in which a design takes place. Our case-based design system is concerned with making such an object world available to designers for reuse based on matching an index of these varied information sources to the designers current informational needs.

Hypermedia can become confusing. Structure within a hypermedia design 'web' comes

locally from the navigational organization of the linked documents. However, if this primary structure becomes jumbled, the gains in understanding that are made through the integration of media within the document set is lost to frustration and confusion on the part of the user [Agogino and Hsi, 1993b]. Thus design documents within the CDIS retain a logical local structure while a more globally defined structure is thrust upon them. The CDIS architecture must provide this global structure through effective similarity matching metrics.

Relational databases can structure information in ways that is useful for more routine design activities. The object world of more abstract conceptual design gives way to sets of components and SBF models which can be effectively circumscribed by the mathematical or logical formalisms typical of the case-based design systems described in Chapter 2. They key within the CDIS is to integrate this structured information within a hypermedia framework. An architecture in which hypermedia and database can inter operate both on functional and referential levels provides the foundation for application development within the CDIS.

4.1 Integrated Hypermedia – Database System

The CDIS architecture must integrate a hypermedia system for storing unstructured or loosely structured information with a database for storing structured information. To be useful for collaboration in organizations ranging from small design teams to corporations, the system is designed within the current client-server idiom for networked applications. Some of the other concerns toward specifying the underlying architecture in which the CDIS is implemented are:

Multi platform support: The choice of the server platform for the CDIS is secondary to the needs of the user community. A multitude of computer platforms and applications are used by designers; the CDIS must not constrain the freedom of the designer to choose the appropriate tool for his design activities.

Inter operation with existing software: The CDIS must provide easy integration with existing software applications. While the ability to control such software remotely would be desirable from a derivational analogy standpoint, it is seen as secondary to the ability to integrate seamlessly the information generated from and directed toward a user-definable set of applications.

Support for network standards: Computer networking is becoming increasingly important for supporting communications both inside and outside engineering organizations. Implementation within current network standards is paramount to the success of the CDIS.

Open architecture: The software architecture should provide for easy integration of a variety of applications for supporting the needs of conceptual design. The CDIS should be implemented as a set of portable tools rather than a monolithic design environment.

Open indexing: The differentiation between structured and unstructured information within the CDIS is one which is essential for deciding where and how a piece of information should be stored. However, it is not a differentiation that should be forced on the user of the system. For this reason, information that is highly structured must be made available to information requests which do not respect or understand the defined structural formalisms; the user should not have to know what exists in the database or how it is structured to gain access to it. Queries which do exploit the structure of the database must also be supported, although they too should not require user understanding of the database representation.

4.2 The CDIS Architecture

An architecture, shown in Figure 4.1, has been defined and implemented which meets the above specifications. It consists of a network standard hypermedia server using the HyperText Transfer Protocol (HTTP), the standard of the World Wide Web (WWW),

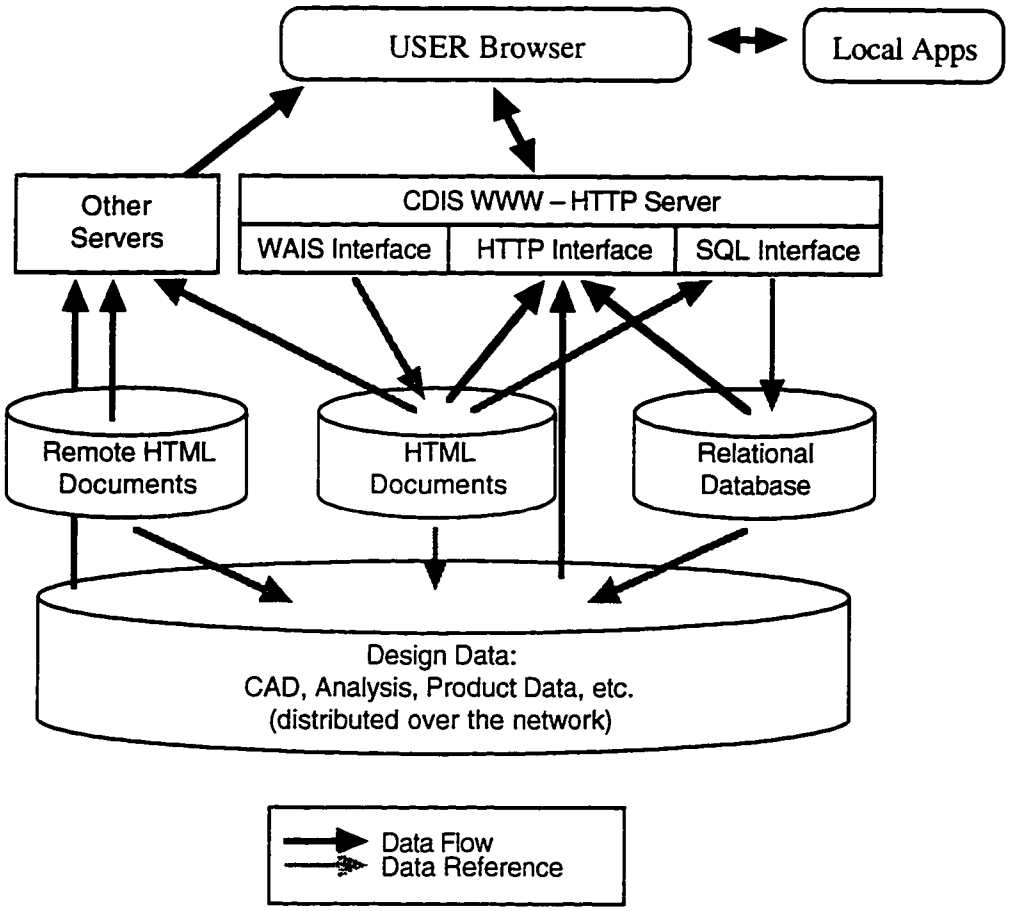


Figure 4.1 CDIS architecture showing open standard client server organization. Databases are served as application with HTML 'front ends'. Structured design information is stored in the Relational Database; unstructured information as a WAIS database (condensed here because the database *is* the set of HTML documents). Links to outside resources like other servers and local applications are shown.

coupled to two database systems. The first database, a Wide Area Information Service (WAIS) server, is used for indexing hypermedia documents based on textual content. The second database, a Structured Query Language (SQL) relational database, is used for storing and indexing structured information within the CDIS. Each of these systems operates on pointers to a set of documents of various media types stored on this or other HTTP servers. A brief description of each of these components is necessary to understand the overall architecture and its power.

4.2.1 HTTP Server

The World Wide Web (WWW) [Berners-Lee et al., 1994] project is rooted in the high energy physics community surrounding CERN. The initial goal of the WWW is the dissemination and linking of research programs distributed throughout the world. High energy physics experiments are expensive and the normal publication process quite slow, so a means of distributing information quickly over networks was devised. HTTP is an internet standard which works in a way similar to simple file transfers. The server listens at a specific network port on an internet connected machine for requests for a document and then sends a byte stream back to the requester. This byte stream can be the contents of a static file, or the result of a program that launched by the server. Additional extensions provide for the input of data for use during program execution and for specifying the remote program which should interpret the output. In addition, HTTP servers are capable of restricting access according to a user's network location, machine name , or username and password. It's simplicity and flexibility has made HTTP service the basis of CommerceNet [Tennenbaum et al., 1995], the main prototype in development for modeling engineering business over the internet. A brief glossary of terms will further elucidate the operation of an HTTP server and the clients which support the protocol. The display of a document typical of the CDIS is shown in Figure 4.2.

URL (Uniform Resource Locator): A URL is an internet standard descriptor used to identify documents (or programs) available over the network. It includes the protocol of the server, the location of the file, and any additional arguments to be passed to the server.

HTML (HyperText Markup Language): HTML is a document type definition within SGML (Standardized General Markup Language). In HTML, ASCII documents contain commands for formatting their display and for linking to other documents via URL. It is called a markup language because the commands (called tags) must be interpreted by an application for displaying the document. Thus, HTML is an

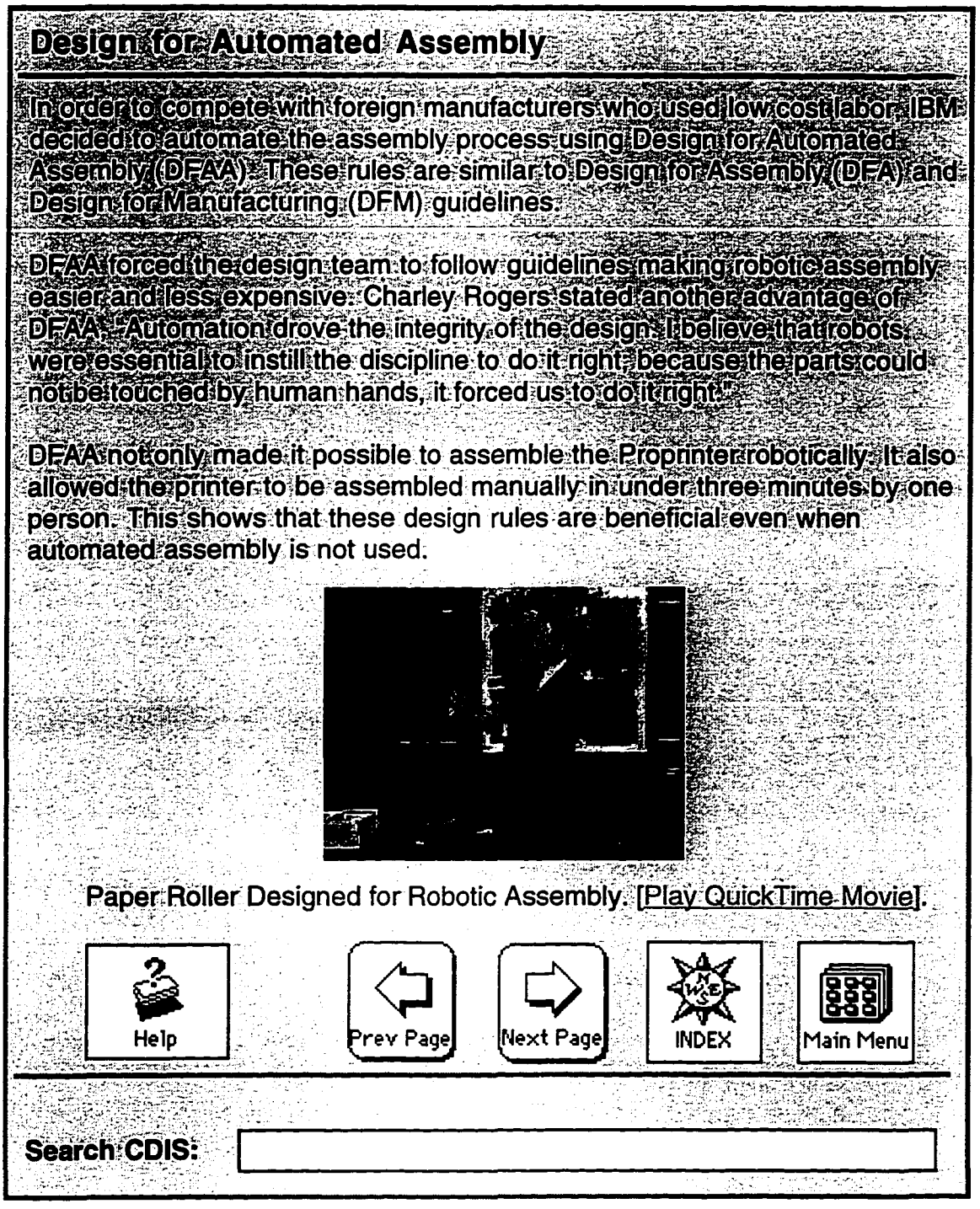


Figure 4.2 HTML representation of a design case study document. Note the hyperlinks (underlined or outlined) to multimedia resources and to local context pages like the main menu, next and previous pages, and index or help page. Virtual hypertext is supplied by the CDIS query interface at the bottom of the page.

```
<TITLE>IBM Proprinter Case Study – DFAA</TITLE>
<H2>Design for Automated Assembly</H2><hr>
In order to compete with foreign manufacturers who used low cost labor, IBM
decided to automate the assembly process using Design for Automated Assembly
(DFAA). These rules are similar to Design for Assembly (DFA) and Design for
Manufacturing (DFM) guidelines.<p>
DFAA forced the design team to follow guidelines making robotic assembly easier
and less expensive. Charley Rogers stated another advantage of DFAA, "Automation
drove the integrity of the design. I believe that robots were essential to instill the
discipline to do it right; because the parts could not be touched by human hands, it
forced us to do it right."<p>
DFAA not only made it possible to assemble the Proprinter robotically. It also
allowed the printer to be assembled manually in under three minutes by one person.
This shows that these design rules are beneficial even when automated assembly is
not used.
<p><center>
<IMG SRC
=HTTP://www.needs.org/proprinter_html/movies.GIF/robotic_assembly_roller.G
IF>
<P> Paper Roller Designed for Robotic Assembly. [<A
HREF=HTTP://www.needs.org/proprinter_html/movies/robotic_assembly_roller.
MOV>Play QuickTime Movie</a>].<p>
<A HREF=HTTP://www.needs.org/proprinter_html/help.html>
<IMG SRC=HTTP://www.needs.org/proprinter_html/icons/help.GIF></A>
<IMG SRC=HTTP://www.needs.org/proprinter_html/icons/small_blank.GIF>
<A HREF=HTTP://www.needs.org/proprinter_html/after_CE.html>
<IMG SRC=HTTP://www.needs.org/proprinter_html/icons/previous.GIF></A>
<A HREF=HTTP://www.needs.org/proprinter_html/DFA_layered_design.html>
<IMG SRC=HTTP://www.needs.org/proprinter_html/icons/next.GIF></A>
<A HREF=HTTP://www.needs.org/proprinter_html/timeline.html>
<IMG SRC=HTTP://www.needs.org/proprinter_html/icons/timeline.GIF></A>
<A HREF=HTTP://www.needs.org/proprinter_html/main_menu.html>
<IMG SRC=HTTP://www.needs.org/proprinter_html/icons/main_menu.GIF></A>
</center><hr>
<FORM METHOD=POST ACTION=HTTP://www.needs.org/cgi-bin/cdis.cgi>
<b>Search CDIS:</b><INPUT TYPE="text" NAME="search_term"
SIZE=40></FORM><HR>
```

Figure 4.3 HTML commands used to generate Figure 4.2. Note that much of the document is textual. HTML tags are the commands embedded in brackets (e.g. "<img src = *URL*>" is the HTML command to display an image found at *URL*). User input is supplied through a FORM tag which denotes (with another URL) what program to run and provides the means for sending it user input. Here the program is the CDIS query processor.

open standard that has encouraged the development of a great number of freely

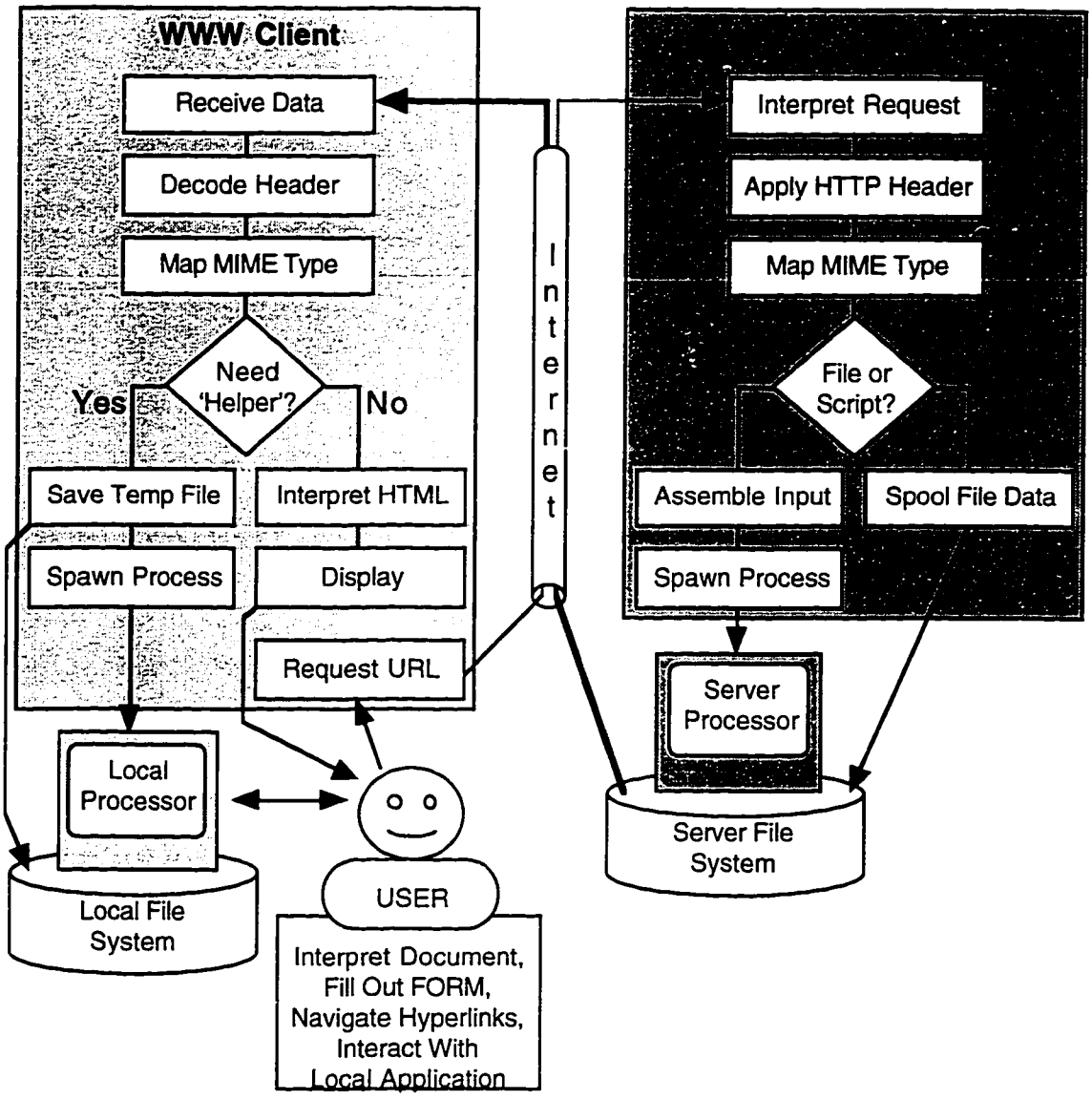


Figure 4.4 Client-Server interaction in the WWW. System is 'stateless' under normal operation, user does not maintain an active communication path to the server. Server can send any MIME-compliant data to be interpreted as necessary on the client side. HTML (along with some other data types) is interpreted within the client. HTML FORMS provides a non-interactive user interface to collect input to server-side applications (e.g. database queries).

distributed viewing applications (called WWW browsers). The HTML file which generated the display of Figure 4.2 is shown in Figure 4.3.

MIME (Multipurpose Internet Mail Extensions): MIME defines types of data to be

distributed over the network (originally as extensions to electronic mail messages). A mime type is sent from the HTTP server to the WWW browser so that the browser can direct the ensuing document to the correct application. As an example, a document labeled as "text/html" by the HTTP server is typically interpreted by the browser itself while one of the type "video/quicktime" would be sent to a user

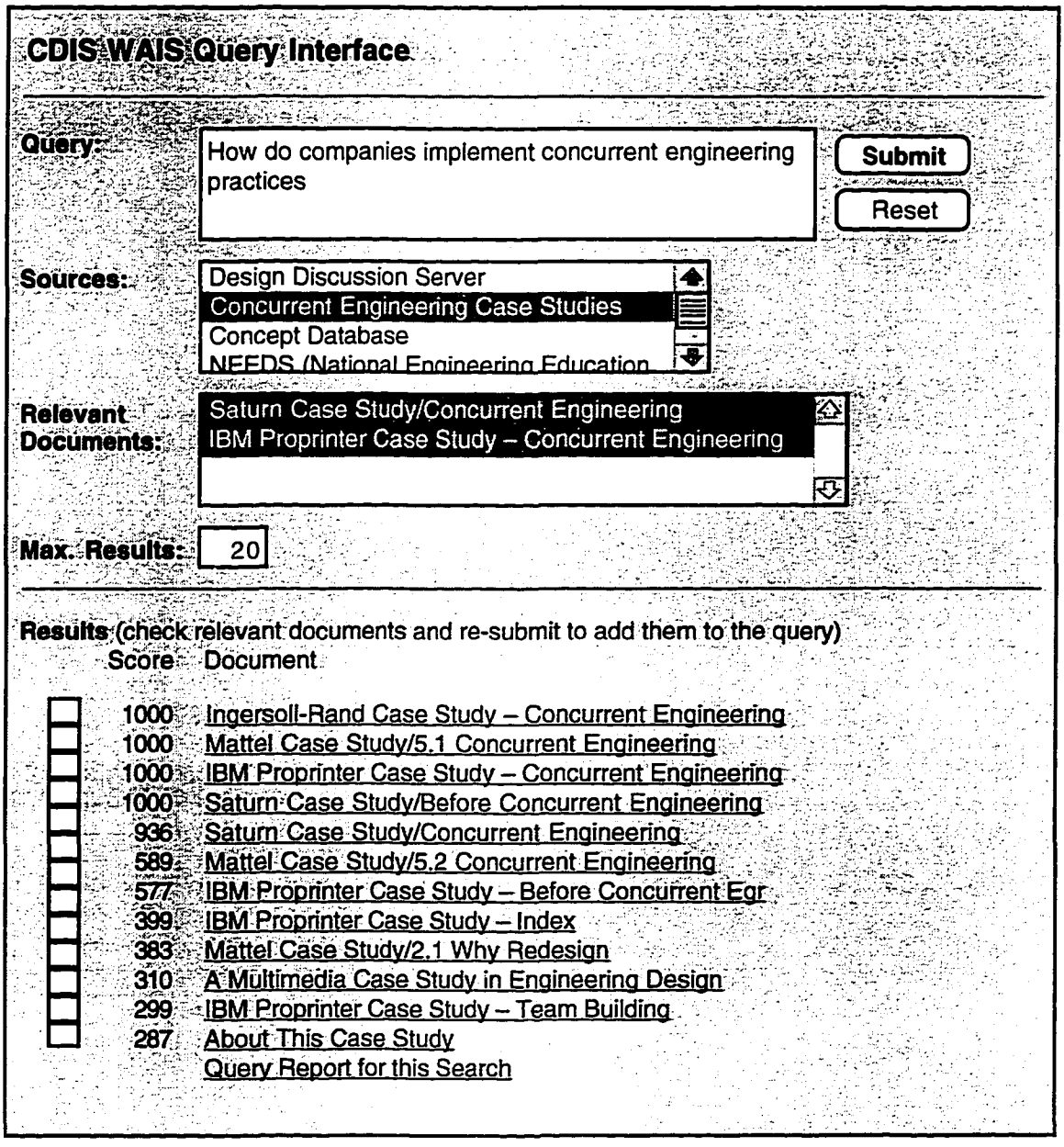


Figure 4.5 Typical user interface to WAIS. User enters a query, selects a source and the server(s) return a scored document set. In this case, two documents from a previous search have been added to the query as relevance feedback.

defined application that can play a Quicktime movie. While there are a limited number of predefined MIME document types, these are often extended using experimental types (e.g. x-application/autocad might be used for AutoCAD drawing files).

FORMS: The original functionality for gathering input from HTML documents was limited due to the initial scope of the HTTP definition. To enhance the functionality of HTTP servers, a small set of forms to be used for input were defined. These forms allow for selections from lists of options, radio buttons, and input of text. Many of the CDIS documents make use of these FORMS extensions.

Figure 4.4 demonstrates the typical operation of an HTTP client/server pair.

4.2.2 WAIS Server

Wide Area Information Service defines another protocol for serving documents in a networked environment. While HTTP servers are developed specifically to enable hypertext links among documents, WAIS servers are primarily used to provide for content-based searching over ASCII documents (really any format of document, but only ASCII content is currently available for indexing). WAIS implements basic information retrieval tasks, comparing terms from a user query to those distilled from a set of documents. Documents that share a larger proportion of the words from the query are given a higher score, the highest scoring documents are returned to the user as a list. As in HTTP servers, WAIS exploits MIME typing to match documents to applications. An example session with a WAIS server is shown in Figure 4.5.

As general purpose open architecture information retrieval engines, WAIS servers have become a prime candidate for on-line publishing services: Britannica Online [Vaulauskas, 1995] among others has been implemented within WAIS. The original architecture was developed within a joint project among Thinking Machines Inc., Apple Computer, Dow Jones, The Wall Street Journal and KPMG Peat Marwick as a means of delivering

information to financial managers [Kahle et al., 1993].

4.2.3 Relational Database

The relational database portion of the CDIS architecture is used for storing structured information. SQL is the de-facto standard for querying such databases, although the actual query interfaces should largely be transparent to users of the CDIS. The relational database is able to model the state of product models to be represented within the system. Its use with respect to three core applications will be more completely discussed in Chapter 5. Indexing based on deterministic matching, as done by SQL, will be discussed in Chapter 6 as one of the two main indexing schemes used within the CDIS. In general, the relational database system operates as a general purpose data store on top of which more complex database applications with WWW interfaces can be built. The user interface to the applications is accomplished through links forged between the relational database system and the hypermedia server.

4.3 CDIS System Integration

Each of the above components is capable of providing part of the infrastructure necessary to support the goals of the CDIS. The hypermedia server provides access to design information structured on a local navigational basis. The WAIS server allows access into this document base through an unstructured index based on the content of the document. In addition, the relational database can function as a data store for structured design information. The three core technologies mesh into a strong architecture for the CDIS. Chapter 5 discusses the three core applications implemented within this architecture, along with some extensions related to the CDIS.

Chapter 5

CDIS Applications

The CDIS infrastructure described in the previous chapter is the basis for three distinct CDIS applications: 1) Concurrent Engineering Case Studies present industry 'best practices' in the context of real design projects and real products; 2) The Design Discussion Server affords an asynchronous communication medium through which members of a design team can resolve the natural conflicts among issues important to each of the stakeholders in a project; 3) The Concept Database provides access to a set of modeling and evaluation 'templates' which can be applied in the selection of engineering components from a heterogeneous component database. These applications are differentiated by the nature and extent of the design information they represent. Case studies provide an overall context of design projects not only with respect to the artifact being designed, but also with respect to the design process and the project organization effective for implementing concurrent engineering. Design discussions follow more closely the process of defining and resolving design issues, capturing the design process in greater detail than that afforded in the case study format. The concept database provides the final link to the concrete, taking design issues and encoding them into evaluation metrics which drive the selection of specific components.

Taken together these three applications span a broad range of concerns in conceptual design. They embody case knowledge from the abstract to the concrete, from the process to

the product. Implemented in the WWW, the user interface to each of these applications enables easy linking among and within them. While each application deals with a different aspect of design, the integration of all three within the networked hypermedia server architecture outlined in Chapter 4 provides a common interface. The rest of this chapter deals with the implementation details and example content of the Concurrent Engineering Case Studies and the Design Discussion Server. Chapter 6 examines the Concept Database and the indexing issues that arise from the need to deal effectively with component abstraction.

5.1 Concurrent Engineering Case Studies

The first 'core' application of the CDIS to be considered is a set of hypermedia case studies of concurrent engineering as successfully applied in industry. Concurrent engineering – opening up the design process to include more input from manufacturing specialists – and the related life cycle engineering – modeling all design stakeholders as 'customers' – are simple concepts which have proven effective strategies for compressing time-to-market and increasing product quality. However, even simple concepts can be difficult to implement. Industry has fostered large, entrenched infrastructures which resist the application of concurrent and life cycle engineering principles. Organizationally, design departments are more closely related to marketing departments than to manufacturing. Each has different accountability and responsibility. This has led to a philosophy of separation rather than integration. Physical location is also a factor. Manufacturing engineers are rarely collocated with designers or marketers. This hampers communication, one of the primary precepts of concurrent engineering. Even greater physical (and contextual) space exists between those stakeholders that are in house – like marketing, design, and manufacturing – and those that are generally external – service, retirement, regulation, etc.

The concurrent engineering case studies not only present successful results of applying the concept, they present interviews with major participants along the engineering cycle. They demonstrate, through the eyes of the participants, the effective organization of

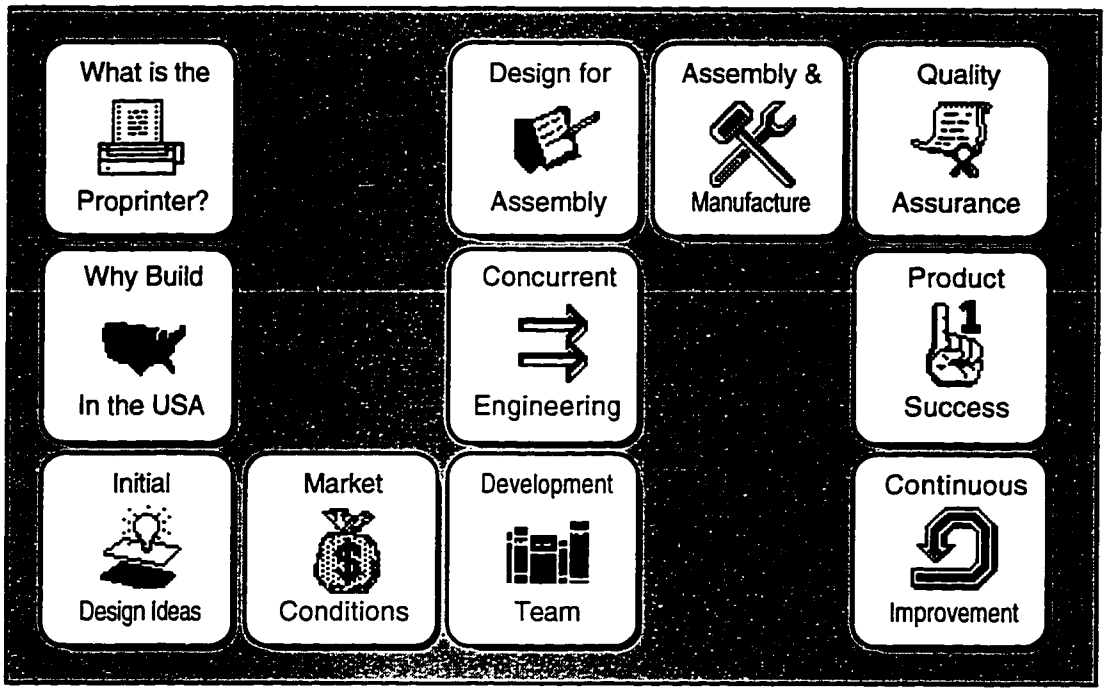


Figure 5.1 Navigational Backbone for the IBM Proprinter Case Study.

multifunctional teams (teams of engineers representing the major stakeholders in the design process – the makeup of these teams vary by product and industry). They also give overall background on the nature of the company doing the design and the market in which the product competes – this is the true design context.

From the process standpoint, the major decisions involved in the engineering of a successful product are reviewed. This does not approach the level of detail that might be required to redesign the product, but it does convey the overall development philosophy. Facets of the final design that represent this philosophy are highlighted as well. The general tenor of the case studies is centered on the context layer of representation, with abstractions of process and product representations mixed in to amplify the lessons to be learned.

5.1.1 Local Organization

Each case study is a standalone hypermedia presentation organized loosely around the product design cycle, as demonstrated by the main navigational screen from the IBM Proprinter Case Study, shown in Figure 5.1. This may seem ironic in that concurrent

engineering strives toward integration of the design-manufacture process. In practice the design cycle remains relatively unchanged although the participants and activities at each stage are transformed.

Other organizations exist within the case studies. Most contain a timeline of events in the engineering process. Some have brief personal descriptions of participants. Others contain historical or technical reference material. In general the design of these case studies errs on the side of structure in information to prevent the user from getting 'lost in hypermedia space'. This structure has helped to focus the content material presented within the case study. Each case contains over 50 hypermedia documents (similar to the 'lessons' in Archie III but collectively more numerous and broader in scope). A brief review of the design project and the concurrent engineering lessons from each case follows.

5.1.2 IBM Proprinter

This case study chronicles the development of a low cost printer designed to compete in the personal computer peripheral market [Agogino and Evans, 1993]. The project was IBM's first application of concurrent engineering in product development. The case study traces the project from marketing motivation – the Proprinter was seen as an acid test for IBM's ability to compete with overseas producers in high volume low cost computer manufacture – through multifunctional team formation, design, development, and manufacture. Major lessons include:

Design for Automated Assembly: To compete with lower labor costs overseas, IBM decided to develop the Proprinter for automated assembly. This quickly resulted in a motto adopted by the development team – "no screws, no belts, no springs" – because such parts are difficult for robots to manipulate. These parts were replaced, respectively, by molded-in snap fit fasteners, a helical lead screw for positioning the print head, and compliance elements molded into plastic components.

In addition to eliminating components that are difficult for robots to handle, other

constraints were placed on the design to ease assembly. The product is designed to be built up in a series of layers. Each assembly task 'approaches' the Proprinter from above, eliminating the need to manipulate the assembly fixturing. Each component snaps into place in such a way that force measurement is not required to determine if it is installed properly. Finally, the number of parts to be assembled is minimized.

Multifunctional Teams: In order to eliminate difficult to assemble components, manufacturing engineering played a large role in the multifunctional design team. This role was more active in the conceptual design phase where manufacturing engineers were required to commit to develop new processes for injection molding the lead screw and for handling carbon fiber introduced into plastic components to drain static charge from the paper handler. In turn, the mechanical design had to devise complex moldings to eliminate the components banned for poor robotic assembly behavior.

These are the two main ideas presented in the case study, but other lessons come through as well. While the Proprinter was designed for automated assembly, IBM has since gutted the several mile long flexible assembly plant in favor of a hundred foot long manual assembly line. The flexible assembly line required that components be placed in 'totes' so that the robot would be guaranteed precise knowledge of position and orientation. It turns out that the manual labor to place components in these totes is approximately the same as that required to assemble the components into a Proprinter. Additionally, the Proprinter was designed as a family of printers based around the same paper handling components. Such modular design techniques further reduce manufacturing costs. Finally, the success of the Proprinter has led to the spread of concurrent engineering techniques throughout IBM and to the company's manufacturing competitiveness (although overall success depends on good marketing decisions as well – an unintended lesson of the case study).

5.1.3 Mattel Toys

Mattel competes in a market segment vastly different from IBM, but has embraced concurrent engineering practices just the same. In the toy industry, marketing information is often inaccurate. Children (and the parents who actually buy the toys) have quickly changing tastes. Because of this, little development effort is expended on a product until its market success is assured – approximately the time it takes for the original injection molds to wear out. This case study focuses on the redesign of a product, the ColorSpin baby toy, in its transition from initial offering to core product status. In addition, lessons learned from other toy lines are integrated.

The ColorSpin was based on a toy that had proven successful in the Japanese market. Initially the design was licensed from the producer of this toy and altered to reduce its size and cost/price. With minimal consumption of Mattel's engineering resources, the ColorSpin went into production and was met with more than modest success. At this point, it was examined for redesign as a core product.

Design for Assembly: Success in the market proved the ColorSpin to be functionally well conceived. The case study traces the redesign of the product to reduce its cost. (The price was in the $10 - $15 range in the late 1980's.) Redesigning the ColorSpin to be more easily assembled was the main effort toward reducing manufacturing cost. Parts that had been slightly asymmetric were redesigned so that it is obvious which assembly orientation is correct. These improvements also helped reduce the number of different parts that make up a ColorSpin.

Part Count Reduction: This aspect of design is extremely important in low cost products like toys where the overhead associated with each part is amplified. Taking $.01 out of a Proprinter might not be considered worth the engineering time it might require. For a product produced in the millions each year, that penny can make a huge difference. Packaging costs are likewise amplified in toys, so the box that a toy comes in must be engineered for 'play value' as well. In the ColorSpin,

part count reduction was accomplished by: using more complex moldings to integrate gears into rotating components that share the same axis, exploiting symmetry to substitute a single part for previously left and right handed pairs, making the gear train from a uniform pitch, reducing the number of gears needed.

Component Sourcing: An important decision in any design is that of sourcing a component from an outside supplier or producing it in-house. In the ColorSpin case, the major functional components were not manufactured by Mattel. In order to reduce costs for the second generation, the redesign effort focused on these components. The internal mechanism was redesigned in technology available within Mattel – the precision of the gears and mechanisms used in the toy were relaxed to make injection molding production possible. This redesign also produced the largest impact in the part reduction effort.

Rapid Prototyping: Competitiveness in the toy industry relies heavily on the speed with which design concepts can enter production. New designs move quickly from artists concept through the model shop and into marketing focus groups. If customers respond favorably to the design concept, it goes into production. Time pressure results in minimal re-engineering of the design model for large scale manufacture. For this reason, Mattel integrates much of its production machinery into the model building process. Also, education of the model builders in design for manufacture produces more manufacturable models, reducing the re-engineering costs.

The ColorSpin case study emphasizes the application of concurrent engineering principles in a design process that accomplishes a successful balance of engineering development and rapid product realization. Lessons learned are based in the reality of the toy industry: effective communication among all stakeholders in a design project is extremely important, development of prototypes using production techniques (and actual manufacturing facilities) provides assurance that the design will be manufacturable, and design for

assembly is extremely important in low margin industries where assembly costs are the main variable in the profit/loss equation.

5.1.4 Saturn Automobiles

Concurrent engineering as presented in the Saturn Automobiles case study is adapted to the manufacture of high volume, high price products of considerable engineering complexity [Zook, 1994]. The Saturn project is an experiment in manufacturing efficiency under the constraints of unionized labor in the United States. As such, the case study presents the perspective of the original 'Group of 99' members. Multifunctional teams in the other case studies discussed here number on the order of five to twenty members; for Saturn, marketing, design, manufacturing, labor, and service disciplines were all represented. The entire automobile life cycle was the object of a major redesign effort, incorporating concurrent engineering concepts like:

Team Building and Partnerships: Central to the success of Saturn is the way in which the product development process was managed. Teams are omnipresent in the Saturn story – from the original 'Group of 99' to assembly teams to partnerships with component suppliers, sales organizations, and labor unions. Expending the effort to build such teams demonstrates the vital importance of communication in the engineering process. Design must take place in a larger context than the drafting studio. Interaction in teams of stakeholders improves the quality of design decisions.

Flexible Assembly: Saturns, like most compact cars, are offered with a choice of transmission type – automatic or manual. Out of the hundreds of components that make up each transmission, components common to the two might number in the tens. 'Just in Time' delivery of transmissions from their assembly line into the main assembly line requires that a mix of transmission be available. Saturn accomplished this mix by using a single, flexible assembly line for both transmissions. The two were designed with common fixturing points (which also eliminates different

transmission mounts in the car) and common locating points for the internal assemblies. Thus, final assembly can take place on the same line, enhancing plant space utilization and reducing the (quality reducing) inventory of 'work in progress' transmissions.

Part Count Reduction: In addition to the elimination of fasteners, Saturn employs a novel casting method, lost foam casting, that allows many formerly external fasteners to be integrated into the engine block. The same precision that allows for fastener integration also reduces the amount of finish machining required to finish the engine block.

Ergonomic Assembly Line: The Saturn production line is atypical in that assembly takes place on 'skillets' that adjust the height of the vehicle in response to the assembly operation being performed. Workers can ride these skillets while performing assembly tasks. This improves assembly quality because individual workers can optimize the flowing work environment in which they must perform.

The engineering concepts that do not fall into the above categorizations are as important as those that do. The Saturn case study emphasizes that design is a social activity which involves satisfying multiple views of design requirements. Bringing voices typically excluded from design into the process produces much better results for all.

5.1.5 Ingersoll-Rand Cyclone Grinder

This case study [Carlstrom, 1993] shares much with the other three. The main difference is in the type of product and market in which concurrent engineering product development was used. Grinders are industrial tools used in industry to clean welds, finish castings, or polish surfaces. The market is a mature one; increasing the market share of a product requires reducing the market share of another. In such competition, price and features are the major points of differentiation (assuming an equal amount of 'good will' among market competitors). Thus, this case study represents more emphasis on including the customer in

the concurrent engineering process than the other cases. Specific lessons include:

Design for Serviceability: The Cyclone grinder was redesigned using part count reduction and design for assembly techniques like those described above. In the Saturn case, design for serviceability amounts to color coding maintenance items in the engine compartment. Grinders require near-complete disassembly for servicing on a regular (weekly) basis. Thus design for serviceability rules for the Cyclone grinder are similar to design for assembly rules. The step of disassembly is an important change, though; snap fits like those used in the Proprinter are difficult to disassemble and wear out after few cycles. In addition to disassembly, part count reduction is highlighted; the new design has half the parts of the old one. The new design prevents serviceability problems by eliminating small parts and making most parts such that improper assembly is nearly impossible. More durable materials were also exploited to reduce the servicing requirements.

Customer Recognition: Ingersoll-Rand studied the chain of customers from the point the grinder is shipped form the factory until it is used on the shop floor. They recognized that a major group of distributors were the first customers, followed by purchasing managers, safety inspectors, and finally the end users. Each customer has specific, but different concerns that influence the design of the product, but the primary customer is still the end user.

Observation of Use: Every member of the concurrent engineering team was required to call on customers to better understand all of the various applications for the grinder. These visits pointed out major failings of the entire grinder market in terms of ergonomics. Grinders often had safety interlocks removed because they were difficult to activate with gloved hands and the constant pressure they required caused user fatigue. In addition, grinders were often modified by the application of tape to prevent the user's hand from slipping into the abrasive blade. Also, since grinders were uniformly made of metal they provided little thermal or shock

insulation to the user, again causing fatigue. None of these concerns had been voiced by the customer in typical marketing surveys; only through observation of the application environment did the team understand the importance of ergonomics in the grinder design.

Outside Help: Partially due to this recognition, Ingersoll-Rand also realized that it did not have the expertise in-house to quickly produce a set of design improvement concepts. They called in an industrial design company and within a month had settled on two prototypes – better designs and more quickly done than the 'engineering-based' design that Ingersoll-Rand had been using.

Supplier Involvement: The ergonomically improved design solution featured a plastic-steel composite housing unlike anything Ingersoll-Rand had manufactured in-house. As a result, component suppliers were brought into the project team to develop the Cyclone grinder design within the overall manufacturing scheme. This improved time-to-market and quality over the company's previous method of dealing with vendors.

This set of case studies represents the application of concurrent engineering techniques in widely varying market and organization contexts. In each, the ideas are adapted to specific applications and integrated into conceptual design. The message is clear: concurrent engineering is a vital tool for competitiveness in design. Another message also comes through: concurrent engineering experience is difficult to circumscribe. Many of the lessons outlined in the case studies are difficult (perhaps impossible) to represent adequately as structured rules.

Additional case studies are under development. A view of computer disk drive development outlines issues in precision manufacturing and highlights continuous improvement and quality. A more business oriented set of case studies includes sketches of three to four different manufacturing concerns and the secrets of their success from the standpoint of various members of a multidisciplinary team of managers and engineers. The Boeing 777

project, "the paperless plane", will demonstrate the additional impact of technology on concurrent engineering in very complex design and manufacturing operations. Small scale custom design is outlined in studies of the U.C. Berkeley Human Powered Vehicle and Solar Powered Car projects.

5.2 Design Discussion Server

The second core application of the CDIS, the design discussion server, provides an infrastructure through which members of a design team can collaborate by communicating through multimedia. The system has its roots in Rittel's IBIS concept that design is a negotiation or argumentation process. From its inception, a design project is broken down into issues that are to be resolved (the first issue often being that of deciding where/how to start the process). Proposals are offered to resolve these issues. Often these give rise to new issues which must be resolved. Eventually, conflicts among requirements are identified, tradeoffs negotiated, and decisions made. Because design issues are rarely isolated (decoupled), the discussion naturally takes on a 'tangled' aspect with discussion progressing on several fronts at once and focus changing in response to the results of decisions and evaluations.

Chapter 3 discusses several design systems based on this type of interaction. Here we will discuss a framework in which each can be accomplished. The design discussion server is organized around a relational database which stores information about users, issues, design comments, and transitions among comments. The structure of the relational database that supports the server is shown in Figure 5.2. Users of the discussion server are free to use predefined transitions from current comments to add to the design discussion. Entry into the discussion thread is through a hierarchical representation of comments and transitions. This implementation borrows heavily from that of gIBIS (graphical IBIS [Conklin and Begeman, 1988]) but is somewhat less graphical. A primary criticism of gIBIS is that while it provides a rich representation for capturing the design process, access is hindered by the inability to query these contents. In order to navigate a gIBIS design representation,

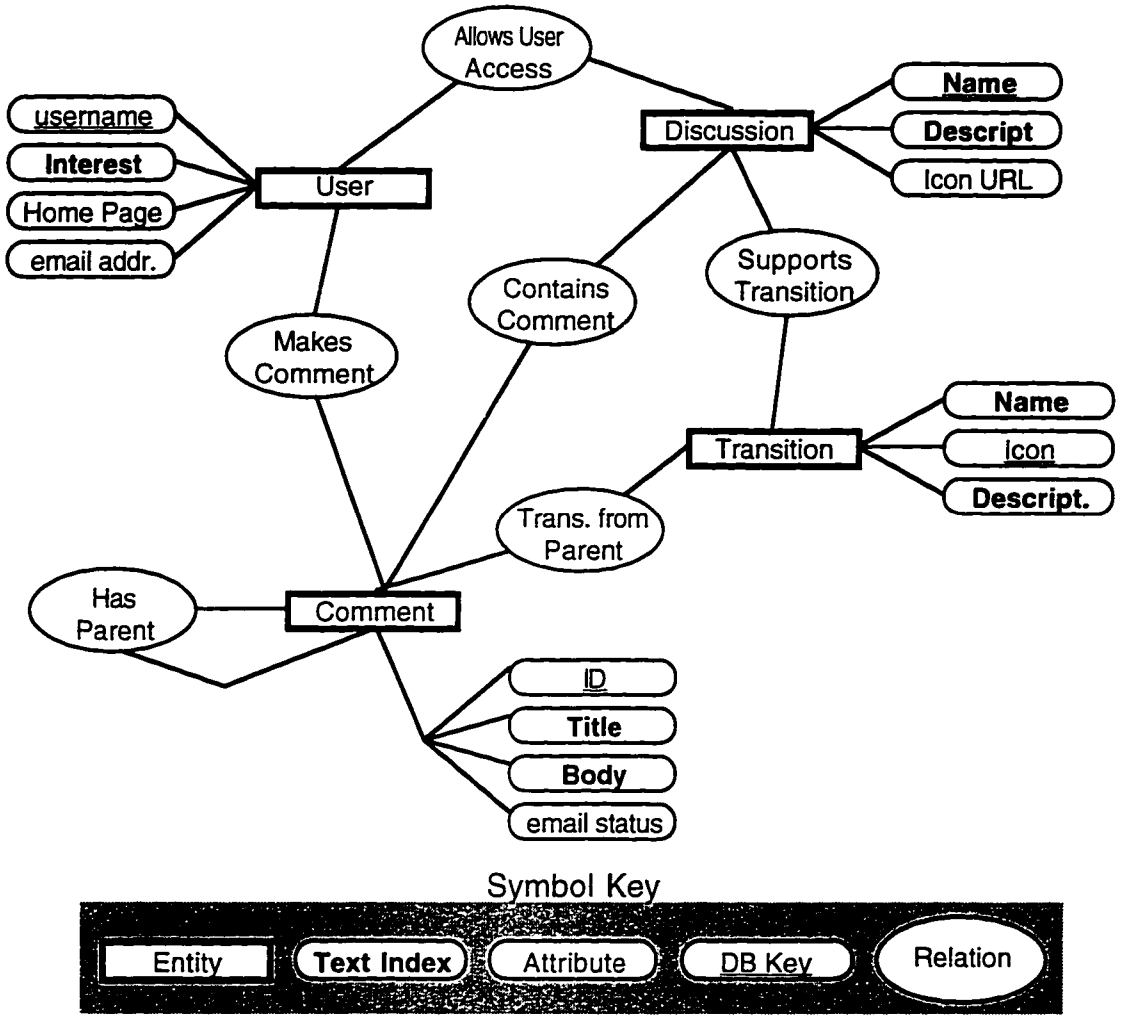


Figure 5.2 Entity Relation Diagram for the Design Discussion Server.

the user must be thoroughly familiar with the design. By supporting a structured representation in which argument trees contract and expand hierarchically, the design discussion server partitions design information so that specific threads (trees) are not obscured in the overall design discussion (forest). In addition, the CDIS provides additional access is through a design-tuned content index which will be described in Chapter 7.

The design discussion server provides a means of understanding the evolution of a design on an issue by issue basis. Discussants are free to enter their comments in hypertext form, referring to comments from other threads or documents anywhere on the World Wide Web.

The MADEFAST Seeker Design Scenario:

The test of success will be the ability to track a light through a range of motion. The light source will be a laser pointer or some other source with sufficient illumination. The source will be manually scanned in a random direction around a wall of a conference room. The demonstration will be held in a medium size lecture room (50x50—75x75). The tracker will need to track the source and display in some fashion (i.e., if using a visible light camera monitoring the video out of the camera should show the results of the centerline of the camera with the light source). Finished unit could rest on a small cart (computer for control could rest below). A second test will involve manually moving the tracker and tracking a moving light source. Open question remains about sustainable slew rates.

The tracker will acquire the light-source at a fixed point, and if the signal is lost the tracker should return to this home point for re-acquisition.

Low cost, short development time is key, as is flexibility. Overall system should be designed for reconfiguration. For example, redesign with additional space constraints, or with a new sensor that might change the weight and require some redesign of components, or for a larger production buy of 100 units. The design should be robust enough to allow these scenarios. Low cost, commercial components should be considered.

Add Comment

- ▼ Issue – MRC: What are typical optical tracking systems?
  - ▼ Position – LP: Pointer to a CCD System
    - Argument – SS: Our CCD Vision System
  - Position – SS: Other Video Tracking Systems – Non CCD
  - ▼ Position – RH: We have used PSD devices for light tracking
    - Argument – VDL: Robot arm tracking using PSD units
  - Position – VDL: How about human motion studies?
  - Position – JW: Ovation Cyclops pointer tracking computer mouse
  - ▼ Issue – GT: Error specs for sensor?
    - Position – MRC: PSD evaluation and testing program
  - ▼ Decision – MRC: PSD sensor for control input
    - ▼ Issue – What is the best optical design for the PSD system?
      - Position – JC: Using a lense-based system
      - ▼ Position – JW: Single mirror design
        - Argument – JW: Several arguments in favor of a single mirror option
        - Argument – GT: Inertia, shadow
        - Argument – JC: Single Mirror Aberration, etc.
        - Argument – JC: Aberration for a Parabolic Single Mirror
      - ▶ Position – JC: Two mirror design
    - ▶ Issue – GT: What will we use for demonstration
    - ▶ Issue – GT: What will we use for demonstration of tracking error?
  - ▶ Issue – VDL: Track pointer position, or laser dot?

Figure 5.3 Design Thread from Design Discussion Server. Vertical triangles denote nodes that have not been 'opened', horizontal ones show an 'open' thread. Underlined phrases are hyperlinks from the current page.

MADEFAST Seeker Design: Position – JC: Two Mirror Design

Issue – MRC: What are typical optical tracking systems?
Decision – MRC: PSD sensor for control input
Issue – What is the best optical design for the PSD system?
Position – JC: Two mirror design

Two mirrors can be used – the PSD and CCD can be mounted behind the primary mirror and see the source through a secondary mirror.

Design Configuration:
Primary spherical mirror with curvature C1.
Flat secondary mirror (curvature = C2 = 0)
Distance between mirrors = D
Back focal length (distance between the secondary mirror and the final focal point) = B

Equations:
C1 = (B-F)/(2DF)
C2 = (B+D-F)/(2DB)

These are the major equations, see the DesignSheet model for more information.

Design:
Considering that we are using 10mmx10mm PSD, and the view angle is about 5 degrees, we can select focal length of about 120 mm. That leaves us with B+D = 120 mm. We can take D = 60mm and B = 60 mm (we can make them bigger if focal length needs to be increased)

This leaves us with C1 = 1/240 i.e., radius of the primary mirror is 240 mm.

Aberrations:
Given the mirror diameter is about 80 mm, and the object is at infinity (a reasonable assumption), the total third order aberrations are as follows:

TSC = Transverse spherical aberration = 0.556 mm
CC = Coma = 0.139 mm
TAC = transverse Astigmatism = 0.035 mm
TPC = transverse Petzval curvature sum = 0.035 mm

Effective third order aberration (conservative) = 0.765 mm

Make a Comment Related to this one:

Issue Position Argument Decision

Current Descendants:
Issue – JC: Is the blur of 0.765 mm bad enough to ignore this option?

Figure 5.4 Design Comment from the Design Discussion Server. Comments are displayed in the context of their specific design 'thread'. Hyperlinks from comments are supported as are the various branching options for adding a comment 'below' this one.

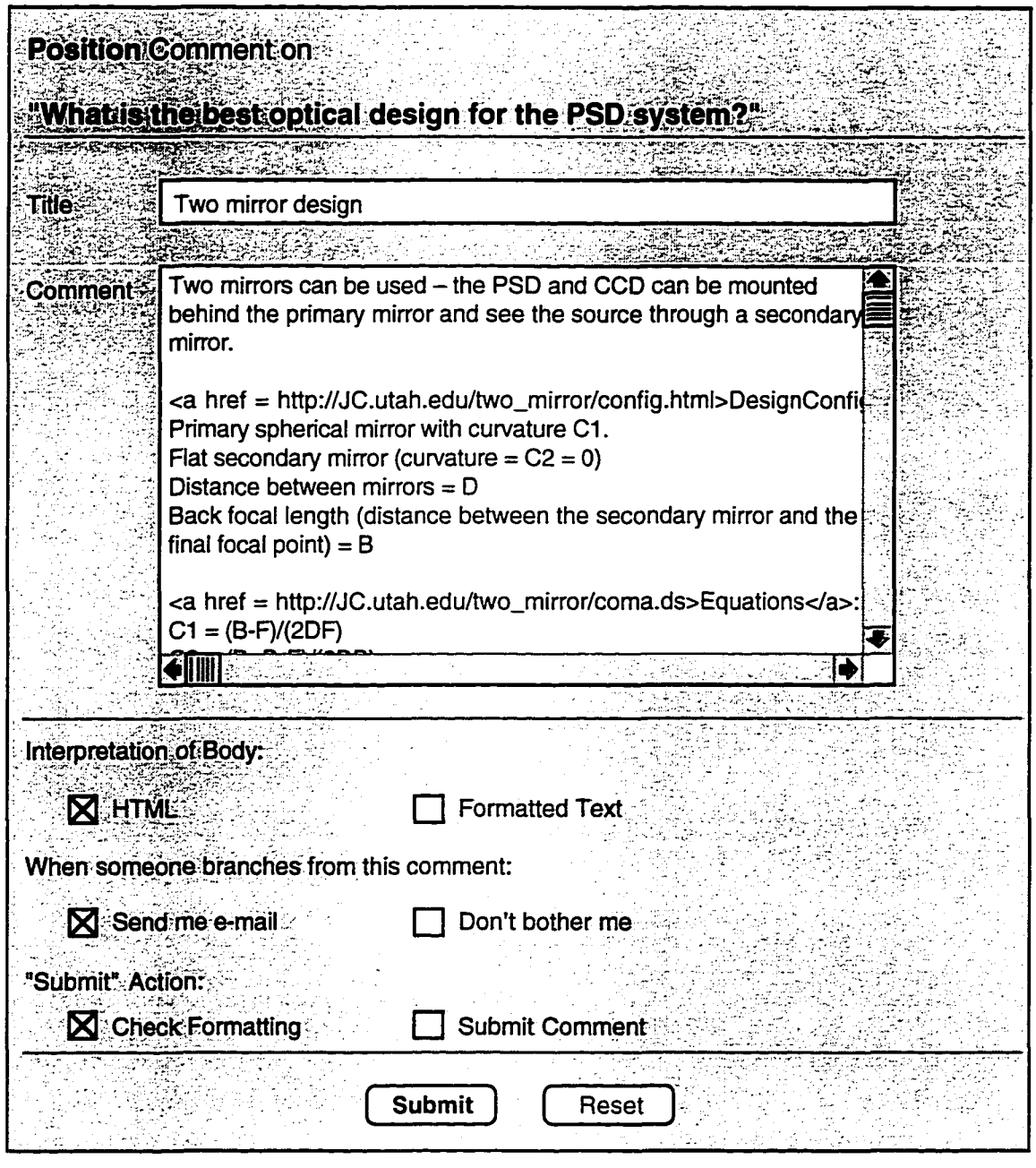


Figure 5.5 Design Discussion Server Comment Entry Form.

This flexibility allows the inclusion not only of textual arguments but also of analysis input, output , or traces. In addition, arguments can center on full multimedia representations of information. Confusion or controversy can possibly be averted through enriching the communication channel. Figure 5.3 demonstrates a typical discussion thread, Figure 5.4 shows the text of a comment and the transitions available from that comment, and Figure

5.5 shows the typical comment entry interface.

Good communication among multifunctional team members is paramount to the successful application of concurrent engineering principles. In the case studies discussed above, better communication is fostered by collocating team members – replacing disciplinary organization with task oriented organization. The design discussion server provides a hypermedia communication tool which fosters virtual collocation over the computer network. Also increasing the communication channel is the ability to mark a comment such that transitions executed from notify the author. This enhances discussion turnaround time while not forcing the resource consumption attributed to synchronous communication. Since the World Wide Web architecture is open to any MIME typed document (and effectively any document type the discussion group can agree upon), synchronous communication can also be supported (although logging of such activity is not supported).

The design discussion server improves on the generic computer news / bulletin board idiom within the implementation constraints of current network interfaces like electronic mail and the WWW. Enhancements include: increasing the communication bandwidth through hypermedia, supporting multiple hierarchical threads, providing more descriptive transitions among comments, and adding user definable argument 'e-mail paging' to keep discussants up to date on important threads. Because the much of the system information is stored in a relational database, future applications can query over many aspect of a design discussion. Customized reports can also be used as design documentation.

It is interesting to note that the design discussion server is but one variant of a common theme for WWW-based collaboration. Similar tools are being investigated in the education community as a potentially powerful means of fostering collaborative learning. As a progenitor of the design discussion server, the Multimedia Forum Kiosk (MFK) [Hoadley et al., 1995a&b] provides a more general-purpose discussion interface which offers transitions among comments based on conjunctions (e.g. AND, BUT, OR, etc.) which help identify the connection between arguments. Another general argumentation tool,

WebCAMILE [Turns et al., 1995], uses transitions isomorphic to those supported by the MFK system. WebCAMILE (Web Collaborative and Multimedia Interactive Learning Environment) has been deployed in a design education context (i.e. design project course) with mixed results. It did not foster the kind of design collaboration that had been hoped. Part of the difficulty involved the user interface, part a difficulty accessing the system through the network, part a failure to continually engage the project participants in the discussion process. By integrating the design discussion server within the daily work tools – especially electronic mail – and by providing an accurate search interface, the design discussion server should address some of the shortcomings of WebCaMILE.

The design discussion server must also be contrasted with other networked design tools. Because it strives to improve support for concurrent engineering, the tool is focused more on communication than actually on tasks involved in the act of 'designing'. To enter comments into the system, the user must have some knowledge of HTML. For those comments to be linked to design tools, the user must have some knowledge of how to address network objects (and must also have access to a WWW server on which to post the information). A system like PENS (Personal Electronic Notes System) [Hong et al., 1994], improves the WWW interface somewhat by providing a local interface for organizing design information which can then be posted to a server. Henry [Silva and Katz, 1993&95], a networked design notebook, integrates common VLSI design tools to organize a single user's design activity. SHARE [Toye et al., 1993] takes the similar metaphor of a design notebook. These efforts are seen as complimentary to discussion tools like the design discussion server because they focus on the part of design which is more individual. Availing such individual effort to the overall design discussion meshes well with the argumentation process that is the prevalent mode of interaction in the design discussion server. Integrating argumentation with personal reflection in the form of a design notebook can only better represent the design process. Implementation of both types of system in the WWW makes the strengths of each easily available through a simple URL.

5.3 Concept Database

As discussed above, design discussion inevitably leads to the need to establish concrete representations of design options and evaluate tradeoffs among them. The Concept Database integrates a structured representation of components and the mathematical equations used to model their behavior with a hypermedia database system used to select and combine these equations into usable models. In addition, component supplier databases are included to provide effective links from these models to real components. Finally, the Concept Database creates representations suitable for input to an advanced evaluation tool which is launched from the hypermedia interface.

The initial domain of emphasis for the Concept Database is electric motors. These components accomplish a wide variety of tasks and are ubiquitous in our modern environs (Donald Norman [1988] uses the number of motors one comes into contact with as a measure of the integration of technology into daily life). The wide variety of applications for motors renders their selection nontrivial; models used to evaluate the performance of motor depend on the specific needs of the application. Straightforward selection by speed or torque gives way to concerns like heat gain, position resolution, efficiency, and/or ripple torque as applications range from the mundane to the exotic. The database must respond with representations that recognize this variety in modeling abstraction (or accuracy). The next chapter provides a thorough discussion of the Concept Database, its implementation, and the indexing methods which bring component selection into case-based conceptual design.

5.4 Summary

This chapter has addressed the basic applications that have been implemented within the hypermedia-database infrastructure provided by the CDIS architecture. Case studies of the application of concurrent engineering techniques in industry demonstrate the need to provide loosely structured information to the conceptual designer. The design discussion server supports communication among teams of designers based on the well established examples of IBIS in design, and computer bulletin boards in general. In addition,

communication is augmented through the support of full hypermedia in the discussion process. Finally, the concept database offers a range of analytical support for the design decision process. Access is provided to catalogs of engineering components, sets of engineering relations, and fully developed analytical models.

These three core application of the CDIS provide a demonstration of case-based reasoning in a more flexible and powerful paradigm than those previously used in design. This approach has been taken to foster reasoning by derivational analogy in addition to the transformational analogy typical of case-based design systems. That much of the design process is difficult to represent as logical constructs leads to the varying representational bias used in the system. The concurrent engineering case studies are completely unbiased, allowing virtually any kind of information to be represented. The design discussion server places the bias toward IBIS-style argumentation on design information. Finally the Concept database allows the representation of design issues through mathematical models and that of components as sets of parameter-value pairs.

The CDIS architecture extends the navigational possibilities in typical WWW documents like those in the case studies by using a database to store user-modifiable connections among documents and to generate documents on demand. This provides a very powerful local indexing methodology for the CDIS. Additional applications involving the indexing and distribution of education courseware have been implemented in the CDIS architecture [Wood and Agogino, 1996]. The next chapter demonstrates a more complex database application – the Concept Database. In Chapter 7, the global indexing scheme used to integrate all of the design information resources of the CDIS is discussed in detail and its performance evaluated.

Chapter 6

Case-Based Reasoning for Component Selection in the CDIS

As discussed in Chapter 2, applications of case based reasoning to design have exploited parameterizations of example artifacts to provide models for engineering analysis where such models are absent or too costly to compute. While the argument of Chapter 3 is that these cases do not encode enough of the design context to support fully all stages of conceptual design, they provide powerful tools for prototype selection and evaluation during uncertain and ambiguous stages of conceptual design. This chapter focuses on two different ways in which artifacts can be treated as cases in the context of the CDIS. In the first, we try to capture the selection process by storing cases of component selection and evaluation within the Concept Database. The Concept Database bridges the space between the formal mathematical models used for design evaluation, the component catalogs which provide the design space for selection, and informal descriptions of the design strategies involved in the selection process. Indexing the informal aspects of information stored within the Concept Database is left to the next chapter; this chapter presents the way in which formal models are represented and indexed to provide a framework for arbitrarily many design 'templates' in the domain of electric motor selection.

The latter portion of the chapter presents methods for using the performance parameters cataloged for each motor to create a design space which at once supports the ambiguity necessary for conceptual design along with useful strategies for mapping this ambiguity to

uncertainty toward exploiting normative methods for conceptual design. IRTD has been previously proposed as a normative method for design based on decision and information value theory. We couple this methodology with the knowledge to structure the component database as well as knowledge derived from its contents. Because conceptual design involves the active manipulation of the representational abstraction of an artifact, we present a formal method for deciding the appropriate level of abstraction based on an expressed (possibly very loosely expressed) set of objectives and constraints. Before we lay out the case-based reasoning implementation within the CDIS, it is appropriate to discuss design representation in conceptual design and it's impact on the conceptual design process.

6.1 Modeling is an Engineering Decision

A major focus of engineering is the creation and manipulation of models for 'real world' behavior. It is a primary task of engineering to reduce the cost of development by being able to predict accurately (enough) the behavior of artifacts that have not yet been constructed. Petroski [1994] traces the evolution of structural models in engineering design in a generate-and-test paradigm where a model is proposed, repeatedly applied, and accepted after success. As this model is applied more aggressively (i.e., the applied safety factor is reduced) imperfections in it inevitably produce failure. Once failure occurs in the model, two courses of action are available: acknowledge the imperfection of the model and continue to use it with an increase in the factor of safety or undertake to improve the model (at an unknown cost). Which of these courses should be followed is an engineering decision based on the economics of developing new models and those of sacrificing efficiency with a less complete or accurate model. Such modeling decisions are routinely made outside of the bounds of failure; the development of newer and possibly more accurate models is a prime activity in engineering research.

In conceptual design, the level of abstraction of design specifications often overwhelms the accuracy of existing models or precludes the use of more accurate ones. Modeling must adapt to the level of abstraction demanded by the current state of information in the

problem. This problem is isomorphic to that typical of diagnostic systems where enough information is applied to a problem to be able to reach a conclusion. In decision theoretic systems this conclusion has with it an associated value, possible improvements in which must be traded against the cost of acquiring more information.

Automated model selection is applied in the Design Modeling Environment (DME) [Iwasaki and Levy, 1993], part of the 'How things Work' project [Fikes et al., 1991], which supports variable abstraction in modeling component behavior. In this case, modeling is directed toward explaining the behavior of a system in a diagnostic framework. DME operates over a 'scenario' (an observation of current system operation), chaining 'facts' about how the system is supposed to function together by using 'rules' about how components normally operate and how they might fail. In doing so, it selects models based on the data 'facts' presented to it and not necessarily on whether their inclusion provides a better model of system behavior. Operating in a diagnostic mode under the influence of several domain models of varying abstraction, DME chooses the level of abstraction that is appropriate to the given set of facts and produces an explanation which is qualified by the possible application of more information. The facts themselves determine the level of abstraction for the explanation; in response to this explanation, the user can add facts to the current scenario in an effort to improve the explanation.

Conceptual design demands a synthetic process, not a diagnostic one as embodied by DME. In order to synthesize a design solution, it is necessary to provide a model of objectives along with multiple models of solution behavior. These objectives are typically not as straightforward as 'tell me what's going on in the system'; neither are they likely to be as static. The selection of modeling abstraction must match the abstraction of the design specification and evolve with it, helping to transform representations of artifact, objective and constraints from the ambiguous or uncertain to the operational functions which can support formal design evaluation. Simply offering a number of representational abstractions is not enough to ensure that conceptual design activities are properly supported. It is also necessary to introduce guidance as to when and how to change

abstraction level in the design process.

An important tenet of conceptual design is to "preserve ambiguity"[Leifer, 1994]; that is, to not make unconsidered decisions without realizing the tradeoff between design decisions and the need for later design changes. Too often, designers commit to concrete design representations too early in the conceptual design process simply because they are much easier to deal with. One can't easily look up a 'motor' to find out typical performance or physical characteristics. It is much easier to instead decide on a 'DC Brushless, Rare Earth, Servomotor' because *that* can be looked up in a catalog and manipulated with a set of standard design functions and evaluated. Many discuss weaknesses that result from prematurely reducing ambiguity. In Suh's [1990] treatise on engineering design, it is recommended that under a change in the design specification (which is the constant state in conceptual design) *all* design decisions must be revisited to assure compliance with his design axiom. A measured approach of incrementally establishing design requirements is more efficient than jumping to conclusions since small perturbations in requirement space can result in very large perturbations in design space. Ward et al. [1994] report on Toyota's successful application of this 'least commitment' principle in the case of the designation of 'hard points' in automotive body design. Toyota carries much more ambiguity (in the form of spatial intervals for various component locations on a car body) later into the design process than any other car manufacturer and makes significant gains in reducing production engineering change orders in the process. The question that is difficult to resolve is: When is the representational efficiency of making such a possibly premature decision to reduce the design space more valuable than dealing with less operational models in a larger design space. Here we take a stab at formalizing this question in the framework of IRTD.

Intelligent Real-Time Design (IRTD) recognizes that uncertainty in any system is reduced at a cost. It is the job of an engineer to balance this cost against possible improvements better information[1] might foster in the design. In the IRTD framework, design is cast as an information gathering activity starting with relative uncertainty about specifications,

[1] Better from the standpoint of IRTD is the information value concept of perfect information – knowing exact numerical values for design parameters.

continuing through rough models for preliminary concept generation, iterating to define more completely both problem and solution, and finally selecting from among a small set of candidate solutions. At each step, IRTD directs this information gathering by determining what information has the most expected impact on the overall objective (which includes improving the model of the objective itself [Bradley 1993]). First, we discuss support for traditional engineering models in the CDIS as design cases that can be reused. It is intended that these cases themselves embody IRTD methods with respect to the selection of components from catalogs. Once the traditional model for catalog selection is established we then take a step toward automating conceptual design using case-based design models and IRTD.

6.2 Structured Indices for Identifying and Reusing Relevant Design Instances

Relational databases provide a means of establishing and representing information about the relationships of entities, in this case information used to select and model engineering components. Because the database used in the CDIS is a commercial relational database (Sybase), we will not discuss the indexing functionality internal to the database itself. Instead, focus is directed toward the mechanisms encoded within the database relations to support the multiple levels of abstraction needed to effectively model decision making in conceptual design.

Much of the art in engineering is in deciding at what level of abstraction a problem should be modeled to balance most effectively the need for a solution with the resources available for arriving at that solution. Certainly one of the most effective resources available to an engineer is direct case experience. Chapter 2 discusses various AI-based systems in which representational abstraction plays a major role: the choice model representation determines the conceptual bounds in which a case-based system operates. Of the case-based design systems discussed, Scheme Builder[Chaplin et al., 1994] provides the most varied representation of case information. Scheme Builder allows for components to be represented at three abstraction levels: simplified single or two-port dynamic models, more

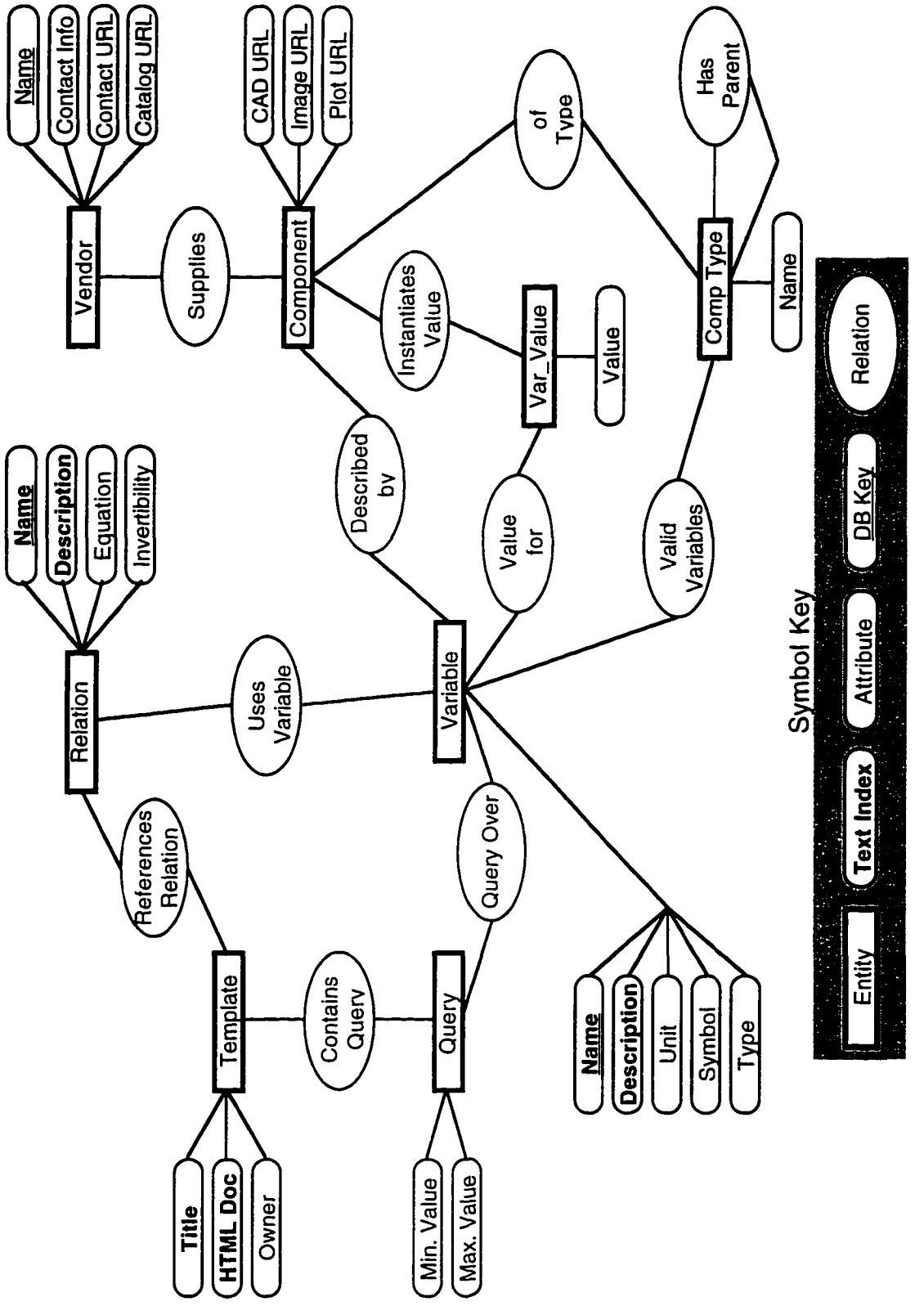


Figure 6.1 Entity-relation diagram for the Concept Database.

more complete subsystem models, or its physical embodiment.

The Concept Database offers greater diversity of representation than is available in Scheme Builder and the design process and analysis tools used for design evaluation differ greatly. Scheme Builder is concerned with dynamic systems, constructing designs by recalling and linking together stored MATLAB models for dynamic systems based on simple 'black box' input/output relationships defined by the user. The result is an AutoCAD representation of the design produced by conjoining geometric analogues to the theoretical models used in the construction phase. Evaluation is still left largely to the user. Unlike Scheme Builder, the Concept Database is not tied to a specific analysis tool or design domain; although it currently supports Design Sheet [Buckley et al., 1992], the generic mathematical expressions used to encode design models can be used by virtually any 'solver'. The Concept Database also supports more levels of abstraction in representation through component type abstraction hierarchies derived from the operating principles and performance parameters of components. The main concern is how to store models of design decision-making activity in a way that is modular enough to offer the opportunity for reuse and can be indexed so that relevant design cases can be identified. The idea is to provide a system that learns from user activity in component selection, collecting cases of design activity which can then be reapplied in a manner similar to Howard et al.'s DDIS [1989]: loosely structured categorizing based on abstract design requirements preprocesses activity toward set 'design plans' which can then be executed as an expert system.

For the Concept Database, design plans are called *templates*: component selection cases which embody constraints and objectives used in past designs, indexed formally by the relations and variables used in queries and analysis and informally by textual descriptions of the template application, relations, and variables. Informal indexing is accomplished using methods described in the next chapter. The database design that supports the structured design case indexing in the Concept Database is now outlined.

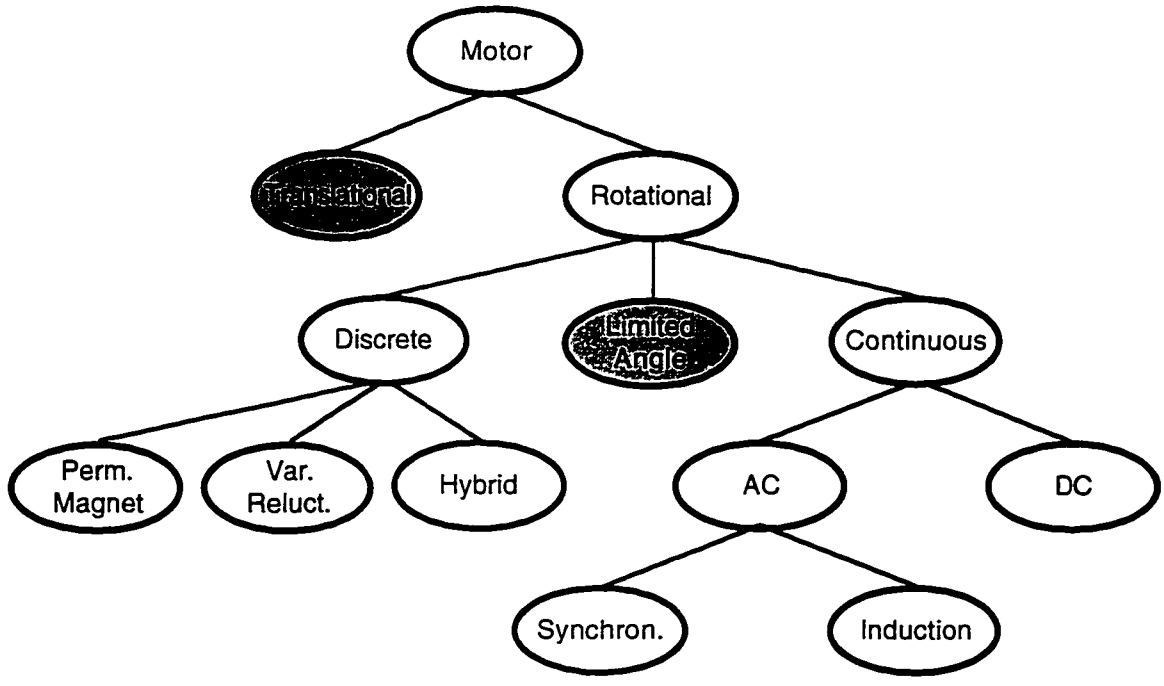


Figure 6.2 Component type hierarchy based on operating principles for subsets of the class 'Motor'. Shaded nodes are not expanded.

6.2.1 Indexing Templates as Design Cases

Figure 6.1 shows the entity-relation diagram for the Concept Database. Central to indexing abstraction level within the database is the association of various entities (variables, relations) within a hierarchy of component types. This hierarchy establishes the scope of any particular variable or relation through simple inheritance – as the hierarchy is descended, the set of valid variables and relations grows with the addition of those connected to each node. In the example case of motors, a partial hierarchy is shown in Figure 6.2 with concomitant variable and relation mappings exemplified in Table 6.1. Geometric quantities like diameter, length, shaft size, etc. are germane to all motor types because all are physical objects. Some performance parameters like efficiency are common to all motor components as well; in this case, the relations used to derive this quantity (from motor attributes variables) differ among motor classes. In addition, multiple relations for calculating the same variable can be defined at various levels of abstraction; the set of variables used in the calculation changing to provide more accurate models at less abstract component types. Classification variables are also used to manipulate the abstraction level

Component Type	Catalog Variables	Performance Variables	Classification Variables
Motor	Dimensions, Weight	Heat Generation	Frame Type
Translation...			
Rotation	Max Torque		
Limited Angle ...	Max Angle, Max Torque		Winding Type
Discrete	Pull-in Torque, Pulse Rate		
Permanent Mag.			
Var. Reluct.			
Hybrid			
Continuous	Stall Torque	Efficiency, Power	
AC			
Induction			
Synchronous			
DC	Max Cogging Torque, No Load Speed, etc.	Ripple Torque	Field Excitation, Commutation

Table 6.1 Variable assignments to component type hierarchy for 'Motor', shown in Figure 6.2.

of a component. The main distinction between classification made in component type hierarchies and those made using classification variables is that while the former describes the operating principles of the component type and thus determines which variables and relations are valid, the latter describe implementational differences which divide the design space. For instance, the CDIS component types do not distinguish between frame or frameless motors since all relations are valid for both. However, design issues represented in the objective or constraints might favor all frameless motors over framed ones (e.g. if there is a strong preference for compact design dimensions). So, while it is useful to divide the design space among operating principles of components, it is likewise useful to apply implementational distinctions so that design focus can match the abstraction level necessary at any point during conceptual design.

Within the Concept Database, system relations are defined which can be used to derive all of the common variable quantities. The system strives to provide all of the necessary building blocks for design synthesis and evaluation. By organizing these relations along the component type hierarchies, a set of abstractions is defined which lends conceptual structure to the variables and relations from which user-defined analysis models are compiled. These sets of system relations define the basic design issues related to the

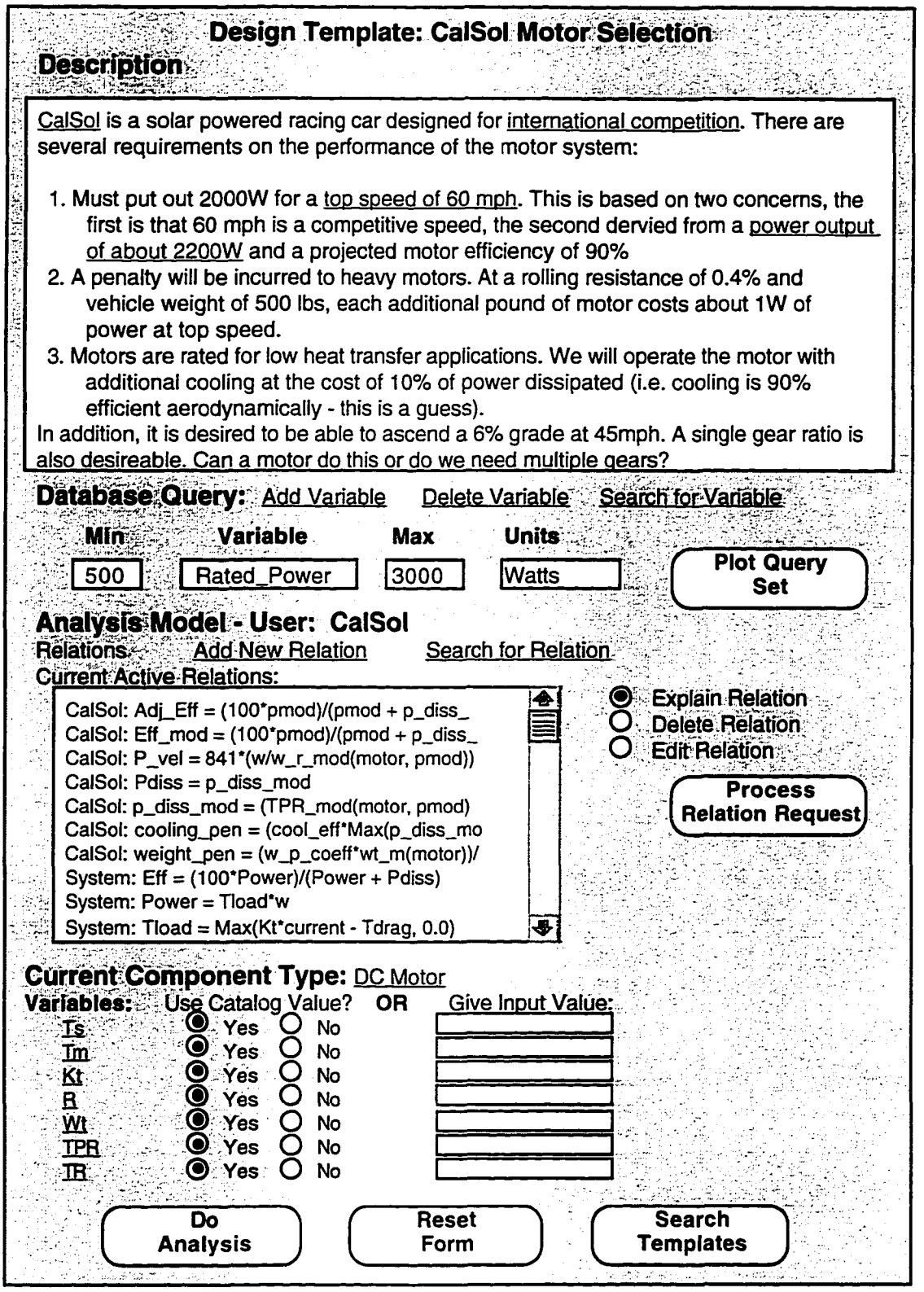


Figure 6.3 Design template of motor selection for a solar car. Three main sections are Description, Query and Model. Note hyperlinks throughout the document (shown as underscored text or buttons).

evaluation of a component. Their composition into a model in response to the goals of a particular design instance (called a template in the Concept Database) provides a means of capturing and storing the design rationale so that it can be reused in similar design instances. It is the job of the database to provide indices by which similar cases can be identified.

Figure 6.3 shows a Concept Database template, a case embedded in a hypertext web including a contextual description, a set of engineering relations over a predefined set of variables, and a query to be performed over the database of catalog components. This shows a vital link between the side of design that is difficult to structure – descriptions of design intent and its link to the decision making rationale – and the side that is more easily structured – sets of equations used to evaluate design alternatives and mathematical descriptions of the alternatives themselves. Entering the Concept Database template through an unstructured query (perhaps including an abstract design description of issues and motivations) leads to grounding the user in more structured representations. The means of accomplishing this type of informal mapping is discussed in the next chapter.

Here we discuss users entering the system in a more structured manner, perhaps building a model for a new design problem. Users can query over designs using similar mathematical relationships and perhaps gain insight into the ultimate usefulness of such relationships by browsing over models that apply them. By providing access to documentation in forms that are useful within these separate motivations, the Concept Database supports design exploration from first principles as well as case-based reasoning over the actual design models used to produce a 'case' that is espoused so strongly by Petroski [1994]. These design models are presented in context with statements of simplifying assumptions about the design problem and the rationale that lead to such assumptions.

6.2.2 Design of the Concept Database

Having discussed the link between structured and unstructured sides of the Concept Database, it is useful to identify the types of queries that the E-R diagram of Figure 6.1 is

designed to support. The entities within the database and the relationships that structure them and support different query modes are as follows:

First the analysis side:

Variables: A variable represents some parameter of performance or a descriptive attribute of an engineering component. Variables are separated by type for various purposes within the Concept Database:

Classification Variables: These are distinguished from other variables in that by instantiating a classification variable, broad classes of components are eliminated. Classification variables are significant in that they can be applied in various orders and combined with the component type to set the abstraction level of a component. For example, whether a motor comes with a frame or as frameless components is an important design consideration which is separate from the operating principles which are encoded in the component classes. Classification variables are discrete; some examples are frame type, commutation type, magnet type.

Catalog Variables: These are measured attributes by which the performance or physical dimensions of a component are described. The set of catalog variables for a component type defines the part of the set of queriable features for the component. Examples for the motor domain are: outside diameter, peak torque, rated power, etc.

Performance Variables: These are variables that are defined as being standard descriptions of operation or measures of performance for a component. Relations defined as system relations can derive performance variables from combinations of catalog and performance variables. A simple performance variable might be motor speed. From it and appropriate catalog variables, other performance variables such as continuous torque output, motor efficiency, or

drag torque can be derived.

User Variables: These are similar to performance variables but are non-standard. The component type classification of user variables is inferred to be that of the least abstract type that 'covers' all of the variables used to derive it (possibly in multiple relations).

Relations: Relations describe mathematical operations over the set of variables. They can contain any mix of performance parameters and physical attributes, the main purpose being to map from motor attributes to performance parameter. The database stores a description of the relation along with a property associated with the invertiblity of the relation – needed for efficiency in the modeling package chosen for the demonstration system.

Queries: Queries provide a means for establishing basic set of component alternatives. A query is defined by the attributes of interest within the design and a set of attributes value ranges which constrain the result set. This result set becomes a lookup table that relates components and their descriptive attributes for evaluation.

Templates: A template is essentially a small design case or design plan. It is modified by a description of the particular design instance, i.e. a hypermedia document that might be linked to a design discussion (upstream links) and some design evaluation documents from the latter parts of the life cycle (downstream links). This description is designed to contextualize the template within the overall design web since individual component selection instances hardly make up a complete design case. Templates contain a query to provide a set of components to form the basis for decision making and a model used to analyze tradeoffs among alternatives. It is interesting to note that as the underlying component database changes, the results of using the template might also change.

Now the component side:

Component: A component is an engineering entity to be evaluated as a single device. It could be as simple as a screw or a complicated as a larger assembly of components (like a motor or more complex subsystem). Components are treated as discrete objects in this sense, described by a set of variable (attribute) pairs which are the basis for the queries and lookup tables described above. In addition an identification number, supplier, and hypertext links to more complete information like a CAD file or catalog page are also stored.

Component Class: Components are grouped into component classes which are hierarchically defined as shown in Figure 6.2. Inheritance of relations, variables, and classification variables are implemented through standard queries. Note that component class typically deals with general operating principles of a device and as such is not arbitrary. Component class is sufficient for defining the variables over which database queries can be made, more abstract types providing fewer queriable variables.

Relating Analysis to Components:

Component Class - Variable: A mapping is made between a component class and the variables that are germane to that class. The total set of variables that are valid within a component class are those related directly to that class augmented by those related to any superior class.

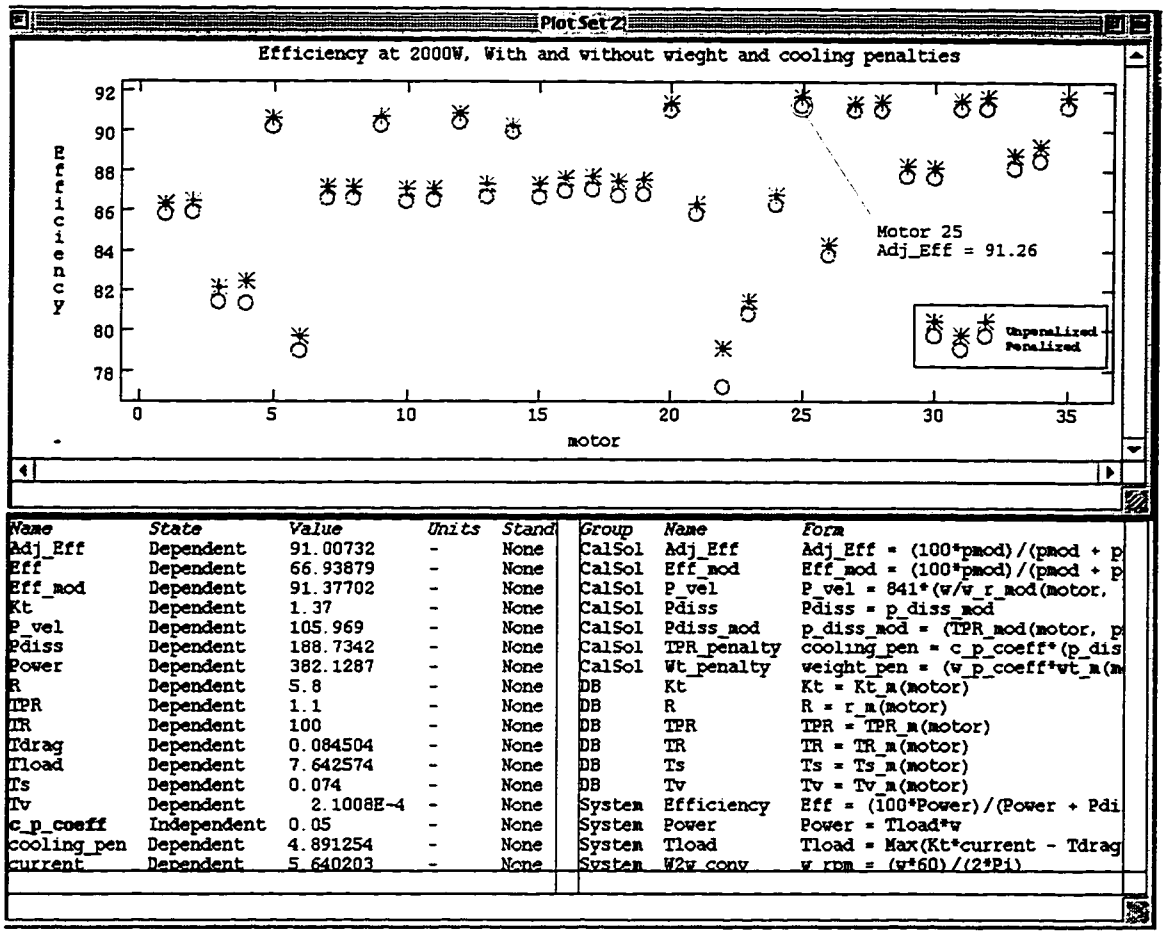
Component Class - Relation, Query, Template: No explicit mapping is made among relations, models, queries, or templates and the hierarchy of component classes. Instead, this mapping is accomplished through the variables that constitute these entities. A query on relations valid for a component class is made simply by finding all of the relations which contain only variables valid within that class. Likewise, the other entities can be related to the component class hierarchy.

Abstraction within the Concept Database is directly encoded within the component type hierarchy. Each level of abstraction is made over the variables that are valid at that point in the hierarchy. For instance, at the top of the hierarchy, a motor can be described as an object with physical dimension, weight, and a rated power. Descending to the DC motor level, the model can be enriched by adding information about speed - torque relationships, decreasing the level of abstraction. Rotor inertial effects can be used to study coupling with the driven system. Electrical quantities like inductance can be used to evaluate electronic drive circuitry. In each of these cases, a specific design decision drives the representational level of the motor. The hierarchy shown in figure 6.2 was developed to describe motors by operating principle. This bias is important for determining which sets of relations can be applied, but may not reduce the design space effectively based on a specific design objective. In this case, the abstraction differences offered by the classifying variables can help to eliminate unnecessary descent (with the analogous commitment to less abstraction in component type) The hierarchy depends largely on discrete, classifying variables like brush vs. brushless, rare-earth vs. AlNiCo, or frame vs. frameless, their order shaping the resulting hierarchy.

6.2.3 Linking Design Templates to Analysis Tools

A design template consists of three main sections: a description of the design context for the template, a set of relations which compose a mathematical model to represent this context, and a set of intervals defined over catalog variables which serve to reduce the design space. Templates contribute to design evaluation mainly through this latter screening process which is used to define a discrete set of alternatives whose suitability can be assessed using an external application. In our prototype this application is Design Sheet, a general purpose symbolic and numerical solver capable of operating over very large design models.

Let us consider the case of the prototype template shown in Figure 6.3 above. The application in question is the construction of a solar powered racing car to be used in international competition. Because such vehicles have limited power generation and storage



Name	State	Value	Units	Stand
Adj_Eff	Dependent	91.00732	-	None
Eff	Dependent	66.93879	-	None
Eff_mod	Dependent	91.37702	-	None
Kt	Dependent	1.37	-	None
P_vel	Dependent	105.969	-	None
Pdiss	Dependent	188.7342	-	None
Power	Dependent	382.1287	-	None
R	Dependent	5.8	-	None
TPR	Dependent	1.1	-	None
TR	Dependent	100	-	None
Tdrag	Dependent	0.084504	-	None
Tload	Dependent	7.642574	-	None
Ts	Dependent	0.074	-	None
Tv	Dependent	2.1008E-4	-	None
c_p_coeff	Independent	0.05	-	None
cooling_pen	Dependent	4.891254	-	None
current	Dependent	5.640203	-	None

Group	Name	Form
CalSol	Adj_Eff	Adj_Eff = (100*pmod)/(pmod + p
CalSol	Eff_mod	Eff_mod = (100*pmod)/(pmod + p
CalSol	P_vel	P_vel = 841*(w/w_r_mod(motor,
CalSol	Pdiss	Pdiss = p_diss_mod
CalSol	Pdiss_mod	p_diss_mod = (TPR_mod(motor, p
CalSol	TPR_penalty	cooling_pen = c_p_coeff*(p_dis
CalSol	Wt_penalty	weight_pen = (w_p_coeff*wt_m(m
DB	Kt	Kt = Kt_m(motor)
DB	R	R = r_m(motor)
DB	TPR	TPR = TPR_m(motor)
DB	TR	TR = TR_m(motor)
DB	Ts	Ts = Ts_m(motor)
DB	Tv	Tv = Tv_m(motor)
System	Efficiency	Eff = (100*Power)/(Power + Pdi
System	Power	Power = Tload*w
System	Tload	Tload = Max(Kt*current - Tdrag
System	W2w_conv	w_rpm = (w*60)/(2*Pi)

Figure 6.4 Design Sheet session using design model from the template shown in Figure 6.3. Here, a motor is selected from candidates based on an objective function maximizing efficiency at an output of 2000 Watts. Penalties are applied to motors for power dissipated by the car due to carrying extra weight and providing additional cooling. Circles denote adjusted values, asterisks the 'raw' performance of the motor.

facilities, efficiency is the main concern in the design template. Weight is also of concern because a heavier car has increased rolling resistance which becomes significant at high speeds. In the model used here, a 1 Watt power dissipation penalty is exacted for each pound of motor weight. To help reduce motor weight, the team considers using additional cooling which can be supplied to motors with a smaller nominal rating than the 2000 Watts necessary to achieve 60 mph. Such cooling can boost the performance rating of the motor but also exacts a power dissipation cost to reject the additional heat. Here, 10% of the necessary additional heat rejection is applied to the power dissipation of the motor. These relations are all encoded in the template of Figure 6.3 as custom relations specific to the

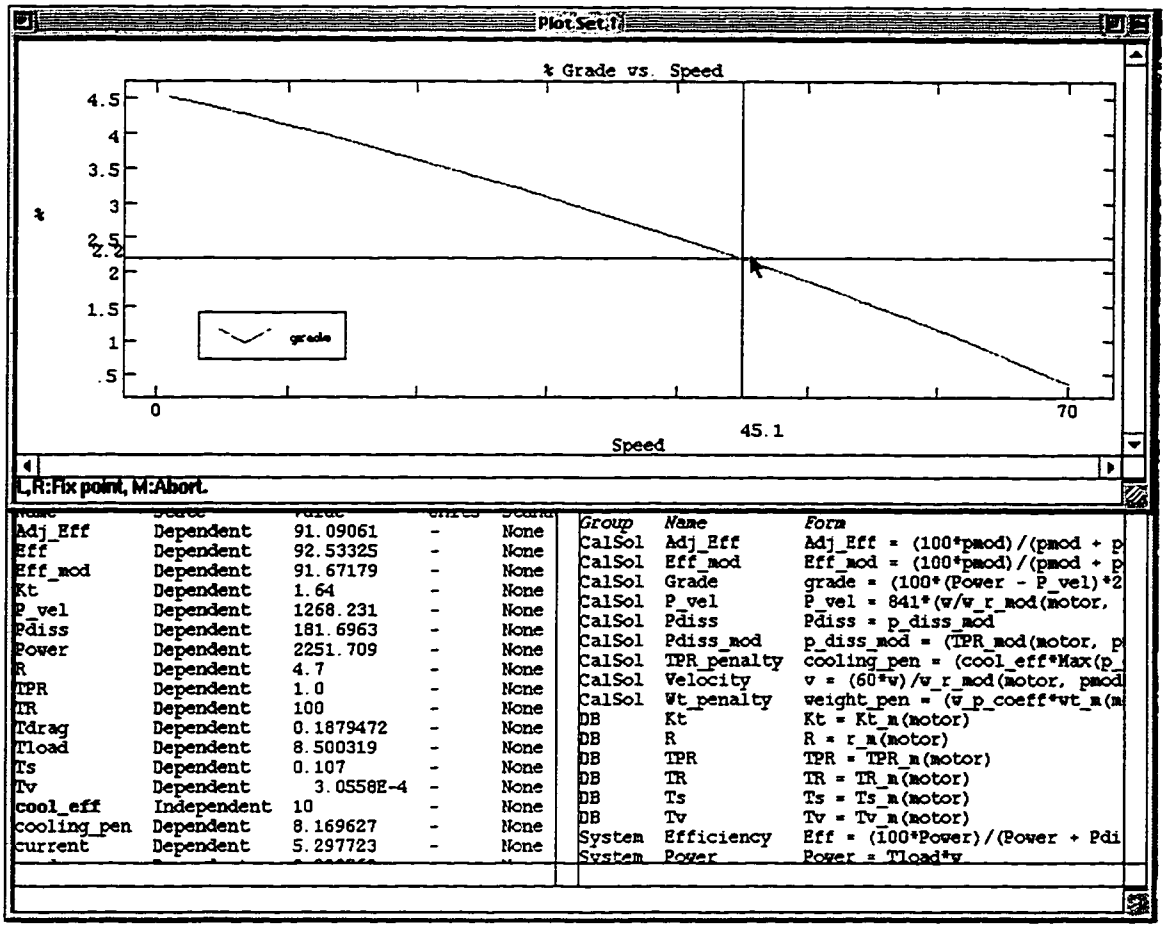


Figure 6.5 The motor selected in Figure 6.4 is evaluated for the maximum grade that can be ascended using a single gear fixed for the motor power peak to occur at 70 mph. The 6% grade requirement is violated, an additional 'climbing' gear is required.

user *CalSol*. Additional relations needed to calculate motor torque and power are selected from a set of system relations supplied by the component manufacturer. Together, these custom and standard relations make up the design model that is passed to Design Sheet, shown in Figure 6.4. In this figure, motors screened by the template query are evaluated with respect to the objective and the best candidate selected for further evaluation. It should be noted that none of the activity which takes place within Design Sheet is passed back to the Concept Database template. However, files generated during analysis can be saved and referenced by URL in the template description.

Figure 6.5 shows additional analysis taking place in Design Sheet using the same

engineering model from the template of Figure 6.3. Because the model is only the starting point for analysis, Design Sheet provides a great deal of flexibility in evaluating design possibilities offered by the component group and engineering model sent to it. Here, the performance of the selected motor on a grade is evaluated so that the team can decide whether a single speed gearbox can be used. As is shown in the graph, the grade requirement of 6% is violated in a single gear system, so an additional climbing ratio must be added. This information can be fed back into the design template and the model updated to reflect a second gear ratio.

6.2.4 Discussion

We have shown the Concept Database to be a useful and flexible tool for capturing and structuring design procedures typically used in component selection tasks. The design of the Concept Database supports the formalism of case based reasoning by offering a modular set of design cases which can be searched for similarity along several dimensions. Of course the search indices supplied to the user do not guarantee similarity, they merely offer an efficient mechanism for presenting possible candidates which can be 'replayed' in a new context or from which the user can glean 'interesting' components to combine into a new template.

These indices from which similarity is derived in the Concept Database fall into both formal and informal categories. In a relational database, similarity can be calculated through shared reference. Two templates which share a large set of system relations or variables are certainly similar on some level. In addition, templates with similar descriptions are also related. This type of similarity will be discussed in the next chapter.

Concept Database design templates represent component selection as modular design cases which can be identified and reconfigured in response to user queries. These cases are a valuable resource because they encode the mathematical formalisms that apply the designer's rationale to the component selection process. Their format encourages both formal and informal indexing in a representation accessible to the other CDIS component

applications. Integrating HTML with a structured database interface, design templates can themselves be contextualized within a design discussion or case study. As building blocks for conceptual design, the evolution of templates within a project discussion thread can formally trace the thinking of the designer in a way that closely models the IBIS extensions proposed by Nagy et al. [1992]. Operating within a standardized component abstraction hierarchy, templates reshape the underlying component database according to the current design need. We now turn to methods that can help determine when and how to navigate the component abstraction hierarchy.

6.3 Intelligent Real-Time Design (IRTD) Applied to Managing Design Abstraction

Conceptual design is an information gathering process directed toward the optimal use of available resources. Typical resources include designer knowledge, general design rules, analysis models, and catalogs of previous or available designs and components (e.g. company archives, vendor catalogs, a designer's personal experience etc.). Starting from an abstract problem statement, a designer is charged with optimizing the total overall design cost – the cost of the design itself (weighted by manufacturing volume) plus the cost of the design process necessary to specify the manufacture, distributions, and service. Casting design within this cost framework, the design process for inexpensive mass produced items differs greatly from that used for large low-volume items. The depth of analysis at any stage of design, the evaluation of the quality of the results of this analysis, and the degree to which the design space navigation are all driven by tradeoffs between the expense of performing analysis and the degree to which such an analysis can decrease the overall cost of the design.

At the heart of IRTD is a decision theoretic approach which attempts to direct the design process by examining the value of the many alternative information gathering and decision making options available to the designer. Couched in a traditional, nonlinear programming problem statement including an objective function and constraints, IRTD extends the problem statement to include uncertainty (which can formally represent ambiguity) in the

problem statement and applies information value theory to predict the value of reducing this uncertainty. Any gains to be made in the quality of a design are relative to the stated design objective (which incorporates the multiple objectives of all life cycle 'customers'). The validity of obtaining such design improvement is left as a decision for the designer[2]. The general requirements for applying IRTD to a design problem are:

Measurement of Design Objectives: A means of establishing the value of a proposed design instance.

Statement of Uncertain Parameters: Those parameters in which there is a degree of uncertainty (that might be reduced by the designer through some activity external to the IRTD system) are identified so that the value of reducing this uncertainty, reflected in expected changes in the measurement of objectives, can be calculated.

Set of Alternatives: A finite set of alternatives must be present for consideration. In standard IRTD catalog selection formulations, the discrete alternatives are distinct component entries. In the following sections we will explore abstraction level as the discrete decision – whether to descend the component type hierarchy or apply a classification variable that has not yet been determined.

These requirements provide the impetus for the application of a combination of information value theory and decision theory. In addition, Bradley and Agogino [1993a&b] introduces two general types of decisions for which IRTD can be applied.

Catalog selection problems in which the design parameters are uncertain and the performance criteria can be calculated for each of many distinct catalog options.

Prototype selection problems in which optimal designs using several configurations, each

2 In theory the objective could include engineering costs directly; however these costs are often very difficult to assess without a great deal of experience in the specific design context. This is beyond the scope of the current discussion as is the development of the original objective function.

with its own design model, are derived from using the same set of uncertain design parameters.

We will focus on the problem of deciding the appropriate design abstraction level given the current state of information about the design problem. As in the similar prototype selection problem, a class of designs makes up the discrete decision variable. Also the actions include adopting a specific abstraction level or obtaining better information about uncertain design parameters. Unlike the standard prototype selection problem, we are dealing with a 'sliding scale' of prototypes. Decisions can alter the abstraction any number of times, moving up or down the component type hierarchy or fixing (or freeing) any of classification variables valid in the current abstraction. The next section describes a method for creating uncertain design models in our domain which focuses on a set of canonical design variables (like all variables these are tied to specific abstraction levels). Because this method presents a model linking the objective function to a discrete set of design variables, the IRTD method can be applied to the task of finding those variables (under the current design objective) whose consideration will most positively effect the reduction of design abstraction. In addition, IRTD can estimate the current value of reducing design abstraction by evaluating the expected value of the objective function for each of the unconstrained classification variables. Thus, IRTD can be used both to determine the best current level of design abstraction and define the most profitable path toward reducing abstraction. Linked to measures of the probable gain to be realized by reducing design abstraction, this can provide a roadmap toward more optimal conceptual design with sympathy for initial design abstraction. The basis for IRTD is a calculation of the expected value of the current objective for the current decision and for each possible next step:

$$\mathrm{E}[obj|dec_i,\mathbf{c},\mathbf{v}] = \int_{\Omega_\mathbf{v}} obj(dec_i,\mathbf{c},\mathbf{v}) * P(obj|dec_i,\mathbf{c},\mathbf{v})d\mathbf{v} \qquad (6.3.1)$$

where:

obj is the value of the objective function
dec_i is one of the set of possible decisions
c represents deterministic constraints
v represents imprecise constraints and uncertain design variables
$\Omega\mathbf{v}$ is the state space of the uncertainty, **v**, in the problem

This can be used by the designer to decide if the current state of information warrants a decision to reduce design abstraction by constraining a classification variable or changing component class. That is, will choosing a component subtype likely increase the objective enough that the resulting reduction of the design space will not adversely effect the final design? In order to make this decision, a designer must be cognizant not only of how accurately the objective reflects the true design objective but also of how accurate the current design model is in the assessment of value. While computation cannot read the designer's mind, it can help to determine if gathering more information (in this case, applying constraint to a design variable) might be a better course of action. The IRTD method addresses the possibility of improving the current state of information by assessing the maximum possible benefit of constraining a design variable. The expected value of perfect information (EVPI) is calculated for each unconstrained variable (how this set is determined will be discussed in the next section):

$$\mathrm{EVPI}(v_j) = \int_{\mathrm{var}_j} \left\{ \max_i \left(E[obj|dec_i, v_j, \mathbf{c}] \right) - E[obj|dec^*, v_j, \mathbf{c}] \right\} P(v_j, \mathbf{c}) dv_j \tag{6.3.2}$$

where:

v_j is an uncertain design variable
obj is the value of the objective function
dec^* is the current (best) decision
dec_i is one of the set of possible decisions
c is the constraint vector
$P(v_j, \mathbf{c})$ is the probability of the value of the design variable and constraint set

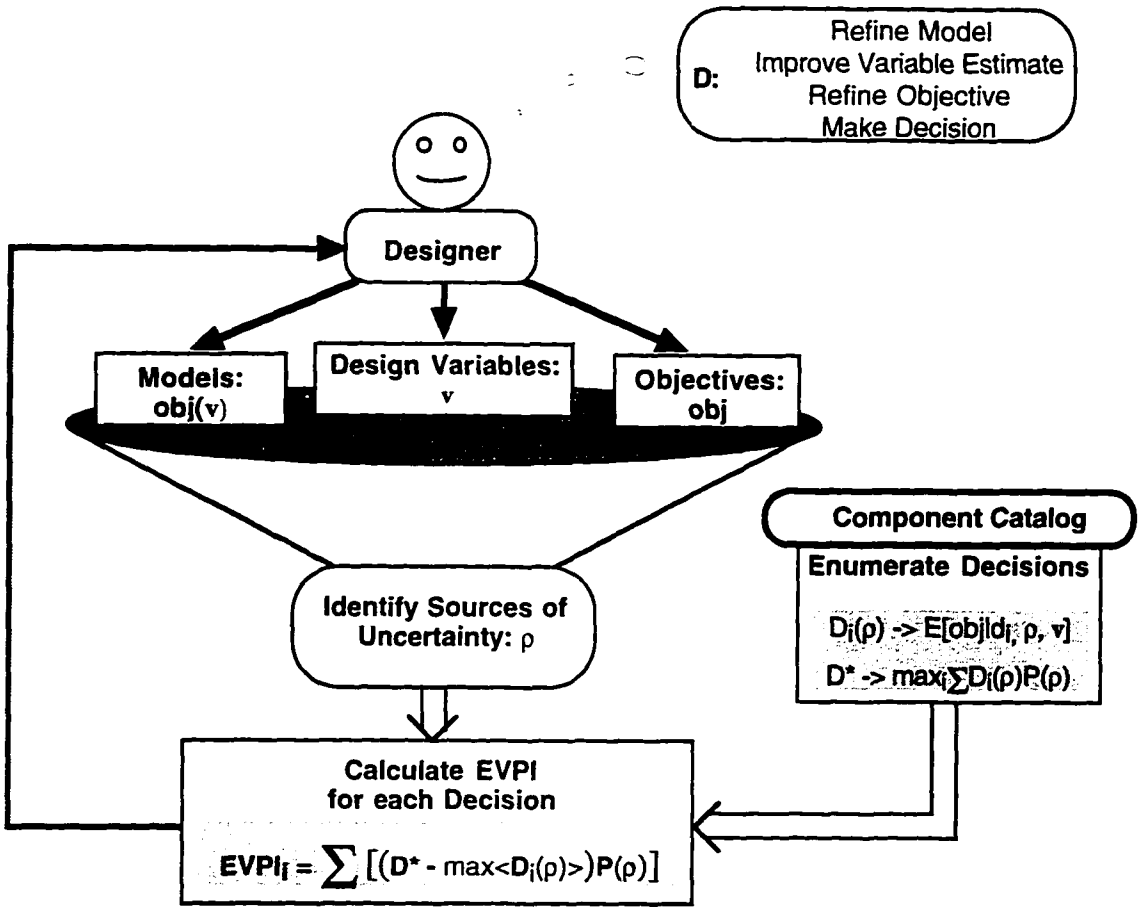


Figure 6.6 Diagram of the IRTD method showing the typical design loop in which a designer provides approximate models, variablles, and objectives which are combined with catalog-based decisions to direct the design process.

EVPI of a design variable measures the expected effect of selecting an exact value for the variable according to its predefined probability distribution. If, in the current design, the EVPI for a variable is lower than the cost of improving the accuracy of its estimate, then it should be removed from consideration (temporarily). By trading off the effort in improving design variable estimate against its expected impact, the designer can manage design progression as an information gathering process. Figure 6.6 shows a schematic for reasoning using IRTD. Combining EVPI with the expected value of a design decision, IRTD trades off two aspects of conceptual design:

1. Model accuracy. Does either the accuracy or the content of the less abstract model provide for an increase in the objective function? If so, the designer must judge whether such an increase in the objective function warrants commitment to a component subtype and thus a higher level of constraint on the design space.

2. Ambiguity. Do variables currently not present in the design model or objective contribute to the reduction of design abstraction (i.e. if there currently is little benefit in reducing design abstraction, would the value of another variable produce a difference that might make abstraction reduction valuable)? Here we deviate from the typical specification of IRTD where formal mathematical models are used to describe system behavior, all uncertain parameters are predefined. In the following section we discuss the creation of design models in which performance variables are related through probability distributions instead of mathematical expressions.

As shown in the formal analysis, a great deal of information is needed before a designer can even begin an IRTD analysis. An objective function must be defined, a model suitable for calculating this objective with respect to design constraints must be specified, and probability distributions must be given for each uncertain design variable. For early conceptual design, it might be inefficient or infeasible to assemble an objective that is useful, especially when the components themselves have the complexity of motors where each motor expresses a set of design tradeoffs among the various operating parameters. A template like the one listed in figure 6.3 might be a good starting point if it is sufficiently close to the current design context. Here, the design model can be adopted without modification and new parameter uncertainties adopted to reflect the new context. A difficult problem in conceptual design is that of identifying the motor which most closely embodies a very loose set of design preferences. The decision support system must provide the feedback from the design space necessary to define more completely these preferences. The next section demonstrates how a set of design instantiations can be combined into

operational models for conceptual design so that the IRTD method might be applied under the significant objective and model uncertainty prevalent in conceptual design.

6.4 Deriving Engineering Models from Artifact Cases

Through the component relations and analysis models constructed from them, the Concept Database provides rich support for the use of mathematical models of engineering devices. Despite their statement as formal mathematical relations, they remain models for the actual behavior of the components. As discussed in Chapter 2, for many CBR systems cases form the basis of an ad-hoc modeling system for complex components. In demonstrating the foundations for the Concept Database, we store information about electric motors; each of these is, in fact, a case embodying the performance of an engineering device derived according to a particular manufacturer's preference system. While many of the performance parameters for a motor can be derived using the mathematical models stored in the Concept Database, many important motor attributes of the design have no closed form derivation. How does one come up with a mathematical model in which package size or weight is derived from the first principles of electromechanics? Any such model must be an approximation and, to be of general use, be one based on other important derivable parameters. The problem is that the importance of a particular parameter is not static – it is intimately tied to the particular design application and the preferences of the designer.

In the discussion of artifact-centered case-based reasoning in Chapter 2, the CONCEPTOR system of Maher and Li [1994] was presented. We return to this example because it motivates the formation of engineering models from cases. In the EFD (empirical formula discovery) subsystem, design models are produced as (single dimensional) linear regressions over the clustered data. Within each domain of similarity, clustered through the analysis of discrete features (e.g. cable suspension bridges, cantilever bridges, soil conditions at column bases, etc.), relations among continuous design features like span length, deck width, and section depth are modeled. The models also record the mean squared error (R^2) for each relation to provide a measure of fit. There are some inherent

limitations to this approach:

1. Extrapolation: With relationships encoded linearly, the system has abstracted the case data into a form in which reasoning beyond the bounds of experience becomes possible. The R^2 value represents the strength of the relationship but cannot determine how 'far' from the underlying data one is operating. It is this form of modeling that Petroski specifically warns against.

2. Bias: Regression is pairwise among the various design parameters, compromising possible accuracy gained by higher dimensionality of regression. In assuming a first order relationship, the models have a strong bias which discards or weakens relationships which are not linear.

3. Directionality: The system chooses the order in which information is applied. For bridge design, there is a well defined set of environmental parameters from which the system attempts to produce both design type classification and propose a preliminary set of design variable instantiations. In most conceptual design a large part of the process is in determining which environmental factors (i.e., design issues) are important *in this design instance* and then creating an objective based on them.

However, in the conceptual design phase, questions like "how much does a motor which has a power output of about 45 Watts weigh?" are quite important. The designer is often 'feeling his way' through all of the design configurations and sets of parameters. This question is not only ambiguous as to what 'about' means, but also does not define 'motor' clearly. Nonetheless it is an important part of the decision making process in general and one which specifically does *not* ask "find me a motor of 45 Watts and tell me how much it weighs". By asking the former question, the designer might be trying to find out how much of the design space (with respect to motor length) is eliminated by assuming a 45 Watt motor – has a significant decision been made by limiting the power requirement to 45

Watts? Alternatively, the designer might be seeking a default value to be used in further analysis of the packaging aspects of the project. The quality of this default value will depend on how ambiguous the designer is about other unknown constraints. The design model must adapt to these and other design decision making needs.

6.4.1 Learning Engineering Models from Concept Database Component Data

The design space for motors is a high dimensional one in which each motor instance defines a distinct set of tradeoffs and objectives. We have applied CBR to the design space modeling problem through the use of a probabilistic neural network [Specht 1988, Tseng 1991, Agogino, Tseng et al., 1992] which learns over design instances to capture the 'knowledge' they represent by deriving a joint probability distribution over all design variables. The general formulation is derived from a Parzen estimator for probability distributions and is expressed by Specht as follows:

$$P(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{d}{2}}\sigma^{d}} \frac{1}{n} \sum_{i=1}^{n} e^{\frac{(\mathbf{x}-\mathbf{x}_i)(\mathbf{x}-\mathbf{x}_i)^T}{2\sigma^2}} \qquad (6.4.1.1)$$

where:

- $\mathbf{x}$ is the vector whose probability is being assessed
- $\mathbf{x}_i$ is the i^{th} training point
- n is the total number of training patterns
- d is the dimension of the input patterns
- σ is the distribution variance (smoothing parameter)

Further in a [0,1] normalized design space:

$$\sigma = cn^{-\frac{1}{(d+4)}}$$
$$c = [0.01, 0.35] \qquad (6.4.1.2)$$

This 'network' provides the means for representing relationships among continuous design

variables (Boolean values are encoded as extremes in the vector space, their distributions do not overlap significantly along the classification axis of the space). Generalization occurs according to a single 'focusing' parameter - **c**. If **c** is set to a high value, individual samples are obscured by a softening of focus. As **c** is reduced, peaks around design instances become more pronounced as a discrete world of individual motors appears. In testing over known probability distributions, Tseng [1991] provides a range over which the integrated relative error between the learned and reference distributions is quite low (on the order of 1%). Within this range, a wide variety of generalization can occur. In **c** values slightly below this range (**c** = 0.005), the system shows little generalization from the underlying data. Thus, Specht's model provides an interesting means for representing a suite of generalizations over the same training data. The main disadvantage of this model is that both memory requirements and computation increase at the order of the number of samples. Tseng proposes the Self Organized Probabilistic Neural Network (SOPNN), which limits the memory required to a fixed value by applying a *k-means* clustering preprocessor which assigns an instance **x** to a cluster *j* by a nearest neighbor approach:

$$C(\mathbf{x}) = j \text{ s.t. } \min_{k}\left(\left\|\mathbf{x} - \frac{\left[nc_j\mathbf{x}_j + (\mathbf{x} - \mathbf{x}_j)\right]}{nc_j + 1}\right\|\right) = j \qquad (6.4.1.3)$$

where:

x is the input vector

$\mathbf{x}_j$ is the j^{th} cluster

nc_j is number of training patterns associated with cluster *j*

These cluster centroids $\mathbf{x}_j$ are then applied as a weighted sum in equation 6.4.1.1 to form the new probability distribution estimate:

$$P(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{d}{2}}\sigma^d}\frac{1}{n}\sum_{j=1}^{k} nc_j e^{-\frac{(\mathbf{x}-\mathbf{x}_j)(\mathbf{x}-\mathbf{x}_j)^T}{2\sigma^2}} \qquad (6.4.1.4)$$

where (deviations from the above notation):

$\mathbf{x}_j$ is the j^{th} cluster center

nc_j is the number of input patterns associated with cluster j

Basing our case-based models on probability distributions learned using these methods rectifies many of the disadvantages discussed above for the linear regression scheme:

1. Extrapolation: At the 'edges' of the design space, the model degrades gracefully into making less 'sure' statements about the underlying data. When applied within the IRTD framework, this degradation of certainty is reflected in a de-emphasis of unlikely events influencing the design because the probability of an event is directly represented in calculating both the expected value of the objective for each decision and the EVPI for each design variable.

2. Bias: Any relation among variables can be expressed as a joint probability distribution. Thus bias is removed from the design knowledge base; any arbitrary relationship among the design variables and objective can be encoded, including uncertain relationships. This gain in generality is at the cost of computational efficiency offered by the representational abstraction of empirical equations. In domains where these empirical relations are weak for key performance variables (e.g. cost, weight, size) this tradeoff is a good one. In domains which are highly linear, the jump to probability distributions still helps to define the boundaries of the design space during the application of IRTD.

3. Directionality: Any of the joint distributions can be expressed as a conditional distribution so that input/output relationships can easily be constructed for any subset of variables included in the joint. This flexibility encourages easy manipulation of the distinction between objectives and constraints. Perhaps the greatest benefit is in showing the designer how decisions effect aspects not directly encoded in the design model, objectives, or constraints. Starting with an

unconstrained model with a simple objective, the designer can incrementally apply constraints and improve the accuracy of the objective, all within the normative IRTD framework.

6.4.2 Examples of Probability Distributions Learned from Catalogs

Figures 6.7-6.10, below, demonstrate the use of both Specht's model and the SOPNN for creating probabilistic models for component behavior. In this case, a joint distribution of motor weight and maximum continuous torque is created for each method using the Inland Motor Company Catalog [1995] for the underlying motor data. These are represented as the contour plots in Figures 6.7 and 6.9. Overlaid on the joint distribution is the expected value of motor weight plotted as a function of torque, derived from the joint distribution by the following relationship:

$$E[weight|torque] = \int_{\Omega_{weight}} \frac{weight * P(weight \wedge torque)}{P(torque)} dweight \qquad (6.4.2.1)$$

where:

$$P(torque) = \int_{\Omega_{weight}} P(weight \wedge torque) dweight \qquad (6.4.2.2)$$

Aspects of the design space which are not shown in the current model (the overall design space has 10 variables) are simply ignored as aspects of the cluster centroids. Because no generalization among data points takes place in Specht's model, it is as if these aspects of the component do not exist. However in the SOPNN, unseen aspects 'bleed through' into the simplified model because clustering has been based at least partly on variables not shown in the joint of reduced dimensionality. Additional constraints on the SOPNN clustering algorithm are imposed by the nature of the discrete classification variables. Because the minimum difference between members of different classifications is set to the maximum single axis difference in the design space, clusters naturally fall along the lines of

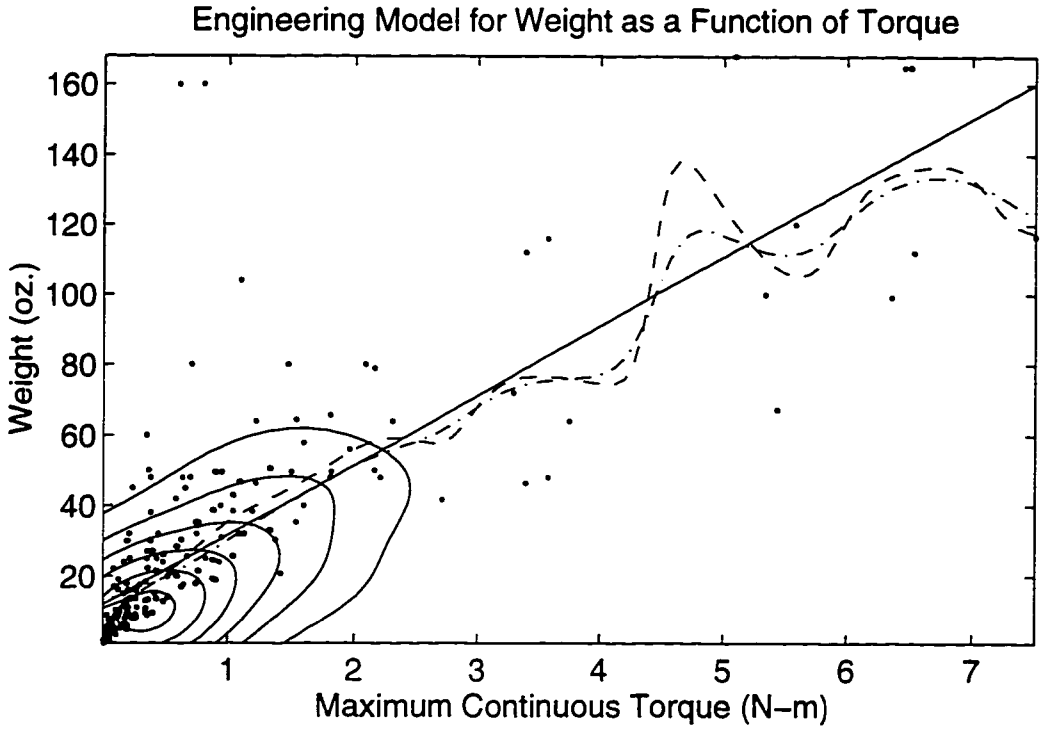


Figure 6.7 Engineering model generated using Specht's method. Contour plot shows joint probability density, overlays show model abstractions. The dashed line uses a smoothing factor of 0.01, the dot-dash line 0.05. Regression line overlaid has R^2 of 0.635. Underlying data is shown as '.' marks.

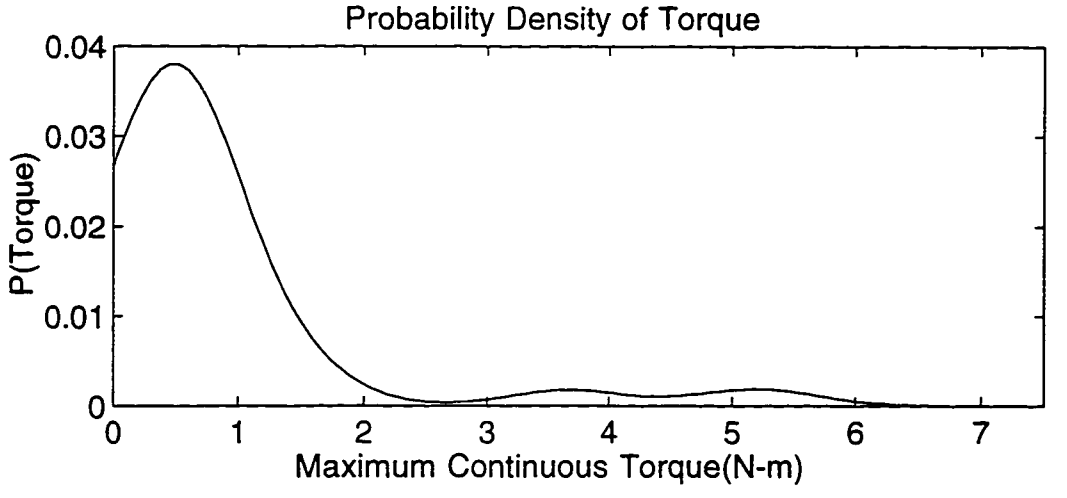


Figure 6.8 Probability density of Torque using Specht's model. Note that erratic aspects of the probabilistic model occur at low probability density. This behavior will be filtered out during IRTD analysis.

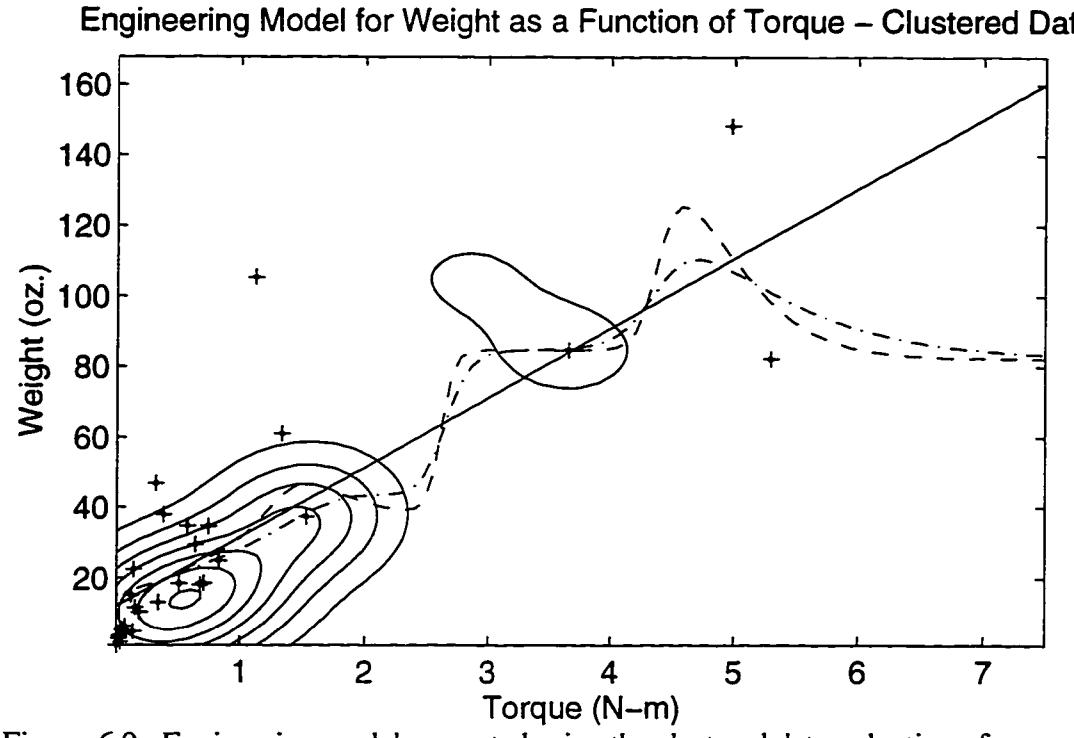


Figure 6.9 Engineering model generated using the clustered data reduction of SOPNN. Contour plot shows joint probability density, overlays show model abstractions. The dashed line uses a smoothing factor of 0.01, the dot-dash 0.05. Regression line overlaid has R^2 of 0.635. Cluster centroids are marked with '+'. Note the low probability cluster centered near (3.5,90) and its effect on the expected value plots that run through it.

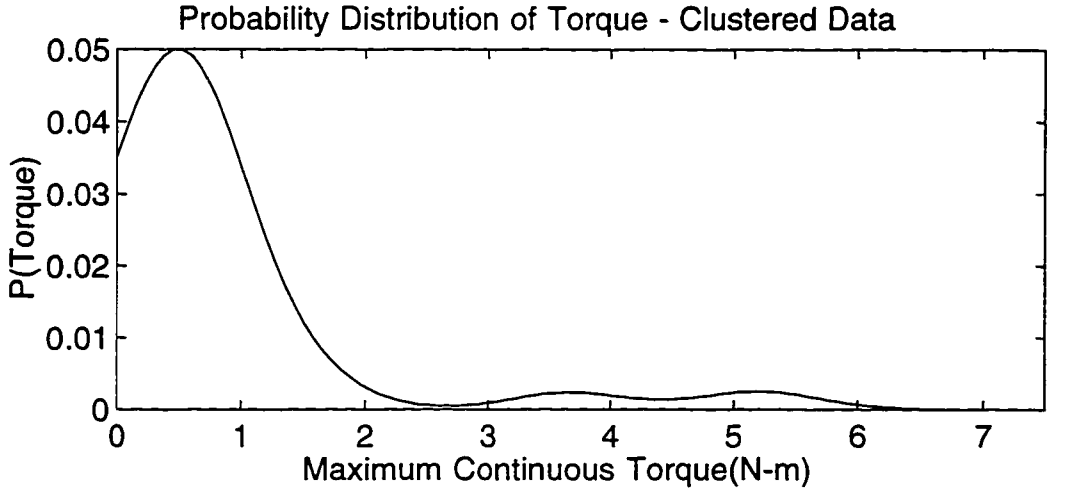


Figure 6.10 Probability density of Torque using SOPNN. Note that, as is the case with Specht's model, erratic aspects of the probabilistic model of Figure 6.7 occur at low probability. Note the 'bumps' in the distribution near the outlying cluster from Figure 6.9.

similar motor classes. Even when these classes are not included as constraints on the design, their effect is felt through the partitioning of cluster centroids by component subclass.

The main point of Figures 6.7-6.10 is derived from the final overlay on the joint distribution contour: the regression line used by Maher and Li's [1994] CONCEPTOR. Although this line closely echoes the probabilistic model for the relationship between torque and weight for the motors under discussion here, CONCEPTOR itself might choose to ignore it due to the relatively low regression confidence over the entire domain. By employing an approach where an empirical relationship can be derived from data along with the extent of the domain over which it is valid, we end up with a much stronger representation for artifact-based engineering models. This is as true in cases where the relationship has a strong linear character as it is in the more general case. Where relationships among significant design variables are not naturally linear, the probabilistic method still provides accurate model representation. We now return to IRTD as the means of controlling design search over our probabilistic performance models.

6.4.3 IRTD Control of Reasoning in Artifact-Based CBR

The main drawback of this approach to modeling is that because no empirical relations are derived from it, typical analysis tools are less useful than might be desirable. While this might be a significant drawback for later stages of design, objectives and models in the early conceptual stages tend to be difficult to model formally. The engineering model is simply a large joint probability distribution which can be manipulated to form conditional probability distributions appropriate to estimate component performance under uncertain constraints (among other performance variables). This mediates a big problem in modeling the transition between abstraction levels in conceptual design: even when approximate models can be applied, it is often difficult to quantify their accuracy with respect to more complete models. It is hard for the designer to know when enough information has been assessed in order to make a decision. As seen in Equations 6.3.1 and 6.3.2, IRTD always

operates with respect to a set of possible decisions. Here, potential gains in the objective function are offered in exchange for commitment by the designer to reduce design abstraction by setting a constraint on a classification or design variable. Because classification variables narrow the scope of samples significant to the distribution to subsets of the design space which are more closely related, the resulting probability distributions are more homogeneous (and likely more accurate) the more abstract models which combine them.

Ambiguity in the design arises as unconstrained performance variables present in the overall joint distribution. In each case, these variables are not completely free. Their portions of the design space joint are conditioned on design constraints just in the same way that variables appearing in the objective are. Ambiguity on the designer's part thus takes the form of not constraining certain design variables. This cannot be confused with the adoption of a uniform distribution of design variables currently unseen in the design model.

The derived probabilistic model offers the advantage of supporting not only the transition among models of varying complexity (more or fewer variables can be used to condition the variables included in the objective) but also of supplying information about the accuracy of these models. With IRTD providing the framework for handling uncertainty – specifically for deciding how uncertainty affects decision making, we can easily tradeoff commitment among various design axes. For IRTD to generate optimal strategies for progressing with the design (i.e. reducing design abstraction and design ambiguity), objectives must be represented by a utility function, 'dimensions' must be supplied along which ambiguity in the current design state can be manipulated, and discrete steps in design abstraction must exist. An example of the evolution of motor selection loose objective to committed design follows.

6.4.4 Illustrative Example

Recall the case of the 45 Watt motor. In a previous section, a designer asked 'How much

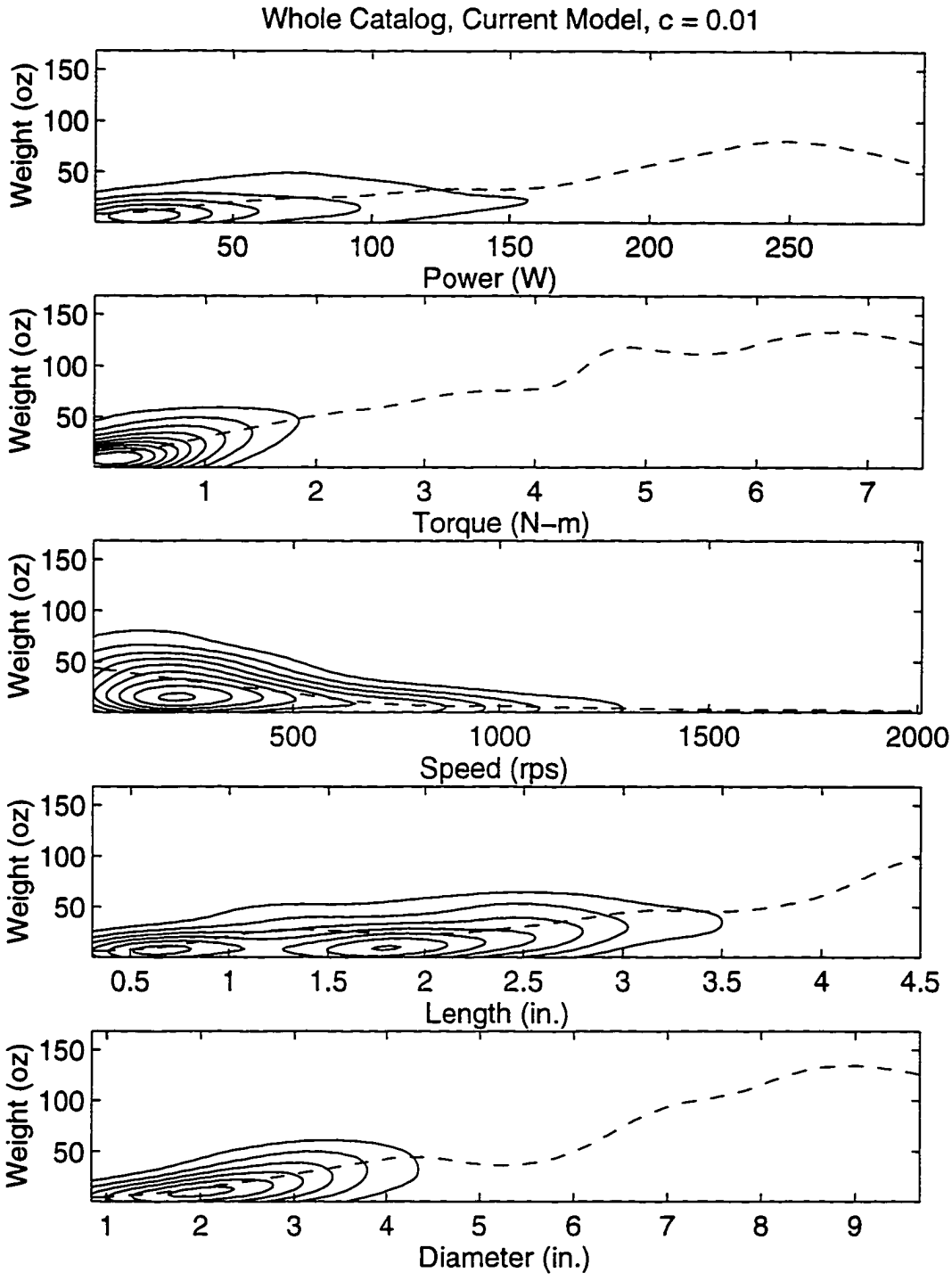


Figure 6.11 Probabilistic Engineering Model for Motor Catalog. Five key design parameters are shown as joint distribution contour plots overlaid with dashed lines showing the expected value of the current objective, weight. Note that the less smooth protion s of the expected value lie at low overall probability.

Power	Torque	Speed	Length	Diameter
12.45	8.09	7.2	12.25	9.6

Table 6.2 EVPI for design variables with respect to current decision: DC Motor with an expected weight of 26.5 oz.

does a 45 Watt motor weigh?" We have presented a method whereby the answer to that question is a probability distribution which depends on the level of commitment to abstraction level by the designer. The design process stemming from this question evolves as the answer is continually evaluated. At the current state of commitment (neither a definition of 'motor' nor constraints on variables aside from power have been given), our probabilistic framework transform the question into "what is the probability of weight for any motor type given the rated power is 45 Watts". If this query returns a uniform distribution of motors or one that shows acceptable performance, the designer might set 45 Watts as a constraint or perhaps see how the proposed constraint has affected other significant design variables like cost, weight, maximum torque, etc. Eventually we have to get the designer to commit to the design objectives which spawned the initial question. For instance, the question might express an underlying need to package a 45 Watt motor in a dynamic system where weight or size might be constraints and cost the main objective. This is the nature of conceptual design, a search for what is possible followed by the generation of evaluation metrics that can narrow the search.

So we begin by examining models of weight for motors with respect to power and some other key engineering parameters: torque, speed, length, and diameter. Figure 6.11 shows a set of engineering models for these parameters generated using Specht's probabilistic model over a catalog of about 300 motors and a smoothing factor of $\mathbf{c} = 0.01$. These distributions show the overall design space. The EVPI for each of the unconstrained variables is shown in Table 6.2, validating power as an important design variable with the

Torque	Speed	Length	Diameter
11.34	8.38	12.53	8.34

Table 6.3 EVPI for design variables with respect to current decision: DC Motor with rated power of 45 Watts and an expected weight of 19.1 oz.

highest EVPI measure; it is expected that by constraining power within the current range, a decisions (no single decision need be dominant over the whole domain of power) can be made which will reduce motor weight by about 12.45 oz. The motor power constraint set at 45 Watts, Figure 6.12 shows the resulting joint distributions of the remaining unconstrained variables and the objective, motor weight. Table 6.3 shows the EVPI for each of these variables, recalculated to reflect the current constraint set. To see how these EVPI values evolve, we must analyze the possible decisions based on each parameter. Figures 6.13 - 6.16 below show plots of the expected value for each of the possible abstraction reduction constraints on classification variables. At the top of each plot set is the expected value of weight for the variable under the current set of constraints, along with an dot-dash overlay representing the probability distribution of the variable. As for decision making, note that in the case of only one variable – torque – is the decision (shown as '*' marks overlaying the expected value plot of the most advantageous classification variable) as to the best next step dominated by a single classification variable. For all others, the best way to reduce design abstraction depends on how the variable in question is constrained. This leads to a natural progression in the design of investigating constraints an d how they effect potential design decisions. This is a normative model for the conceptual design process.

6.4.5 Discussion

As demonstrated above, once an objective has been defined IRTD can be used to operate over the hierarchical subdivisions of motor performance to quantify potential increases in the design objective to be gained by limiting consideration to a subset of motors. In addition, design variables are analyzed for their impact on reducing design abstraction. Here, the expected impact on the decision making process of adding constraints to the current model can be calculated. In many cases, these decision are multidimensional – cost, size, weight, and performance are all rolled into a single decision. Some applications might even warrant negotiation of a customized motor. Recall the CalSol motor selection template. It is clear that changing design parameters not generally represented as catalog

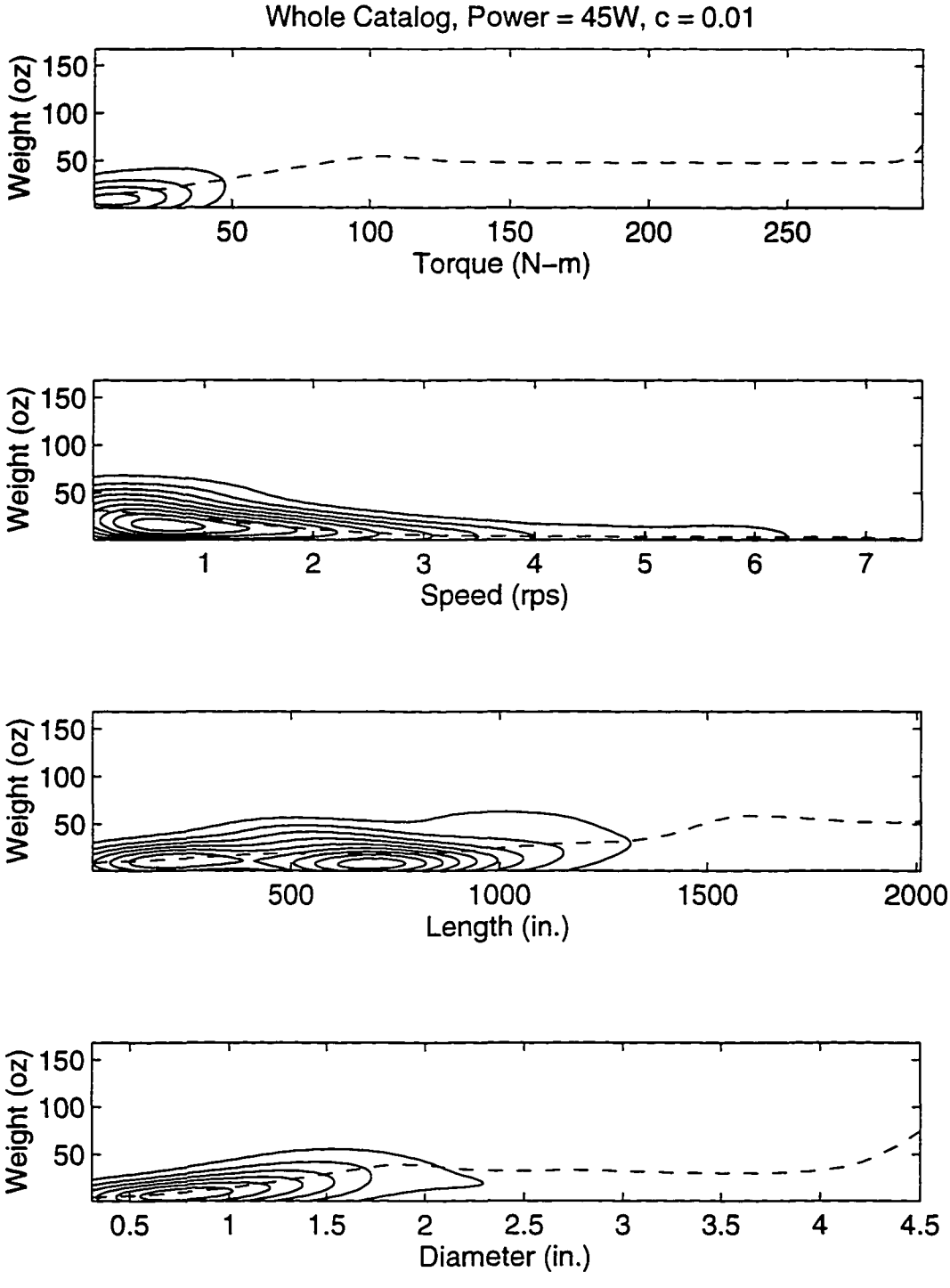


Figure 6.12 Probabilistic Engineering Model for Motor Catalog, after the constraint of 45 Watts has been applied. Because of the commitment, only four design parameters are 'free' variables. Again, expected value of weight is given as a dashed line. In comparison to Figure 6.11, distributions here are more tightly defined, a result of applying constraint to the design space.

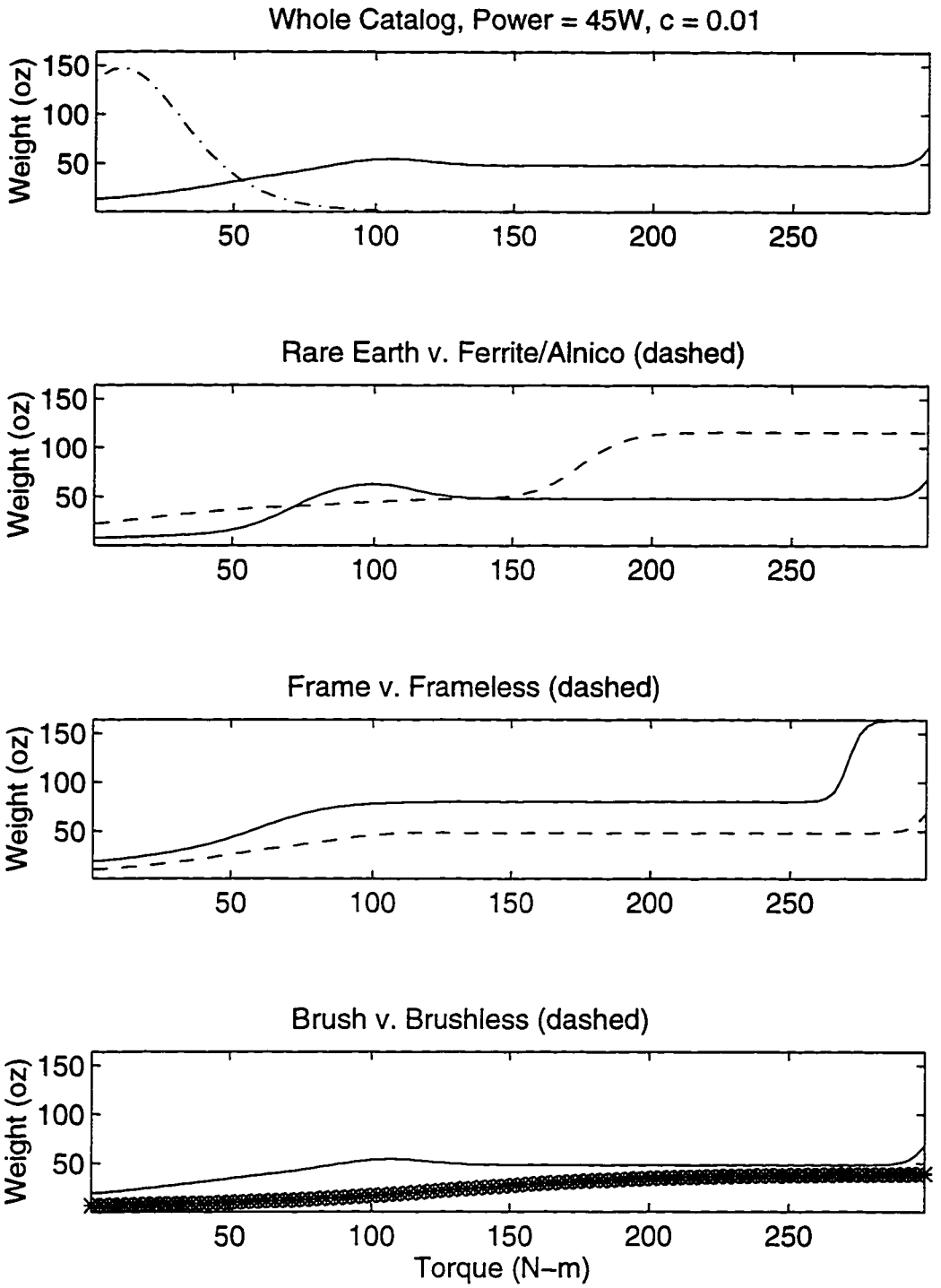


Figure 6.13 Decision Making Based on Motor Torque. The top plot shows the expected value of weight plotted against torque. Overlaid is the probability that a motor in the constrained domain will have a given torque (not to scale). The optimal decision (denoted by a '*' overlaying the expected value plot) is a Brushless Motor.

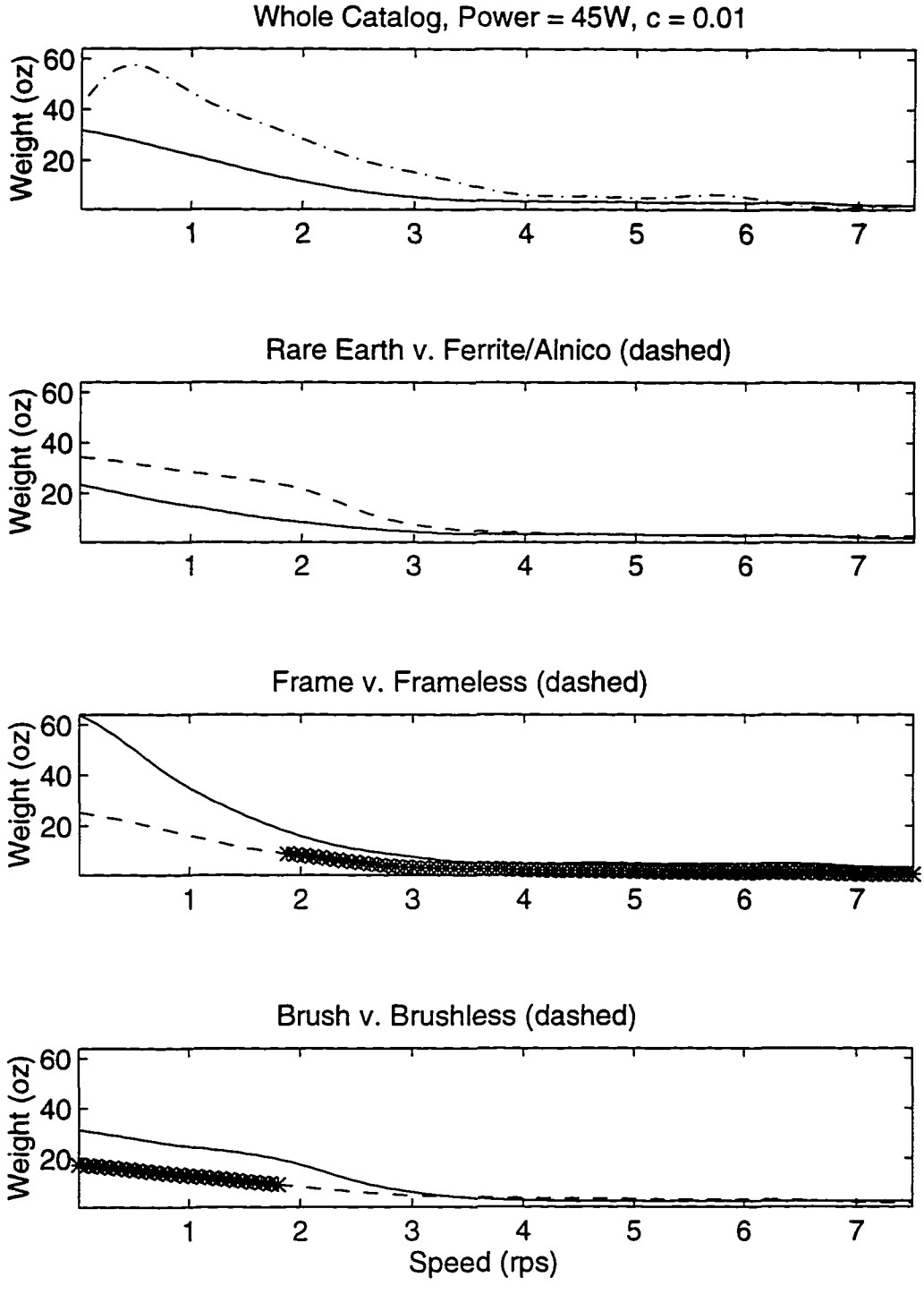


Figure 6.14 Decision Making Based on Motor Speed. The top plot shows the expected value of weight plotted against speed. Probability of weight given speed is overlaid (not to scale). The optimal decision ('*') changes at about 500 rps from Brushless below to Frameless above.

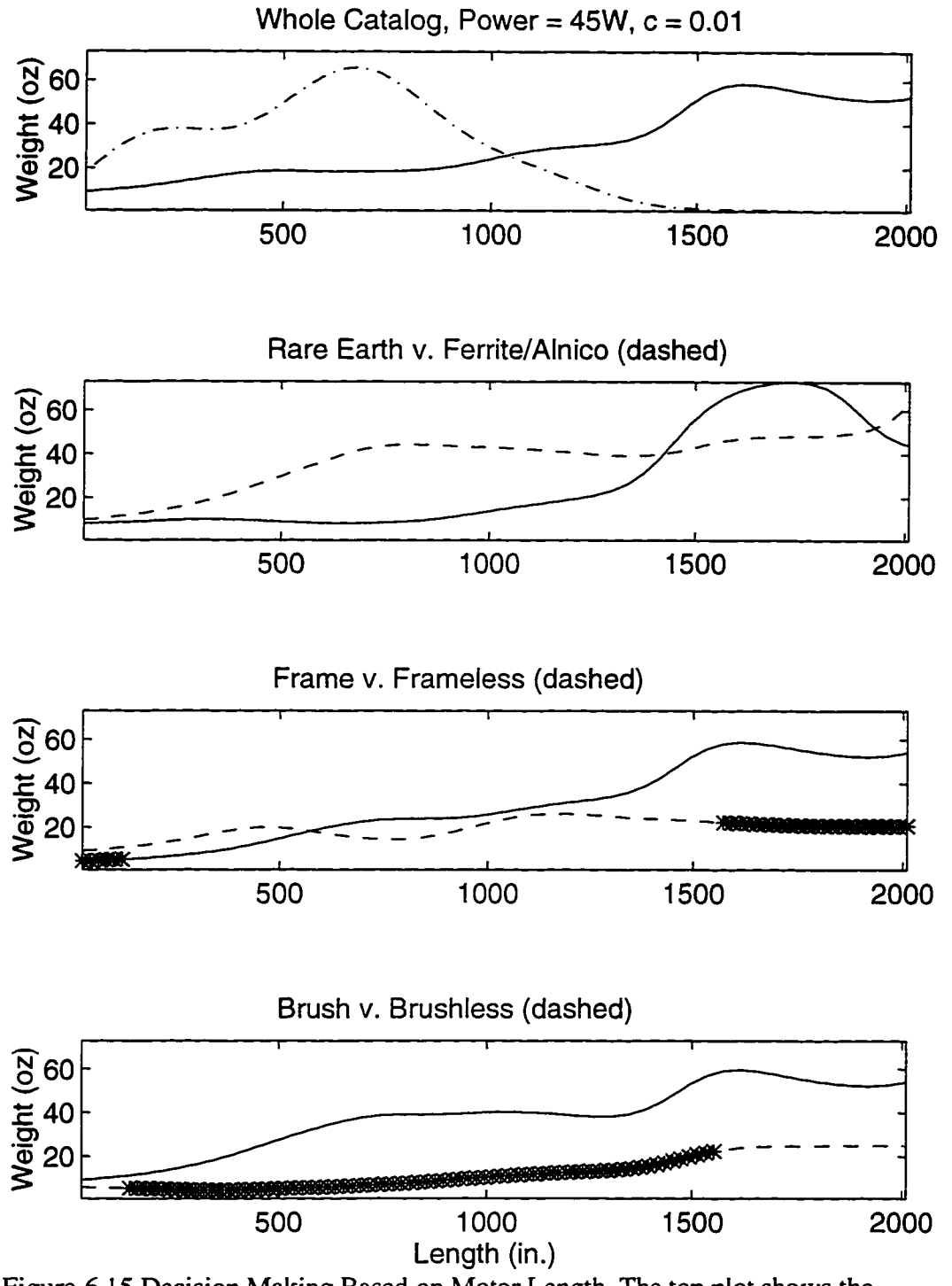


Figure 6.15 Decision Making Based on Motor Length. The top plot shows the expected value of weight plotted against length. Probability of weight given length is overlaid (not to scale). The optimal decision ('*') changes from Frameless for short motors to Brushless for most other lengths.

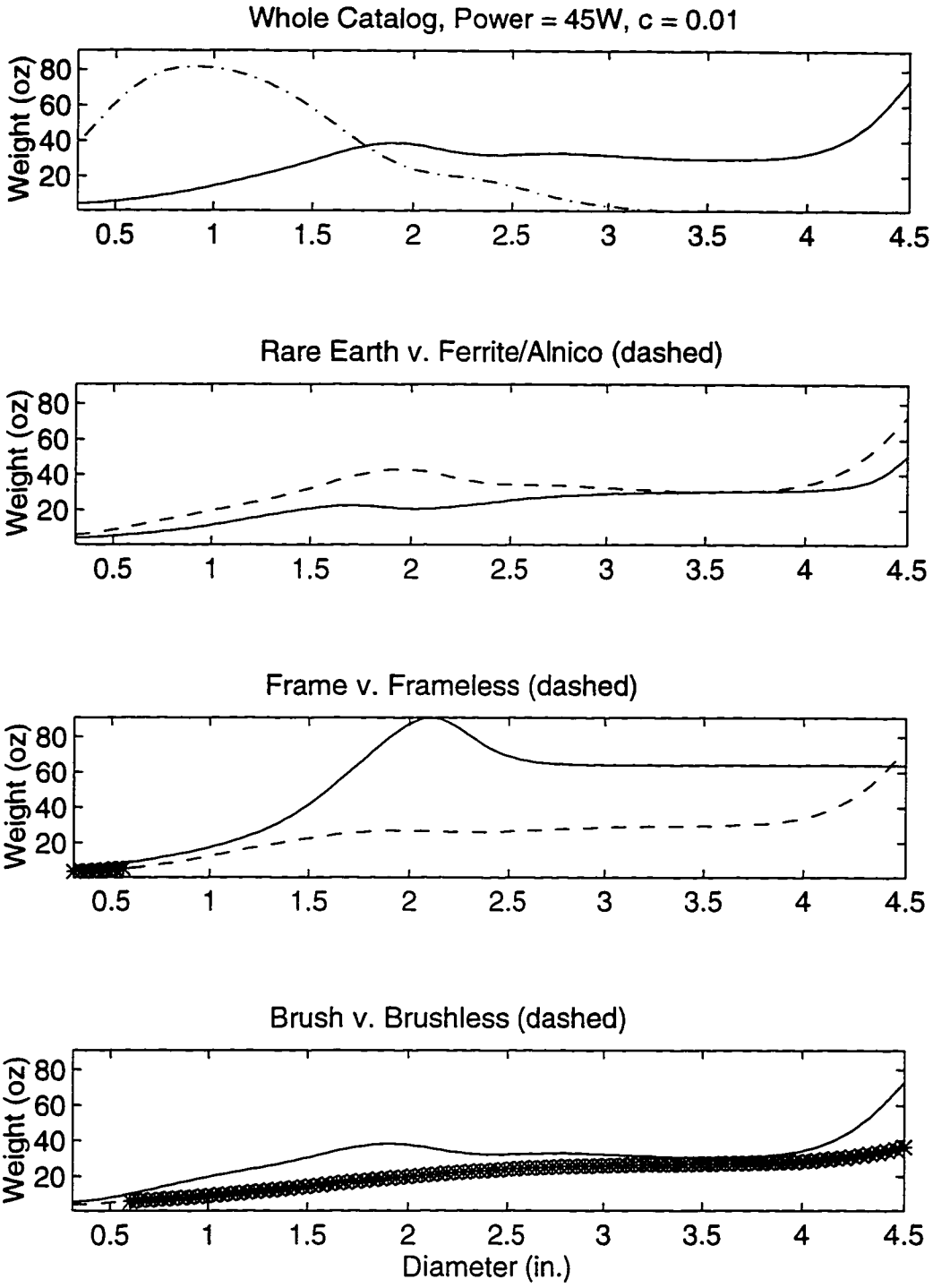


Figure 6.16 Decision Making Based on Motor Diameter. The top plot shows the expected value of weight plotted against diameter. Probability of weight given diameter is overlaid (not to scale). As in the Length variable, the optimal decision changes from Frameless for small motors to Brushless larger ones.

variables becomes of interest. For example, increasing heat rejection of a motor can significantly improve key performance aspects like power/weight ratio and cost (motors are typically rated at close to worst-case heat transfer – conservative for most applications). Extending the performance characteristics of a motor was done in the CalSol template on a customized basis. While the method above was demonstrated in the context of design variables which are direct catalog lookup variables, it does extend to functions of these variables and other parameters as well. Any static set of data that represents the performance of a component can be woven into the artifact-based CBR/IRTD framework described here.

Of course these general purpose functions extend to design objectives as well. Perhaps the main remaining question is how to handle inequality or interval constraints. Here we rely on the database search to create a subset of data over which learning can take place. Experiments were carried out substituting for the 45 Watt constraint a constraint on the underlying data set of power to be within the range of 30 to 60 Watts. In addition, the 45W constraint was added back into the system to assess query ranges as a means of reducing calculation time by eliminating portions of the design space only weakly linked to the current context. A detailed analysis of these can be found in Appendix A, the basic findings are:

1. The constraint is reflected in changes in the model and decision making using the whole catalog as the underlying motor knowledge base. The constrained model shows a uniform decrease in noise at every level, the distribution showing a tighter grouping than that produced without constraint. Decision making also reflects the constraint; narrowing the design space with the constraint modifies optimal design decisions (e.g., the whole catalog model suggests a brushless motor for minimizing weight for a power output of 45W, the model constrained to 45W indicates that speed is an important consideration in deciding whether to settle on brushless motors or frameless ones first.)

2. Modeling and decision making based on applying constraint through conditioning the joint distribution are consistent. Whether the whole catalog or a subset of it is used, the decision making is the same. Reduction in the computation time is linear with respect to reducing the size of the database over which learning takes place.

3. The application of constraint through database subsets generated from screening query intervals yields results similar to those of applying a nominal value constraint. Unconstrained vs. constrained modeling and decision making produce largely consistent results. Inconsistencies are limited to the extremes of the design variable ranges where neither method is likely producing accurate results. As a result, information value among constrained and unconstrained subset methods are almost exactly the same.

In addition , Appendix A addresses the changes that results from varying the focusing parameter (**c**). Varying **c**, which changes the variance around each motor center, allows the user to manipulate the amount of generalization that takes place in the model. At larger values of **c** the models become less defined the expected values become straight lines, and decisions tend to be uniform across all design variables. This is especially so in the constrained cases. At the smallest values of **c** individual motors can be seen as 'islands' emerge from models that had looked fairly continuous. Decision making here reflects these islands as specific classes of motors represent distinct tradeoffs among the design variables. The transition from general to specific takes place gradually in all cases, indicating that **c** is a useful parameter that can control reasoning.

Finally a discussion of the application of SOPNN is in order. Again, Appendix A elaborates on the basic result: SOPNN degrades system performance in virtually all aspects. It results in erratic decision making and information valuation due to over-generalizing the underlying design instances. Perhaps this is to be expected for the catalog in question represents the product offerings of a single manufacturer; duplication of the type which SOPNN is designed to generalize should be absent in a well-conceived product

line. The application of SOPNN must wait until the overlap generated by the combination of multiple vendor catalogs can be exploited by combining similar design instances. At this point, the best choice for gaining computational performance seems to be to use design variable intervals to reduce the design space before decision making begins.

6.4.6 A Note on Motor Customization

The above discussion does not take into account an important aspect of component selection – customization. We have described a learning algorithm which generalizes over cases through a logarithmically weighted interpolation function – the Gaussian distribution. While such interpolation of performance characteristics is realistic for most of the properties of a component, there is one key property for which this assumption breaks down: cost. In general, deviating from standard catalog components results in increased cost. This increase is also dependent on the specific axis on which the deviation takes place. In our domain of DC motors, most of the performance parameters have ignored electrical properties like voltage input. This is because several voltages are supported for each motor frame, typically the frame is selected for its mechanical performance properties and then an appropriate winding chosen. Custom windings can bridge the gaps between standard windings with little additional cost.

More costly changes to a motor design occur when its physical dimensions need to adapt to specific design needs. Here we have two main options – changing the motor diameter or changing its length. The former is a costly alteration since the entire motor must essentially be redesigned; the latter is a minor change which can be accomplished by making the standard motor longer (shorter) by adding (subtracting) plates from the magnetic core. Certainly a similar situation exists with other 'standard' components which one might want to customize. In this case, a simple solution might be to provide a cost penalty function which can be applied as the design deviates from each standard motor. This penalty function might also vary with a particular supplier's willingness to customize his offerings (in the case of high performance motors, many vendors offer catalogs containing 'stock'

motors to use as starting points; many do not keep any inventory of most models).

The implication is more serious when database 'samples' are generalized into clusters using the SOPNN method described by Tseng. While useful for limiting the number of sample points represented in the distribution and thus the amount of computation necessary to analyze a component model, clustering *forgets* the original samples themselves, synthesizing them into representative samples. Customization costs that might be modeled as a simple penalty function applied at the center of a single motor sample cannot be likewise appleid to a cluster centroid because this cluster does not represent an actual component – it is a composite of several. In order to find a basis for customization and apply the cost of deviation from the standard, modeling must revert (at least over this reduced design subspace) to methods that recognize individual components. Thus direct mathematical modeling and typical IRTD component selection from among the set 'near' the desired cluster centers can be used. Alternatively, a reduced form of Specht's model can represent generalization in the design space reduced by screening queries, exploiting the computational efficiency of generalizing the larger design space using SOPNNs and then reverting to the sample based distribution as the abstraction level decreases.

6.5 Summary

Typical of systems described in Chapter 2, design cases (i.e., artifacts and the performance data that describes them) are clustered by an external assumption of similarity. What is not typical is that this external similarity bias can be manipulated automatically within the system. Each node in the hierarchy defines a set of parameters that are valid for similarity measurements, along with a set of relations that can be used to build evaluation models. In addition, the nodes represent distinct sets of discrete variables which are generally interchangeable. Thus, motors (and motor catalogs) take on an array of abstraction levels that match the specific information need of the current design instance. Design Sheet can operate over sets of relations established as an evaluation of potential candidates to help the user select a motor from a set of motors (of possibly heterogeneous types). Where

mathematical models are absent (e.g. power as a function of cost; power, speed, and package length related) behavior based on the set of motor 'cases' in the database that are similar to the current design instance can be synthesized into an approximate model. This model can be directly synthesized from a database or could be based on subjective probabilities[3] offered by the user.

We have proposed a hierarchically defined component type collection coupled to analysis relations learned from component data over which IRTD is used to formalize strategy in case-based conceptual design. We also recognize that IRTD, with its mathematical formalisms, is not the only appropriate strategy for the early stages of conceptual design. In this stage, design cases can provide valuable resources to the designer. Queries against a library of cases with specifications similar to those in his current project can direct these early stages of conceptual design by uncovering important issues and suggesting possible solution paths. All of this must be done over a representation that does not provide useful structure and fights interpretation by machine through its use of abstract, informal language (free text). The next chapter focuses on this very question – the creation of a global index for affording access to the less structured side of the CDIS.

[3] These could be evaluations of the probability of using a specific motor type, a distribution of classification variables, a distribution of constraint variables, or even a joint distribution of an objective and several performance parameters.

Chapter 7

Global Indexing: A Framework for Unifying Information Access in the Conceptual Design Information Server

The preceding chapters have laid out a foundation for the CDIS consisting of three primary components: Concurrent Engineering Hypermedia Case Studies, Design Discussions, and the Concept Database. Each of these subsystems provides strong local indexing; navigation from a starting point is straightforward with a natural interface individual to each. Case studies are indexed locally through a concept map which has proven effective for a broad range of users. It is easy to trace through a design discussion by navigating the design issue hierarchy. The implementation of the design discussion server within a database provides the capability of additional project 'views': a chronological view can replay the evolution of the design from problem statement to solution, a user-centered view can reveal the role of any participant. The preceding chapter focused on local indexing within the Concept Database, centering on the process of creating and evaluating design templates based on prior designs and generic system components. Through the navigational links they offer, the hypermedia documents in all three subsystems express strong local indexing.

The locality of reference within the three CDIS components provides a natural means of navigating within a design context once it is found, but matching the context of a design

case to the current design problem remains a difficult task. All three components of the CDIS also share a representational scheme based strongly on textual expression of information. Chapter 3 argues that the textual representation places the least bias on the case information so that it can be most effective in communicating real design information to the user. But relaxing bias from a formal representation removes the domain structure often implicitly stated as part of the representational abstraction. This chapter focuses on a method for externally applying structure to the free-text, unstructured information spread throughout the CDIS so that it might be effectively applied throughout the process of conceptual design.

Basic to this goal of information access is the assumption that there exists a commonality among the components that is not already expressed in the local indexing. Case studies of a single industrial project must be woven together to reveal how various issues in concurrent engineering are manifested in different contexts. Design discussions are represented in the design discussion server as tree structures: clearly, in the concurrent engineering conceptual design process, discussions are more appropriately modeled as a tangled web of interacting issues and concerns among the many stakeholders in the project. These issues also transcend the discussion in which they arise, linking discussions among many design contexts to provide a case base of experience which the user can interpret under the context of his current design concern. Finally, component selection templates do not represent the specification of a single component but also place its application within an overall design. Here, valuable information about component function along with the preferences which shaped the selection process enriches the design context far beyond a simple statement of modeling equations and query ranges. So, through the design issues that shape the concurrent engineering process in the case studies and drive conceptual design discussion, to the specific functions that arise from these issues, and into the concrete representation of these functions into components, the components of the CDIS represent an overlapping continuum of conceptual design context.

Having established a common ground for information reference, consideration turns

pragmatically to implementational issues. The proposal is to create a set of 'virtual hypertext' documents [McCall et al., 1991] in which navigation can take place not only along the locally defined paths but also along paths that are dynamically determined by the current user context and the contents of the CDIS case base. The goal is to provide an external structure which can recognize the current design context and then automatically provide navigational links to those contexts within the CDIS relevant to the current recognized context.

The remainder of the chapter is structured as follows. The library model of assigning subject context is discussed toward elucidating the main issues involved in the process of cataloging information by subject (context). Experience in library user interface development underlines the importance of 'systems' view of information access and retrieval. This experience provides the impetus for the design of the indexing scheme, placing constraints on the implementation of the CDIS global index. The technology used in the implementation, WAIS, is introduced as a generic implementation of paradigmatic information retrieval. The library and information retrieval models are then conjoined through the development of a set of hierarchical design contexts used as a controlled vocabulary and applied to the information retrieval process. Design issue and function abstraction hierarchies are developed which, along with the component hierarchies defined in the previous chapter, combine to pinpoint design context. A method for automatic indexing and query expansion incorporating these abstraction hierarchies in to the standard information retrieval paradigm is then proposed. Evaluation of experiments run within and outside of the CDIS information base are evaluated toward the creation of the CDIS global indexing interface.

7.1 Background: Library Methods for Indexing Locally Structured Information

The library science community is charged with a task isomorphic to that of the CDIS global index: to provide an external contextual structure for information represented in the relatively unstructured format of free text. Two general solutions have gained acceptance

here: subject tagging using a controlled vocabulary and indexing derived from the textual content of the information or a surrogate of it (e.g., title, abstract, summary, etc.). Various different controlled vocabularies have been developed to encode the informational context of documents within many different fields (i.e., Medline, Library of Congress, Inspec, Engineering Index). In each of these the controlled vocabulary is not a 'flat' list of subject headings but is expressed as a hierarchy; broad terms are connected to narrower ones. Because such hierarchies unnecessarily bias the expression of knowledge as trees, related terms are used to 'tangle the branches' in a more natural contextual arrangement (terms are also repeated among multiple hierarchies, encoding further 'tangles').

Given a controlled vocabulary in which to categorize documents, professional catalogers must analyze document content and assign its context by associating with it a number (often limited to three as a matter of policy) of subject headings from the prescribed set. In addition to these subject headings, the document title, author, publisher, abstract, and perhaps a brief description are combined to form a surrogate used to represent the full document it in the library catalog system. OCLC (On-line Computer Library Center), the major cataloging entity in the U.S. outside of the Library of Congress, sponsored an exhaustive study of subject-based retrieval in library indexing systems performed by Drabenscott [1995][1] . Building on recommendations from prior studies of user transactions within on-line public access catalogs (OPACs) and on the analysis of a new set of user search trees, Drabenscott finds the following major failings of subject searching:

1. Subject queries posted by users frequently do not match controlled vocabulary terms or incorrectly match terms outside of the intent of the query.

2. Controlled vocabulary terms are not available to the user to provide cues for searching – typically a series of interactions with the system (generally two to three queries) is required to identify valid subject headings.

[1] It should be noted that the impetus for this study comes from a finding that while subject cataloging is resource-intensive, it is also the least utilized mode of access to library collections. Users prefer author and title searches for unknown resources; many search trees are to find resources that are already known to exist.

3. In many instances when a user query does match a controlled vocabulary term (usually geographical information or proper names), the result set generated by the query is too large to be useful.

These results indicate that there is much room for improvement in subject-based searching in library systems. Another interesting result is the nature of the queries used for the study – samples were taken from user interactions with numerous on-line systems. Many of the queries consist of very short exchanges between the user and the system – most are fewer than three words. Three words to express the context of a search is quite small, the explanation given by the authors being that perhaps users are attempting to match what they know are short phrases in the controlled vocabulary. As the search tree continues, users tend to add in more contextual information to narrow searches that return too many results and try alternate terminology in searches that return too few. This strategy of manipulating the query in response to the result set returned is a valuable one that could be better supported if the subject hierarchies were available directly to the user.

Studying the handling of subject searches from those developing library information systems helps to define the best ways in which to use an indexing framework for information retrieval. It is equally interesting to note how such subject indexes are created. It is common practice for libraries and other organizations endeavoring to index collections of documents is to develop an abstraction framework that can be used classify the content of members of the collection. This development effort is responsive to both the type and breadth of the collection as well as the needs of the assumed user group. For this reason, Library of Congress subject headings, useful for large collections, often give way to more domain-specific indexes like the Engineering Index or Inspec. Whatever the chosen contextual framework, the classification of the document is done by human catalogers who are armed with some general knowledge of the subject area, a hierarchical set of classifications[2] and a set of guidelines for language, key phrases, synonyms, etc. to give further contextual clues to each of these classifications. Studies within various collections

[2] While most subject indexes are hierarchical, their organization is generally not directly expressed as such to the user and is barely represented to the cataloger.

have determined that human catalogers/indexers do not perform consistently across organizations – in general less than 20% of the time [Chan, 1989] is the same item given a consistent set of indexing terms from different indexing agents. If one considers the space of indexing terms from the same controlled vocabulary ascribed to an item, fewer than 50% of the terms are common across indexing agencies. This poor performance is perhaps not that shocking; in a study on the naming concepts, Collantes [1995] finds very poor agreement among user-supplied names (as low as 2% agreement) and similar performance to the above (about 20% agreement) when users are required to place a concept into Library of Congress Subject Headings (LCSH). It is not clear whether this poor performance is due to the nature of the LCSH controlled vocabulary used in the studies or simply due to differences in interpretation among the human subjects. Finally, Ellis et al. [1994] find that link consistency among human subjects in assembling a hypertext presentation using the same materials is also surprisingly low – on the order of 25%.

Cooper [1968] points out that while this indexing consistency is low, consistently *bad* indexing is even worse for an information retrieval system than inconsistent indexing. Again, it must be noted that indexing policy generally sets the number of subject assignments to be made for any given resource. Perhaps one remedy is to relax indexing policies (which may have been set in a time when computer space was much more expensive and computers much slower in processing deterministic queries) too restrictive with respect to the number of contexts assigned to an item. Plaunt and Norgard [1995] find that indexing consistency increases as more subjects are assigned. Nonetheless, the overall goal of any indexing system remains that of providing consistent, accurate classification of resources within an abstraction framework that is appropriate to both the information base and the defined user community.

Several influences have spawned a change in both the nature of documents to be indexed and the amount of information available about them. These influences derive from the availability of more powerful computers, inexpensive storage, and widespread networking. The 'paperless office' and 'virtual publishing' movements demonstrate a shift in paradigms

for information distribution and dissemination away from hard copy toward electronic means. Because much information is available over computer networks in its entirety, the need to create the library-style surrogate document is greatly diminished; the document itself can be used in a self-indexing mode. The concern is that any such self-indexing provide consistency at least equivalent to that expected from human catalogers.

Two key technologies are available as generic tools for information sharing and for indexing: Hypertext Transport Protocol and Hypertext Markup Language, the command and document display languages of the World-Wide-Web; and Wide Area Information Service, a generic distributed information retrieval engine. It is serendipitous that they interact so well; information in HTML is stored as ASCII text files, the format also chosen for indexing/retrieval in WAIS. As discussed in Chapters 4 and 5, these are the standards chosen for use in the CDIS. Within the CDIS architecture, typical library-based search can be implemented (the NEEDS bibliographic server discussed in Chapter 5 demonstrates the ease with which library-style applications can be developed using WAIS and the WWW). The main question remains, how much of the search should be based on content, and how much on deterministic indexing over a restricted vocabulary as used in library systems?

To help answer this question, a distinction between the type of information found in the CDIS and that typically used in a library information retrieval system must be drawn. The user of a library-based system probably has a notion of searching over items like books or journals. The information in the CDIS is broken into much smaller pieces. The average length of text within a Concurrent Engineering Case Studies hypertext 'page' is less than 500 words. This is comparable in size to a brief summary of a book or the abstract of a paper from a journal. In the design discussion server, comments range from a few sentences to a few pages, templates in the Concept Database can range from a few sentences to several pages including a description of design context coupled to descriptions of relations and variables. The relatively many, short entries in the CDIS makes it infeasible to index each by hand. What is needed is an indexing system that can provide a framework for automatically indexing information and, following Drabenscott's recommendations for

improving the utility of the index, to make that framework available to the user so that it might be exploited for better query performance. The above discussion is now summarized into a set of design requirements for the CDIS global indexing system:

The global index must be automatic: While judgment of the user is an important resource, user intervention should not be *required* in finding design documents of importance to the described design context.

The index must be visible: The user must be able to intervene in the indexing process to enhance the consistency of the automatic index.

CDIS information sources must be modular: Documents from the three components of the CDIS must be searchable within their own context as well as the overall design context which might combine them.

The CDIS indexing system must index unknown, outside information sources: The system must not only be open to information provided as external resources, it must also enhance the 'design retrieval' performance searches over them.

Extending the design context vocabulary must be modular: Addition of indexing contexts must result in automatic update of indexing system. Previously indexed items must respond to the new contexts when appropriate.

Extending the internal CDIS information bases must be modular: New contents added to the CDIS must immediately be made available to the indexing scheme and, through new queries using it, to users.

The user interface must be consistent among structured and unstructured information seeking: Indexing should be done in such a way that access to structured and unstructured information be based on the same general scheme. Whether the context for a specific resource has been assigned dynamically by the automatic indexing

system or by a human cataloger/indexer, access should appear to be the same.

The system must follow network-standards: Indexing methods should operate through standard interfaces so that information sources external to the CDIS can be easily integrated. It is of utmost importance that the CDIS be able to draw on resources outside of its direct purview. Implementing an indexing method based on network-standard interfaces is the best way to accomplish this.

Some of these requirements are met through the choice of implementation. Using the WWW as the document format, information is naturally modular on a page basis. With the design webs segregated into the three components of the CDIS, modularity of information source is accomplished. If, when new pages are added to any of these subsystems[3] they are automatically indexed, the modularity of the information base is assured. Many of the requirements focusing on open, network-standard information retrieval are accomplished through the use of WAIS in the implementation of the system. We now discuss WAIS in the context of the information retrieval paradigm it embodies.

7.1.1 WAIS: The Content Indexing Engine of the CDIS

WAIS (Wide Area Information Service) is comprised of a network-standard information search and retrieval protocol coupled to a generic tool for accomplishing text-based content indexing and query processing. Because it is primarily a communications protocol, the implementation of information retrieval techniques within WAIS varies among the several available server packages (i.e., WAIS prescribes that text-based queries be processed and sets standards for how the results of these queries are communicated to the user but does not prescribe methods for evaluating document - query matches). The varied implementations do not stray far from information retrieval canon; Salton (1988) provides an overview of the basics of information retrieval, summarized as follows :

[3] The Design Discussion Server and the Concept Database constantly accrue new documents. Concurrent Engineering Case Studies undergo periodic updates in addition to the introduction of new cases.

Document Representation: Each document is represented by the set of terms derived from its textual content. A dictionary of all such terms is created from the document corpus. A vector of term weights represents a document – higher term weights indicate stronger correspondence. Term weights can be derived from several features: the number of occurrences of a in the document, the number of times it occurs in the document title (i.e., "headline"[4]), the number of times it occurs in the overall corpus, and the number of documents from the corpus that contain the term (the inverse document frequency or *idf*). Weighting functions from (a) Salton [1988] and SMART [Buckley et al., 1995], (b) freeWAIS-sf 2.0 [Pfeifer, 1995], (c) freeWAIS 0.3 [CNIDR, 1995] are:

$$w_{i,k} = \frac{\left(\log\left(f_{i,k}\right)+1\right) * \log\left(N/n_k\right)}{\sqrt{\sum_j w_{i,j}^2}} \qquad (7.1.1.1a)$$

$$w_{i,k} = 0.5 + 0.5 * \left(\frac{f_{i,k}}{1 + \max_j\left(f_{i,j}\right)} \right) \qquad (7.1.1.1b)$$

$$w_{i,k} = \frac{\left(\log\left(10 * f_{i,k}^{hl} + 5 * f_{i,k}^{body}\right)+10\right)}{\sum_j \left(f_{i,j}^{hl} + f_{i,j}^{body}\right)} \qquad (7.1.1.1c)$$

where

$w_{i,k}$ is the weight of term k in document (query) i

$f_{i,k}$ is the frequency of term k in document (query) i

$f_{i,k}^{hl}$ is the frequency of term k in the headline of document (query) i

$f_{i,k}^{body}$ is the frequency of term k in the body of document (query) i

N is the number of documents in the collection

n_k is the number of documents in the collection in which term k appears

[4] Extensive work in retrieving information from news stories has resulted in artifacts of their structure appearing in discussions of document representations. For an HTML document, the <title> tag is interpreted as the document's 'headline'.

In each case, $w_{i,k}$ is set to zero when the term is absent from the document.

Query Representation: Queries are represented in the same scheme as documents are represented. Because queries have no headline and are assumed to be context-free (i.e., no 'corpus' of queries is defined), weight calculation is based only on the number of times a term occurs in a query and the number of terms in a query (i.e., $f^{hl} = 0$, $N = 1$, and $n_k = 1$).

Similarity Measure: Query-Document similarity is based on a function of the weights in the vector representations. Several typical similarity measures for comparing query Q_i to document D_j are shown below:

$$Sim(Q_i, D_j) = \mathbf{w}_i \cdot \mathbf{w}_j \qquad (7.1.1.2a)$$

$$Sim(Q_i, D_j) = \frac{\mathbf{w}_i \cdot \mathbf{w}_j}{\|\mathbf{w}_i\| * \|\mathbf{w}_j\|} \qquad (7.1.1.2b,c)$$

these measures are of the class of similarity function called *cosine* measures by Salton[1988] because they represent the 'angle' between two term 'vectors'.

In addition to these basics, some common extensions may make the information retrieval more effective:

Stopwords: Within a document collection, words that have high frequencies of occurrence in each document (e.g., *and, or, the, was, it,* etc.) do not serve any purpose in discriminating among documents in the collection (i.e., their *idf* is quite high causing very low term weight). As such, they contribute little to the information retrieval task and are eliminated from the document representation – the system treats them as if they were not in the original document.

Synonyms/Term Grouping: Where multiple terms represent the same concept, it is desirable to count any occurrence among the set as that of a single term. In this way, term weighting more accurately reflects the content of the document and not just the statement of that content. The idea of expression versus meaning is basic to effective information retrieval, synonym substitution is a simple step toward accomplishing this. Unfortunately, there really exist very few strict synonyms that are of design import. The proclivity for engineers to use acronyms creates special problems in the typical vector representation because these acronyms encode multiple terms, each of which is represented separately. No simple substitution can be made in this case.

Query Scoring: WAIS provides support for Boolean operations over subqueries. The standard similarity measures shown above provide the score for each subquery, these scores are then combined according to logical connective (max[A,B] for A or B, min[A,B] for A and B). In addition to Boolean operations literal search using term adjacency is supported within subqueries. Scores for documents are computed to lie in the range [0, 1000][5].

Relevance Feedback: WAIS also implements an extremely effective means of expressing context to the system. Documents that the user has found to be relevant can be submitted to the system as examples which augment or replace the user query (in general when augmenting a query, relevant document terms receive lower weights than the terms of the query). This also tends to combat a philosophical discrepancy in representing queries and documents in the same space when they are typically quite different. While some question the representation of documents and queries in the same vector space [Bollman-Sdorra and Raghavan, 1993], these reservations do not extend to the common representation among documents. In fact, Salton casts the information retrieval process as one of attempting to find queries

[5] These scores resemble probabilities however the use of probability theory is interestingly absent from much of the information retrieval literature. A particularly persuasive argument against using simple applications of probability theory is offered by Cooper [1994].

which return at least a few relevant documents and then using document-document similarity to produce better and better results sets.

Salton provides some empirical results from information retrieval studies of several modifications to the basic vector space model in order to prioritize the deployment in the CDIS:

Indexing Method	Impact on Performance
Basic single-term automatic indexing	benchmark
Use of thesaurus to group related terms in the given topic area	+10% to +20%
Use of automatically derived term associations obtained from joint term assignments found in sample document collection	-10% to 0%
Use of automatically derived term phrases obtained by using co-occuring terms found in the texts of sample collections	+5% to +10%
Use of one iteration of relevance feedback to add new query terms extracted from previously retrieved relevant documents	+30% to +60%

The upshot of these findings is that we must preserve (or improve) relevance feedback within the system. Creating a design thesaurus also seems promising for improving information retrieval performance, especially if this thesaurus includes frequently occurring phrases.

This summary of information retrieval has been abbreviated but includes all that is relevant to the current discussion. In summary, documents and queries are interpreted as sets of terms whose number of occurrence is used to form a measure of weight representing *aboutness*. Similarity between query and document is derived from these weights, documents highly ranked according to this similarity are returned to the user. Because documents and queries share representation, documents themselves can be submitted so

that the system might find similar documents using relevance feedback. Throughout the discussion there is little mention of concepts like interpretation or meaning. The choice was made not to bias the representation of information in the CDIS, this is still true from the standpoint of the information that is provided to the user. However, in indexing the information we are applying a bias, it is thus appropriate to discuss the impact that the simplistic vector space document representation has on document interpretation.

7.1.2 Context and Meaning

The above discussion, with the exception of the use of synonyms, focuses on the textual content of documents and queries without bringing up the real focus of information retrieval – finding documents which contain information that is relevant to the user's current information gathering needs. There is a fine distinction here between textual content – which WAIS attempts to index, and information content – the meaning of the document to the user. It is finding documents that are relevant according to the latter which is the goal of an information retrieval system, the former is merely an assumption which simplifies system implementation. The primary alternative to information retrieval methods and the bias they place on document representation are those of the natural language processing (NLP) community. There is a general feeling that NLP, once accomplished, will obviate the information retrieval techniques implemented within the CDIS[6].

Natural language understanding and information retrieval have similar goals – to distill the meaning of a document so that that meaning might be made useful to a user at some level. It is in the representation of results that the two differ greatly. The goal of information retrieval is never a complete understanding of a document but rather an understanding that is 'good enough' to identify documents relevant to the user's current context. This has resulted in the representational simplification of the vector space document model discussed above. Natural language understanding attempts to derive, from a textual document (or speech recognized version), all of the factual information that it entails (i.e., document interpretation or understanding). The methodology here focuses on micro-level structure:

[6] Wilensky often refers to information retrieval as an 'NLP-Complete' problem.

parse sentences and fragments into noun and verb phrases and attempt to garner some logical meaning in terms of actors, actions, objects of actions, declarations, etc. A main issue in NLP is *word sense disambiguation*; determining which, from a set of several possible meanings, is the most appropriate to the given textual context. Of course, this context is influenced by the overall meaning of the document which itself is derived from the (possibly ambiguous) words that constitute it. Word sense disambiguation is precisely the tool needed to infer context from user queries in our information retrieval task, but is a full-blown NLP application necessary? To decide this, we will discuss word sense disambiguation in natural language and then present its application to our information retrieval task. The common ground between the two approaches presents a clear direction for improving the effectiveness of information retrieval within the CDIS with possible future extensions to NLP.

The basic problem: given an utterance or fragment which contains words or phrases with ambiguous meaning – "the butcher has kidneys today" [Wilensky, 1992] – determine the most appropriate facts discernible from it. There are a number of choices for interpretation which imply a like number of contexts in which the fragment occurs. In this example, further information about surgery or an ailment of the kidney might promote an interpretation in which the butcher received a transplant. In a discourse about food, perhaps referring to a dish which contains kidneys, the interpretation is that of the butcher wanting to sell kidneys as food. The simplest interpretation is that which is self evident from the fragment – that the butcher's body, like that of almost all other people, contains organs called kidneys at the current moment. The latter, while being the most likely logical entailment of the sentence, is probably the least likely to actually be *communicated.*

So what is the best way to discern among these various meanings? Charniak and Goldman [1988] espouse a method by which text is interpreted in the context of a semantic model of the world. In this case, interpretation proceeds by adding facts as they occur in a fragment (in this case probabilistic 'facts') to the accumulated knowledge base which includes a large amount of background knowledge about the world. This produces a somewhat greedy

algorithm which might prematurely see the consistency between a person, of which the set of butchers is a subset butcher, having organs similar to most people and interpret the simplest consistent fact. Wilensky [1992] points out that such a method fails to distinguish that the general intent of any discourse is not to restate evident facts but to communicate new facts in an effective manner. Thus it is important *not* to be greedy about interpretation, but to use the entire document to establish the context of an utterance in order to produce the most likely interpretation. It is possible that the explanation for an expression that is most likely is also so self-evident that it is unlikely anyone would attempt to communicate it. Consideration not only of the whole of the document context but also of the purpose of discourse itself is the crux of the above example and a primary concern in word sense disambiguation.

So how do NLP and information retrieval relate to each other? Because they must focus on overall document context and not on local sentence structure, techniques for word sense disambiguation are quite similar to those used in information retrieval. Schutze [1992] uses the vector space representation common in information retrieval to distinguish the various contexts which aid interpretation. In his investigation, the local presence (within a paragraph) of as few as three additional terms defines the context, through the cosine function discussed above, in which different senses of the same word are used. Relaxing the representation of a textual document from that of natural language understanding to the less expressive representation of the term vector space preserves the document's overall context to the point that word sense can be determined. Similar representations are operated upon by different similarity measures in investigations using activation networks [Gallant, 1991], semantic networks [Vorhees, 1994], and dictionaries/thesauruses [Vorhees, 1993]. Thus, instead of being replaced by NLP, information retrieval techniques are vital to its success.

Perhaps the main difference between information retrieval and NLP techniques can be seen in the scale of the undertaking. For CYC [Lenat, 1995], a system designed to capture 'human reality', text is read into the system and interpreted. The system then responds by

restating its interpretation of the text, leading to human correction and patching of the knowledge base. This is among the most ambitious AI projects ever undertaken, CYC includes thousands of semantic relationships in order to create a knowledge model that knows enough to be able to interpret and learn from text. In contrast WordNet [Miller, 1995], while containing several hundred thousand words, provides only six semantic relations among them: synonymy (similar), antonymy (opposite), hyponymy (subordinate), meronymy (part of), troponymy (manner), and entailment. While this has proven effective for word sense disambiguation and as a means of augmenting queries for information retrieval it falls far short of Lenat's stated goals. So while information retrieval may be an 'NLP-complete' problem, NLP itself is far from solved at this point. However CYC does raise the issue of learning context from texts a discussion of which follows.

7.1.3 Learning Context

The main distinction between the approaches taken in the natural language literature and that within the information retrieval literature is the means by which the background knowledge used to define context is derived. In general, natural language understanding applications interpret text into a set of logical constructs that extend a knowledge base of previously known or interpreted 'facts'. The goal is induction; generate theories that most consistently account for both the structure of the discourse (syntax) and its meaning (semantics). Information retrieval techniques for learning context from textual corpi often exploit parallel language translations to label word context/sense mappings to create an effective training set. Here, a word whose interpretation is ambiguous in one language might be represented in a second language text by a set of contextually unambiguous words (e.g., Canadian government documents distributed in both French and English are used by [Schutze and Pederson, 1994]).

Machine learning in a non-hierarchical abstraction framework has been explored in experiments for the Text Retrieval Conferences (TREC) which focus on the retrieval of information from various news sources. Here a set of predefined topics, or 'routing

queries'[7], are used as benchmarks to evaluate the query performance of participating systems over a known corpus. This corpus is labeled with relevance mappings from document to topic so that the retrieval set can be evaluated as to how relevant its results are to standard tasks. This labeling of the document space provides the necessary impetus for applying machine learning techniques. Many different experiments were performed with a variety of parsing strategies (e.g., standard vector parsing, noun phrase extraction, full NLP-type parsing), query elaboration, and scoring metrics. The surprising result of the overall experiment is that the varied methods yield increasingly similar results. One experiment in particular shows the nature of the task. For TREC topics like that shown as Figure 7.1, research groups build on the topic descriptions to create an appropriate query; Cooper [1994] compares this method to one based on machine learning and another in which human subjects assembled queries. The finding is that perhaps the TREC topics are

Topic 111 Narrative: A relevant document will report on efforts by the UN's International Atomic Energy Agency to monitor compliance with the Nuclear Non-proliferation Treaty, or, report on efforts by the United States, Britain, France, USSR, India, or China to control the transfer of technology, equipment, materials, or delivery systems to nations suspected of nuclear weapons development programs, or, a relevant document will report any nuclear activities by Argentina, Brazil, Iraq, North Korea, Pakistan, South Africa, of Iran (all suspected proliferators)

Topic 122 Narrative: A relevant document will report on any phase in the worldwide process of bringing new cancer fighting drugs to market, from conceptualization to government marketing approval. The laboratory or company responsible for the drug project, the specific type of cancer(s) which the drug is designed to counter, and the chemical/medical properties of the drug must be identified.

Topic 131 Narrative: A relevant document will provide specified data on contracts awarded to McDonell Douglas for the production of military aircraft for any nation, any military service, or of any aircraft type (interceptor, fighter-bomber, helicopter, etc.). The contract must be for completed airframes, NOT contracts for aircraft development, factory tooling, components, spare parts, services, etc. To be relevant, the document also must specify the number of aircraft to be delivered, the dollar size of the contract, and the aircraft type sought.

Figure 7.1 Typical TREC Topic Narratives. Note that each of these narratives is an expression of the conditions under which a document will be judged relevant.

[7] So called because they simulate the routing of messages from information source to those concerned with tracking a specific topic.

not a descriptive enough base from which to build, that introducing context from hand-coded queries is a better starting point.

Having established that the CDIS will employ a hierarchical controlled vocabulary for capturing design context, we now move from the generic setting to one more closely resembling the CDIS. An interesting application of labeled learning using the vector representations of the information retrieval paradigm is discussed by Plaunt and Norgard [1995]. Human indexed document surrogates – title, author, and abstract of computer science technical reports from INSPEC are used to generate co-occurrence patterns among terms in like-indexed items. Strong co-occurences determine a set of contextual clues that can be used to automatically index new documents. Basic single term-index mappings result in indexing consistency measures in line with that expected from human catalogers. (Again it is important to note that consistency here is non-judgmental; a poorly indexed training set will directly impact the quality of automatic indexing.) Recommendations from the results of this study address two considerable handicaps discovered: the hierarchical nature of controlled vocabulary terms is not expressed in the representation of these terms; it does not exploit combinations of terms, significant since many important concepts can only be expressed by multiple terms (think of how many three letter acronyms are important in science in general and computer science in particular).

7.1.4 Discussion

Vector space representation is useful in determining the context of a document whether that context is to be used for natural language interpretation or simply for document scoring in information retrieval. In both cases, context is taken to be the idea behind expression. It is a useful simplification of the actual content of a document. Wilensky maintains that the purpose of discourse is is to transfer ideas or information to others and thus it is not generally practiced with deception. Perhaps then, the simplistic concept of meaning derived from a vector representation is enough to represent document context for our purposes. The main distinction between information retrieval and natural language understanding is that

while the former attempts to present the user with information sources, the latter strives to interpret this information. The nuance of design information and design process that is encoded in natural language representations of design cases presents a formidable challenge to a system trying to actively interpret it. Regardless of whether design interpretation is realizable or not, implementation of the CDIS takes the first step toward meeting this challenge by concentrating on document context from an information retrieval standpoint.

In the previous chapter, the utility of a hierarchically defined collection of component types is demonstrated as a means for manipulating design abstraction. IRTD is coupled to this hierarchy to help the designer direct the reduction of ambiguity by focusing attention on portions of the design model that make significant improvements in the desired design objective. In addition, probability-based design space modeling relaxes some of the mathematical formalism required for using IRTD in general. While the information navigation model for the CDIS global index resists the mathematical modeling used in the previous chapter, the utility of defining design hierarchies to model the transition in abstraction that permeates the conceptual design process remains. The strategy that IRTD encodes, that of assessing the information necessary to make objective design decisions under uncertainty, remains valid even when that uncertainty cannot be quantified. Instead of using artifact values from the component database to make this decision, the CDIS turns to the collective experience of design cases; queries against a library of cases with specifications similar to those in his current project can direct the early stages of conceptual design by uncovering important issues and suggesting possible solution paths. All of this must be done over a representation that does itself provide useful structure and fights interpretation by machine through its use of abstract, informal language (free text). Therefore, structure that was added to the Concept Database through an abstraction hierarchy of component types is extended throughout the rest of the CDIS through similar abstraction hierarchies based on concurrent engineering design issue and on design function.

7.2 Creating the Conceptual Design Controlled Vocabulary

Having proposed that a hierarchical set of controlled vocabulary terms is useful for indexing design information within the CDIS, the derivation of this set of terms is an important issue. With respect to typical case-based design applications, this set of terms is analogous to the parameter axes chosen to characterize a design and upon which case similarity is measured. The goal of the CDIS is to extend this artifact-centered case-based reasoning into conceptual design by operating over design process information (how) in addition to design artifact information (what). Also, because a primary activity in conceptual design is synthesis, the artifact-centered information must be made available at the varying levels of abstraction used in it synthesis. The CDIS strives to capture the abstraction level at which this information is applied in it own design so that new conceptual design instances are not encumbered by overly concrete (or abstract) representations and evaluations.

In order to accomplish this, the CDIS controlled vocabulary is faceted: it contains artifact-based function hierarchies, component hierarchies, and design issue hierarchies. These separate concept classification trees are used to 'triangulate' the context of documents from the three main components of the CDIS. Function hierarchies are used to represent the system design which is encapsulated by the design discussion server, providing points of reference from which a user can navigate through local design processes. Component hierarchies, as noted previously, help the concept database organize the abstraction level for mathematical models used to evaluate/analyze components within a design context. By augmenting the component taxonomy developed in the previous chapter with descriptions of these models (often including important parameter and issue definitions), a valuable, user-extendable source of background knowledge can be exploited.Finally , design issue hierarchies provide a concurrent engineering framework that is the basis for the engineering case studies portion of the CDIS. While each of these indexing facets appears to be associated with a single component of the CDIS, this is not the case. The issue hierarchy also helps to organize rationale and evaluation within the design discussion server and the concept database templates by classifying multidisciplinary constraints and evaluation metrics. Clearly the function hierarchy extends into case studies of engineered products as

well as into the selection processes which seek to encode performance metrics by which the degree to which a particular function is accomplished can be objectively scored. Certainly engineering component permeate the design process. However, it is this process hierarchy that is the real value-added of the CDIS since it is the main axis by which one can index into 'generic' design discussions or case studies and abstract away useful information about the design process – extending the case-base reasoning beyond artifacts and into design.

The development of the component and function hierarchies is relatively straightforward (although not simple). They differ in both the information base to be indexed and the method by which the abstraction hierarchy is constructed. This difference is illustrated in the two basic classes of indexing methods in use in information retrieval: *assigned* and *derived* indexing [Paijmans, 1993]. Assigned indexing does not generally take into account the nature or extent of the information in any particular database, it attempts to capture and organize generic world 'knowledge'. An example of this is the Library of Congress Subject Headings: these are consistent across all libraries in the country regardless of the individual collection content within any one library. This is effective for making information available in a standard form, enabling easy access to all resources by supplying a standard contextual interface – this is the goal of the component index. Derived indexing is done on more specialized collections whose various contexts are not effectively captured *a priori*. The collection must be analyzed and a conceptual framework can be derived from its content: documents within the collection are then indexed based on this derived set of contexts. The function index uses this derived indexing because function is so context dependent.

7.2.1 The Component Index

In many design instances, the task of the designer is to assemble and integrate these previously defined components into a system that satisfies design constraints while providing some measure of optimality with respect to desired performance. This process is particularly prevalent in the target domain for the CDIS – mechatronic design. In this

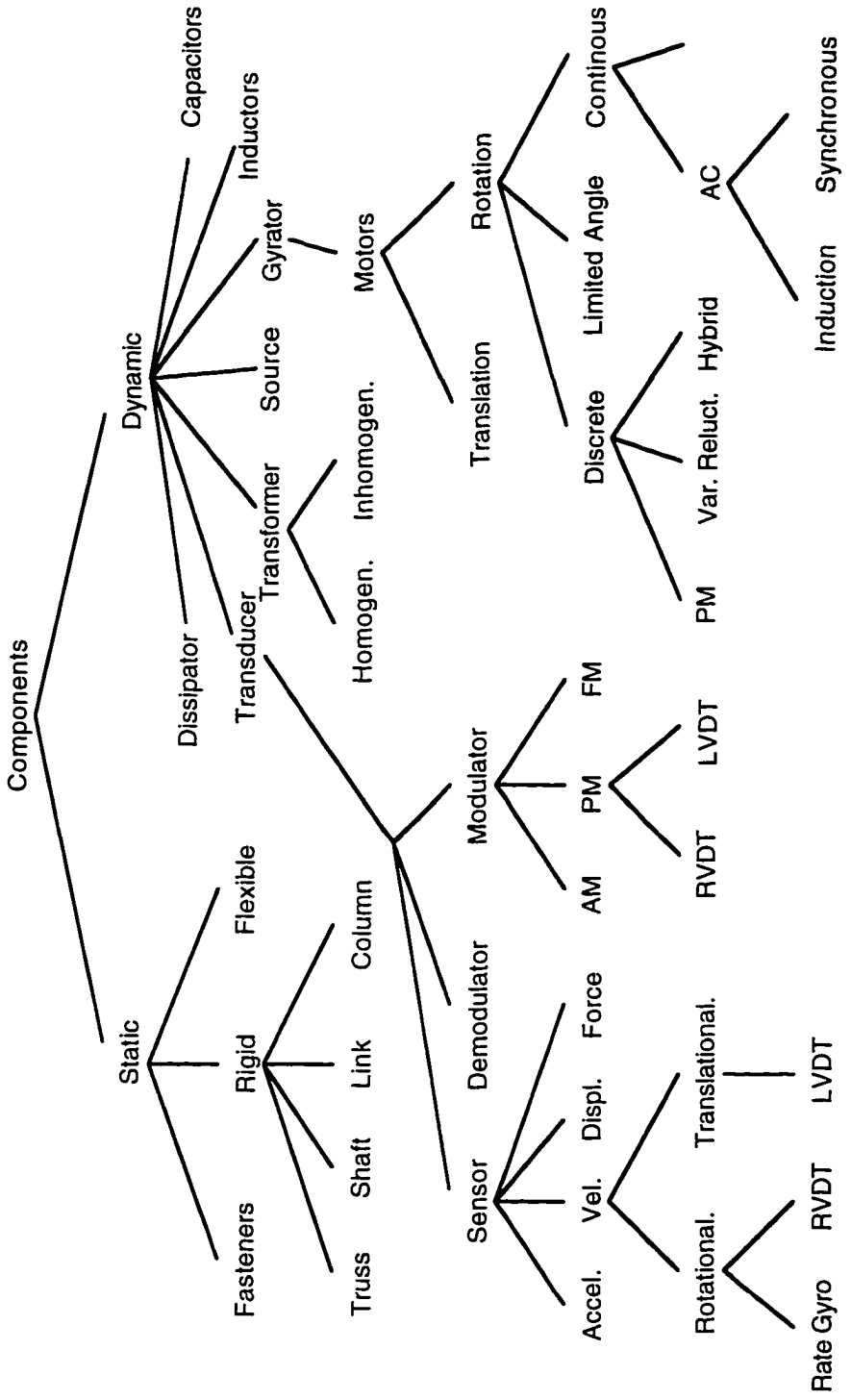


Figure 7.2 Expanded Component Hierarchy. Motor hierarchy from previous chapter (in upper right corner) is put in context within the CDIS component abstraction hierarchy.

domain, standard sensing and actuating elements are combined with some sort of intelligent controller (typically a standard computer or off-the shelf controller – at least in the conceptual prototyping phase of design). The set of components available for use in a mechatronic design is large but known (at least to a database); the controlled vocabulary hierarchy used to classify this set can take advantage of assigned indexing. The basis for this component index and its implementation of component abstraction are discussed in the previous chapter. The relationship used to manage abstraction within the component index is the operating principles of the components themselves – these make up the parent-child relationships that encode the hierarchy itself. Augmenting the hierarchy is a description of the component along with the related design variables involved in its specification and evaluation. Descriptions of these variables tend to overlap somewhat with the function and issue indexes, providing an automatic semantic link among the three hierarchically separate indexes. An expansion of the component hierarchy featured in the previous chapter is shown as Figure 7.2.

7.2.2 The Function Index

Function is an important aspect in conceptual design. A great deal of the design activity revolves around the transformation of loosely specified design requirements into a functional specification. Tomiyama [1995] proposed a model for design consisting of a formal mapping between requirement and artifact, eschewing the functional element that existed intuitively but seemed extraneous. After performing a set of design protocol studies this position has changed: function was found to be of primary importance in the design process and cannot be ignored as an intermediate representation between design requirements and artifact attributes. Gero [1990] has proposed a design indexing scheme in which a mapping between structure, behavior, and function (SBF) defines an artifact. In the CDIS, behavior is indexed through the modeling relations stored for a component in the Concept Database. In much of mechanical design, structure accomplishes specific functional tasks, therefore the CDIS folds much of structure into the functional index along with all of the other functional specifications. By combining contexts from both the

function and component indexes, Gero's SFB mappings can be modeled effectively.

Relationships among the constitutive elements of the design are not subject to *a priori* classification beyond the sensor / actuator / controller / housing classes described in the component index discussion above. It is the main purpose of the function index to capture higher level relationships which have shaped the design process in any particular instance. In order to capture and index these relationships, an index must be derived for each design known to the CDIS to describe the hierarchical functional decomposition of the artifact. This index provides the mapping from artifact attributes back into the design process that created it. Functional decomposition also partially describes the design process through the varying level of abstraction that is captured in the hierarchy: at the top are loose design specifications which give way to more distinct functions until leaf nodes specify distinct behavior of a subsystem.

This derived index can be used to search a design discussion – primarily a design process-centered hyperdocument – along typical artifact characterization. Because the decomposition hierarchy has 'leaf' terms that are common to the generic component hierarchy (most mechatronic functions end up being accomplished by components), the two indices form a natural link between the design discussion server documents and the concept database documents. Most important in this mapping is a 'reverse' link that leads from component selection back into the context of the selected components to provide insights toward the design rationale of the application of a component. While much of this information should be captured in the Concept Database design template, searching the function index allows the user to backtrack through the design abstraction levels leading up to component selection. Figure 7.3 shows a typical decomposition for the MADEFAST Seeker design discussion used as a test bed for the design discussion server.

A similar method of artifact-based derived indexing scheme is employed in the DEDAL project [Baya, et al., 1992]. Here, a decomposition hierarchy is used as a model for the design and linked into graphical representations of the final artifact which is used as a

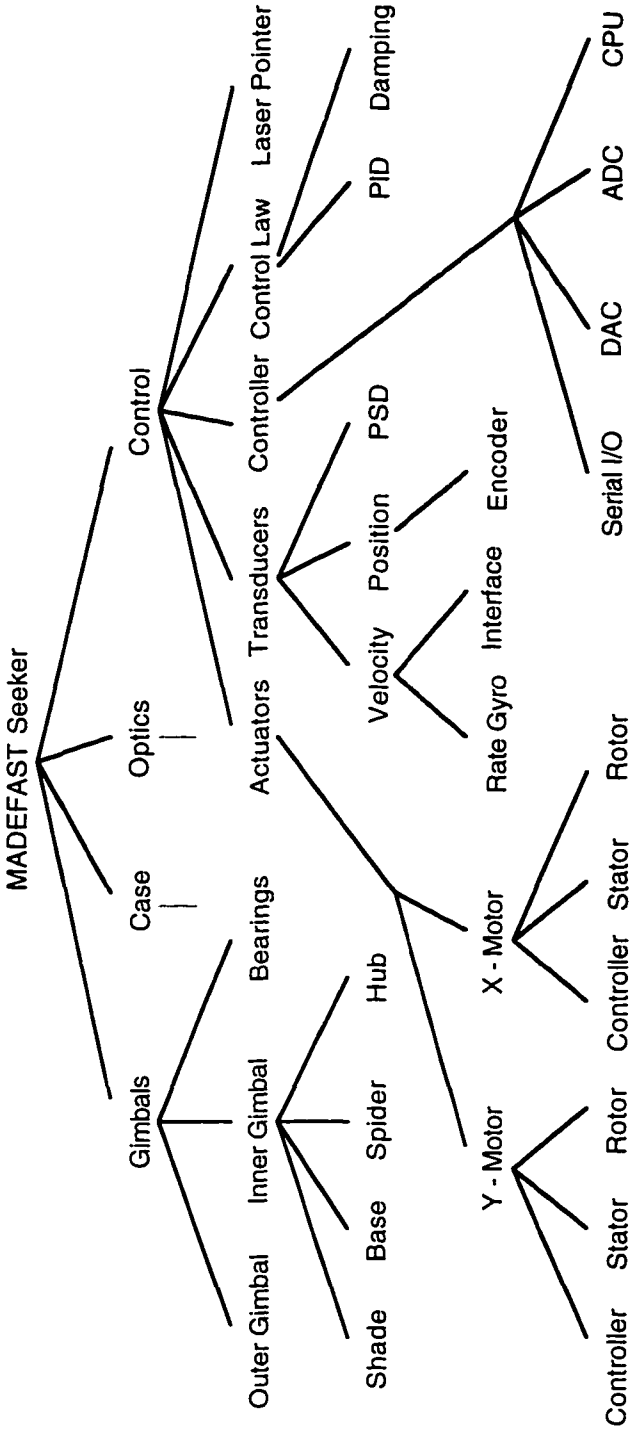


Figure 7.3 Functional Decomposition Hierarchy for MADEFAST Seeker. Each system is broken into subsystems whose function is described in the database.

navigational backbone. The main focus of DEDAL is to create a set of design documents useful for redesign of an artifact (essentially case-based reasoning where the case base has only one member – the previous design). Based on studies of information requests from designers, the developers offer a second index into design documentation based on the level of detail expected (in decreasing level of abstraction): conceptual, configuration, detail. The design documents DEDAL indexes are (loosely) constrained to contain information in the following format which provides a design 'snapshot':

Description - An unformatted section used for a free-text description of the element, including product information for the final of instantiation of a component.

Function - Textual description of the function of the element in question, including system decomposition information and hyperlinks to documents discussion these decomposed elements.

Requirements - More concrete representations of design requirements which include parameter - requirement - rationale (here rationale refers to the reasoning behind the requirement) triples, in formatted but not machine understandable representations. These requirements are indexed into Functional, Physical, and Other categories.

Technical Alternatives - Description of the set of alternatives examined for use in the application. This includes component class distinctions along with alternatives within the chosen component class. In some cases, this component class is not analogous to a 'leaf' of the CDIS concept database component hierarchy loosely expressing abstraction.

Decisions and Rationales - Major decisions made in the selection of the final design along with the reasoning behind these decisions.

This document format is essentially a prescription for a design report, ostensibly done at the conclusion of a design (or the conclusion of the specification of a particular subsystem)

but potentially evolving with the design as 'living' documentation[8] . In addition to artifact decomposition and information abstraction level, documents are indexed according to media type and, for the above mentioned document structure, design information category (i.e., Description, Function, Requirements, Technical Alternatives, Rationale). This indexing is not automatic, it is performed by hand. In addition, the documents that are indexed are not generally those created in the act of designing but retrospective summaries of design decisions. The design discussion server version of the same set of documents provides further insight into the design process, insight that might be garnered from DEDAL if links back into the design evolution were supported. (Note: will give several URLs here to support the DDS vs. DEDAL vs. Hypermail archives as a representation). As DEDAL currently stands, the evolution of design issues is not adequately captured nor are the strategies used by the designers for navigating the design space. The design discussion server captures both by supporting the discussion that shapes design decisions. Augmenting the standard IBIS argumentation transitions of Issue, Position, Argument with a Summary statement similar to those indexed under DEDAL seems to provide the best of both; this is the current design discussion server implementation.

7.2.3 The Issue Index

We have described two indexes this far. The component index is an assigned one in which the underlying documents are categorized according to a rigid standard – the set of mechatronic components. The function index is derived from the document base of the design discussion server because of the extreme context-sensitivity of system and subsystem function. As such, function can be defined as a design proceeds, a relatively easy task when the abstraction hierarchy is a simple as that shown in Figure 7.3 above. This exercise is also not without value for it expresses information that is not readily discernible from the design discussion but is relevant throughout the design process. Indexing in the design discussion server provides information access *during* the design

[8] In fact, by tuning the discussion transitions to match the document specifications outlined above, the CDIS can directly support the creation of DEDAL documents during design. The implementation has been developed to allow the design leader (discussion owner) to impress prescriptive practices of varying rigor, from IBIS to DEDAL and beyond.

process, making the current design available as a design case even before it is complete.

We have climbed the ladder of abstraction from discrete components to function and now must tackle the task of creating an abstraction hierarchy that models concurrent engineering design issues that shape the overall process of conceptual design. These issues are related to the context-sensistive issues in that they represent the generic functional aspects of a design from the standpoints of the varied project stakeholders. For instance, serviceability concerns like lubrication, inspection, calibration/adjustment are generic and are included in the issue index while the mapping from these issues to subsystems that implement them would be pertinent to the function index.

The CDIS supports a design process modeled as two primary activities: IRTD – the accumulation of design information in response to decision making under uncertainty involving performance metrics, constraints, and a finite set of design alternatives; and IBIS – the use of argumentation to drive the formation of these concrete design requirements and objectives from abstract design issues. The previous chapter implements IRTD through the storage of component selection and modeling information from prior designs and further applies the methodology toward navigation within the component abstraction hierarchy using information from the CDIS Concept Database. Along with the above described functional and component abstraction hierarchies, these methods help to structure information closely related to artifacts.The process-oriented index must bear a close relationship to the design process framework chosen for implementation in the CDIS – IBIS.

Recall that IBIS stands for Issue Based Information System. It seems natural then, that a third abstraction hierarchy be used to encode the design issues at the center of the IBIS design process model. In the transformation of case-based reasoning to a case-based design assistant, the main addition to ARCHIE [Domeshek et al., 1994] is just that, a sensitivity to design issues so that experience from one design process can be transferred to designs which revolve around similar issues but may be concerned with completely different

- Life Cycle >
  - Design >
    - Concept Generation >
      - Brainstorming >
      - Issue Identification >
      - Decomposition
    - Service Conditions >
      - Load >
      - Environment >
    - Dynamics >
      - Vibration >
      - Control >
    - Statics >
      - Strength
      - Deflection >
      - Load Type >
    - System Reliability >
      - Mean Time Between Failure
      - Repair Rate
      - Availability
      - Fault Tree >
    - Failure Mode >
      - Environmental >
      - Wear >
      - Fatigue >
      - Plastic Deformation >
      - Elastic Deflection
      - Load Instability >
      - Combination >
      - Impact >
  - Marketing >
    - Identifying Customer Needs >
      - Customers >
      - Eliciting Product Needs >
    - Competitive Benchmarking >
      - Performance Testing
      - Reverse Engineering >
      - Competition Map
      - Niche Identification
    - Market Prediction
  - Distribution >
    - Sale >
      - Packaging >
      - Pricing
    - Delivery >
      - Lead Time
      - Inventory
      - Acceptance Procedures
  - Manufacturing >
    - Forming Processes >
      - Primary Forming Processes >
      - Secondary Forming Processes >
    - Assembly Processes >
      - Fastening >
      - Fixturing >
      - Time - Motion
      - Part Count >
      - Part Insertion >
      - Part Handling >
      - Assembly Line >
      - Assembly Method >
    - Process Control >
      - Work Station >
      - Buffer
      - Quality Control >
      - Transfer Device
    - Finishing Processes >
      - Cleaning >
      - Coating >
  - Service >
    - Maintenance >
      - Lubrication
      - Adjustment
      - Cleaning
      - Access
      - Identification
    - Repair >
      - Diagnosis >
      - Repair Action >
      - Diagnosability
    - Upgrade >
      - Component Replacement
      - Modularity
  - Retirement >
    - Waste Stream
    - Recycling >
      - Manufacturing Waste >
      - Packaging Waste >
      - Service Waste >
      - Used Product Waste >
    - Disposal

Figure 7.4 Partial Issue Abstraction Hierarchy. Shown in outline form for clarity, the issue hierarchy expands further at each leaf demarked with a ">". Over 500 nodes are used to contextualize design issues.

function or artifact specifications. Issue indexing requires an organization of design issues in a way similar to that by which components and artifacts are organized but does not suggest a straightforward method for accomplishing this organization. Instead of functional decomposition or component abstraction that can be based directly on the class of design artifacts in question, the issue index must embody those topics which have shaped the design of these (and other) artifacts. Perhaps the best starting point for such an index is the current industry 'best practice' of concurrent engineering. Here, the typical concerns of all of the design stakeholders are captured and decomposed in the same way that function was decomposed for artifacts. Figure 7.4 shows several levels of the issue index used in the CDIS. Perhaps an indicator of the spirit of this index is shown by the incorporation of teaming and collaboration issues in addition to the more technical concerns of the various stakeholders.

Showing the index is one thing, but how can we be sure that the index is valid? It was relatively straightforward to create component indexes – all that was needed was a survey of existing components and some organizational backbone around which to hang them. For design functions, it was an easy decision to require a functional decomposition for all designs stored within the CDIS; this can even be done fairly early in the design process making functional indexing available almost from the onset of a design. But what is the source for the issue index? Because they were created for use in an academic setting where practical concerns are sometimes under-emphasized, standard indexes like INSPEC and the Engineering Index make little mention of concurrent engineering or other important aspects of engineering practice. The issue index must be constructed 'from scratch' specifically for the CDIS. And because most of concurrent engineering focus has been on 'Design for X' (a.k.a. DFX, where *X* can be any part of the product life cycle: manufacture, recycling, maintenance, etc.), the resources must be gathered from disparate sources [Boothroyd and Dewhurst, 1990; Boothroyd et al. 1991, Ruff and Paasch, 1993; Marks et al., 1993; Ishii et al., 1992]. Again while the *X* is generic, the way in which the DFX concerns are integrated into the traditional design function are highlighted in the function index.

The basic issue in any of these DFX efforts is that the designer understand (or at least be sensitive to) the processes that are required to realize, use, and reuse the design he is creating. For building the part of the index that concentrates on traditional design and manufacturing process issues, the primary information source is Kutz' *Mechanical Engineer's Handbook*[1986], a standard reference. These two important stakeholders are represented by hierarchies of issues created by condensing the subheadings of the various pertinent chapters. Murty and Jain [1995] successfully use similar document headings to form a conceptual map for clustering technical books in a library collection, so the notion that section headings can comprise an index is substantiated. For some of the less-traditional stakeholders, technology has outstripped this standard reference so the issue hierarchies for them have been augmented using resources like textbooks [Ulrich and Eppinger, 1995], design guides, journal publications [see DFX references above], or common sense[9]. In some instances, material from the concurrent engineering case studies has been used to augment the issue abstraction hierarchy. Sources like the Engineering Index Vocabulary [1990] and the NASA Technical Thesaurus [1988], much more general than our concurrent engineering issue hierarchy, are still useful for standardizing terms and mapping among technical synonyms. The general structure is a synthesis of those local abstraction indexes generated from information sources for each topic.

The CDIS is designed to support collaborative conceptual design based on concurrent engineering principles. The core of the hierarchical design process vocabulary is based around generic issues from throughout the design life cycle. In addition to this multidisciplinary core, more disciplinary design issues are introduced (in the case of our demonstration system these are limited in scope to the current design information base). As with the two other indexing vocabulary systems, the taxonomy ranges from the abstract (e.g., manufacturing issues) into the concrete (e.g., using part symmetry to reduce part count and/or ease assembly tasks). Figure 7.4 shows design issues organized around concurrent engineering and life cycle design; this index accomplishes much the same

[9] For issues like serviceability, very little material exists beyond the common sense issues like adjustment, maintenance, wear, etc.

functionality as the hand-built indices in the Archie system or the local indexing of the concurrent engineering case studies. Lessons from previous designs can be accessed by stakeholder and issue simply by highlighting desired aspects of the design life cycle[10] in the CDIS global index.

7.2.4 Discussion

Another question is: What is there within the design discussion server or the concurrent engineering that can be applied to new design instances? How the design process is cast helps determine the type of indexing that should be used. In DEDAL, the process is hidden behind a hypermedia design report structure. In the CDIS the process *is* the design report, whether in standard IBIS or IBIS augmented to include DEDAL-type decision summaries. The goal of the artifact index is to structure the process information into a more usable form after the design is complete. While the bulk of the design process information is embedded in the local structure of the design document, it is still important to be able to identify important issues from among a large set of such design documents.

The development of three abstraction hierarchies has been undertaken with the goal of describing design context. The tacit assumption is that by placing a designer at a particular point in each of the hierarchies not only will his current position be captured but also the possible directions of travel. Another assumption is that the three design hierarchies triangulate this position accurately enough that they can be used as the basis for an information search over the resources of the CDIS to retrieve useful or relevant design information.

In this section, the two main strategies for document indexing were introduced – derived vs. assigned indexing. Their use was demonstrated in the development of the three abstraction hierarchies used to index design context. While it is clear that assigned indexing is an important aspect of the CDIS global index, WAIS in particular and information

[10] Additional structured indices can be added to the design discussion server to classify user's and their comments along stakeholder lines as well.

retrieval in general embody a very simple derived indexing system where terms collected from the contents of the database are the only indices offered for retrieval. No terms outside of the particular collection are used as indexing, everything comes from within it. Because the WAIS engine already supports derived indexing, we turn to the steps necessary to apply the controlled vocabulary described above for assigned indexing in the CDIS. The next section describes the steps taken to transform WAIS from a derived indexing system into one which accomplishes the necessary mix of derived and assigned indexing.

7.3 Augmenting WAIS to Create the CDIS Global Index

The architecture of the CDIS includes WAIS as the information retrieval engine. The declarative, assigned indexing approach to context representation must be woven into the WAIS protocol which is primarily a derived indexing system. WAIS implements the basic information retrieval services using vector term representations for documents and queries. In addition , queries can be structured using Boolean and positional functions. At some point in the process of using the CDIS global indexing system from the time a free-text query is entered to when it is used to calculate query-document similarity measures, the context matching process must take place.

The canon of *Saltonian* information retrieval may not pay adequate attention to the difference between the nature of queries and the documents that are indexed. In the above discussion of information retrieval techniques, the basic idea of document similarity is patched so that words that don't discriminate among documents can be discarded from the representation. Another patch is the transformation of the representation to handle strict synonyms. But are these patches enough to justify the assumption that queries *should* be represented in the same way that documents are. After all, Salton himself maintains that the main purpose of an initial query is to find at least one relevant document that can then be used, through similarity matching, to find others. Recently many of the canons of information retrieval have come under scrutiny, not the least of which is query representation. Bollman-Sdorra and Raghavan discuss the implication of using the

document vector space as the common representational medium. The main question is how to map queries from their natural representation into a form that more closely resembles documents. Alternatively, should documents be cast into query space or is there an intermediate space that more effectively represents both?

Two main concepts in information retrieval deal address alternative representations for queries and documents. Working toward mapping queries into document space, elaboration of the content of a query is performed to map it to a representation that maintains context while expanding the query vector to a size useful for comparison to documents. Transforming the document into query space – automatic indexing – assigns to each document a context from a controlled set. Queries can then be generated from these contexts (this is essentially the same as library subject searches except that the documents are assigned contexts automatically instead of by hand).

Within the CDIS, hand indexing each document presents potential problems due to the number of documents and the degree of engineering expertise that a cataloger would need to hold in order to properly assign context. This is in contrast to the degree of expertise typically required to categorize library holdings because their controlled vocabulary subject headings are coarse compared to those in a specialized index like that of the CDIS (e.g., they are able to distinguish between neural networks and television networks, but not among various implementations or applications of neural networks themselves). Library catalogers are armed with some general knowledge – they are probably a good approximation of the average library user – the list of controlled vocabulary words, synonyms, related terms, and an additional list of 'catch' words for each heading. For our application, concurrent engineering conceptual design, a similar set of information could be made available to a cataloger but it would be difficult (perhaps just too expensive) to include detailed knowledge of engineering in this arsenal.

The approach taken in the CDIS to perform indexing is to use a specialized WAIS database as the document 'cataloger': a hierarchical set of controlled vocabulary terms which is

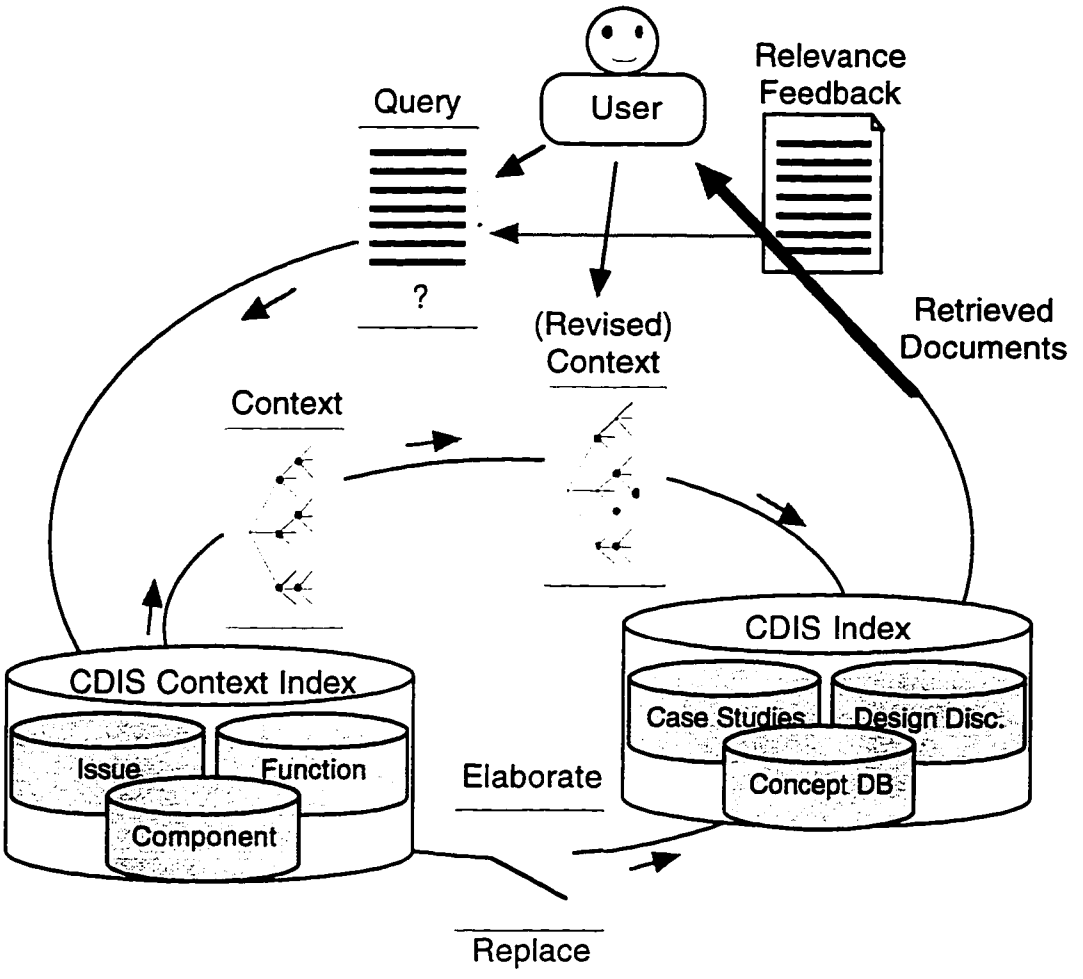


Figure 7.5 Flow of information in CDIS global index. User Query generates context, which can be edited. Context can then be used to replace or to elaborate original query when searching CDIS information database.

augmented by related terms and key phrases. An interior 'loop' that additionally identifies a query context using this same WAIS database is added to the retrieval process. Figure 7.5 demonstrates the operation modes of this loop: the user enters a query (perhaps a whole document), the system matches terms from this query to terms in the controlled vocabulary, those terms are evaluated for likely subjects and these subjects along with related terms and phrases appended to the query as literal search terms. The augmented query is proposed to the user, who can alter it before sending it to the CDIS content server which treats it as if it were a 'normal' WAIS query. A discussion on query elaboration in general and its implementation on the CDIS follows.

7.3.1 Query Elaboration in the CDIS

Query elaboration takes on many forms. Gauch and Smith [1993] have applied expert system technology, decomposing the user query into Boolean operations among subqueries over various available document indices. These subqueries are elaborated upon using a 7,000+ term thesaurus to add in contextually similar terms which might provide a better mapping into the vector document space. Voorhees has taken a similar tack, first using a semantic network thesaurus, WordNet [1993] to elaborate user queries and later adding a part of speech tagger to enrich the vector representation [1994]. Qiu and Frei [1993] apply clustering techniques to term occurrences in a corpus, attempting to learn 'concepts' (sets of related terms) that are then used to elaborate queries based on a typical IR similarity scheme. A recent explosion in these methods has resulted from the format used for the TREC-3 conference. As previously mentioned, for the routing queries TREC defines a set of standard subjects into which incoming data is parsed. With TREC-3, a large knowledge base left over from TREC-2 provided a set of document-subject relevance mappings over which similar conceptual learning can be used. Performance among all participants increased. Analyzing the performance increase, several of the groups realized that terms used in query elaboration have little correlation to those used in the actual subject description. For this reason, Cooper and Chen [1994] abandoned the original subject statements as a basis for query elaboration, turning to human expertise for the creation of query elaboration 'concepts' for which no previously determined mapping existed (i.e these were new subjects added after TREC-2). The performance of this system was better than most of the more complicated methods for handling the new subjects.

The CDIS database, at this point, does not include a predefined set of document-context mappings from which to learn. Query elaboration blends the ad-hoc methods Cooper found successful with the more structured, thesaurus-based methods. Here, the abstraction hierarchies provide both information sources. Descriptions of each context include keywords which encode heuristics about terms that tend to distinguish a particular context from others. The structure of the hierarchy can be exploited toward this effort as well –

term 'parents' and 'children' supply additional context-defining terms that can be added to a query. Finally, a thesaurus in which synonym mappings can take place over phrases is supplied. With the application of this additional knowledge toward fleshing out a more complete representation of a query's intended context, the CDIS approaches a uniform representation for documents and queries. But this is just one way in which indexing in the CDIS can be improved, automatic indexing of documents is an important method as well.

7.3.2 Automatic Indexing in the CDIS

Having an established set of terms to define the context of design information, the CDIS can exploit automated indexing technique by which documents are assigned a set of contexts through a WAIS query. The difference between the this and the query elaboration techniques is subtle. Basically it is a matter of when the elaboration takes place: automatic indexing assigns a design context to a document which can then be searched deterministically, query elaboration adds terms into a typical free-form information retrieval query. In a sense, automatic indexing is applied during query elaboration to contextualize the query so that additional terms can be added. In document-document similarity comparisons, both can be mapped into the design context space where the context terms from the three abstraction indices are used to calculate *design* similarity.

7.3.3 An Intermediate Representation for Design Context in the CDIS

In Chapter 3 the bias used to represent cases for CBR design systems bias was emphasized as an important factor in their application, we now rejoin that discussion. The controlled vocabulary created for the CDIS as the set of three abstraction hierarchies represents a bias toward querying over the contexts represented in it. As with any representational bias, it can be both a boon and a curse. It is a boon in terms of distilling out relevant *design* features from both queries and documents. It can be a curse if it does not adequately represent the important features. However, because this bias is applied under the supervision of the user and can be augmented by additional natural language. The language of the query manipulates the standard bias to cover topics not foreseen in the development

of the controlled vocabulary. For non-design queries, the system performs like a standard information retrieval system. For design queries, a specific context set is used to determine similarity of context from query to document. Basically, the global index of the CDIS provides an intermediate representation system which bridges the gap between the vector representation of the full text of the documents in the database and that of queries. Transforming similarity along the expressed representational bias of the design context set transforms information retrieval to *design* information retrieval.

7.3.4 Implications of the CDIS Global Index

Because the global index is implemented within the framework of classic information retrieval systems, the methods that is employs are useful for information sources outside of the direct purview of the system. Because WAIS provides a network-standard protocol for information retrieval, the controlled vocabulary can help create design queries on any WAIS (or some WAIS-like) database(s). Examples of information sources of interest here are a design handbook or outside case studies; results of system operation over these database are provided in the next chapter. In such cases, it acts as a 'design context filter' by emphasizing design in databases with potentially far flung subject matter.

Chapter 8 discusses some empirical results of applying this technique within several document sets, here we discuss how the various methods described above can be integrated into an interface which supports context classification and navigation for design information retrieval.

7.4 Measuring the Effectiveness of the CDIS Global Index

Due to the informal design representation used by portions of the Conceptual Design Information Server (CDIS), the retrieval aspect of case-based reasoning has been emphasized. The previous section introduced methods for applying a bias to this representation for the purpose of information retrieval. By mapping design documents into a set of hierarchical conceptual abstractions pertaining to design issues, system function,

and elemental components the CDIS strives to ascertain conceptual design 'similarity' among design documents. In theory, once this mapping is done the context of a design document is captured in a standard language that can be used to enhance the retrieval of pertinent design examples.

Two alternatives for distilling the contextual bias of a design document have been discussed – natural language processing and information retrieval. While the former method promises to overtake the latter in effectiveness, this can only happen at the great cost of producing a generic knowledge base for conceptual design – a formidable task. Information retrieval was chosen as the framework for indexing CDIS documents because it promises the distinct benefits of automatic indexing of design information and automated determination of design context when applied to a predefined framework of design abstraction – the CDIS global index. In addition, the development of this abstraction framework is a necessary a building block toward applying more rigorous NLP methods because interpretation of a design document must be done with respect to an identified design context. The indexing methods suggested in the previous section applied will be evaluated in the context of information retrieval. To do this, standard metrics for measuring the performance of information retrieval are introduced, a testing methodology proposed, a number of indexing tests performed, and the results presented toward implementation of a user interface for interaction with the design case base.

7.4.1 Testing Index Effectiveness in Information Retrieval

Because computerized information retrieval dates to the late 60's, fairly rigorous evaluation metrics have been established. While many of these have focused (perhaps unfortunately) on the performance of the indexing system, the overall *retrieval* system encompasses a larger framework for evaluation. Boyce et al. [1994] recommend the following framework for measuring the overall effectiveness of an information retrieval system:

1. Develop a statement of the user's information needs

2. Apply this statement to the system by transforming it into a suitable query (i.e., a statement of information need that the system can 'understand')

3. Collect the response of the system

4. Assess the appropriateness of this response with reference to the entire collection of information available

Each of these steps must be interpreted into the context of a specific information retrieval system in order to prescribe a rigorous method for testing. Only the collection step is even remotely straightforward, the others must be addressed with respect to the goals of the CDIS.

The CDIS user is an engineer who is in the process of developing a conceptual design for a mechatronic device. Three document sets have been established over which the system operates: concurrent engineering case studies, conceptual design discussions, and templates for component selection. Additional resources like handbooks, texts, more case studies, or perhaps unknown WWW resources can also be brought into the information search process. The model of the design process applied within the CDIS highlights the need, within the conceptual design phase, for the manipulation of design abstraction to manage navigation of the design space. We propose that abstraction begins at a high level with the design specifications which are transformed into distinct requirements (functional, physical, economic, etc.) by integrating issues of importance to all design stakeholders. These issues are not necessarily readily identifiable a priori, they must be developed through discussion to arrive at a set of issue tradeoffs that express some form of optimality in the current design context. This process involves the purposeful manipulation of design abstraction in an iterative process that includes search and evaluation. Within the CDIS, we provide a framework for search through the design context abstraction hierarchies. Through representations of design discussions and component selection, evaluation metrics at varying level of abstractions are available as design information. Basically the information needs of a designer during conceptual design are determined by the current

design activity, a point reinforced by the literature [Baya and Leifer, 1994; Hirose, et al., Stauffer and Ullman, 1988]. The proposal for the CDIS is to determine this state by tracking the current design context within a defined abstraction framework.

How do we transform design context into a set of benchmark queries for evaluation? Typically, sets of user queries are collected from an on-line system and analyzed to determine an appropriate set; this has been done extensively in the library community [Drabenscott, 1995]. This is a laborious but fairly straightforward task in which user requests are interpreted to provide a framework of relevance which can then be used to evaluate the performance of the system in response to the users stated queries. This method has been used to create testing samples from a large stream of information requests. Within the CDIS it is not as straightforward to come up with a standard set of information requests. Perhaps the foremost reason for this is because the system is under development and not under heavy use like that of on-line library catalogs, any sample of user requests would be skewed by the small sample space. We therefore make the assumption that the documents within the CDIS represent typical user queries[11]. This is reasonable if the primary activity within the system is to be that of identifying possible navigational links from the current context – in line with case-based reasoning's reliance on design features for similarity matching. Thus, the main study will be that of identifying similar documents within the document set of the CDIS.

In another concession to pragmatism due to the difficulty of defining a user query set and quantifying its conceptual 'coverage' of the information space of a system, Salton maintains that system effectiveness can best be derived from the clustering capabilities it holds. This removes ambiguities involved both in stating an information demand and in translating that statement into a format that is the best possible query. If one knew exactly what one wanted to find, searching would become quite straightforward. The problem is that users typically do not know exactly for what they are searching and even if they do,

[11] This assumption places the system in a monitoring and critiquing mode [McCall et al. 1991] where stated design thoughts are submitted to the system which provides collateral information support by attempting to find similar (from a design context standpoint) documents.

they are not sure that it exists (or how close to it they can get). This is born out in the subject query studies of Drabenscott mentioned above. It is only through examination of search 'trees' that the context of an initial query can be ascertained. Users typically apply the experience from these initial queries to refine their information demand statement. Salton dismisses the importance of actual user queries, proposing that their primary purpose is to find a relevant document and to use that to identify a cluster of similar documents that, through vector space similarity measures, are highly likely to be relevant. Evaluation of system performance through document clustering is thus well supported. While one cannot dismiss the interface design issues related to Boyce's steps 1 and 2, evaluation of the indexing system can be separated from evaluation of the interface.

The model for user interaction for the CDIS global index set, the data collection and evaluation must be defined. Collecting the results is somewhat trivial, simply supply the query to the system and collect the returned document rankings. Interpreting these rankings is less so. Here a Boolean assumption is made: items identified by the information retrieval system are labeled as 'relevant' or 'irrelevant'. Within the CDIS, WAIS is the underlying information retrieval engine and provides a score for relevance, not a Boolean decision. Thresholds for this score can be used to take data points for evaluation by the defined set of metrics. By studying thresholds of WAIS scoring, it might be possible to define thresholds for document similarity below which few relevant documents are likely to be found. This is valuable information in the design of both the retrieval system and its user interface.

Given the above metrics and whether the result set is Boolean or ranked, determining relevance is still the main focus of assessing the performance of an information retrieval system. But relevance itself is a difficult concept, one which the information retrieval literature assumes can be measured but does not really address as an issue (or ever actually measure formally). The main problem is that relevance is highly subjective – one must have both a complete understanding of the user's need for information and complete knowledge of the content of the system's database in order to begin to make relevance judgments. But even then, relevance is subjective. The answer seems to be that it is impossible to derive

the set of relevant documents directly. The best compromise is to study relevance subjectively, applying a variety of search methods to accrue a superset of the relevant documents and then evaluating this set by hand to ultimately determine relevance. This is what is done for the TREC [Harman, 1994] efforts; for each topic, all systems are run and the resulting retrieval set accumulated, this set is then evaluated by an expert panel and Boolean relevance assigned to each topic/retrieved document pair. The assumption here is that the various systems all perform slightly differently; the union of all retrieved documents is likely to contain most of the relevant documents in the database by relaxing local performance biases inherent to each system by agglomeration.

Once this set of relevant documents has been identified, two primary metrics are used to quantify the performance of the system. They are based on components graphically depicted in Figure 7.6 and described conceptually and mathematically as:

Precision - A measure of the validity of the results set. It is defined as the number of relevant documents retrieved divided by the total number of documents retrieved.

$$P = \frac{\text{Retrieved} \wedge \text{Relevant}}{\text{Retrieved}} \qquad (7.4.1.1)$$

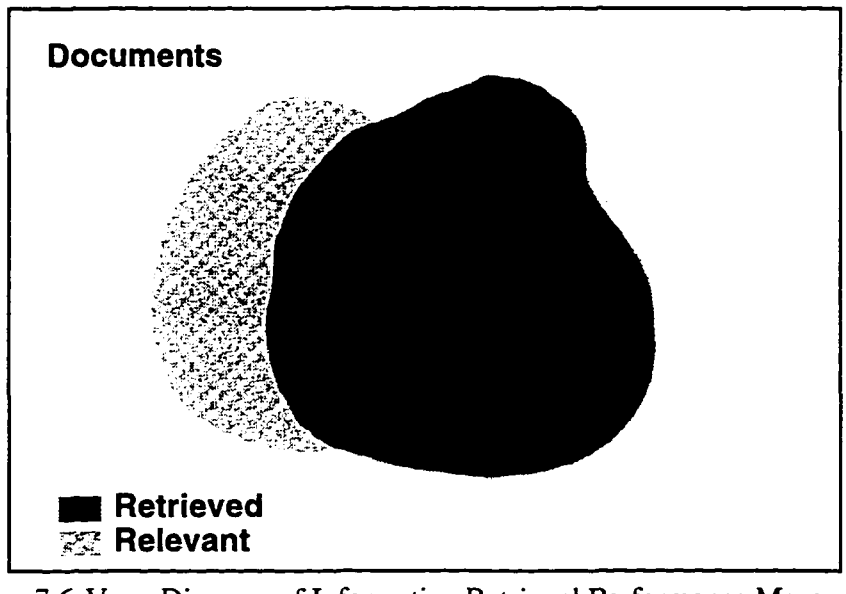


Figure 7.6 Venn Diagram of Information Retrieval Performance Measures.

Recall - A measure of the effectiveness of the retrieval with respect to the overall database. It is defined by the number of relevant documents retrieved divided by the number of relevant documents in the database.

$$R = \frac{\text{Retrieved} \wedge \text{Relevant}}{\text{Relevant}} \qquad (7.4.1.2)$$

These metrics measure the two main concerns of information retrieval: how relevant is the retrieved set and did it retrieve everything that is relevant. The two conceptually related to each other – too broad a search (i.e., adding in search terms unrelated to the context) and the recall might be good but the precision poor, too narrow a search (i.e., not adding a search term closely related to the context) and the precision might be good but the recall poor. They have been shown to be empirically related by [Lancaster et al., 1994]:

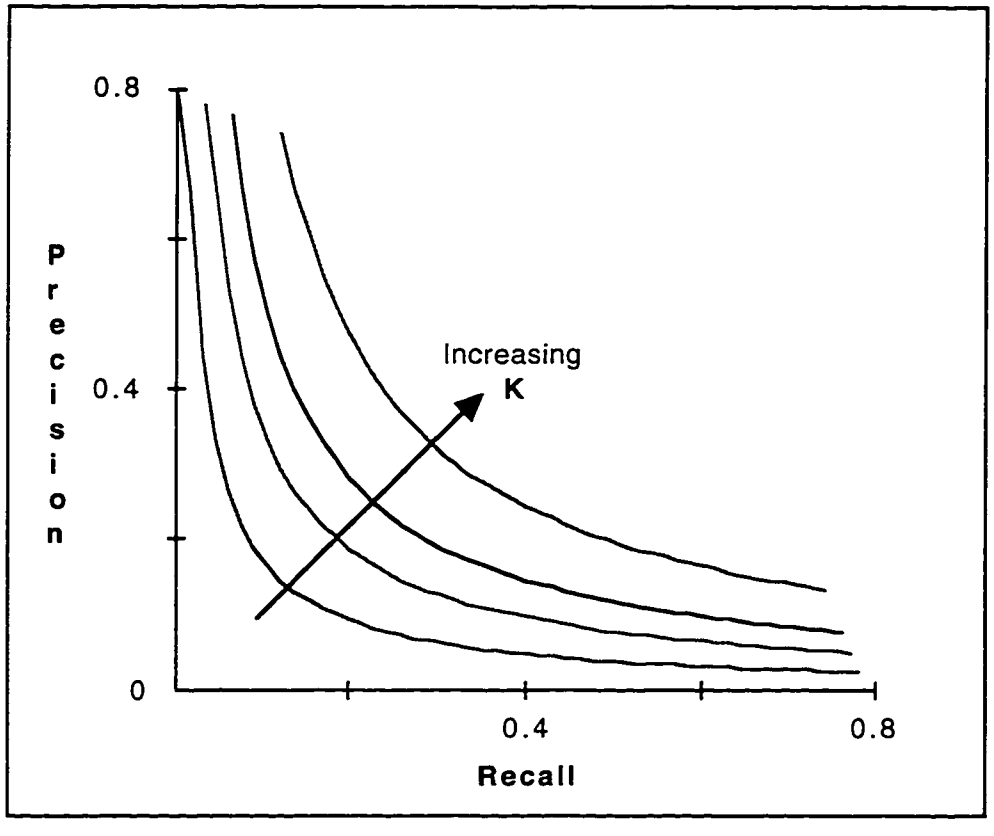


Figure 7.7 Empirical Relation Between Precision and Recall. An increase in the combined performance measure, **K**, provides better system performance without penalty to either Recall or Precision.

$$\mathbf{K} = P * R \qquad (7.4.1.3)$$

and presented graphically in Figure 7.7. Thus, a Pareto-optimal[12] condition exists in ideal information retrieval systems: assuming a fixed **K**, improving recall must come at a cost in precision and vice-versa. Extracting the maximum general performance from an information retrieval system is thus analogous to increasing **K**; clearly one can do things to the information search process, like adding search terms that are unrelated to the search context, which degrade both precision and recall (reducing **K**). The main focus in designing an information retrieval system is to create a situation where the user can actively manipulate precision and recall along a Pareto line of maximum **K**, expanding the search for greater recall when too few documents are found and restricting it for greater precision when too many are found – following exactly the human searchers' strategy anecdotally derived from the library search trees by Drabenscott. The approach is twofold: provide an interface where the user can actively manipulate the search process but does not have to work too hard to do so, and implement the system in such a way that representation and thus retrieval is biased toward relevant information.

The TREC efforts provide a slightly different view of precision with regard to retrieval system which rank their retrieval sets instead of making Boolean decisions as to relevance [Harman, 1994]. Here, documents are scored with regard to relevance and their position within the ranked retrieval set:

$$P(i) = R(i)/i \qquad (7.4.1.4)$$

This metric does tend to reward precision over recall for systems with large sets of relevant documents, so it will be used in combination with the traditional precision and recall to measure system performance.

The performance measures described above will be used to evaluate several methods

[12] A Pareto-optimal surface is defined for multiple objectives (like precision and recall) as that surface where no improvement in one objective can be made without a degradation in the other(s).

introduced into the WAIS query cycle for increasing performance of the CDIS global index. The testing suite consists of first evaluating the performance of the automatic indexing component (i.e., does the system correctly identify the appropriate CDIS global index design context of a document). Then query elaboration is performed, based on the contextual mapping schemes identified as effective. These queries are compared to the standard WAIS queries over databases within and outside of the CDIS.

7.4.2 Automatic Indexing Performance

Automatic indexing in the CDIS is based on WAIS queries over document sets derived from the three design context abstraction hierarchies. The following database were created[13] shown schematically in Table 7.1:

1. Terms and Synonyms: Each document is comprised of the abstraction term and any synonyms. Unlike standard WAIS server support for synonyms, here they can be composed of acronyms or phrases.

2. Terms, Synonyms, and Descriptive Phrases: The above documents are extended to include the phrase used to describe the context.

3. Terms, Synonyms, Descriptions, Parent, and Children: Again each document is extended to include not only the design context description but also the contingent terms. The aim here is to recognize the hierarchical nature of the design context, still centering the context on the term but adding in some additional information on both

Table 7.1 Context Database Constitution

	Local		Parent		Child	
DB	Terms & Synonyms	Phrase	Term & Synonym	Phrase	Terms & Synonyms	Phrase
1	X					
2	X		X			
3	X	X				
4	X	X	X		X	
5		X	X	X	X	X

13 In addition to the various context document options, databases were tested stemmed and unstemmed (stemming refers to the reduction of terms to common roots)

higher and lower abstraction levels.

4. Terms, Synonyms, Descriptions, Parent, Children, Descriptions: The descriptive phrases from the parent and children are added toward contextualizing documents that might be vague or perhaps clash in expression with that encoded in the CDIS.

5. Terms, Synonyms, Parent, Children: Somewhere between databases #1 and #3, the descriptive phrase is omitted in an effort to expand the keyword-only contextual assignment within the context hierarchy.

Tests of context assignment were performed using a set of sample documents drawn from the CDIS database. Forty documents were selected (subjectively) for 'design content' from the concurrent engineering case studies, the design discussion server, and the concept database. Each of these documents was submitted to the automatic indexing portion of the CDIS global index and the resulting design context assignments recorded for each method. A panel of three experts (one case study developer, a design expert, and the author) then evaluated the accuracy of the contextual mapping by assigning a Boolean relevance measure to each member of the (randomized) union of all retrieved design contexts. The context assignments of WAIS searches over each context database were evaluated for precision, recall, and **K** (calculated for Boolean document sets derived from setting various WAIS score thresholds). TREC precision vs. recall as well as precision vs. recall generated from the WAIS threshold studies were also tabulated. Because each single database method is theoretically valid but might yield different results, a combination of the results of searching all databases was evaluated as an alternative to choosing just one of them. The results are as follows:

Context Precision (Figure 7.8[14]): Precision for retrieval sets defined at various WAIS score thresholds (WAIS scores are in [0, 1000] with higher scores indicating stronger similarity) shows that the most precise assignment of context comes from

[14] Please note that the graphs legends are consistent throughout: T - term, S - synonym, Ph - descriptive phrase, P - parent term & synonyms, C - children terms & synonyms.

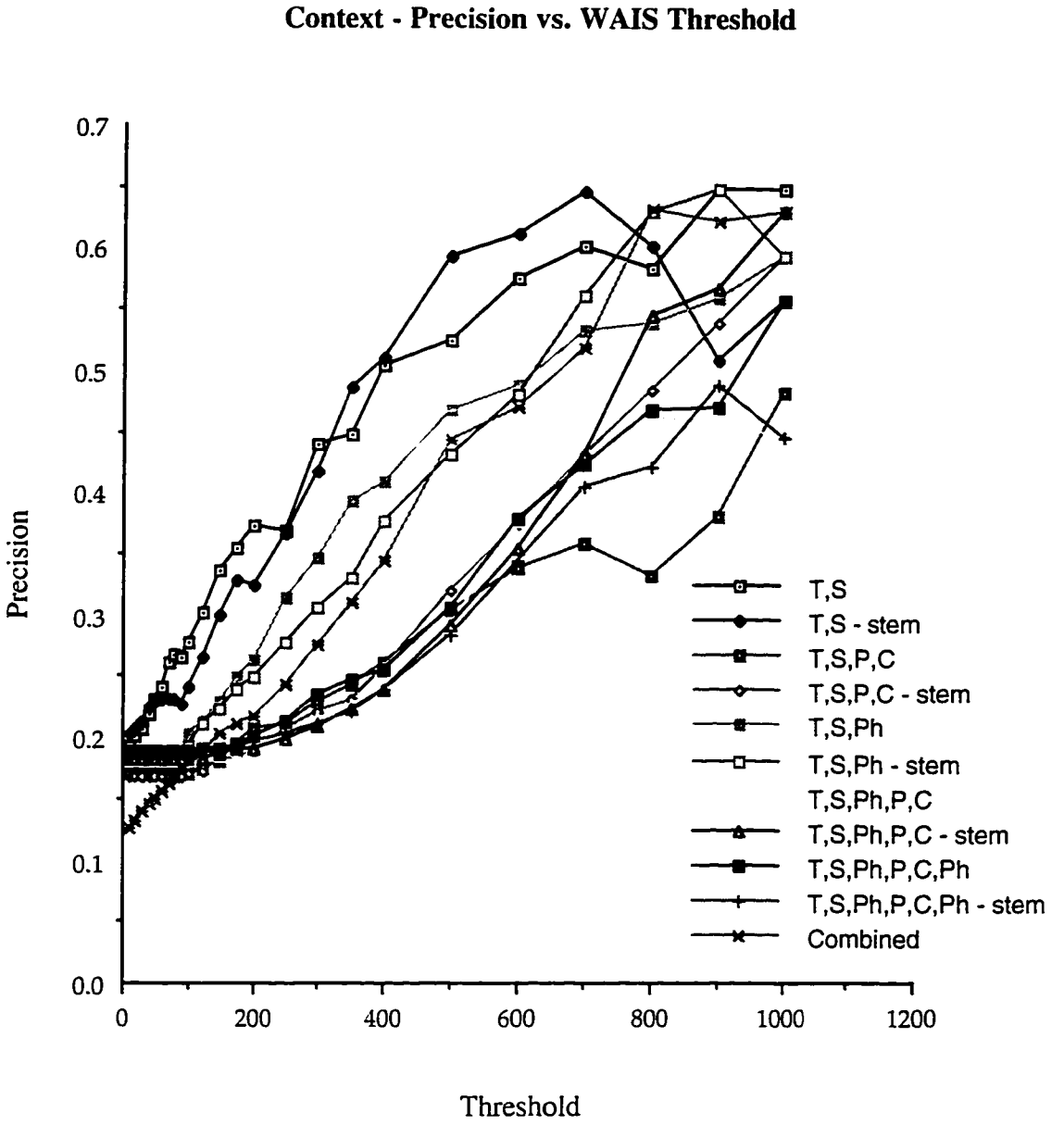


Figure 7.8 Precision vs. WAIS Threshold for Context Assignment.

the unstemmed database built from the context term and its synonyms. Adding the descriptive phrase (stemmed) also helps at high levels of desired precision. As more of the abstraction hierarchy is added into the context, precision is degraded. However, the combination of all databases performs fairly well, second most effective at high precision and generally third or fourth best from mid to low

precision. A WAIS threshold of between 200 and 400 might be appropriate for the more precise context assignments.

Context Recall (Figure 7.9): As one might expect, we get the opposite results here. For better recall one should add more of the abstraction hierarchy into the design context database. It is interesting to note that once again the combined index performs rather

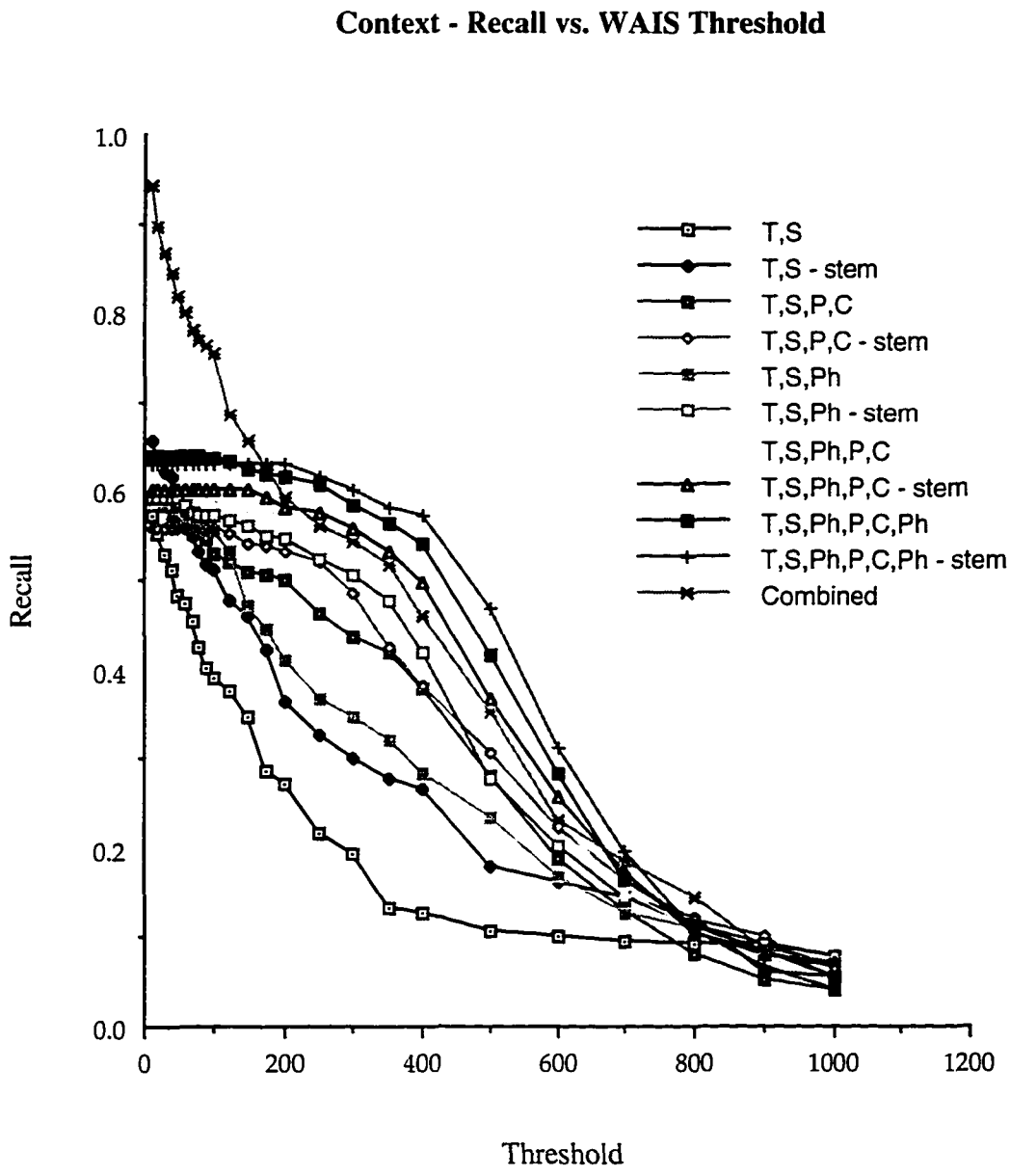


Figure 7.9 Recall vs. WAIS Threshold for Context Assignment.

well, able to achieve close to 100% recall at low WAIS threshold while dropping to the third best option at higher threshold points.

Context **K** (Figure 7.10): Calculated from the above sets of data, **K**, strives to measure the compromise between precision and recall. Note that the combined

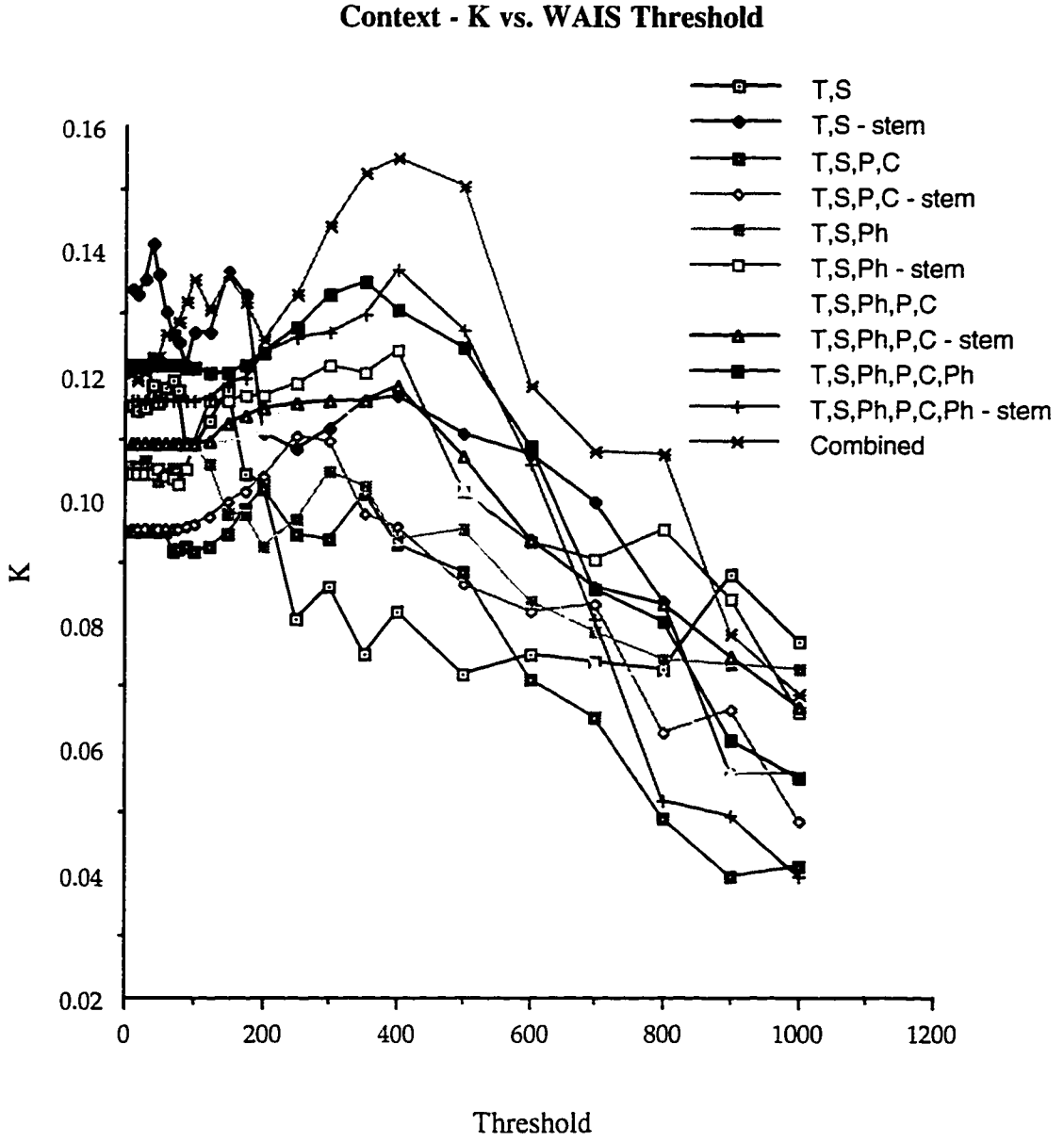


Figure 7.10 **K** vs. WAIS Threshold for Context Assignment.

index seems to be dominant except at extreme values of the WAIS threshold. Also of interest is that the 'extreme' contextual representations seem to be favored over the more moderate ones, with databases #1 and #5 providing the best alternatives to the combined index.

Context TREC Precision vs.Recall (Figure 7.11): While the TREC metric is known to

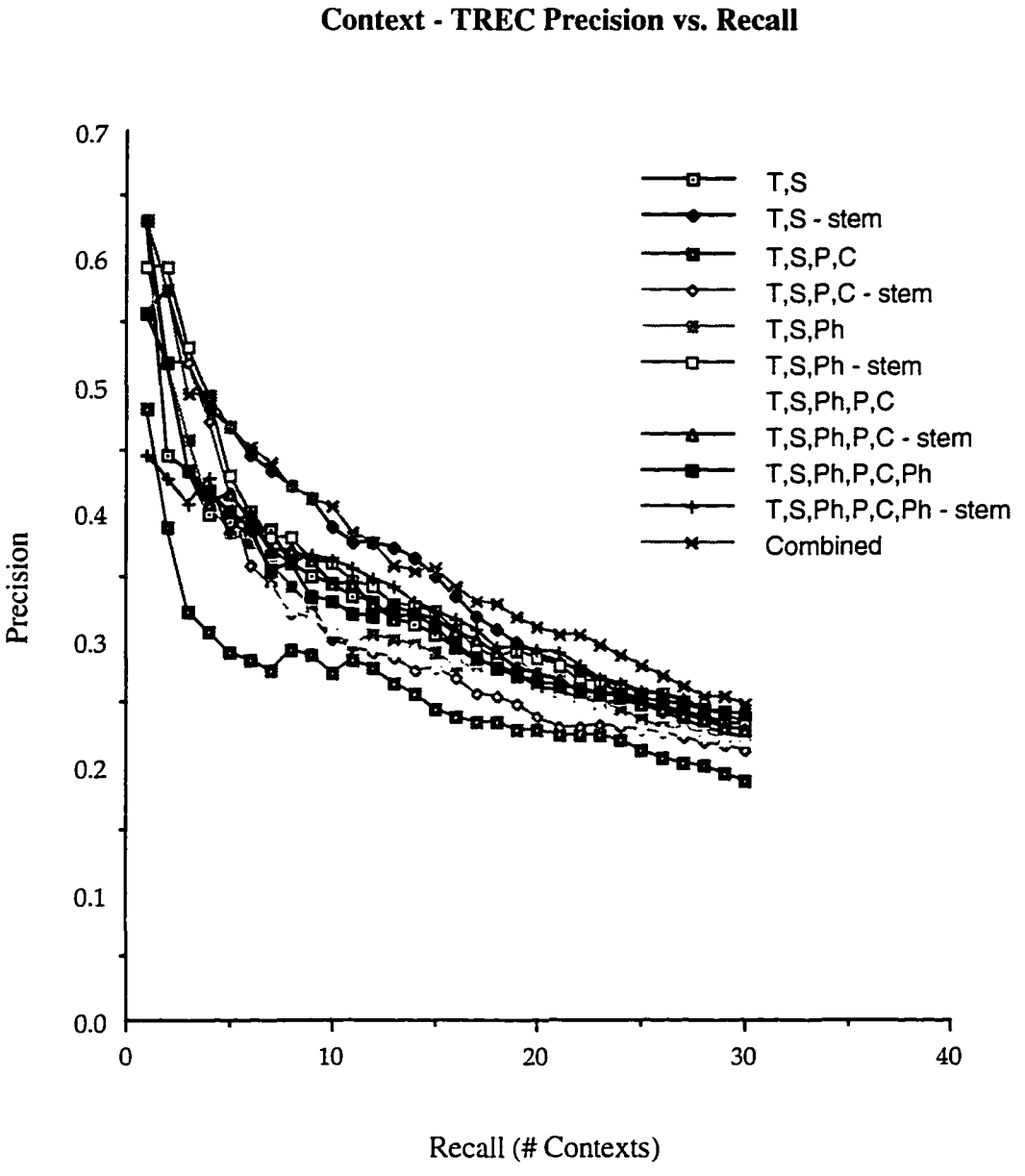


Figure 7.11 TREC Precision vs. Recall for Context Assignment.

favor precision over recall, the results show that the strong **K** performance of the combined scoring method largely overcomes the expected precision advantage of database #1 (stemmed). Also of interest is how the poor recall performance of database #1 (unstemmed) degrades its precision performance when calculated using the TREC metric.

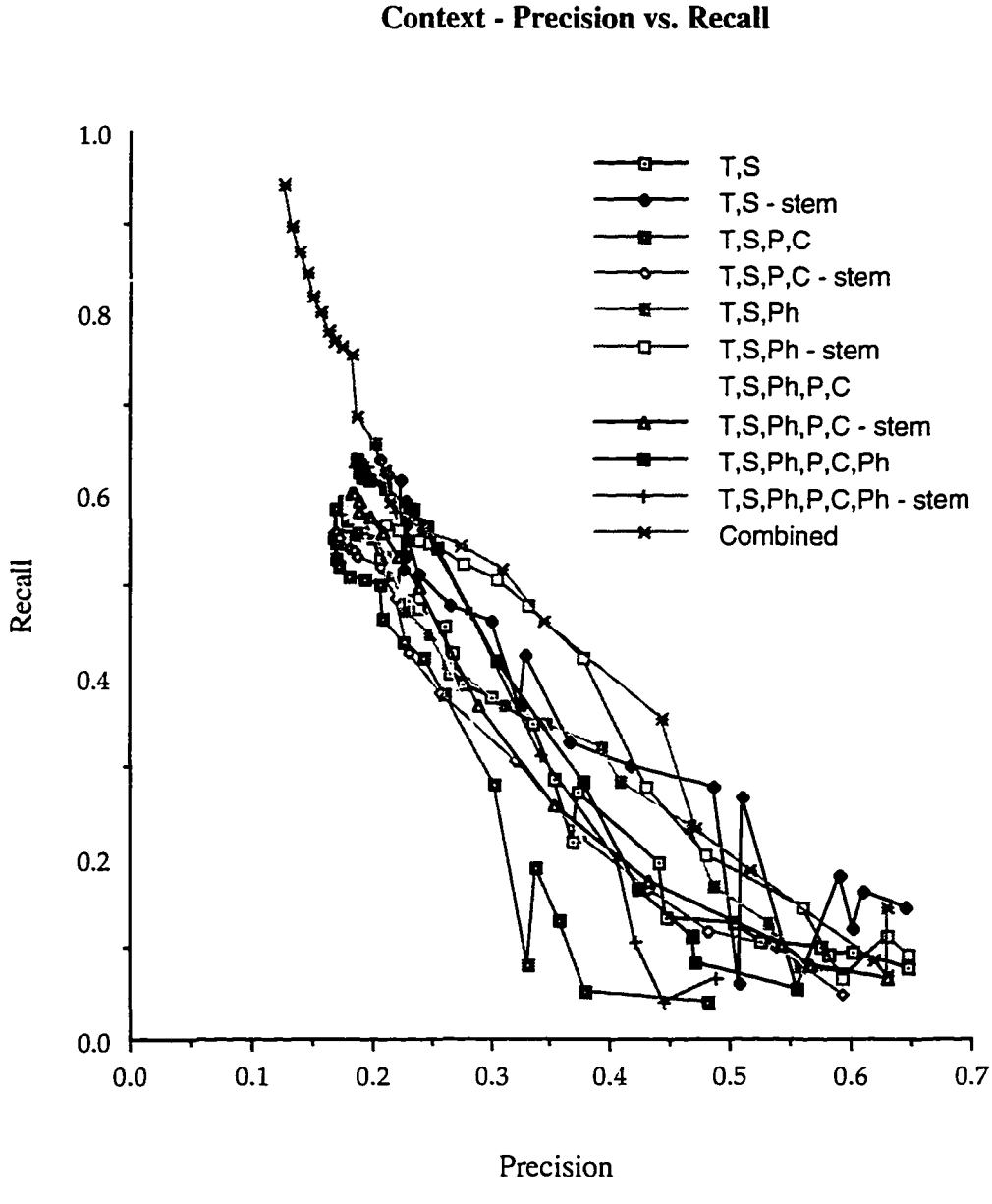


Figure 7.12 Precision vs. Recall for Context Assignment.

Context Precision vs. Recall (Figure 7.12): Plotting the precision points of Figure 7.8 vs. the recall points of Figure 7.9 attempts to recreate Figure 7.7, where variations in **K** demonstrate the overall performance of a retrieval system. Here, the results of the **K** and TREC plots above are echoed. The combined scoring method offers the best compromise of precision and recall of all of those considered. At nearly every recall level, it provides superior contextual assignment precision. Thus, context assignment in the CDIS global index is best done using the combined scoring method.

7.4.3 Automatic Query Elaboration

Having established the best means of assigning context to CDIS queries, we now turn to methods for elaborating these queries in an effort to increase information retrieval performance over that provided by standard WAIS. Because the CDIS global index is intended for generic application to design information, this performance must be considered both within the CDIS information bases and outside of them for more generic engineering resources. Two sets of tests were conducted over internal and external information bases (again stemmed and unstemmed), each of which applied varying levels of query elaboration or replacement[15] at various threshold for context assignment score, shown in Table 7.2 below.

The results for these runs are based on the same set of metrics used in the context assignment experiment discussed above. For clarity, they are separated by information source. In addition, only the top handful of performers out of the 142 test runs for each information source are plotted:

[15] Recall that query here is used loosely. These tests use the same set of documents used for context assignment.

Table 7.2 Query Elaboration/Replacement Sources

Local	Parent		Child	
Terms & Synonyms	Phrase	Term & Synonym	Phrase Terms & Synonyms	Phrase
X				
*				
X		X		
*		*		
X	X			
*	X			
X	X	X	X	
*	X	*	*	
	X	X	X X	X
	X	*	X *	X

X - normal WAIS query
* - WAIS exact phrase query

CDIS Precision (Figure 7.13[16] below): For high precision of information recall the original query applied to an unstemmed database performs the best. There is, in fact, a slight degradation of performance when adding in even the most selective set of extra terms. This seems to indicate that the ~70% precision of context assignment does not help in discriminating among documents in the CDIS internal database. The probable cause of this is that the CDIS database contains documents which, for the most part, are about similar topics and share much of the same language. Because of this, errors in the translation of the query into a more general representation may be exacerbated. Another significant result here is that setting the WAIS threshold for retrieval above 400 ensures relatively high precision in the retrieval set.

CDIS Recall (Figure 7.14 below): The results for recall help remediate the poor precision performance by demonstrating that recall is greatly enhanced (at WAIS thresholds above 200) by substituting the automatically generated design context for the

[16] Here we add to the coding scheme for query elaboration: The leading number indicates the WAIS context assignment threshold; the presence of a 'Q' in the legend indicates that the query was included in the search, its absence indicates query replacement; XT represents a literal phrase search for the term and synonyms.

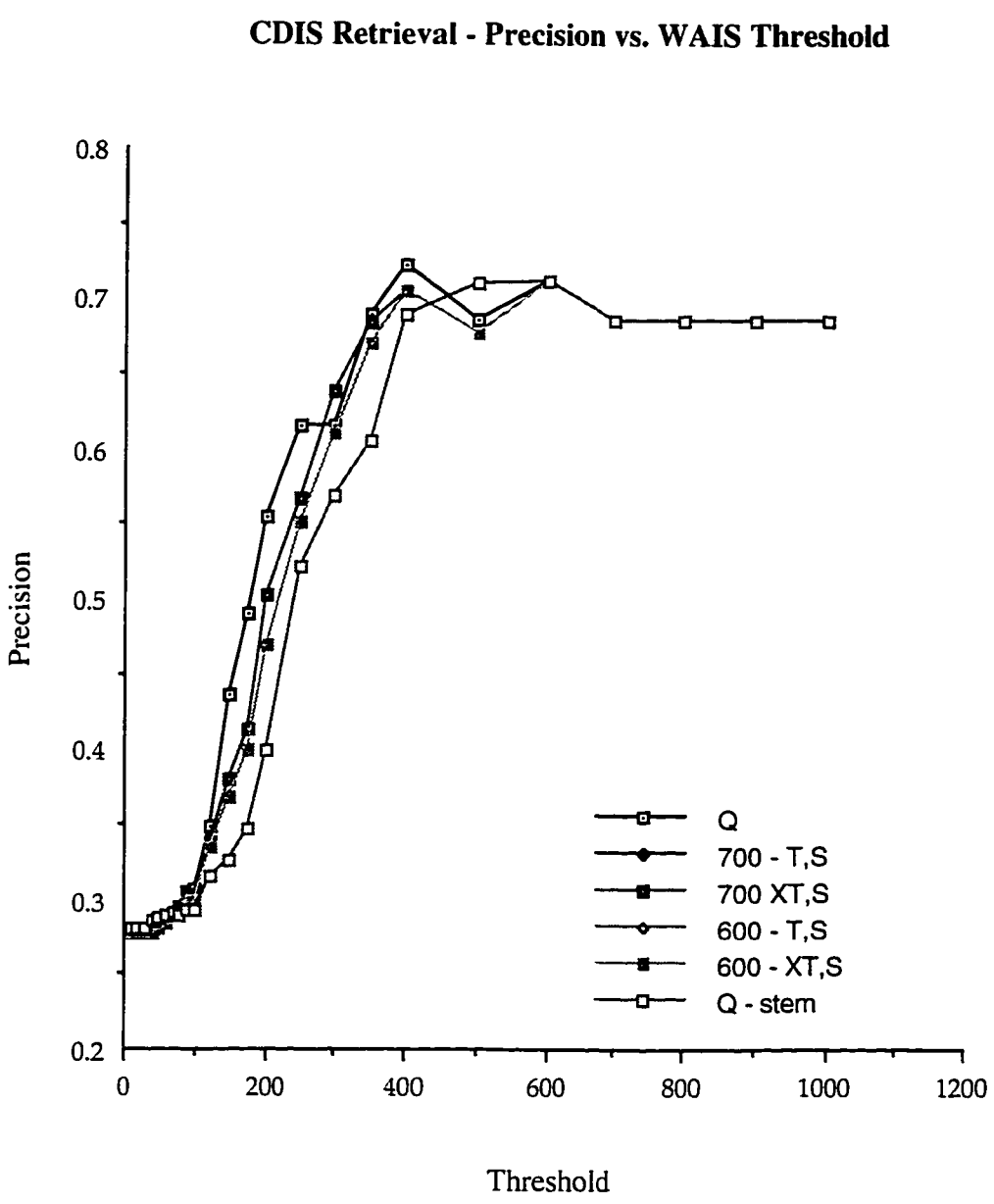


Figure 7.13 Precision vs. WAIS Threshold for Automatic Retrieval over the CDIS Databases.

original query. It is interesting to note the general equivalence among stemmed and unstemmed databases for this query replacement. Also, including the parent and child terms for a context is roughly equivalent to including its descriptive phrase. Applying the exact term phrase also seems to make little difference (perhaps due to single word terms or to some unseen query processing done within WAIS).

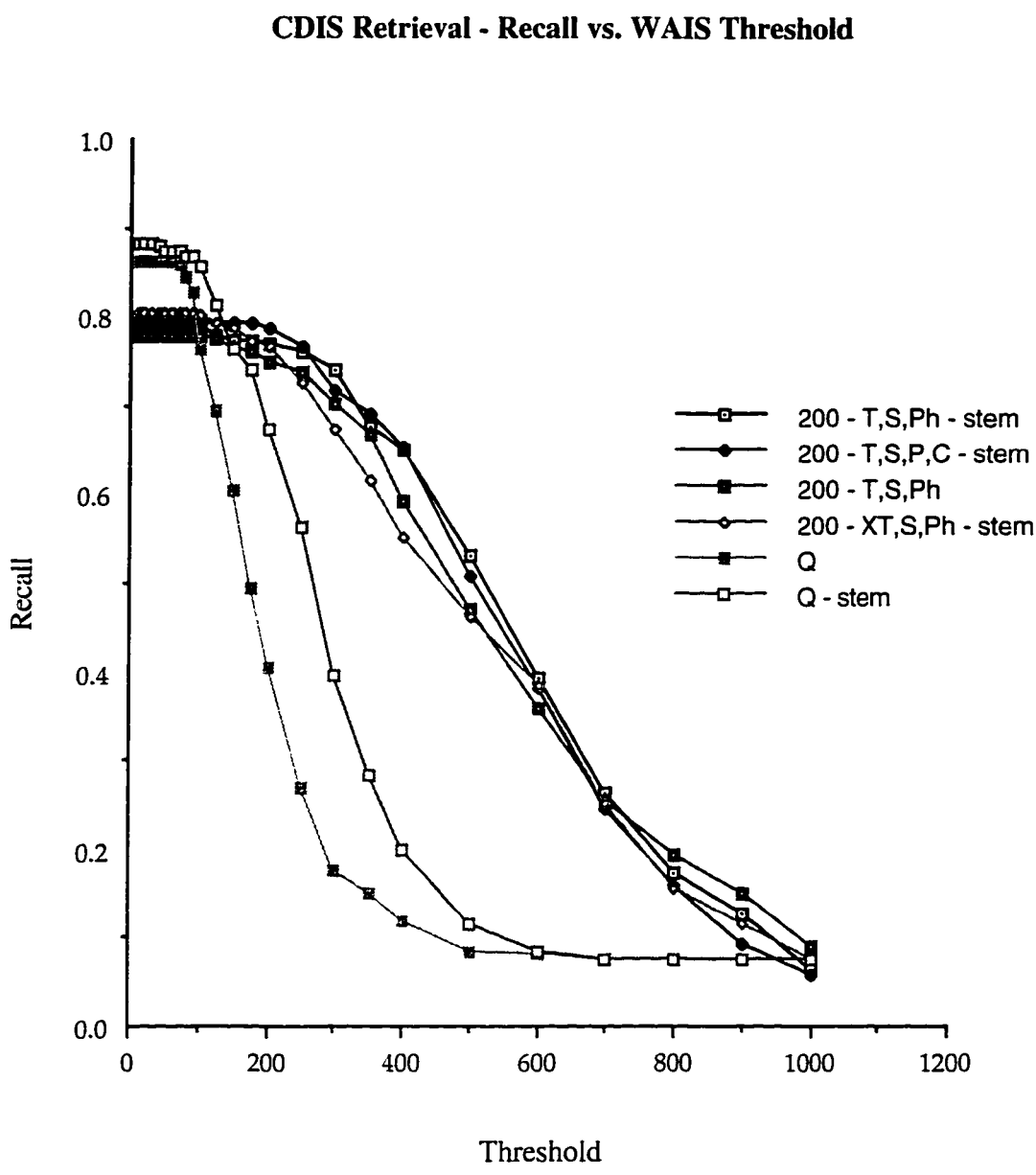


Figure 7.14 Recall vs. WAIS Threshold for Automatic Retrieval over the CDIS Databases.

CDIS **K** (Figure 7.15): A more general measure of retrieval performance reveals a compromise between query replacement and query augmentation. This result seems to recommend a contextualization threshold WAIS score of 300, establishing that retrieval thresholds above 500 are probably sub-optimal for the general case.

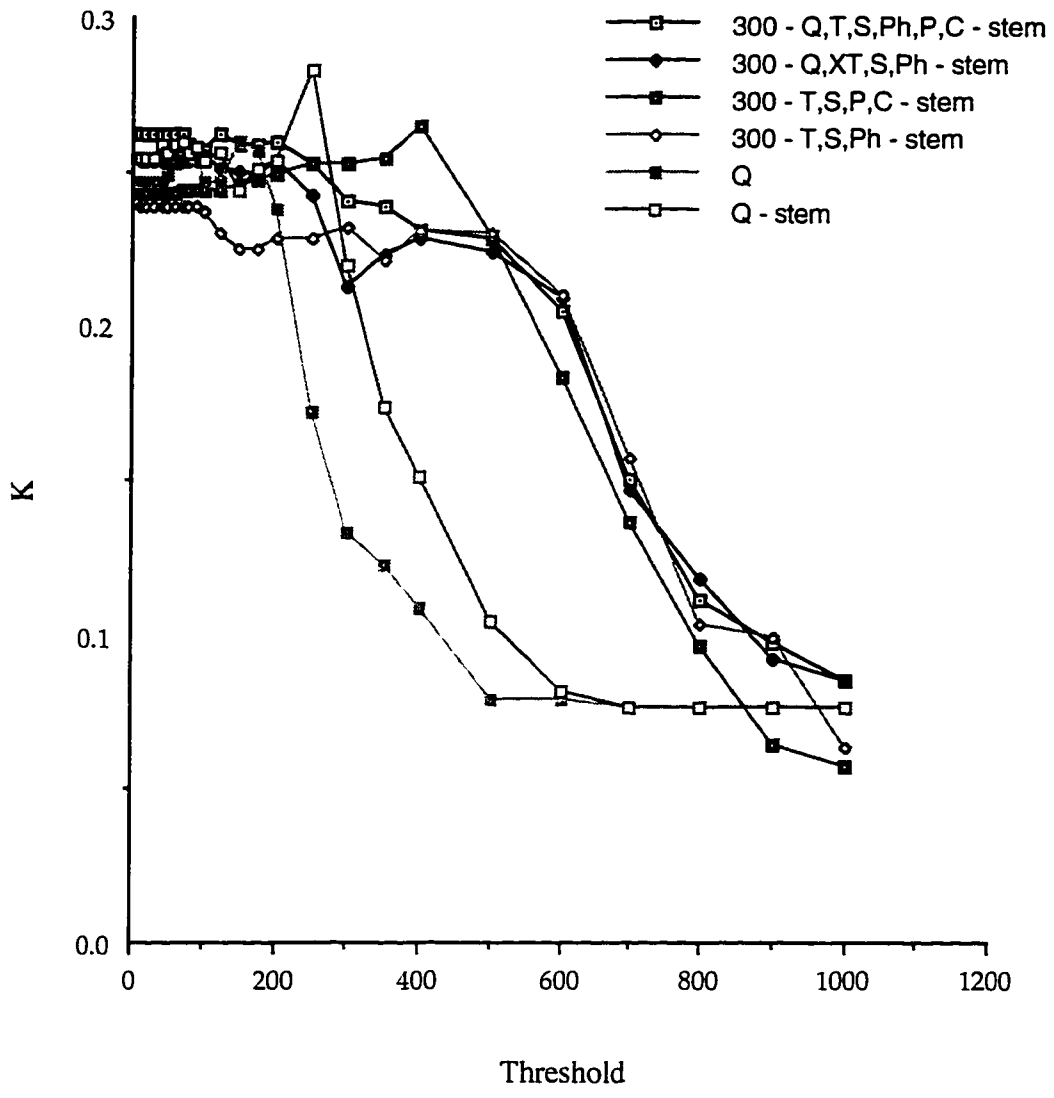


Figure 7.15 **K** vs. WAIS Threshold for Automatic Retrieval over the CDIS Databases.

CDIS TREC Precision vs. Recall (Figure 7.16): For the TREC precision measure, like the generic precision measure, query augmentation instead of replacement is the way to go. Instead of adding in only a few terms, here the best results are gained by including most of the design contexts automatically assigned to the query. It is also clear that stemming the CDIS database is probably best for performance.

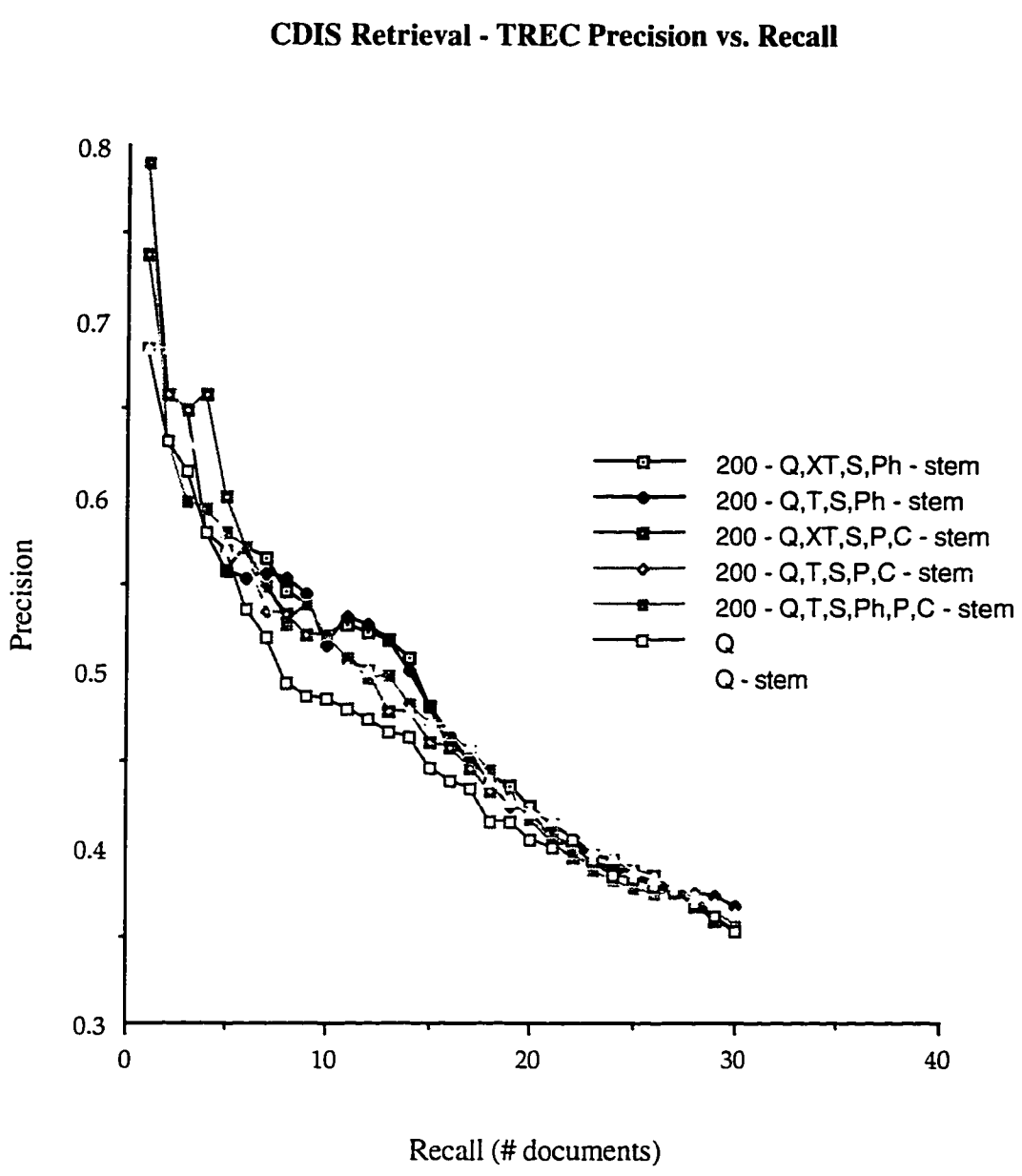


Figure 7.16 TREC Precision vs. Recall for Automatic Retrieval over the CDIS Databases.

CDIS Precision vs. Recall (Figure 7.17): To this point, the CDIS retrieval performance has been interesting in its general contradiction of the first principles of information retrieval. It appears, however, that the generalities from above hold for the overall performance: For better recall, query replacement is the best option. For higher precision, query augmentation improves recall without penalty. For a good

compromise, the standard WAIS methods seem to be best. This interesting result indicates that the information retrieval canon of vector space modeling, stemming, inverse document frequency measures, etc. is well founded when dealing with document-document similarity within a single database. Methods for automatically replacing or elaborating the document certainly do no harm to performance but do not enhance it much, if at all for this case.

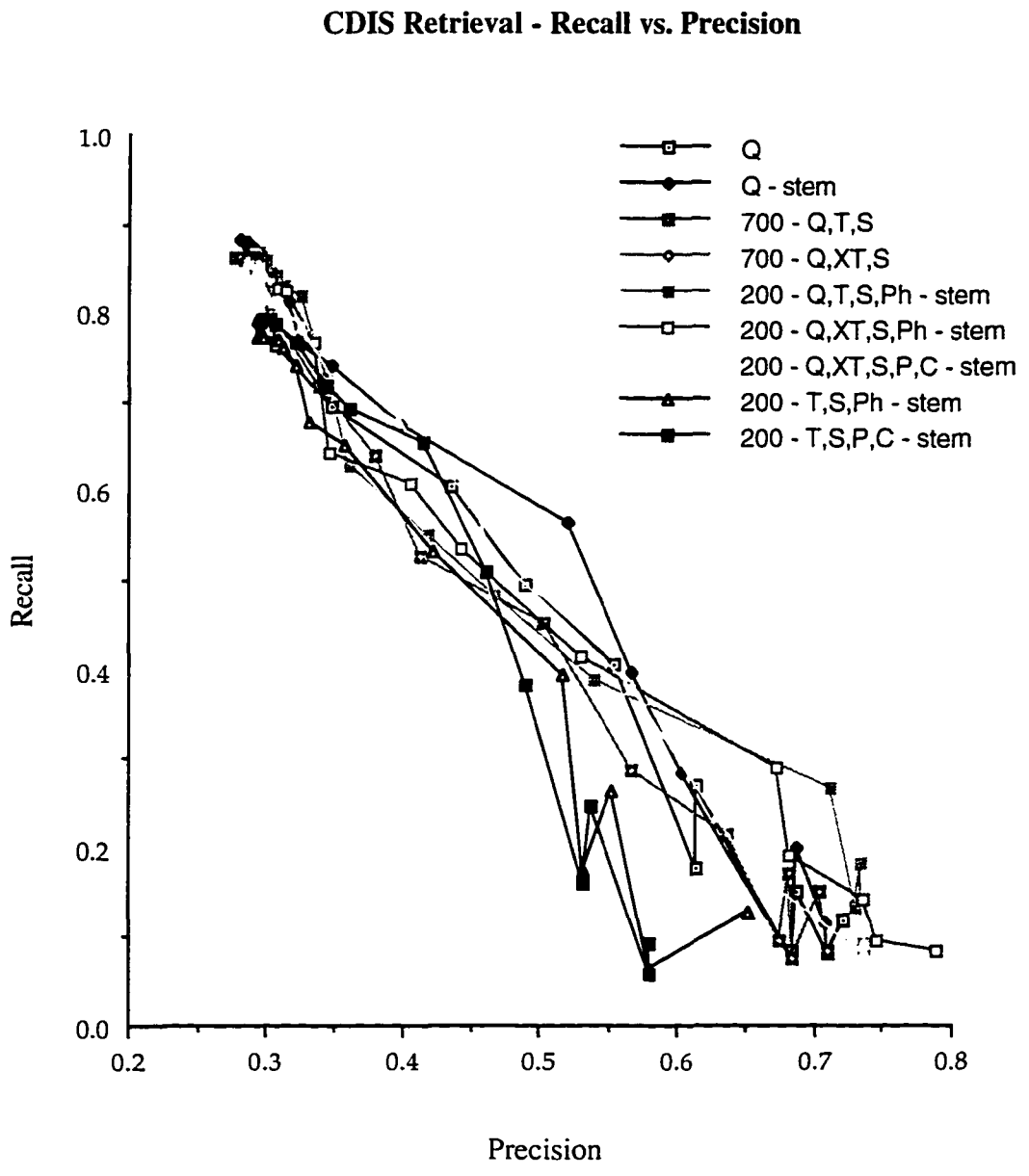


Figure 7.17 Precision vs. Recall for Automatic Retrieval over the CDIS Databases.

The next set of data results from running the same query set over a database consisting of Kutz' *Handbook of Mechanical Engineering*, Juvinall & Marshek's *Fundamentals of Machine Component Design*, and Eichenaur's *Transient Systems Analysis*. The database consists of approximately 3000 scanned pages of text and graphics, processed with optical character recognition (OCR) system to extract the text which was then indexed and served using a WAIS database. These information sources represent a generic mechanical /

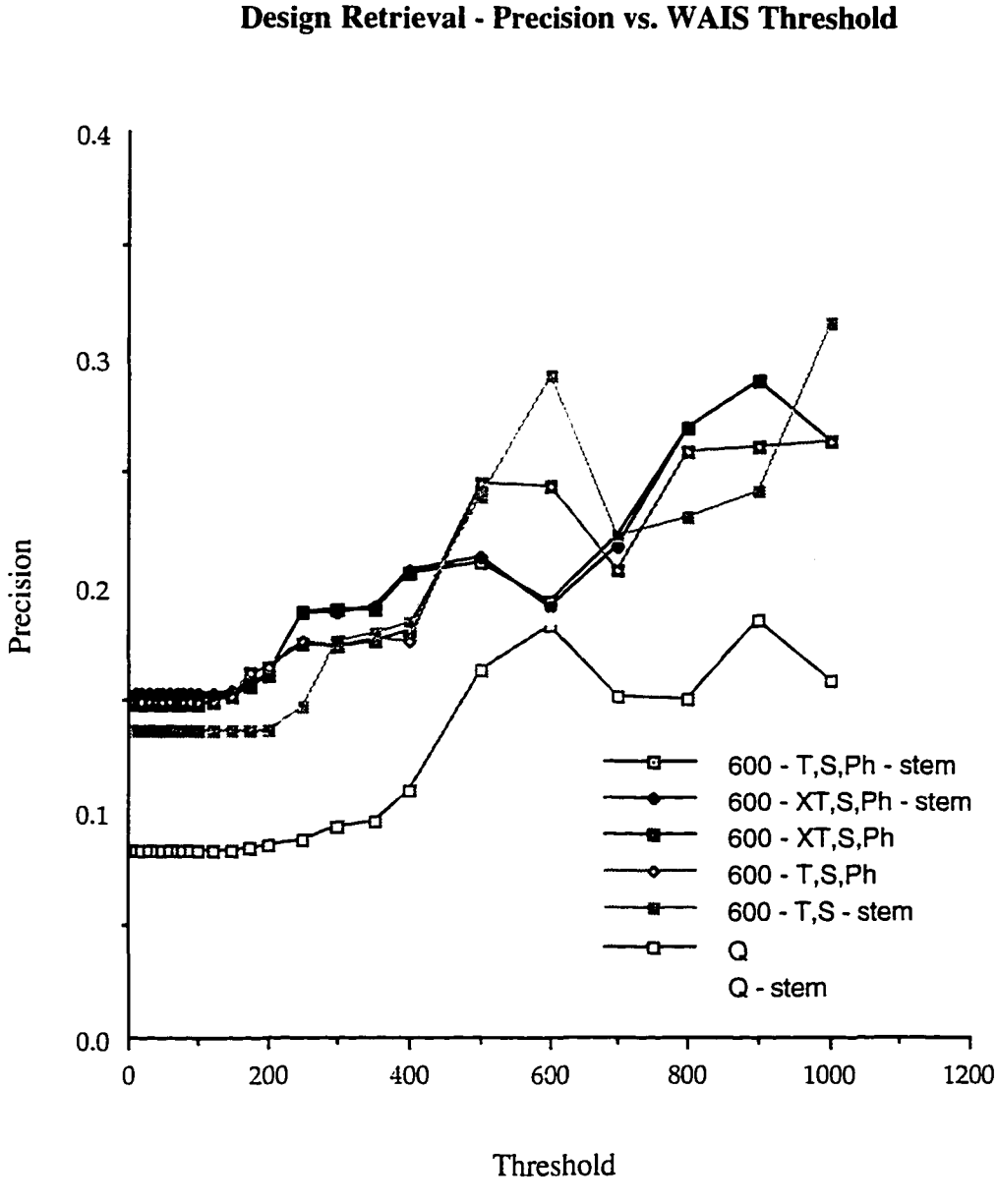


Figure 7.18 Precision vs. WAIS Threshold for Automatic Retrieval over the Design Databases.

mechatronics design resource; results are indicated by the keyword 'Design' in the title of each plot:

Design Precision (Figure 7.18): The increase in retrieval precision expected from the query substitution technique is revealed in this plot. Precision is greater than that produced by the standard WAIS methods, the threshold of 600 for contextualization

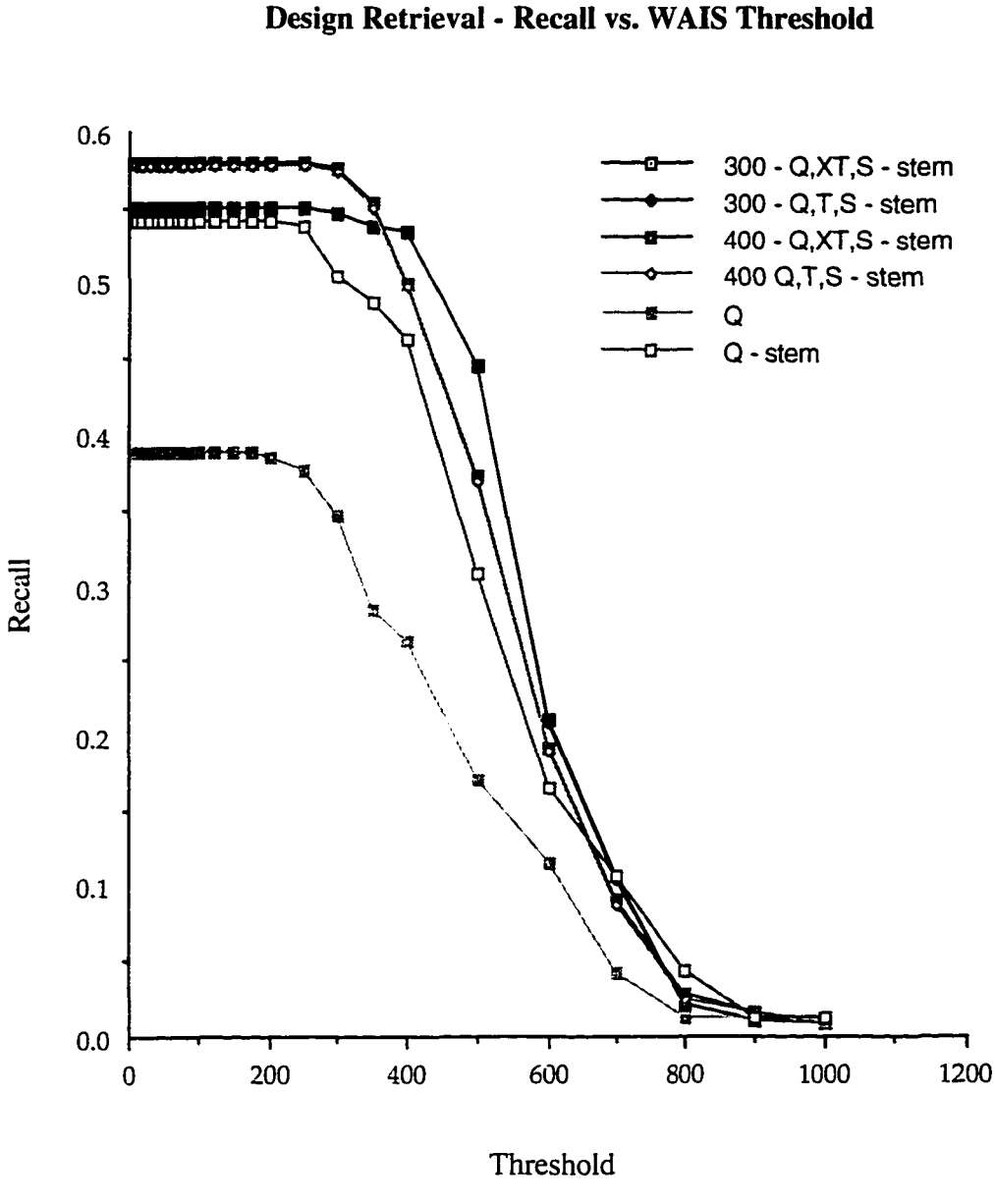


Figure 7.19 Recall vs. WAIS Threshold for Automatic Retrieval over the Design Databases.

represents a compromise between context assignment precision and recall – Figure 7.10 shows 600 be the highest threshold in the high **K** region of contextualization **K** vs. WAIS threshold curve. Conclusions to be drawn here are: 1) query replacement yields better precision; 2) query replacement should probably be made using only information from the local context; and 3) exact term searching and stemming have little impact on precision.

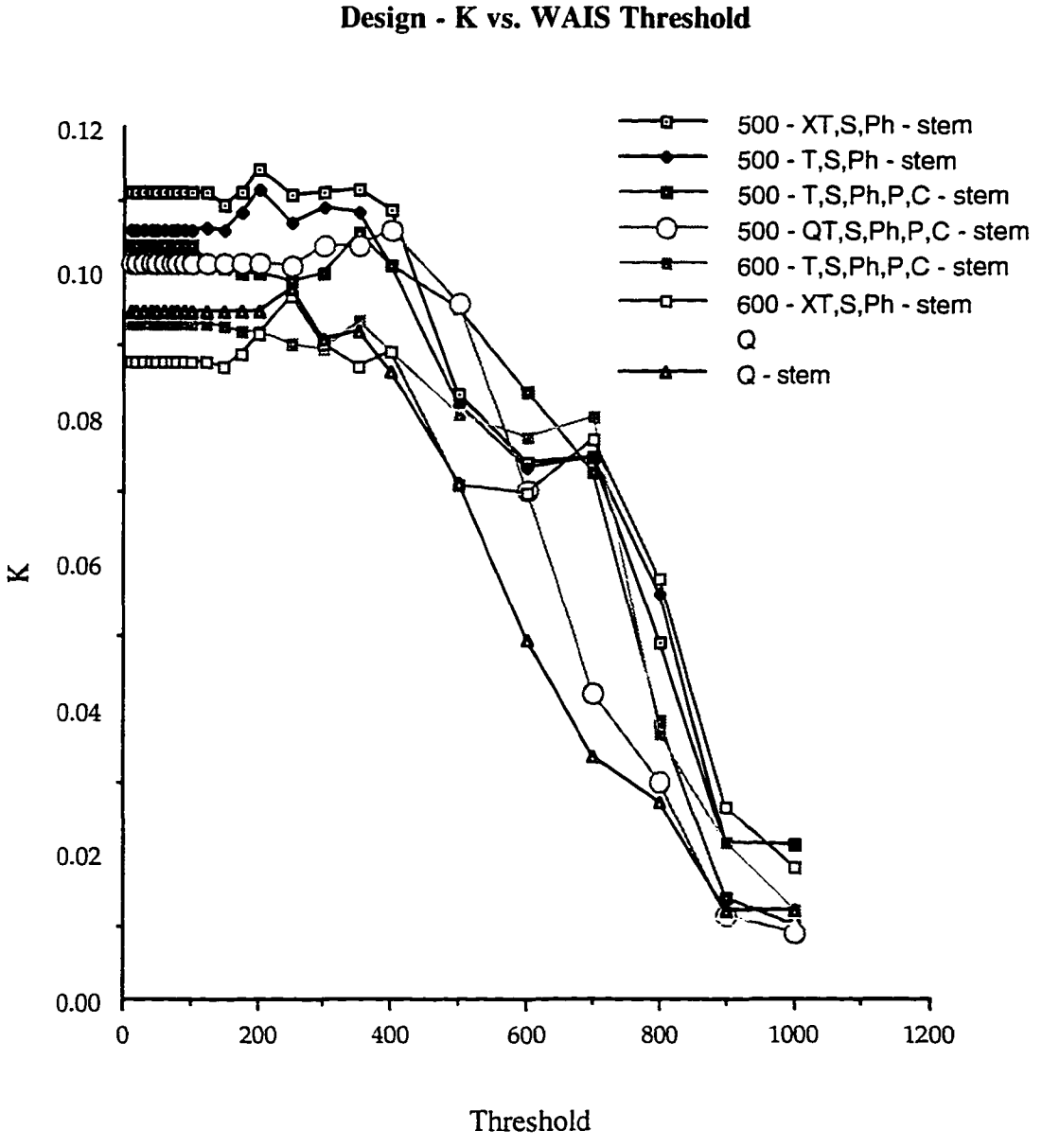


Figure 7.20 **K** vs. WAIS Threshold for Automatic Retrieval over the Design Databases.

Design Recall (Figure 7.19): In results more intuitive than those found for internal document-document searching, recall in searches over the design database is improved by augmenting rather than replacing the original query. In addition, the WAIS context threshold that seems to provide the optimal **K** – between 300 and 400 – here provides the best recall. Also, augmenting the query seems to be best done with only terms and synonyms.

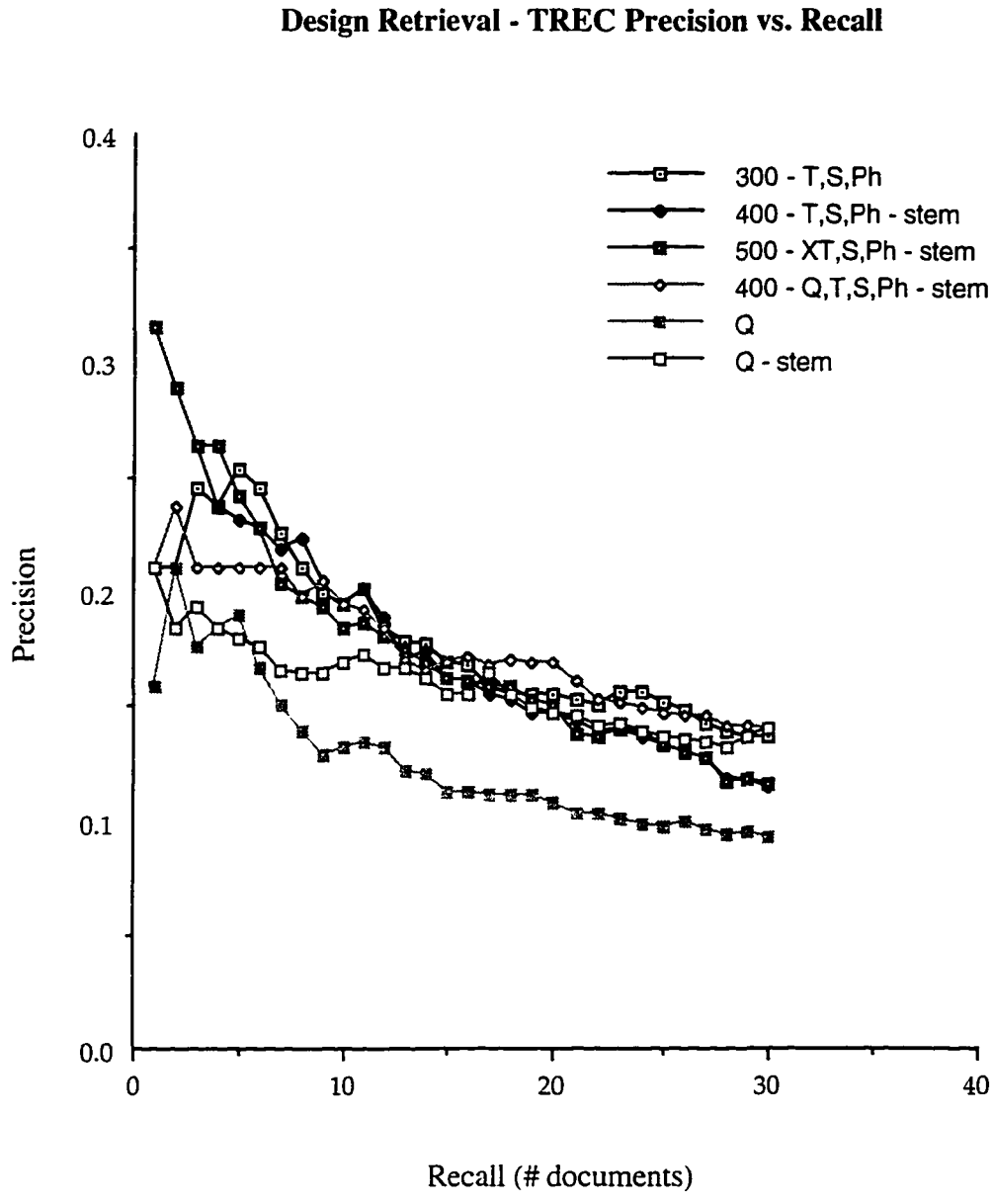


Figure 7.21 TREC Precision vs. Recall for Automatic Retrieval over the Design Databases.

Design **K** (Figure 7.20): Query replacement using a compromise contextualization threshold provides good all around performance. In addition this is the first point at which exact term matching shows performance different from that of the standard term representation, exact matching showing an advantage over most of the range for the consensus optimum contextualization threshold of 500. The utility of

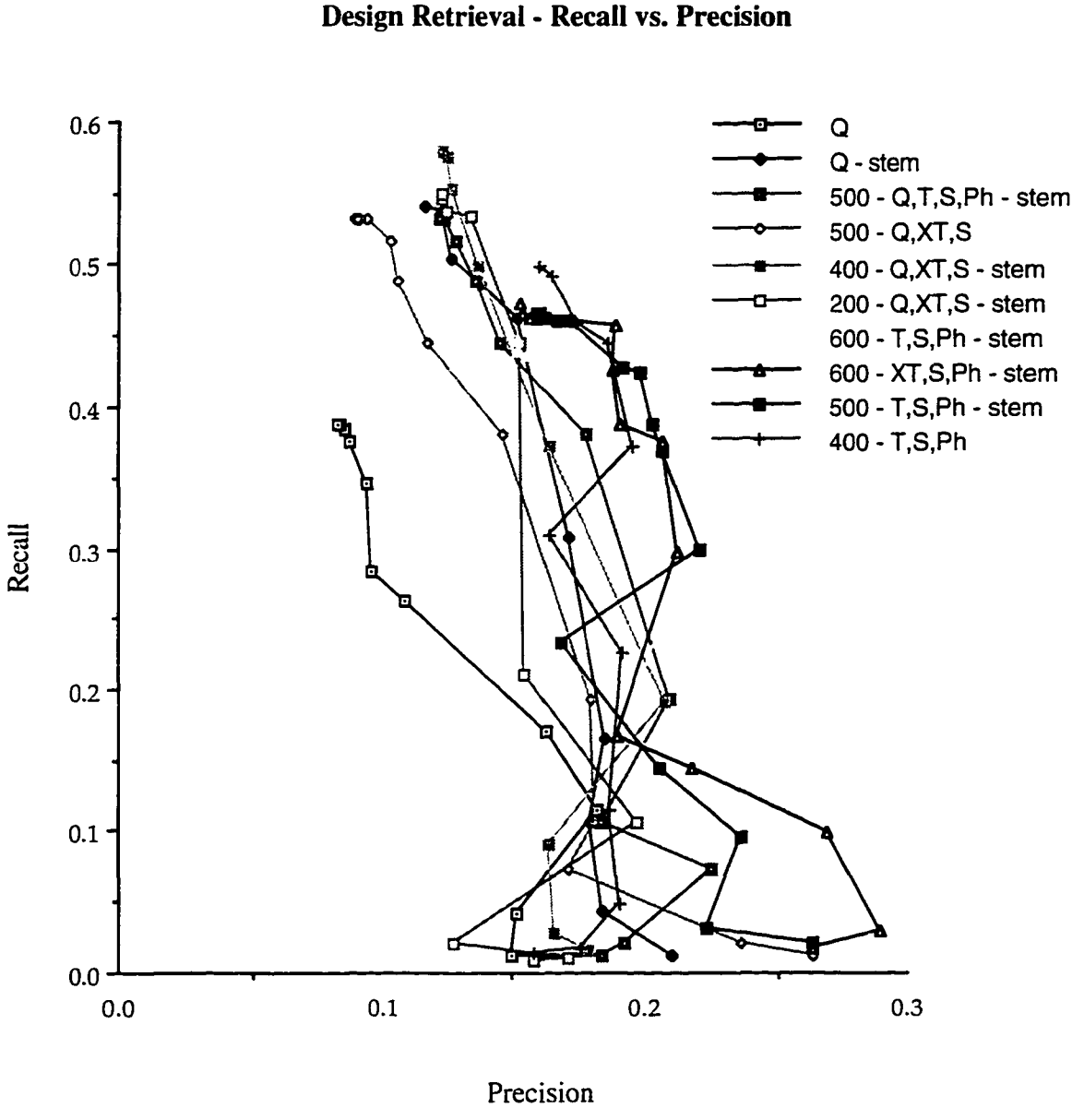


Figure 7.22 Precision vs. Recall for Automatic Retrieval over the Design Databases.

stemming the database is also demonstrated: note that while the standard WAIS query performs reasonably well (**K** of 20-40% below the best query replacement options), applying the same query to an unstemmed database yields abysmal results for all but the smallest retrieval set.

Design TREC Precision vs. Recall (Figure 7.21): Perhaps the final conclusion from the general **K** analysis is a bit misleading. Here, small return sets are significantly more precise than standard WAIS over both stemmed and unstemmed databases. In addition, for high precision within small retrieval sets a contextualization threshold of 400 - 500 applied through the local term only seems to provide the best performance.

Design Precision vs. Recall (Figure 7.22): As the strongest statement of overall system performance for automatic retrieval using the contextualize-elaborate/replace-retrieve loop on which we have been concentrating, this plot shows that: for high recall the original query should be augmented with a (moderate to large) set of contextual terms and synonyms; for high precision the original query should be replaced with a smaller set of contextual terms, synonyms, and their descriptive phrases; in between, there is a general trend of increasing the replacement context threshold which reduces recall and improves precision.

7.4.4 User-Revised Query Elaboration

Thus far, the focus has been on automatic contextualization and its application to automatic retrieval. Drabenscott's recommendations include making the context that is being searched visible to the user so that it might be corrected or manipulated to improve performance. In the next two data sets, the CDIS context assignment is made using the expert-assigned contextualization relevance measures upon which Figures 7.8 - 7.12 were based. In this case, the loop of contextualize-correct-elaborate/replace shown in Figure 7.5 is simulated for various WAIS thresholds of context retrieval. The retrieved context set is reduced to the relevant contexts above each threshold. This set is then used for the same set of query

elaboration / replacement experiments run above. In each case, the results are directly comparable to those in Figures 7.13 - 7.22. Again, the standard WAIS curves are shown as a reference, these can be used to draw comparisons among experiments. First the results within the CDIS databases:

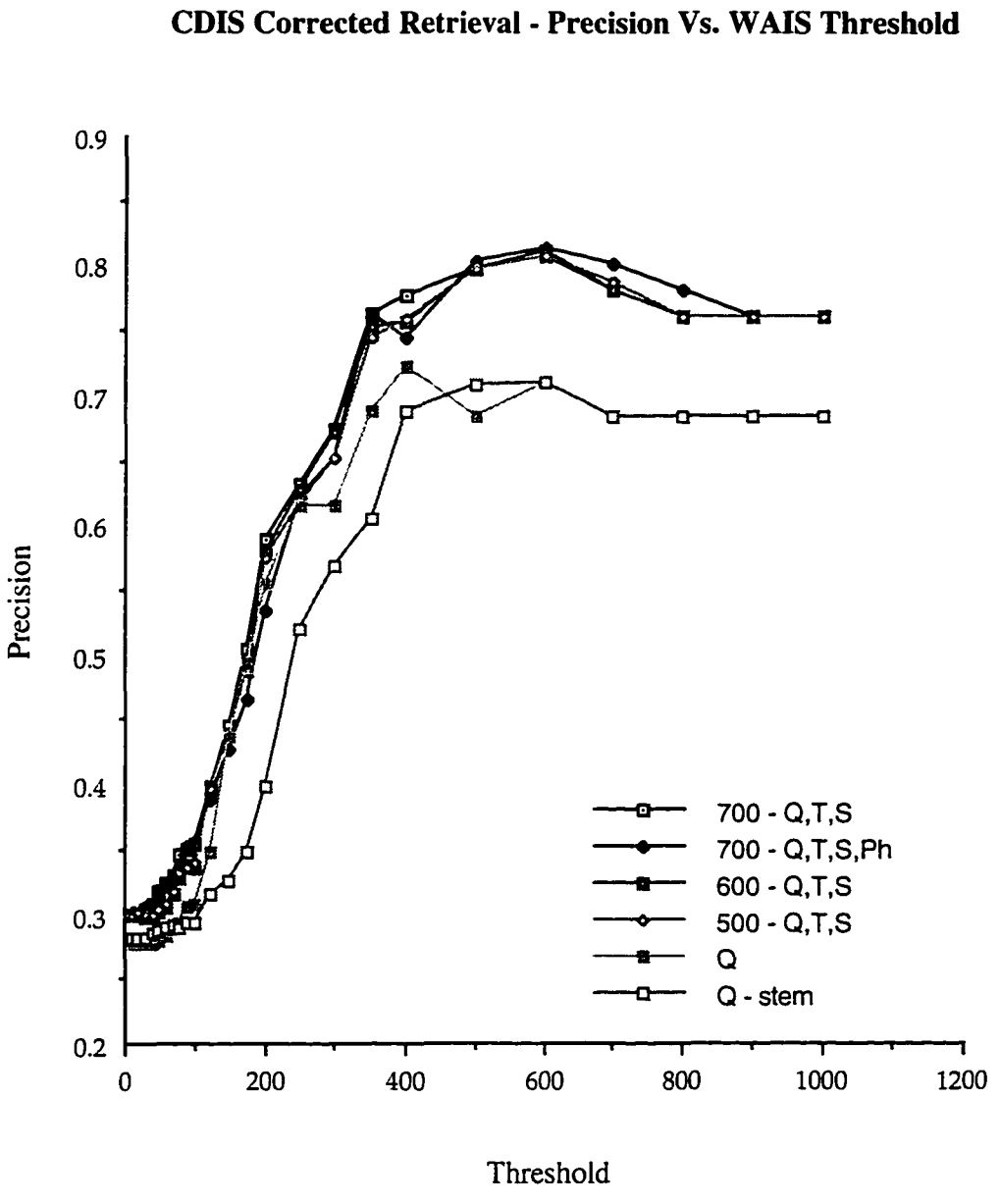


Figure 7.23 Precision vs. WAIS Threshold for Corrected Retrieval over the CDIS Databases.

CDIS Precision (Figure 7.23): While the counter-intuitive results of elaborating a query to enhance precision remains from the prior CDIS internal results, the improvement using the corrected contextual assignment is marked. Again, the most restrictive set of contexts and the most compact statement of them provide the best query elaboration results. Also consistent is the importance of using an unstemmed

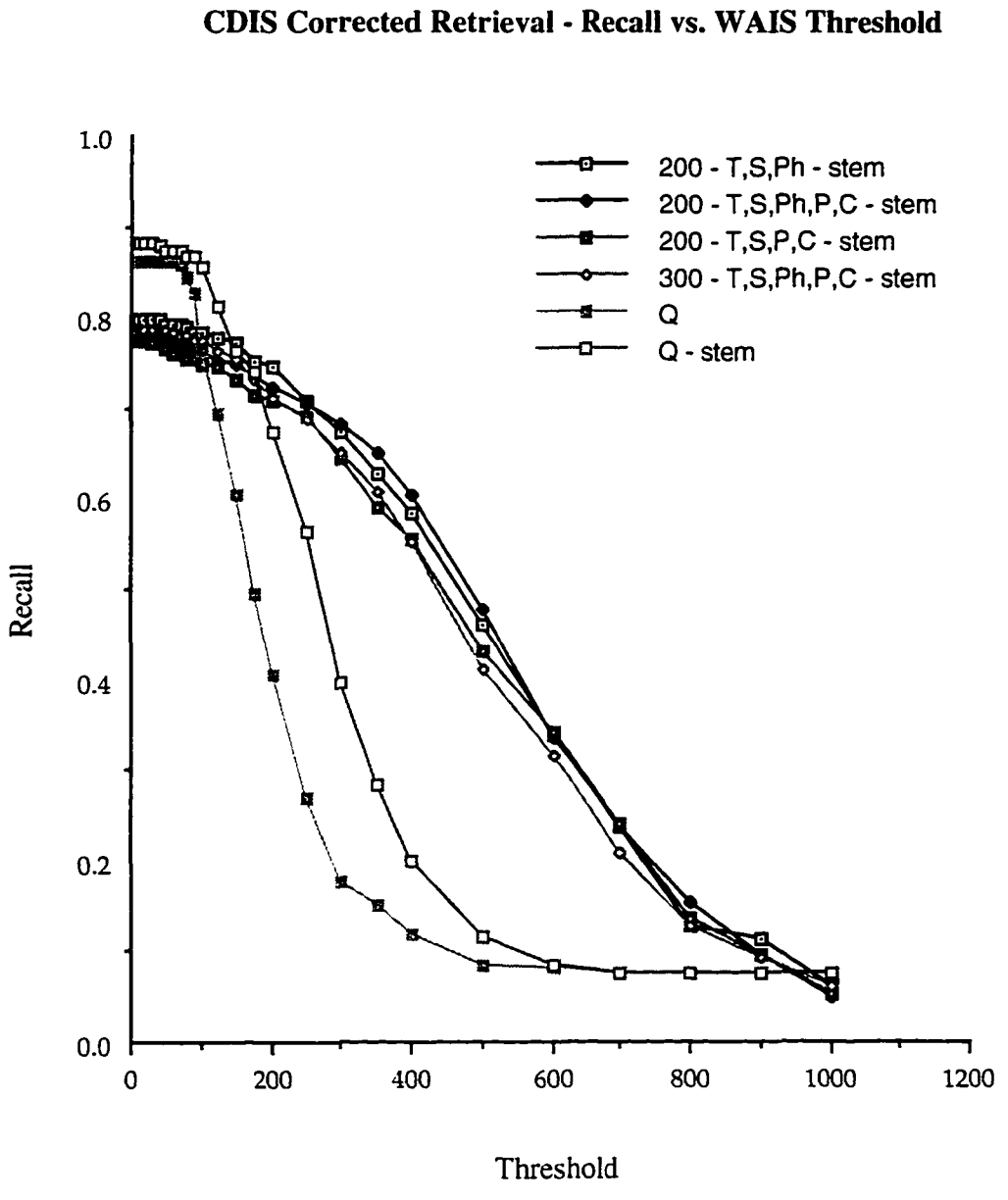


Figure 7.24 Recall vs. WAIS Threshold for Corrected Retrieval over the CDIS Databases.

database for the CDIS document-document precision.

CDIS Recall (Figure 7.24): Assuring correct contextual assignment does not improve recall at the various WAIS threshold measured here. The trend of using query replacement continues from Figure 7.14, as does the recommendation of using a low contextualization threshold and searching over stemmed databases.

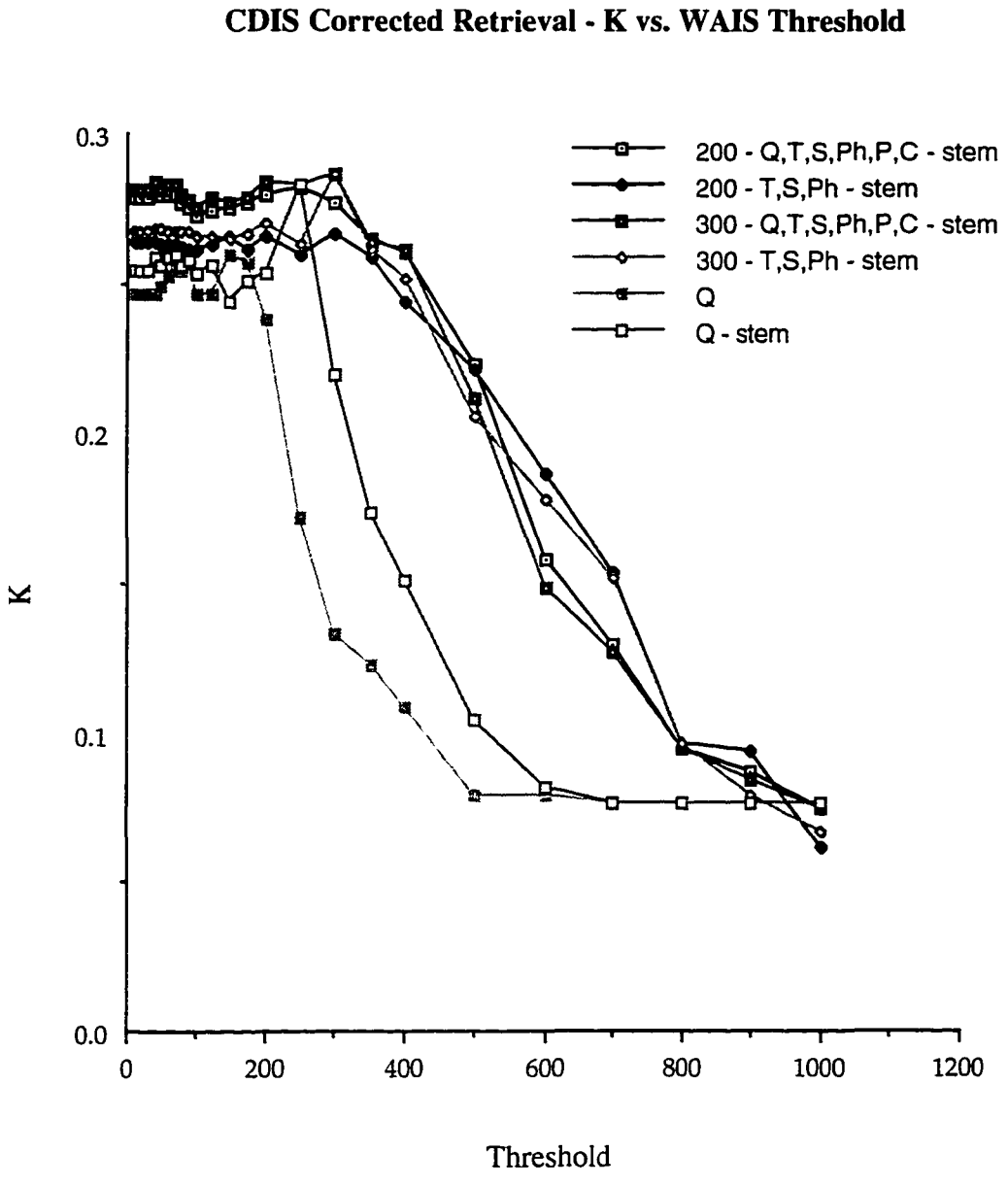


Figure 7.25 **K** vs. WAIS Threshold for Corrected Retrieval over the CDIS Databases.

CDIS **K** (Figure 7.25): The improvement in precision shown in Figure 7.23 improves **K** throughout the range of query modifications. Correcting design context produces a 20% increase in **K** over essentially the same query sets that performed best in the fully automatic experiment. Context threshold should be kept low, databases should be stemmed, queries can be highly elaborated or replaced.

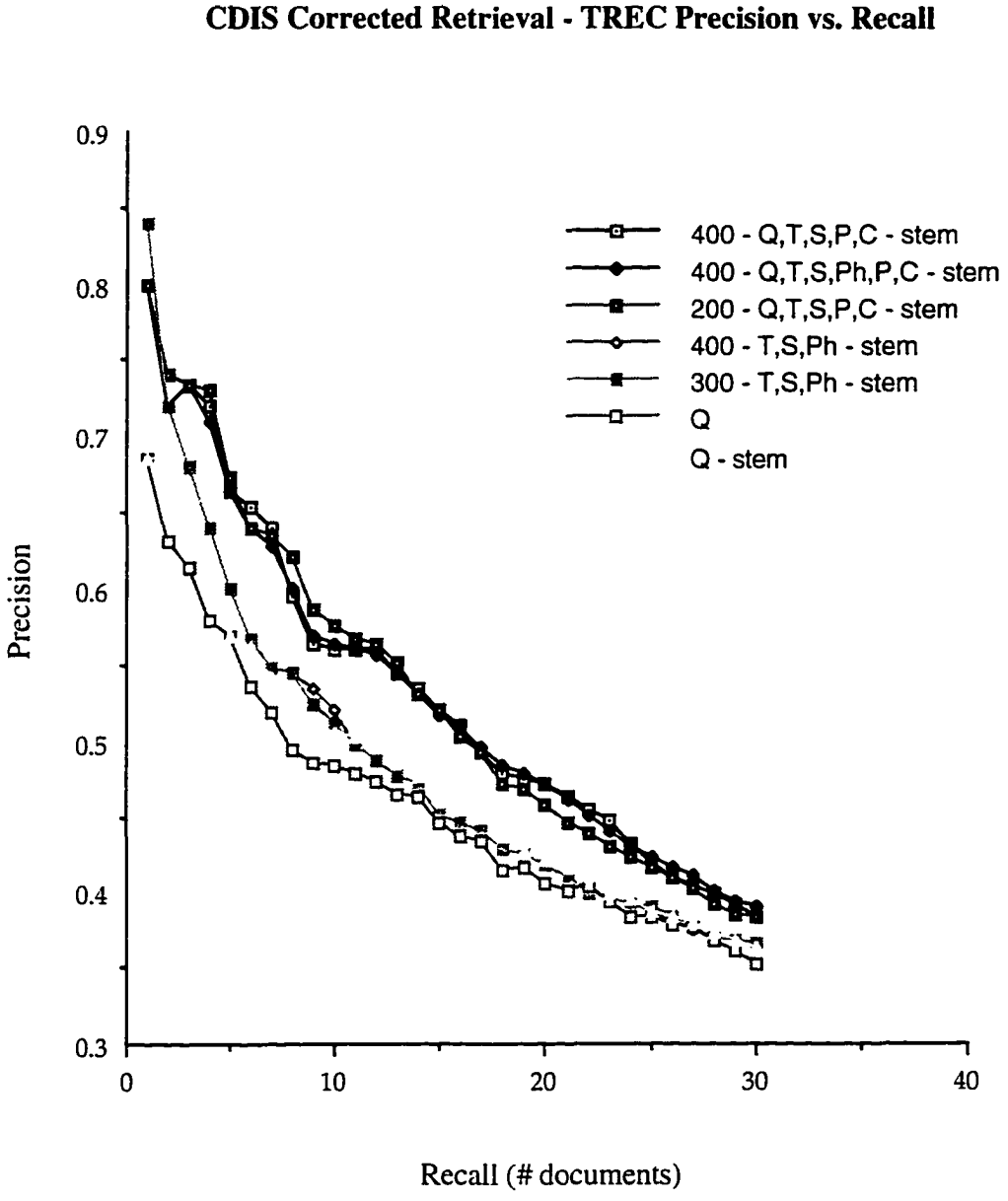


Figure 7.26 TREC Precision vs. Recall for Corrected Retrieval over the CDIS Databases.

CDIS TREC Precision vs. Recall (Figure 7.26): Correction of the design context again shows a marked improvement in the TREC performance measure. A maximum average precision of close to 85% can be achieved by replacing the query with selected terms; query elaboration yields an across-the-board increase in precision of almost 10%.

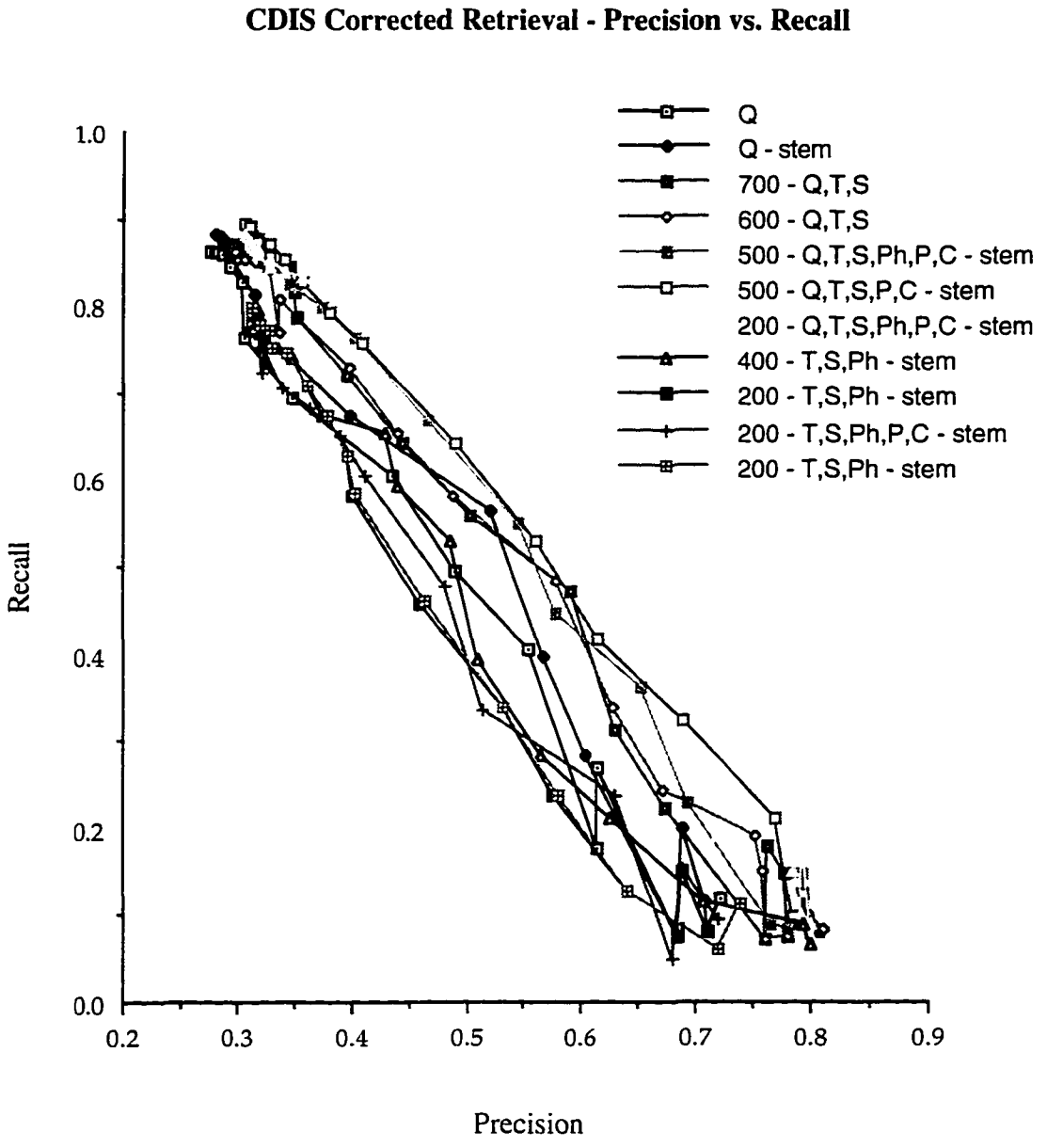


Figure 7.27 TREC Precision vs. Recall for Corrected Retrieval over the CDIS Databases.

CDIS Precision vs. Recall (Figure 7.27): Far from the jumbled results shown in Figure 7.17, here we see that the various methods sort themselves out fairly well. Query elaboration is the best strategy for searching within the CDIS database, the more contexts added and the more information from each are recommended.

Overall, the trends from the experiments in fully automatic context assignment and query

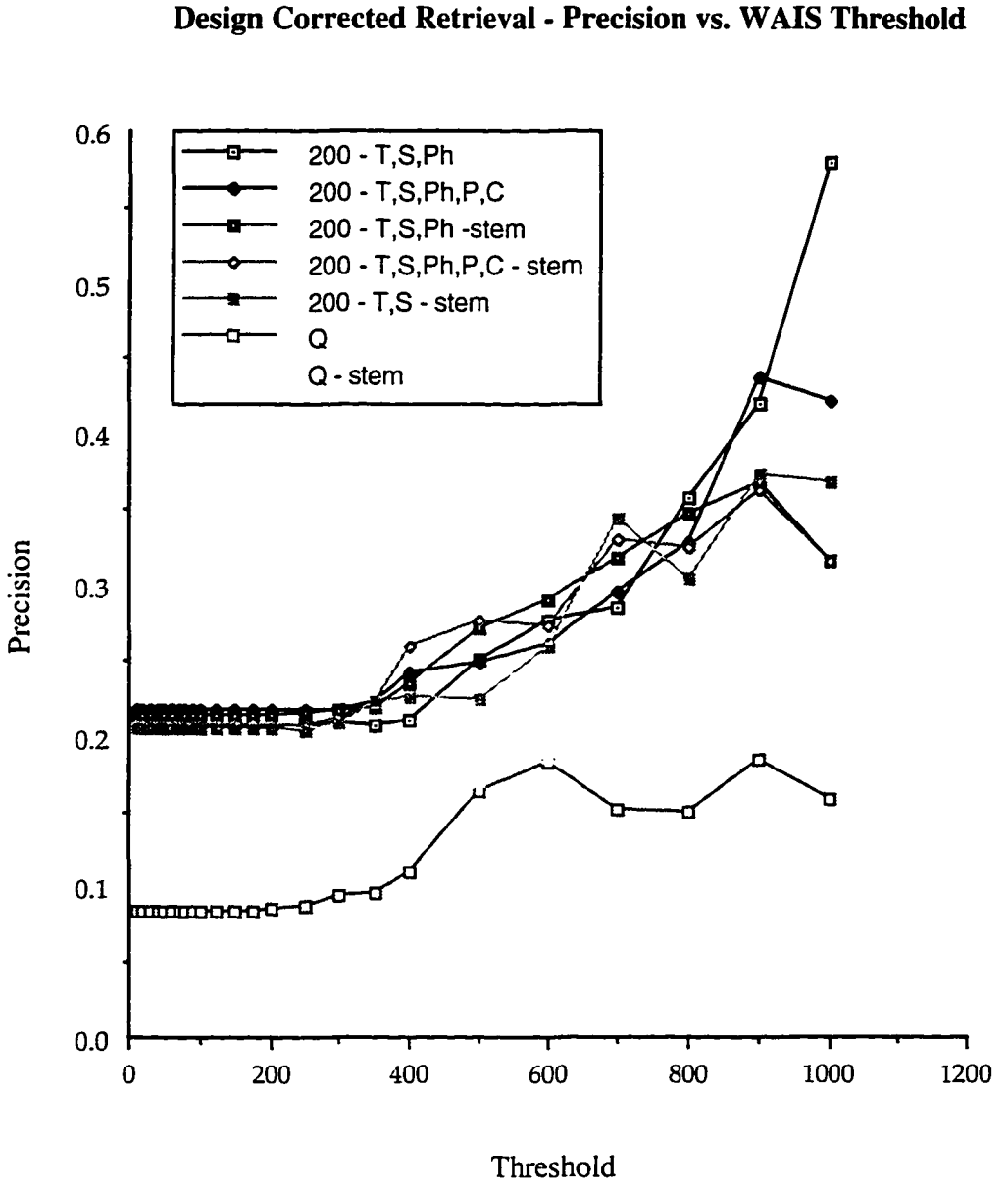


Figure 7.28 Precision vs. WAIS Threshold for Corrected Retrieval over the Design Databases.

modification carry over into the case where the contextualization is supervised. Once again, the CDIS document-document similarity measures confound the intuition that one should replace a query with succinct statements of design contexts to improve precision and conversely to grossly elaborate the query to improve recall. Now, whether a similar improvement in performance searching the design databases must be determined:

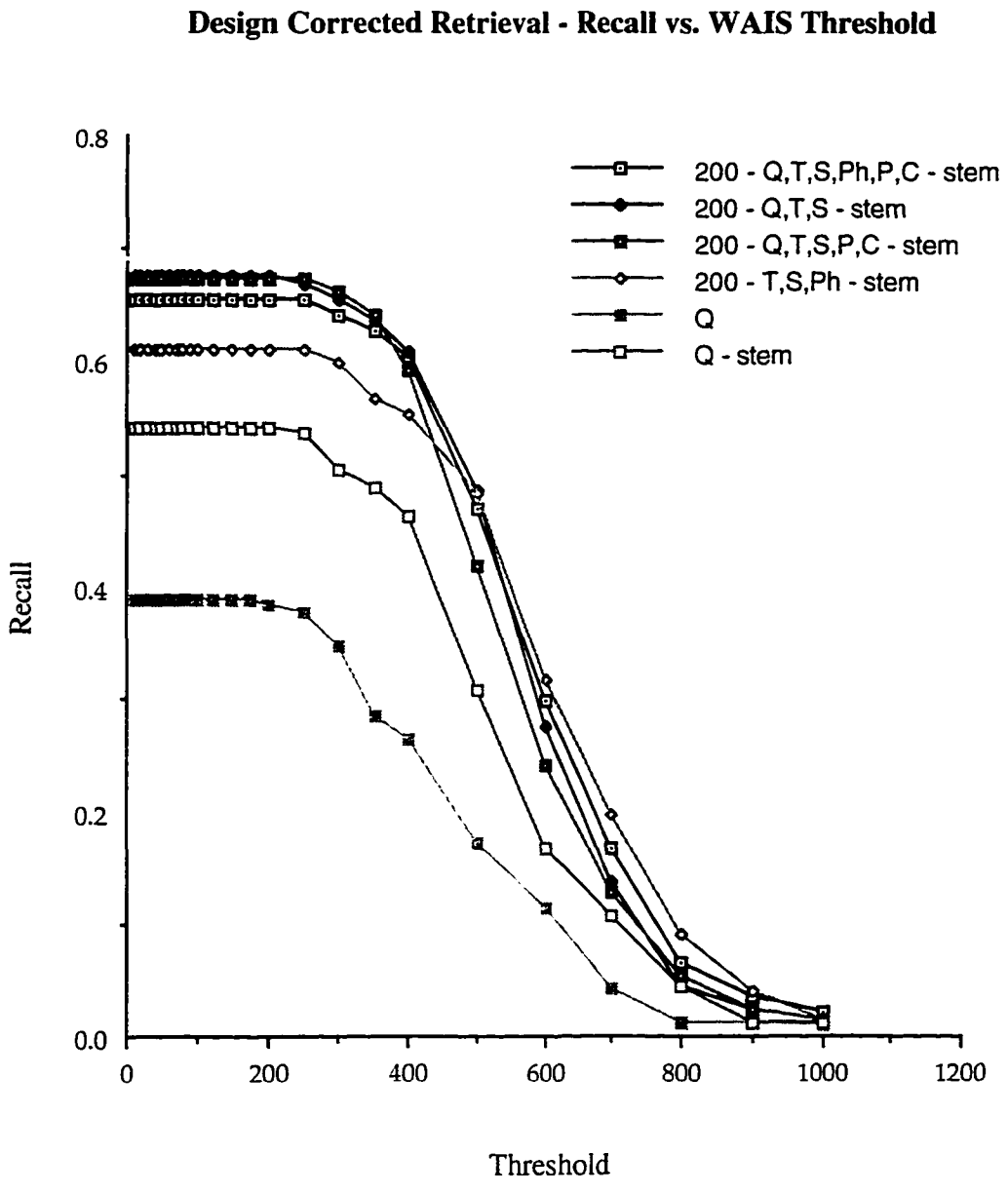


Figure 7.29 Recall vs. WAIS Threshold for Corrected Retrieval over the Design Databases.

Design Precision (Figure 7.28): Query replacement is shown to be the best way to improve precision when searching the design databases. Once again, precision results are mixed with regard to whether the database should be stemmed: an unstemmed database can provide better ultimate precision at the cost of precision lower in the retrieval set (lower WAIS threshold). Precision is enhanced by the addition of the descriptive phrase from the design context, results adding parent and child terms

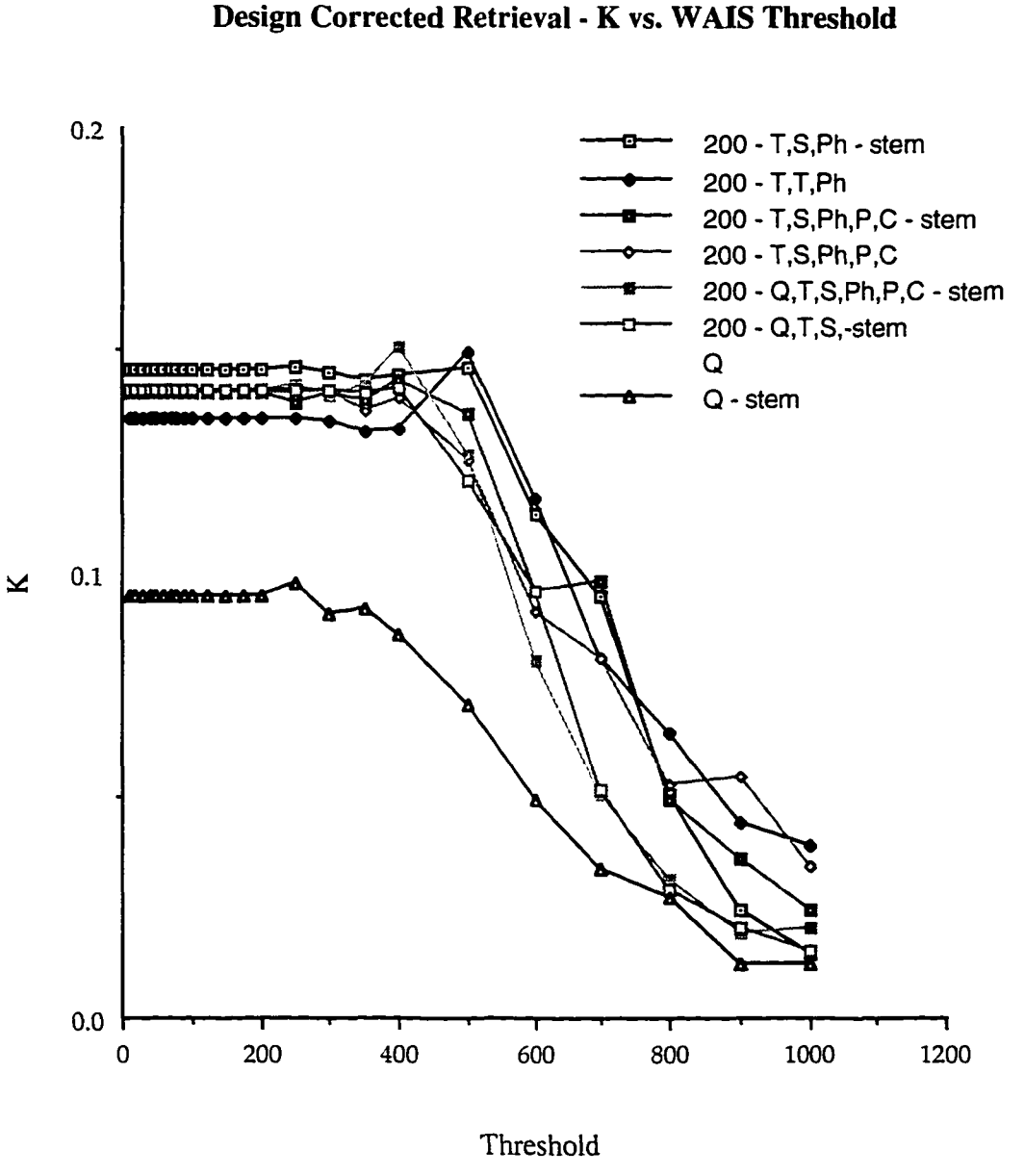


Figure 7.30 **K** vs. WAIS Threshold for Corrected Retrieval over the Design Databases.

enhance baseline precision (increase the intercept at 0 WAIS threshold), but do not have a general positive impact on precision. Overall, correcting the assigned contexts greatly enhances performance (on the order of 70%).

Design Recall (Figure 7.29): Intuition proves correct again for recall in the design database search. A trend toward using a low contextualization threshold indicates that a

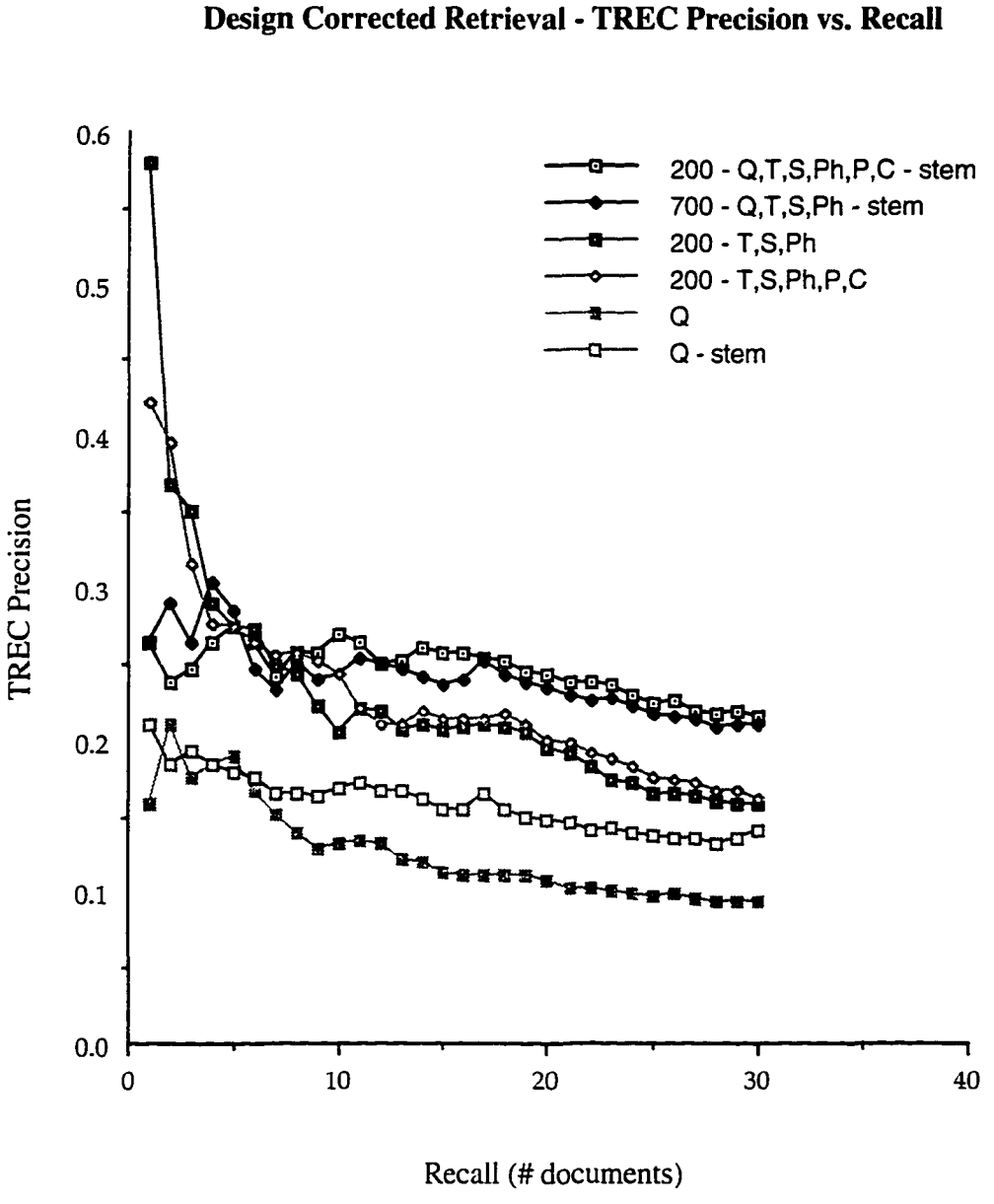


Figure 7.31 TREC Precision vs. Recall for Corrected Retrieval over the Design Databases.

larger number of CDIS design contexts improves performance in both precision and recall. One interesting note here is that in the upper WAIS threshold regions, query replacement outperforms elaboration. This effect does not carry through at lower thresholds where the larger term vector produces lower scores for the most relevant document (not necessarily changing the order), but manages better net recall.

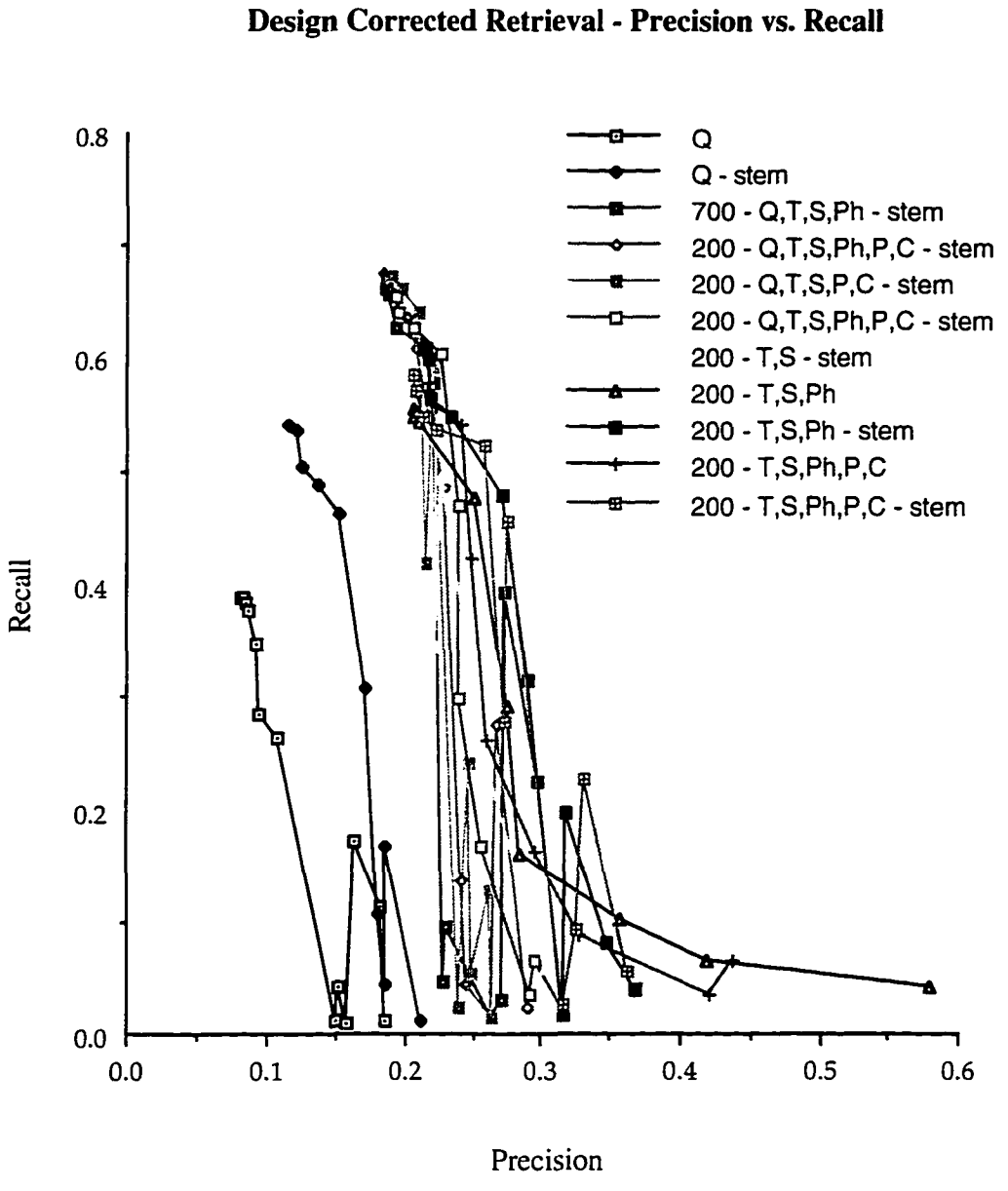


Figure 7.32 Precision vs. Recall for Corrected Retrieval over the Design Databases.

Design **K** (Figure 7.30): As an overall measure of retrieval system performance, **K** shows a marked increase once design contexts are corrected. Recommendations for the best query elaboration strategy are not as clear as those to be garnered from Figure 7.20, however. Here, aside from the trend toward using low contextualization thresholds, results show a mix of query elaboration and replacement as well as a mix of stemmed and unstemmed databases (for query replacement).

Design TREC Precision vs. Recall (Figure 7.31): The TREC presentation of the design retrieval results starts to indicate an emerging strategy: unstemmed query replacement to achieve high precision, stemmed query elaboration for better recall. The difference between the high threshold and low threshold query elaboration demonstrates that precision increases with more concise statement of context while more complete contextual addition produces the best recall.

Design Precision vs. Recall (Figure 7.32): Once again, an analysis of the overall precision vs. recall plot provides the best statement of an optimal strategy for enhancing query performance in the CDIS global index. First, notice the pronounced shift to higher precision that is produced by correcting the assigned design context. Figure 7.32 shows an almost 100% improvement in precision which results from supervising context assignment. The following strategy is indicated by the results: for high recall, query elaboration over a stemmed database is optimal; for high precision, query replacement over an unstemmed database is the way to go; for overall performance, query replacement over a stemmed database is best. In each case, there is little choose between adding terms, synonyms, and descriptive phrase or adding the parent and child terms as well.

7.4.5 Discussion of the CDIS Global Index

The above testing shows significant results from several standpoints. The objective measures of performance derived from the experiments can be used to tune the query

elaboration / automatic indexing and direct the stemming for databases within the CDIS. This is certainly important, but is not the main result to be drawn from the indexing exercise. The primary implication is that an external, hierarchical structure can successfully be applied to an unstructured information base. In fact, most documents within the system can be represented simply by a set of contextual nodes within this external abstraction structure. The interface provided in the CDIS allows a user to understand the bias that is being placed on a query by this external structure and to manipulate this bias in response to information retrieval success. This is an important finding for networked information retrieval because the free-text databases contain all manner of extraneous information (both because the WWW contains intentionally extraneous information and because indexing methods do not use standard stopword lists and indexing methods). This is far from the coherent databases typical of information retrieval research mechanisms in general.

Of particular note is the effectiveness of eliminating the actual query (document itself in this case) when accessing unfamiliar information sources. This is the strongest application of bias and is most appropriate for identifying 'design' relevance by placing an intermediate design representation between the query and the information base. The effectiveness of the CDIS global indexing method on information sources outside of the CDIS systems points out an extremely important aspect of the system design: it is built to operate over *any* free-text search engine by applying an engineering bias to queries *before* they are submitted. The ability to operate through the standard WAIS protocol while improving upon its query performance from outside of the server is significant. Woven into the network standard from the query side, the CDIS global index is a generic tool for design information search. Both the size of the context base (about 1000 design contexts are supplied) and the size of the external design document base (about 3000 design documents of wide-ranging subject matter), the results garnered from the 'design' information recall should extend to resources in general. Considering that much of the information within the CDIS information base is fairly narrow in focus and somewhat interrelated (many of the case studies are quite similar in abstract content), the strong results from applying the CDIS index to the 'design' information base bodes well for application to other information

resources.

Because the unstructured information in the CDIS resists machine-understandable encoding, the main focus of case-based reasoning as performed in the CDIS is similarity matching. The development of taxonomic hierarchies created by explicitly varying levels of abstraction along three separate design 'axes' provides two effects toward supporting this: automatic indexing of documents, and elaboration of queries. Parameters gained from the above described experiments can optimize the application of both automatic indexing and query elaboration strategies. The CDIS query interface design, described in the next section, applies these strategies while taking into account the recommendations of Drabenscott for on-line catalogs.

7.5 The CDIS Design Information Access Interface

The WWW interface to the CDIS global design index is shown in Figure 7.33 below. The designer submits a query to the system in the form of free text, just as in a typical WAIS implementation. This can be a statement of the current problem, be asked in the form of a question, or even be a document submitted to the system just as was done in the previous section's experiments. The CDIS global index uses the text of the document to assign a set of contexts along the three abstraction axes. The designer can use these in an automatic mode just by submitting them, with or without his initial query, to the WAIS databases of the desired portion of the CDIS. Alternatively, a designer can adjust the query elaboration of the system by navigating around the design context hierarchies using the assigned context as a starting point. A simple scale can be tuned by the designer to emphasize precision, recall, or a combination.

The interface directly addresses the most serious of faults described by Drabenscott for subject index searching. It shows which subjects are derived from the user query, lets the user modify these subjects either by changing the query or by navigating among the subjects, and provides for feedback into the system through relevance feedback. In addition, because the interface is implemented on top of a WAIS client, important features

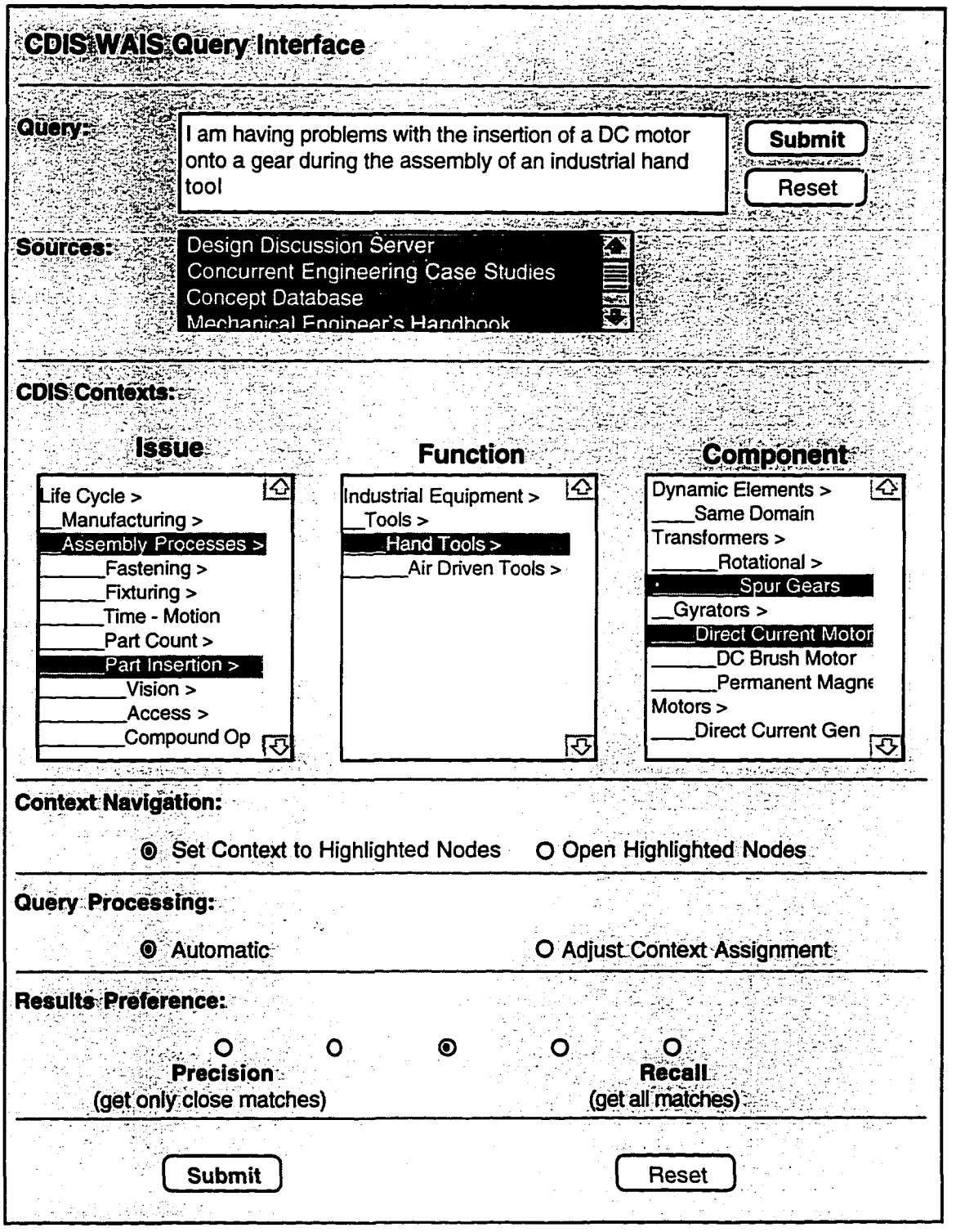


Figure 7.33 User interface for CDIS global indexing system. Similar to the WAIS interface of Figure 4.5, here the query suggests contexts which can be manipulated and re-submitted. Stemming and query elaboration / replacement are controlled through a Precision - Recall scale.

like relevance feedback, and Boolean and literal searches are supported. More importantly, any WAIS database can be queried from a 'design' standpoint without any local modifications.

7.6 Summary

The hypermedia documents for representing concurrent engineering issues within design case studies, ongoing design discussions, and component selection templates are indexed indirectly by both functional decomposition and design issue through the application of an externally expressed design context bias – the CDIS global index. This bias is easily extended through direct coding of more design contexts (required for functional decomposition knowledge). Through performance feedback from users, the system can also tune the programmed contexts using the well-defined methods of the information retrieval community.

The overall goal of the CDIS is to transform simple information retrieval into case-based reasoning that supports derivational analogy. Figure 7.34 below demonstrates the general concept of a user query transforming a set of independently-developed product design case studies into a new 'case study'. In the case of Figure 7.34, manufacturing portions of some of the design case studies elucidate general manufacturing concerns queried for by the user. By moving up and down the abstraction hierarchies, the piece of case study 'pie' becomes larger and more general or smaller and more focused.

The first step of case-based reasoning for design is to determine similar design instances. In light of this need and the chosen representation, we have discussed a unifying indexing scheme where the structure of the domain is used to index structured aspects of design and to enhance the performance of less structured information retrieval methods. Local indexing in CDIS design documents makes it possible to carry out the second step of case-based reasoning: adapting design information from the case library for use in the current design context. In the case of Concept Database documents, direct adaptation is possible by derivational analogy – replaying selection methods that are similar or gathering design

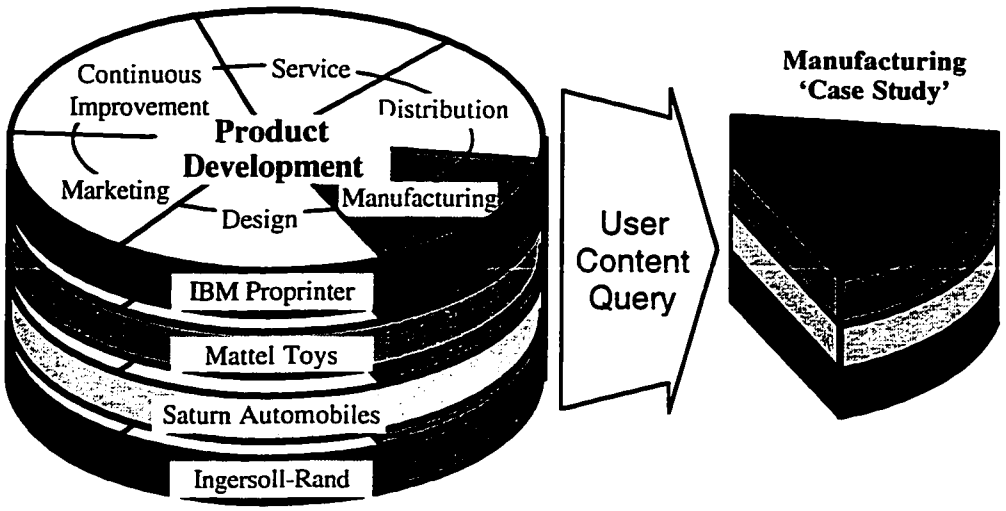


Figure 7.34 Independent Case Studies Transformed and Combined by a Query to the CDIS.

relations from several design templates toward producing a new one. For the less structured information (i.e., the uninterpreted textual information of the design discussion server and the concurrent engineering case studies), adaption of case information is left to the user.

Through a combination of free-text and structured search methods, the CDIS accomplishes the goal of supporting the transformation of the case based reasoning paradigm from narrowly focused, domain dependent applications to an open, generic information retrieval task. By applying an external source of knowledge to bias the information retrieval portion of the CDIS, we have created a design filter that eliminates the need for hand indexing design information and opens up the growing information 'universe' to conceptual design. The results of the experiments of Section 7.4 indicate that a great improvement in query performance of standard WAIS databases is offered through the application of externally applied, user-corrected bias. The next chapter provides a complete discussion of conclusions that can be drawn from this implementation of the CDIS and the directions for future research in case-based conceptual design support in a networked environment.

Chapter 8

Conclusions and Future Directions

The CDIS has been proposed as a tool for supporting the application of concurrent engineering principles from early conceptual design through concept selection. The difficulty of representing the multiple concerns of the concurrent engineering team members in conceptual design has led to a representation that is rich enough to capture design information while introducing minimal bias. Multimedia documents are effective for communicating the context of design decisions and negotiations. In the process of formalizing design decision making IRTD is applied over predefined abstraction hierarchies. This puts control over the representational bias applied to design cases in the hands of the designer who can trade design commitment against its expected value. Finally, design models formalized through this process join hypermedia and mathematic representations which capture both the informality of conceptual design and the rigor of analytical evaluation. Linked within a networked hypermedia server, this combination of representations increases the potential for the reuse of design experience through its contextually complete capture of both design process and product.

8.1 Summary

In Chapter 2, case-based reasoning (CBR) was introduced as a promising means of applying information from past design instances toward solving similar problems.

Typically CBR design systems apply a strong bias to the information stored about a design – the only information available to the system to match similarity and propose new solutions. This bias introduces simplification which allow a straightforward implementation of the CBR paradigm of retrieve, compare, patch, iterate. Most implementations focus, as we have in the CDIS, on the retrieval aspect of CBR; similarity matching is used to form a set of cases from which some general conclusion can be drawn. Any such conclusions are made by some sort of reasoning system: for domains that are not well understood, this system is based on weak methods – retrieved cases are 'averaged' to propose new solutions; in better understood domains the reasoning is more rigorous, cases serve as rule 'chunks' that can be decomposed and reformed to match design requirements (almost always requirements on some abstraction of function).

Conceptual design is not a well understood problem, in fact the main focus of conceptual design is to *try* to understand *what* the problem *is*. Our model for what happens during this process is one of alternate information gathering and evaluation. This design model does not change as the process continues, the representation of design information just becomes more formal. In fact, the process of design is one of manipulating a representation of the design problem from a very abstract level to increasingly more concrete ones until enough information is defined so that an artifact (or process or system) can be realized. In previous work in more concrete representational schemes, this process has been called Intelligent Real-Time Design (IRTD). IRTD has been applied where design requirement can be represented mathematically, design options as a finite list of alternatives. In each case, varied levels of abstraction are supported: intervals, probability distributions, and real values. IRTD provides an algorithm for evaluating, with respect to the design requirements, the most promising area design focus – the best design abstraction to reduce. The CDIS strives to extend this ideological formalism into the design activity that precedes (and in the case of the Concept Database, includes) the stage of design where the IRTD method can provide such guidance.

Case experience is a valuable tool and one which we attempt to exploit for guidance in the

unstructured arena of conceptual design. Three components have been identified as significant means of storing such experience: illustrative case studies which highlight best practices, design histories encoded as design discussions using the IBIS method, and IRTD methods applied to catalog component selection. The first component supplies information about running a design project and general considerations with respect to mitigating the often orthogonal goals of various disciplines represented in the concurrent engineering process. Such organizational insights lead to more artifact oriented discussions of function and alternatives in the collaborative design discussion server. Here, the concurrent engineering team shapes the design through argumentation over design requirements, system functionality, and the embodiment of this functionality in an artifact. The decomposition of this artifact into subsystems which can be objectively evaluated leads directly into the concept database where mathematical models, both new and recalled from case experience, can be applied, optionally in the presence of design alternatives represented as catalog components.

The unifying theme for these CDIS subsystems of disparate purpose is the use of the WWW as the main interface. This accomplishes a couple of functions and has one interesting side effect from the standpoint of indexing and information retrieval. First the functional aspects – using the WWW and HTML documents as a uniform presentation standard provides for easy integration among the three subsystems. Through the inclusion of a link to a URL in one of the other subsystems, the design discussion can easily represent not only the background information used to raise issues or perform subjective evaluation but also the mathematical models used to make objective evaluations of design alternatives. In doing so the design history represented by the discussion moves easily from the abstract to the concrete, shifting representation as is necessary to resolve design issues. Another functional aspect is obvious: all design information is available from a single interface, which through its support of MIME standards, is capable of launching appropriate application programs upon downloading input files linked into the discussion. The interesting side effect is that the effort to communicate design information textually through three different interfaces but through the same medium results in a uniform basis

for indexing that information – text. In addition, this text can be tied to URLs that launch CDIS application modules automatically. Not only is static information indexable in this way, but also more dynamic applications like component database searches. Such searches can be redone to define new component options that have become available, or the prior results used to understand the context of the original decisions.

This brings us to the effectiveness of the indexing system. We have chosen free text as a least biased means of representing design information. This means that anything that can be stated (or even just hinted in a reference to an illustration or application file) can be captured and indexed. This is important from a design reuse standpoint in that no information is lost in the translation to a language that is more 'computable' [Warner]. As is always the case, 'you don't get something for nothing'; the increase in expressiveness of the textual representation bears problems for applying that information. The solution we have proposed within the CDIS is that two representational schemes can offer both the reusability of a structured language and the contextual richness of an unstructured one.

For matching contexts to find appropriate design reuse information, we have proposed a scheme of applying an external bias to the unstructured representation. This external bias is directed toward enhancing the effectiveness of the system in distilling the *design* aspects from a query or document from other considerations that are not important to design. In addition, this bias can be used to manipulate the abstraction level of design issues so that information that is returned is matched to the abstraction level of the query. It is of little use to get information about the mounting dimensions of a particular motor when all that is needed is a general idea of the size of the motor. In design we use abstraction to agglomerate information that is too difficult or not necessary to process in detail. It is vital that the design retrieval system recognize the abstraction level present in the query and try to match that in the documents retrieved. By demonstrating the contextual 'guess' of the system to the user, the CDIS retrieval engine allows the user to tune the abstraction level before querying or in response to poor query performance.

The notion of function is captured through the addition of a hierarchy of functional decomposition to index into the design discussion by function. In terms of component behavior, a more structured approach can be applied through a taxonomy of engineering components. This is demonstrated in the electric motor domain chosen as the application set of the Concept Database. Here, behavioral descriptions can be made of the components at various levels of abstraction. In addition, textual descriptions of defining relationships among various performance parameters are also used to characterize a component. In effect, the function and behavior hierarchies are joined together at the component level where the two become somewhat synonymous (unless you are using a motor as a doorstop, of course). Structural aspects are not directly indexed in the CDIS due to the difficulty of doing so in the chosen representation languages. A discussion of avenues for indexing structure will be taken up in the section on future work.

One of the main goals of the CDIS is to support concurrent engineering. To this end, case studies of the successful application of concurrent engineering techniques are a large portion of the knowledge base of the system. The architecture of the CDIS has been defined partly by the need to support multiple access to design discussions from multiple project participants. Finally, a comprehensive set of issues from various stakeholders along the design life cycle makes up the third contextual hierarchy. This issue-based approach is parallel to the argumentation process implemented in the design discussion server. These design issues resonate throughout the system, appearing as the focus for technical highlights in case studies, as issues throughout the design discussion server, and as constraints and objectives in the concept database. While the function and behavior hierarchies are natural conjoined, the concurrent engineering issue hierarchy is somewhat orthogonal to the other two.

Overall, the CDIS presents an effective tool for conceptual design by integrating the central theme of case-based reasoning in a networked hypermedia framework. Design capture is done in a rich representation formalism and mechanisms for searching and reusing design information from past cases are supported. The entire system is based on network

standards and thus provides ample avenue for expansion in the future.

8.2 Specific Research Contributions

The development of the CDIS has importance to three distinct research communities. Foremost is the mechanical design community to which the following contributions have been made:

- Approximate Design Model Construction: Integrating probabilistic models constructed from databases of component parameters with the mathematical models prevalent in design provides a powerful basis for design modeling and decision making. This technique, in combination with knowledge from hierarchical component classification, has also been shown to be effective for modeling design problems at varying abstraction levels. While specific examples have focused on electric motors, virtually any mechanical component (e.g., power transmissions, bearings, sensors actuators, etc.) can be modeled using the same method. The resulting models readily encode aspects of a design like size, weight, and shape which are often difficult to model analytically but significant to multiple objective, concurrent engineering design prescribed by industry 'best practices'.

- Management of Design Abstraction: The application of decision theoretic and information value techniques in combination with the approximate design models discussed above produce a formal means of analyzing the impact of design abstraction in conceptual design. Conceptual design is a process of synthesizing a set of prototypes and evaluation metrics from an informal, abstract statement of need. Informed as to the impact of a range of possible design decisions on (approximate) design objectives, designers can more effectively trade off the opportunity costs of eliminating design possibilities with the gains likely to be realized from commiting to a design option. Formalizing the manipulation of design abstraction provides not only optimally directed designs, but an optimally directed design process in which abstract reasoning eventually yields to formal analysis and CAD models

- User-Verified Query Elaboration: More general is the improvement in system performance to be gained by including the user in the query elaboration process. When a user corrects context assignment, system performance (measured by any of the typical information retrieval metrics) is virtually doubled. Placing the user in the context assignment-query-retrieve loop is the single most significant variable toward improving retrieval performance.

8.3 Future Directions

The design of the CDIS presents ample opportunity for growth; its basis on open network standards provides forward compatibility for adding new applications and document sets. This discussion will focus on possible directions for future research on the core applications of the CDIS, specifically on improving the case-based reasoning capabilities of the system. The key areas of focus for improvement are in increasing the number of design representations and in enhancing the similarity metrics used to retrieve them.

8.3.1 Representation

Abstraction level has been mentioned throughout this work as a description of the level of detail with which one models a design component or issue; this is informational abstraction. In the IRTD method, abstraction level is treated as a decision variable – reducing abstraction comes at a cost which can be traded off with the quality of the decision to be made. Three abstraction levels are available in IRTD: intervals, probability distributions, real values. The type of analysis that can be carried out effectively at any stage is determined by the level of abstraction at which the problem is modeled.

For the CDIS, the main indexing method relies on an incomplete interpretation of the design information represented in the design documents. It has proven effective for the general purpose of the concurrent engineering issue-based search with a nod to the more established function and behavior search modes through the function and component context hierarchies. However, the representation is not capable of 'understanding' much

about the design problem beyond the general context. The CDIS also takes the approach of providing mathematical models for representing design decisions made objectively within a decomposed problem space within the overall design context. These can be accumulated into larger mathematical models, but assigning meaning to the variables comprising this model is again a matter of textual description, not a usable semantic description.

Perhaps an additional representational formalism should be provided by 'plugging in' a fourth basic module, one similar to the DME[Iwasaki and Low, 1991] where representation of the overall design problem and description can be made in an object-oriented way. Probably the most promising collection point for such a system is an augmentation of the IBIS method with a special transition. Nagy et al. [1992] suggest that by adding a "constraint" transition to the basic IBIS set of issue, position, and argument allows a more formal representation of design issues. The Redux' server [Petrie et al., 1994] relies on such statements to identify conflicts among competing design issues, in addition managing constraint set versioning to ensure that all design options are explored. Others use similar constraint representations to manage negotiation of competing constraints [Bahler, 1994]. The main question is: do we need another representational scheme or can we automatically distill such information from textual descriptions.

In the discussion underlying the selection of the term vector space, it was intimated that such a representation is a valuable first step toward the application of natural language processing techniques in the system. It is necessary to first understand the 'neighborhood' of a statement before attempting to interpret it. In many senses, the current implementation of indexing within the CDIS is similar to methods employed in the message understanding community. The main distinction is that whereas the CDIS stops with the contextual classification, message understanding employs limited natural language processing to distill significant facts from the data stream. An example is the TemplateFiller [Schuldberg et al., 1993] system which fills a prescribed set of semantic 'slots' from newspaper product announcements. Shah and Bliznakov [1993] have proposed that design be done in a Prolog-derived formal language capable of representing typical design objects, concerns,

activities, product descriptions, etc. Rather than prescribe such a language, it seems that it could form the basis for interpreting textual design content using natural language extensions to the current CDIS system. Remembering that design discourse is carried out primarily for communication, it is not unreasonable to assume that textual regularities that are present in messages from other domains might hold for design. Perhaps understanding these regularities can also be used as a query tool toward identifying more exactly topics of interest to users (e.g., What factor of safety is prevalent for aerospace applications?). This does tend to skirt some of the main motivation for a free-text system, that of making the user understand the context of a design decision by interpreting it within the original design discussion. However, moving toward at least cursory text interpretation would begin to close the gap between the CDIS and more typical case-based design systems.

The final aspect of design representation is one has been conspicuous by its absence in the discussion thus far – indexing non-textual media. A large amount of design information in mechanical engineering relates to geometric representations of objects. At this point, the CDIS is capable of indexing textual information including that which can be 'scanned' from graphics (the design documents tested in Chapter 7 consist entirely of text captured from images). While this can point to component tag numbers and general drawing notes and titles, it does not capture the essence of the drawing. Of course this is a very difficult problem. In architecture there a couple of notable efforts for contextualizing graphical elements. The first [Gross et al., 1994] deals with the interpretation of graphical gestures which represent ideas like 'sound dispersion' or 'traffic pattern'. The second is an interesting exploration of the 'gestalt' of a set of graphical elements used in the FABEL project [Schaaf, 1994]. Here, spatial relationships among hierarchically structured graphic elements are used as similarity measures to index into a large set of drawing files. This is, of course, only a part of the case similarity matching scheme but does address significant spatial analogical reasoning issues. These are but a couple of the efforts under way that are interesting from an engineering perspective. Pattern recognition and image understanding is a rich field unto itself and will not be treated in any depth here. The results of these efforts do promise to provide dimensions of similarity for graphical and video objects that will

certainly be of interest in the future.

8.3.2 Case-Based Reasoning

Though its representation of varying component model abstraction level, the CDIS also holds promise as a more typical case-based reasoning system. CADET [Navinchandra, 1988] was discussed in Chapter 2 as a prototype for applying case representations of components and artifacts toward meeting functional specifications for new designs. The approach used in CADET is to establish a set of canonical design function variables and represent components as objects which produce qualitative transformations among a subset of these variables. One of the shortcomings of this method is its simplistic representation scheme based solely on monotonic qualitative transformations. Michelena and Sycara [1994a&b] propose extensions to the methodology that take into account second order effects. The Concept Database subsystem of the CDIS applies a similar tack of establishing a set of design variables and defining relations among them, in this case the relations are mathematical ones. Certainly the first and second order monotonicities useful to CADET can be derived from these relations (at least within some bounds). The additional 'win' here is that components can be modeled at multiple levels of abstraction, allowing the user to begin design specifications with rather abstract functional representations and add more detailed constraints as proposed solution sets are evaluated. For instance, the concept of a linkage can be abstracted to perform many different possible functions; as the functional specification matures, the class of linkage is defined. As this follows the model for conceptual design that has been espoused throughout the development of the CDIS, it adds considerable power to CADET's single abstraction level representation.

8.3.3 Context Issues

A indexing method has been presented that relies on the creation of a set of contextual hierarchies that can be used to apply a bias to design information for the purpose of search. This has been done in the simplest possible way – establishing a set of contexts similar to those applied in message understanding, and arranging them by abstraction level.

Information retrieval techniques are used to match some these contexts to the user query, the contexts selected then applied to the task of retrieving design information from unstructured databases. The user is introduced in the intermediate step to tune the context matching and make choices that can increase either the recall or the precision of the search. This is intended to work as a sort of intermediary representation, similar to an activation network. The query activates nodes in the context network and these nodes, when applied to the search process, activate document nodes in the design databases.

Where the CDIS indexing method differs from an activation network is in the mutual excitation or inhibition that can be used to spread activation around the local design contexts that are 'excited' by terms from the query. In a case-based system used for retrieving relevant videotape lessons Hirichs et al. [1993], has had good success. Because in the case of videotapes no reliable method has been found to distill content from the media, all indexing must be done by hand. An interesting possibility is derived from the work of Plaunt and Norgard [1995] and that of Schutze and Pederson [1994] – learning contexts from free text through word collocation. By having the system 'read' engineering texts unsupervised, it might be possible to induce the term-term relationships that would program a design activation belief network. Dong [1996] reports initial success for learning belief networks representing design concepts within a restricted corpus. Significant is the creation of a contextual representation of design concerns and objects that is more realistically 'tangled' than the hierarchies 'programmed' into the CDIS. There are drawbacks to unsupervised learning with no background knowledge, it could take a very long time for it to learn what it is it *should* be learning – it could just learn that everything is related. If the current design context hierarchies are used as 'seed' contexts (i.e., background knowledge), the process of learning design context relationships might be greatly simplified.

Another approach, also network-based, is the semantic network. In this, relationships among design contexts can be distributed through object oriented techniques like inheritance. For instance, a context that is excited by a query could through inheritance also

excite all of its children. This methodology has been employed through a complex semantic network-based thesaurus in information retrieval with mixed results[Vorhees, 1993]. While it seems to enhance recall without greatly reducing precision (one of the main goals of all information retrieval systems), it is difficult to control abstraction level because of difficulty in limiting the number of deductions made in the network.The creation of a coherent semantic network thesaurus is also an extremely laborious task, far more difficult than our simple context hierarchies. Still, the results are relatively compelling that if done correctly, this method can improve information retrieval. An additional factor is that intermediate representations like ontologies [Gruber, 1993] go a long way toward creating this semantic network. If implemented as an addition to the CDIS they could form the basis for improved indexing effectiveness.

8.4 Conclusions

The Conceptual Design Information Server has been presented as a collection of subsystems, each based on a specific type of design knowledge, which is unified by a common indexing method based on information science methods. The server transforms the typical reasoning done in case-based reasoning systems from that limited to domains using symbolic AI formalisms to reasoning that is more typical of conceptual design. Open, context rich representations are used to encode design in a manner that does not diminish any specific aspect of the process for the expediency of reasoning by machine. This is done in the spirit of providing contextually apt experiential knowledge to the solution of new design problems – the underlying goals of case-based reasoning.

The implementation of the CDIS is future compatible, both in terms of the openness of its implementation and the richness of the information it strives to represent. The multimedia representation scheme provides the contextual richness that provides the main impetus for retrieving experience and adapting it to new situations. This experience must not be limited to factual representations of artifacts but capture the whole of the design process. The CDIS does this. As new methods become available for distilling information from the

hypermedia document base, the CDIS can further its support for the entire machine based retrieve-analize-patch cycle model of case-based design. Machine-based hypermedia retrieval followed by navigation, collection, and adaptation of the retrieved documents by the designer is an important first step toward expanding the horizon of case-based design from limited domains to full multidisciplinary concurrent engineering conceptual design.

Bibliography

Agogino, A.M. and Evans, J., 1993, "Multimedia Case Studies of Design in Industry" Presented at ASME Design Theory and Methodology `93 Conference. (http://pawn.berkeley.edu/~best/abstracts.working.html#9207020)

Agogino, A.M., Hsi, S., 1993a, "Use of Multimedia technology in Teaching Engineering Design", *Proceedings of HCI International '93*, 5th International Conference on Human-Computer Interaction jointly with 9th Symposium on Human Interface, Orlando Florida, pp. 778-783.

Agogino, A.M., Hsi, S., 1993b, "Navigational Issues in Multimedia Case Studies of Engineering Design", *Proceedings of HCI International '93*, 5th International Conference on Human-Computer Interaction jointly with 9th Symposium on Human Interface, Orlando Florida, pp. 764-769.

Agogino, A.M. , Tseng, M.L., and Jain, P., 1992, "Integrating Neural Networks with Influence Diagrams for Power Plant Monitoring and Diagnostics", In Neural Network Computing for the Electric Power Industry: Proceedings of the 1992 INNS Workshop (International Neural Network Society), Lawrence Erlbaum Associates, Publishers, Hillsdale, NJ, pp. 213-216.

Aldrich, K. S., and Stauffer, L. A., 1995, "A Representation for Design Information During the Product Definition Process", *COncurrent Engineering: Research and Applications*, Vol. 3, No. 2, pp. 107-111.

Bahler, D., Dupont, C., and Bowen, J., 1994, "An Axiomatic Approach that Supports Negotiated Resolution of Design Conflicts in Concurrent Engineering", in Gero, J., Sudweeks, F. (eds)*Artificial Intelligence in Design '94*I, Kluwer Academic Publishers,

pp. 363-79.

Balachandran, M., and Gero, J., 1984, "A Comparison of Three Methods for Generating the Pareto Optimal Set," Engineering Optimization, Vol. 7, pp. 319-336.

Bascaran, E., Bannerot, R.B., and Mistree, F., 1989, "Hierarchical Selection Decision Support Problems in COnceptual Design", *Engineering Optimization*, Vol. 14, pp. 207-238.

Baya, V., Gevins, J., Baudin, C., Mabogunje, A., Toye, G. and L. Leifer, 1992, "An Experimental Study of Design Information Reuse" , *Proceedings of Design Theory and Methodology - DTM '94*, DE-Vol. 68, pp. 141-147, ASME.

Baya, V., and L. Leifer, 1994, "A Study of the Information Handling Behavior of Designers During Conceptual Design", *Proceedings of Design Theory and Methodology - DTM '94*, DE-Vol. 68, pp. 153-160, ASME.

Berners-Lee, T., Cailliau, R., Luotonen, A., Frystyk Nielsen, H., et al., 1994, "The World-Wide Web", *Communications of the ACM*, Aug. 1994, Vol.37, No.8, pp. 76-82.

Bhatta, S., and Goel, A., 1994, "Discovery of Physical Principles from Design Experiences", *Artificial INtelligence for Engineering Design, Analysis, and Manufacturing*, Vol. 8, No. 2, pp. 113-124, Cambridge University Press.

Bollmann-Sdorra, P., and Raghavan, V., 1993, "On the Delusiveness of Adopting a Common Space for Modeling IR Objects: Are Queries Documents?", *Journal of the American Society for Informatoin Science*, Vol. 44, No. 10, pp. 383-392.

Boothroyd, G., and Dewhurst, P., 1988, "Product Design for Manufacture and Assembly," *Manufacturing Engineering*, April, pp. 42-46.

Boothroyd, G., Dewhurst, P., 1989, "Design for Assembly: Selecting the Right Method", "Design for Assembly: Manual Assembly", "Design for Assembly: Robots", in *Design for Assembly Handbook*, Boothroyd and Dewhurst Inc.

Boothroyd, G., Dewhurst, P., 1990, "Product Design Decisions Anticipate Robotic Assembly", *Robotics World*, Jan.-Feb. 1990, Vol.8, No.1, pp. 21-3.

Boothroyd, G., Dewhurst, P., and Knight, W.A., 1991, "Research Program on the Selection of Materials and Processes for Component Parts", *International Journal of Advanced Manufacturing Technology*. Vol.6,No.2, pp. 98-111.

Boyce, B.R, Meadow, C.T., and Kraft, D.H., 1994, *Measurement in Information Science,* Academic Press, San Diego, CA.

Bradley, 1993, *Design Optimization under Resource Constraints*, PhD Dissertation, University of California at Berkeley.

Bradley, S., and Agogino, A. M., 1991a, "Intelligent Real Time Design: Application to Prototype Selection", *Artificial Intelligence in Design '91*, (ed. J. Gero), Butterworth-Heinemann Publishers, Oxford, pp. 815-837.

Bradley, S. and Agogino, A. M., 1991b, "An Intelligent Real Time Design Methodology for Catalog Selection," *ASME'91 Design Theory and Methods*, ASME DE-Vol. 31, pp. 201-208.

Bradley, S. and Agogino, A. M., 1991c, "Design Capture and Information Management

for Concurrent Engineering", *The International Journal of Systems Automation: Research and Applications,* Vol. 1, No. 2, pp. 117-141.

Bradley, S., and Agogino, A., 1993, "Catalog Selection with Multiple Objectives," *Design Theory and Methodology - DTM '93,* ASME DE-Vol. 53, pp. 139-147.

Bradley, S., Agogino, A. M., and W.H. Wood,1994, "Intelligent Engineering Component Catalogs", *Artificial Intelligence in Design '94,* (ed. J. Gero), Cambridge University Press.

Broglio, J., Callan, J. P., Croft, W. B., and Nachbar, D. W., 1994, "DOcument Retrieval and Routing Using the INQUERY System", *Text REtrieval Conference : Overview of the third Text REtrieval Conference (TREC-3),* D.K. Harman (ed.), N ational Institute of Standards and Technology, Gaithersburg, MD.

Buckley, C., Salton, G., Allan, J., and Singhal, A., 1995, "Automatic Query Expansion Using SMART: TREC-3", *Text REtrieval Conference : Overview of the third Text REtrieval Conference (TREC-3),* D.K. Harman (ed.), N ational Institute of Standards and Technology, Gaithersburg, MD.

Buckley, M., K. Fertig and D. Smith, 1992, "Design Sheet: An Environment Facilitating Flexible Trade Studies During Conceptual Design," AIAA 92-1191 (1992 Aerospace Design Conference), American Institute for Aeronautics and Astronautics, Washington D.C.

Callan, J., and Croft, W.B., 1993, "An Evaluation of Query Processing Strategies Using the TIPSTER Collection", *SIGIR '93 : proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* ed. Robert Korfhage, pp. 347-355, ACM Press, Baltimore.

Carbonell, J., 1981, "A Computational Model of Problem Solving by Analogy", *Proceedings of the Seventh International Joint COnference on Artificail Intelligence*, IJCA Inc. / Morgan-Kaufman, San Mateo, CA.

Carbonell, J., 1983, "Derivational Analogy and its Role in Problem Solving", *AAAI-83: Proceedings of the Second National Conference on Artificical Intelligence*, AAAI, Menlo Park, CA.

Carlstrom, C. M., 1993, "Development, Testing and Assessment of the Cyclone Grinder Multimedia Case Study," MS Project Report, Department of Mechanical Engineering, University of California at Berkeley 94720.

Carrol, J., Alpert, S., Karat, J., Van Deusen, M., and Rosson, M.B., 1994, "Raison d'Etre: Capturing Design History and Rationale in Multimedia Narratives", *Human Factors in Computing Systems: CHI '94*, pp. 192-197, Boston.

Chan, L. M., 1989, "Inter-Indexer Consistency in Subject Cataloging", *Information Technology and Libraries*, Vol. 8, No. 4, pp. 349-358.

Chaplin, R.V., Li, M., Oh, V.K., Sharpe, J.E.E., and Yan, X.T., 1994, "Integrated Computer Support for Interdisciplinary System Design", in Gero, J., Sudweeks, F. (eds)*Artificial Intelligence in Design '94*I, Kluwer Academic Publishers, pp. 591-608.

Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., and D. Freeman, 1988, "AutoClass: A Bayesian Classification System", MLC-88.

CNIDR (Center for Networked Information Discovery and Retrieval), 1995, "freeWAIS 0.3 Release", (http://www.cnidr.org).

Conklin, J., and Begeman, M., 1988, "gIBIS: A Hypertext Tool for Exploratory Policy Discussion", *Proceedings of the COnference on Cumputer-supported Cooperative Work*, ACM, New York, pp. 140-152.

Cooper, W.S., 1968, "Inter-indexer Consistency in a Hobgoblin?", *American Documentation*, Vol. 20, pp. 268-278.

Cooper, W. S., 1994, "The Formalism of Probability Theory in IR: A Foundation or an Encumbrance?", *SIGIR '94 : proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 242-247, Springer-Verlag, New York.

Cooper, W.S., and Chen, A., 1994, "Experiements in the Probabilistic Retrieval of Full Text Documents", *Text REtrieval Conference : Overview of the third Text REtrieval Conference (TREC-3),* D.K. Harman (ed.), N ational Institute of Standards and Technology, Gaithersburg, MD.

Dasgupta, S., 1994 , 1990, "Testing the Hypothesis Law of Design: The Case of the Britannia Bridge". *Research in Engineering Design*, Vol. 4, No. 1, pp. 39-57, Springer-Verlag, New York.

de Kleer, J., 1993, "A View on Qualitative Physics", *Artificial Intelligence*, Vol.59, No.1-2, pp. 105-14.

Dewhurst, P., and G. Boothroyd, 1987, "Design for Assembly in Action", *Assembly Engineering*, January.

Dieter, G.D., 1991, *Engineering Design: A Materials and Processing Approach*,

McGraw-Hill.

Dixon, J.R., Orelup, M.F., Welch, R.V., 1992, "Computer-Based Models and Representations for Mechanical Design: A Research Progress Report", *Proceedings of the 1992 NSF Design and Manufacturing Systems Conference*, Atlanta Georgia, January 8 - 10, Society of Manufacturing Engineers, Dearborn Michigan.

Dixon, J.R., Orelup, M.F., Welch, R.V., 1993, "A Research Progress Report: Robust Parametric Designs and Conceptual Design Models", *Proceedings of the 1993 NSF Design and Manufacturing Systems Conference*,Charlotte, North Carolina, January 6-8, Society of Manufacturing Engineers, Dearborn Michigan.

Domeshek, E. and Kolodner, J., 1993, "Using the Points of Large Cases", *Artificial Intelligence for Engineering Design, Analysis, and Manufacture*, Vol. 7, No. 2, pp. 87-96.

Domeshek, E.A. and Kolodner, J.L. ,1992, "A Case-Based Design Aid for Architecture", in Gero, J., Sudweeks, F. (eds)*Artificial Intelligence in Design '92*, Kluwer Academic Publishers, pp. 497-516.

Domeshek, E.A., Kolodner, J.L., and Zimring, C.M., 1994, "The Design of a Tool Kit for Case-Based Design Aids", in Gero, J., Sudweeks, F. (eds)*Artificial Intelligence in Design '94*I, Kluwer Academic Publishers, pp. 109-26.

Donaldson, C., 1994, "InQuisiX: An Electronic Catalog for Software Reuse", *SIGIR Forum*, Vol. 28, No. 1, pp. 8-12, Association for Computing Machinery, New York.

Dong, A., and Agogino, A.M., 1996, "Text Analysis for Constructing Design

Representations", to appear in *Artificial Intelligence in Design '96.*

Drabenscott, K.M., (assisted by Burman, C.M., and Weller, M.S), 1995, *Enhancing a New Design for Access to Online Catalogs* , University of Michigan School of Information and Library Science, (ftp://sils.umich.edu/pub/papers/ENHACNE.ps)

Ellis, D., Furner-Hines, J., Willet, P., 1994, "On the Measurement of Inter-Linker Consistency and Retrieval Effectiveness in Hypertext Databases", *SIGIR '94 : proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 51-60, Springer-Verlag, New York.

EI Vocabulary, 1990, Engineering Information Inc., New York.

Fikes, R.E., Hart, P., and Nilsson, N.J., 1972, "Learning and Executing Generalized Robot Plans", *Artificial Intelligence*, VOl. 3, pp. 251-288.

Fikes, R., Gruber, T., Iwasaki, Y., Levy, A., and P. Nayak, 1991, "How Things Work Project Overview"; Knowledge Systems Laboratory Technical Report KSL-91-70; Computer Science Department, Stanford University; November 1991. (ftp://ksl.stanford.edu/pub/KSL_Reports/KSL-91-70.ps)

Forbus, K., Nielsen, P., and Faltings, B., 1991, "Qualitative Spatial reasoning: the COLCK Project", *Artificial Intelligence*, Vol. 51, No. 3.

Gallant, S.I., 1991, "A practical approach for representing context and for performing word sense disambiguation using neural networks", *IJCNN-91-Seattle: International Joint Conference on Neural Networks*, Seattle, WA, USA, 8-14 July 1991, IEEE,

New York, NY, vol.2, pp. 1007-10031.

Garcia, A.C.B., and Howard, H.C., 1992, "Acquiring Design Knowledge Through Design Decision Justification". *Artificial Intelligence for Engineering Design, Analysis, and Manufacture*, Vol. 6, No. 1, pp. 59-71.

Gardiner, D., Riedl, J., and Slagle, J., 1994, "TREC-3: Experience with Conceptual Relations in Information Retrieval", *Text REtrieval Conference : Overview of the third Text REtrieval Conference (TREC-3),* D.K. Harman (ed.), N ational Institute of Standards and Technology, Gaithersburg, MD.

Gauch, S., and Smith, J.B., 1993, "An Expert System for Automatic Query Reformation", *Journal of the American Society for Informatoin Science*, Vol. 44, No. 3, pp. 124-136.

Genesereth, M. R. and Fikes, R. E. (eds), 1992, "Knowledge Interchange Format, Version 3.0 Reference Manual". Computer Science Department, Stanford University, Technical Report Logic-92-1. (http://www-ksl.stanford.edu/knowledge-sharing/papers/README.html#kif)

Gero, J., 1990, "Design Prototypes: A Knowledge Representation Schema for Design", *AI Magazine*, Winter, 26-36, AAAI, Menlo Park, CA.

Goel, V.; Pirolli, P., 1992, "The Structure of Design Problem Spaces", *Cognitive Science*, Vol.16, No.3, pp. 395-429.

Goldman, R., and Charniak, E., 1988, "A Probabilistic ATMS for Plan Recognition", *Proceedings of the Plan-recognition Workshop.*

Gonda, M., K. Fertig and R. Teeter. 1992. "Aerospace Conceptual Vehicle Design Using an Intelligent Design and Analysis Environment: Design Sheet", AIAA Aircraft Design Systems Meeting, Hilton Head, SC.

Gross, M., Zimring, C., Do, E., 1994, "Using DIagrams to Access a Case Base of Architectural Designs", in Gero, J., Sudweeks, F. (eds)*Artificial Intelligence in Design '94*I, Kluwer Academic Publishers, pp. 129-144.

Gruber, T. R., 1993, "A Translation Approach to Portable Ontology Specifications" *Knowledge Acquisition*, Vol. 5, No. 2, pp. 199-200. (http://www-ksl.stanford.edu/knowledge-sharing/papers/README.html)

Harman, D., 1995, "Overview of the Third Text Retrieval Conference (TREC-3)", , *Text REtrieval Conference : Overview of the third Text REtrieval Conference (TREC-3),* D.K. Harman (ed.), N ational Institute of Standards and Technology, Gaithersburg, MD.

Hauge, P.L, and Stauffer, L.A., 1993, "ELK: A Method for Eliciting Knowledge from Customers", *Proceedings of Design Theory and Methodology - DTM '93*, DE-Vol. 53, ASME, pp. 73-81.

Hauser, J., and D. Clausing, 1988, "The House of Quality," *Harvard Business Review*, vol. 66, no. 3, pp. 63-73.

Hennesy, D., and Hinkle, D., 1992, "Applying Case-Based Reasoning to Autocalve Loading", *IEEE Expert*, October 1992, pp. 21-26.

Hinrichs, T.R., Bareiss, R., and Slator, B.M., 1993, "Representation Issues in Multimedia Case Retrieval", *Case-Based Reasoning*, Technical Report WS-93-01,

AAAI Press, Menlo Park, CA.

Hirose, I., Cannon, D., and L. Leifer, 1994, "Development of a Prototype Design Process Recorder Based on Hypergraphs", *Proceedings of Design Theory and Methodology - DTM '94*, DE-Vol. 68, pp. 259-271, ASME.

Hoadley, C. M., Hsi, S., and Berman, B. P., 1995a, "Networked Multimedia for Communication and Collaboration", Paper presented at the Annual Meeting of the American Educational Research Association, San Francisco, California. (http://www.kie.berkeley.edu/kiosk/hoadley-hsi95.html)

Hoadley, C. M., Hsi, S., and Berman, B. P., 1995b, "The Multimedia Forum Kiosk and SpeakEasy", In Zellweger, P. (Ed.) *Proceedings ACM Multimedia '95* pp. 363-364, ACM Press, New York.

Hoch, Rainer, 1994, "Using IR Techniques for Text Classificaqtion in Documetn Analysis", *SIGIR '94 : proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 31-40, Springer-Verlag, New York.

Hong, J, Toye G., Leifer, L. 1994, "Using the WWW for a Team-Based Engineering Design Class", *Electronic Proceedings of the 2nd WWW Conference*, Chicago, IL, October, 1994. (http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/Educ/hong/hong.html)

Hoover, S., and Rinderle, J., 1994, "Abstractions, Design Views and Focusing", *Proceedings of Design Theory and Methodology - DTM '94*, DE-Vol. 68, pp. 115-129, ASME.

Howard, H.C., Wang, J., Daube, F., and Rafiq, T., 1989, "Applying Design-Dependent Knowledge in Structural Design", *Artificial Intelligence in Engineering Design and Manufacturing*, Vol. 3, No. 2, pp. 111-123.

Howe, A.E., Cohen, P.R., Dixon, J.R., and Simmons, M.K., 1986, "Dominic: A Domain-Independent Program for Mechanical Engineering Design", *The International Journal for Artificial Intelligence in Engineering*, Vol. 1 No. 1, pp. 23-28.

Hua, K., and Faltings, B., 1993, "Exploring Case-Based Building design - CADRE", *Artificial Intelligence for Engineering Design, Analysis, and Manufacture*, Vol. 7, No. 2, pp. 135-143.

Hubka, 1988, *Theory of Technical Systems : A Total Concept Theory for Engineering Design*, translated by W. Ernst Eder, Springer-Verlag, Berlin, New York.

Inland Motor (Kollmorgen Corporation). 1995. *Direct Drive DC Motors*, Inland Motor, 501 First Street, Radford, VA 24141.

Institute for Defense Analysis (IDA), 1988, "Unified Life Cycle Design," Technical Reports.

Ishii, K., Eubanks, C>F., and Marks, M.. 1992, "Evaluation Methodology for Post-Manufacturing Issues in Life-Cycle Design". *Concurrent Engineering: Research and Applications*, Vol. 1, pp. 61-68.

Iwasaki, Y., and A. Y. Levy, 1993, "Automated Model Selection for Simulation", *Proceedings of the Eleventh National Conference on Artificial Intelligence: AAAI'93*, AAAI Press, Menlo Park, CA. (ftp://ksl.stanford.edu/pub/KSL_Reports/KSL-93-11.ps)

Iwasaki, Y. and Low, C. M. , 1991, "Model Generation and Simulation of Device Behavior with
Continuous and Discrete Changes." Technical Report, KSL 91-69, Knowledge Systems Laboratory, Stanford University.

Jamalabad, V.R., Langrana, N.A., 1993. "Extraction and Reuse of Design Information", *Proceedings of Design Theory and Methodology - DTM '93*, DE-Vol. 53, ASME, pp. 129-138.

Kahle, B., Morris, H., Goldman, J., Erickson, T., and Curran, J., 1993, "Interfaces for Distributed Systems of Informatoin Servers", *Journal of the American Society for Informatoin Science*, Vol. 44, No. 8, pp. 453-467.

Kannapan, S., Bell, D., and Taylor, D., 1993. "Structuring Information and Coordinating Teams in Product Development", *Proceedings of Design Theory and Methodology - DTM '93*, DE-Vol. 53, pp. 233-242, ASME.

Kholsa, P. , 1995, "Manufacturing Automation and Design Engineering", http://www.arpa.mil/sisto/symp/Overview/MADE.html

Kolodner, J. L., 1993, *Case-based Reasoning* , Morgan Kaufmann Publishers, San Mateo, California

Kolodner, J.L, 1995, "Archie III", personal communication.

Konda, S., Monarch, I., Sargent, P., and Subrahmanian, E., 1992, "Shared Memory in Design". *Research in Engineering Design*, Vol. 4, No. 1, pp. 23-42, Springer-Verlag, New York.

Krovetz, Robert, 1993, "Viewing Morphology as an Inference Process", *SIGIR '93 : proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ed. Robert Korfhage, pp. 191-202, ACM Press, Baltimore.

Kunz, W. and H. Rittel, "Issues as Elements of Information Systems," Studiengruppe Fuer Systemforschung, Heidelberg, Working Paper #131, 1970. (Also described in D. Grant, "How to Use the IBIS as a Procedure for Deliberation and Argument in Environmental Design Planning," *Design Methods and Theories*, vol. 11, no. 4, pp. 185-220, 1977.)

Kutz, M. (ed), 1986, *Mechanical Engineer's Handbook*, John WIley and Sons, New York.

Lakin, F., Wambaugh, J., Leifer, L., Cannon, D., and C. Sivard, 1989, "The Electronic Design Notebook: Performaing Medium and Processing Medium", *The Visual Computer: International Journal of Computer Graphics*, 5(4).

Lancaster, F.W., Elzy, C., Zeter, M.J., Metzler, L., et al., 1994, Searching Databases on CD-ROM: Comparison of the Results of End-User Searching with Results from Two Modes of Searching by Skilled, Intermediaries, *RQ*, Spring 1994, Vol.33, No.3, pp. 370-86.

Lehto, M.R., Zhu, W., 1992, "Provision of Reference Materials to Designers via Hypertext", *Proceedings of the 1992 NSF Design and Manufacturing Systems Conference*, Atlanta Georgia, January 8 - 10, Society of Manufacturing Engineers, Dearborn Michigan.

Lenat, D.B., 1995, "CYC: A Large-Scale Investment in Knowledge Infrastructure", *Communications of the ACM*, Vol. 38, No. 11, pp. 32-38.

Locascio, A., and Thurston, D., 1994, "Quantifying the House of Quality for Optimal Product Design", *Proceedings of Design Theory and Methodology - DTM '94*, DE-Vol. 68, pp. 43-54, ASME.

Maher, M.L., and Li, H., 1994, "Learning Design Concepts Using Machine Learning Techniques", *Artificial INtelligence for Engineering Design, Analysis, and Manufacturing*, Vol. 8, No. 2, pp. 95-111, Cambridge University Press.

Maher, M.L., and Zhang, D.M., 1991, "CADSYN: Using Case and Decomposition Knowledge for Design Synthesis", *AI in Design '91*, Gero, J.S., ed. Butterworth-Heinemann: Oxford, pp. 137-150.

Maher, M.L., and Zhang, D.M., 1993, "CADSYN: A Case-Based Design Process Model", *Artificial Intelligence for Engineering Design, Analysis, and Manufacture*, Vol. 7, No. 2, pp. 97-110.

Marks, M.D., Eubanks, C.F., and Ishii, K., 1993, "Life-Cycle Clumping of Product Designs for Ownership and Retirement", *Design Theory and Methodology - DTM '93*, ASME DE-Vol. 53, pp. 83-90

McCall, R., 1989, "MIKROPOLIS: A Hypetext System for Design", *Design Studies*, Vol. 10, No. 4, pp. 228-238.

McCall, R., 1991, "PHI: A Conceptual Foundation for Design Hypermedia", *Design Studies*, Vol. 12, No. 1, pp. 30-41.

McCall, R., Ostwald, J., and Shipman, F.. 1991. "Supporting Designers' Access to Information through Virtually Structured Hypertext", *IntCAD '91 Preprints*, IFIP WG 5.2 Working COnference on Intelligent CAD.

McGinnis, B. D. , and Ullman, D. G., 1992, "The Evolution of Commitments in the Design of a Component", *Journal of Mechanical Design*, Vol. 114, pp. 1-7.

Michelena, N. and Agogino, A. M., 1993. "Monotonic Influence Diagrams: Extension to Stochastic Programming and Application to Probabilistic Design", *Engineering OPtimization*, Vol. 21, pp. 99-120, Gordon and Breach Science Publishers, S.A.

Michelena, N., and K. Sycara, 1994a, "Performance Representation in Case-Based Design", *Proceedings of Design Theory and Methodology - DTM '94*, DE-Vol. 68, pp. 285-292, ASME.

Michelena, N., and K. Sycara, 1994b, "Physical Synthesis in Case-Based Design", *Proceedings of Design Theory and Methodology - DTM '94*, DE-Vol. 68, pp. 273-284, ASME.

Miller, G.A., 1995, "WordNet: A Lexical Database for English", *Communications of the ACM*, Vol. 38, No. 11, pp. 39-41.

Minton, S., 1988, "Quantitative Results Concerning the Utility of Explanation-Based Learning", *AAAI-88: Proceedings of the Seventh National Conference on Artificical Intelligence*, AAAI, Menlo Park, CA.

Mitchell, T., 1980, "The Need for Biases in Learning Generalizations", Technical Report CbM-TR-117, Department of Computer Science, Rutgers University, May, 1980.

Mitchell, T., 1975, "Generalization as Search", *Artificial Intelligence*, Vol. 18, No. 2, pp. 203-226.

Murty, M.N., and Jain, A.K., 1995, "Knowledge-Based Clustering Scheme for Collection Management and Retrieval of Library Books", *Pattern Recognition*, Vol. 28, No. 7, pp. 949-63.

Nagy, R., Ullman, D., and Dietterich, T., 1992, "A Data Representation for Collaborative Mechanical Design", *Research in Engineering Design*, Vol. 3, No. 4, pp. 233-242, Springer-Verlag, New York.

National Institute of Standards and Technology (NIST), 1995, *Standards Development Project Summary: Standards Office IGES/PDES/STEP*, (http://elib.cme.nist.gov:80/nipde/projects/std00074.html) (http://elib.cme.nist.gov:80/msid/projs/solis/step.gen.html)

National Research Council, 1991,*Improving Engineering Design: Designing for Competitive Advantage*, National Academy Press.

Navinchandra, D., 1988, "Behavioral Synthesis in CADET, a Case-Based Design Tool", *Proceedings of the DARPA Workshop on Case-based Reasoning*, Kolodner, J., Ed., Morgan-Kaufman, pp. 286-301.

NASA Technical Thesaurus, 1988, United States National Aeronautics and Space Administration, Scientific and Technical Information Division.

Neches, R., 1993, "The Knowledge Sharing Effort", http://www-ksl.stanford.edu/knowledge-sharing/papers/kse-overview.text

Norman, Donald A, 1988, *The Design of Everyday Things* , Doubleday/Currency, New York.

O'Conner, L.J., Lan, M.S., Partridge, D.R., Lee, J.M.F., 1992, "A Case-Based Approach to Automated Weld-Process Design", *Applied Artificial Intelligence*, Vol. 6, pp. 315-330.

Oddy, R. N., Robertson, S. E., van Rijsbergen. C. J., and Williams, P. W., (eds), 1981, *Information Retrieval Research*, Butterworths, London.

Pahl, G. , and W. Beitz, 1984, *Engineering Design* , Springer-Verlag, Berlin.

Paijmans, Hans, 1993, "Comparing the Document Representations of Two IR-Systems: CLARIT and TOPIC", *Journal of the American Society for Informatoin Science*, Vol. 44, No. 7, pp. 383-392.

Paynter, H., 1961, *Analysis and Design of Engineering Systems*, MIT Press, Cambridge, MA.

Pearce, M., Goel, A.K., Kolodner, J.L., Zimring, C., Sentosa, L., and Billington, R., 1992, "Case-Based Design Support, A Case Study in Architectural Design", *IEEE Expert*, October 1992, pp. 14-20.

Petrie, C., Cutcosky, M., and Park, H., 1994, "Design Space Navigation as a Collaborative Aid", in Gero, J., Sudweeks, F. (eds)*Artificial Intelligence in Design '94*I, Kluwer Academic Publishers, pp. 437-45.

Petroski, H., 1985, *To Engineer is Human: The Role of Failure in Succesfule Design* , St. Martin's Press, New York.

Petroski, H., 1994, *Design Paradigms : Case Histories of Error and Judgment in Engineering*, Cambridge University Press, New York.

Pfeifer, U., 1995, "freeWAIS-sf: The Enhanced freeWAIS Distribution", http://charly.informatik.uni-dortmund.de/freeWAIS-sf/

Pitts, G., and P. Vadamuttu ,1987, "Retrieval of Component Information for System Designers", *Proceedings of the International Conference on Engineering Design*, Boston.

Plaunt, C. and Norgard B. A., 1995, "An Association Based Method for Automatic Indexing with a Controlled Vocabulary", to appear in *Journal of the American Society for Information Science.*

Quinlan, J., 1986, "Induction of Decision Trees", *Machine Learning*, Vol. 1, Kluwer, Boston.

Qiu, Y., and Frei, H.P., 1993, "Concept Based Query Expansion" , *SIGIR '93 : proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ed. Robert Korfhage, pp. 191-202, ACM Press, Baltimore.

Raghavan, V. V., and Wong, S. K. M., 1986, "A critical Analysis of Vector Space Model for Information Retrieval", *Journal of the American Society for Information Science*, Vol. 37, No. 5, pp. 279-287.

Rittel, H., "The Reasoning of Designers", unpublished manuscript.

Robertson, S.E., and Walker, S., 1994, "Some Simple Effective Approximations to the 2-Poisson Modelfor Probabilistic Weighted Retrieval", *SIGIR '94 : proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 232-241, Springer-Verlag, New York.

Rosenblat, A., and Watson, G. (eds), 1991, "Concurrent Engineering", *IEEE Spectrum*, July 1991.

Ruff, D.N., and Paasch, R.K., 1993, "COnsideration of Failure Diagnosis in COnceptual Design of Mechanical Systems", *Design Theory and Methodology - DTM '93*, ASME DE-Vol. 53, pp. 175-87.

Russel, S., 1988, "Analogy by Similarity", in *Analogy*, edited by David Helman, Dordrecht.

Salton, G., 1988, *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, Reading, MA.

Salzberg, S., and M. Watkins,1990, "Managing Information for Concurrent Engineering: Challenges and Barriers," *Research on Engineering Design*, Vol. 2, No. 1, pp. 35-52, Springer-Verlag, New York.

Salzberg, S., 1991, " A Nearest Hyperrectangle Learning Method", *Machine Learning*, vol.6, (no.3):251-76.

Scardamalia, M. and Bereiter, C., 1993, "Technologies for Knowledge-Building Discourse", *Communications of the ACM*, Vol.36, No.5, pp. 37-41.

Schaaf, J., 1994, "Gestalts in CAD-plans: Analysis of a Similarity Concept", in Gero, J.,

Sudweeks, F. (eds)*Artificial Intelligence in Design '94*I, Kluwer Academic Publishers, pp. 437-45.

Schuldberg, H. K., Macpherson, M., Humphrey, P., and Corley, J., 1993, "Distilling Information from Text: The EDS *TemplateFiller* System", *Journal of the American Society for Informatoin Science*, Vol. 44, No. 9, pp. 493-507.

Schutze, H., 1992, "Context Space", *Probabilistic Approaches to Natural Language*, Technical Report FS-92-05, AAAI Press, Menlo Park, CA, pp. 113-120.

Schutze, H., and Pedersen, J. O., 1994, "A Cooccurrence Based Thesaurus and Two Applications to Information Retrieval", ISTL Technical Report No. ISTL-QCA-1994-03-02, Xerox PARC, Palo Alto, CA.

Seddon, A., and Brereton, P., 1994, "Component Selection Using Temporal Truth Maintenance", *Artificial INtelligence for Engineering Design, Analysis, and Manufacturing*, Vol. 8, No. 1, pp. 13-25, Cambridge University Press.

Segre, A., 1987, "On the Operationality / Generality Tradeoff in Explanation-Based Learning", *Proceedings of the Tenth International Joint onference on Artificial Intelligence: IJCAI '87*, IJCAI, Los Altos, CA.

Shah, J., Bliznakov, P., and Urban, S., 1993, "Development of a Machine Understandable Language for Design Process Representation", *Proceedings of Design Theory and Methodology - DTM '93*, DE-Vol. 53, pp. 15-24, ASME.

Shaw, N., Bloor, S., and de Pennington, A., 1990, "Product Data Models". *Research in Engineering Design*, Vol. 2, No. 1, pp. 43-50, Springer-Verlag, New York.

Silva, M.J.; Katz, R.H., 1993, "Active Documentation: A New Interface for VLSI Design", IN: *Proceedings of the 30th Design Automation Conference* (IEEE Cat. No.93CH3262-3), ACM, pp. 654-60.

Silva, M.J.; Katz, R.H., 1995, "The Case for Design Using the World Wide Web". IN: *Proceedings of the 32nd Design Automation Conference* (IEEE Cat. No.95CH35812), ACM, pp. 579-85.

Slade, S., 1991, "Case-Based Reasoning: A Research Paradigm", *AI Magazine*, Vol.12, No.1, pp. 42-55.

Smoliar, S., and Zhang, H., 1994, "Content-Based Video Indexing and Retrieval", *IEEE Multimedia*, Summer 1994, pp. 62-72, IEEE.Stauffer, L, D. Ullman and T. Dietterich, 1987, "Protocol Analysis of Mechanical Engineering Design," in *Proceedings of 1987 International Conference on Engineering Design (ICED)*, August, pp. 68-73.

Specht, D.F., 1988, "Probabilistic Neural Networks for Classification, Mapping, or Associative Memory", *IEEE International Conf. on Neural Networks*, San Diego, CA, July 24-27, 1988, pp. 525-532.

Stauffer, L. and D. G. Ullman, 1988, "A Comparison of the Results of Empirical Studies into the Mechanical Design Process," *Design Studies*, vol. 9, no. 2, pp. 107-114.

Stevenson, S., Dooley, K., and Anderson, J., 1994, "The Use of Best Design Practices: AN Analysis of US Navy Contractors". *Research in Engineering Design*, Vol. 6, No. 1, pp. 14-24, Springer-Verlag, New York.

Strzalkowski, T., Carballo, J., and Marinescu, M., 1994, "Natrual Language Information Retrieval: TREC-3 Report", *Text REtrieval Conference : Overview of the third Text*

REtrieval Conference (TREC-3), D.K. Harman (ed.), N ational Institute of Standards and Technology, Gaithersburg, MD.

Suh, N., Suh, 1990, *The Principles of Design*, Oxford University Press, New York.

Tenenbaum, J.M., Medich, C., Schiffman, A.M., Wong, W.T., 1995, "CommerceNet: Spontaneous Electronic Commerce on the Internet", in *Digest of Papers, COMPCON '95: Technologies for the Information Superhighway* , IEEE Comput. Soc. Press (Cat. No.95CH35737), pp. 38-43.

Tesauro, G., and Sejnowski, T., 1989, "A Parallel Network That Learns to Play Backgammon", *Artificial Intelligence*, Vol. 39, No. 3, pp. 357-390.

Tomiyama, T., 1995, "A Design Process Model that Unifies General Design Theiry and Empirical Findings", *Design Theory and Methodology - DTM '95*, DE-Vol. 83, Volumne 2, pp. 329 - 340, ASME.

Toye, G., Cutkosky, M., Leifer, L.J., Tenenbaum, J.M., and Glicksman, J., 1993, "SHARE: A Methodology and Environment for Collaborative Product Development", to appear in *Post-Proceedings of the Second Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, IEEE.

Tseng. M.-L., 1991, "Integrating Neural Networks and Influence Diagrams for multiple sensor diagnostic systems," PhD thesis, Department of Mechanical Engineering, University of California at Berkeley.

Turns, J., Mistree, F., Rosen, D., Allen, J.K., et al., 1995, "A Collaborative Multimedia Design Learning Simulator", *Educational Multimedia and Hypermedia: Proceedings of ED-MEDIA 95*, Edited by: Maurer, H., Assoc. Advancement of Comput. Educ,

pp. 654-9.

Ullman, D.G., 1992, "The Evolution of Committments in the Design of a Component", Journal of Mechanical Design, March 1992, Vol 114, p 1-7.

Ullman, D.G., 1993, "The Evolution of Function and Behavior During Mechanical Design", *Proceedings of Design Theory and Methodology - DTM '93*, DE-Vol. 53, pp. 91-103, ASME.

Ullman, D.G., 1994, "Issues Critical to the Development of Design History, Design Rationale, and Design Intent Systems", *Proceedings of Design Theory and Methodology - DTM '94*, DE-Vol. 68, pp. 249-258, ASME.

Ulrich, K., and Pearson, S., 1993, "Does product design really determine 80% of manufacturing cost?", Working Paper WP 3601-93, Sloan School of Management, MIT, Cambridge, MA.

Ulrich, K., 1993 - adding egr models to HoQ.

Ulrich, K. and Seering, W., 1990, "Synthesis of Schematic Descriptions in Mechanical Design", *Research in Engineering Design*, Vol. 2, No. 1, pp. 3-17, Springer-Verlag, New York.

Ulrich, K., and Eppinger, S., 1995, *Product Design and Development* , McGraw-Hill, New York.

Valauskas, E.J., 1995, "Britannica Online: Redefining Encyclopedia for the Next Century, *Database*, Feb.-March 1995, Vol.18, No.1, pp. 14-18, 20.

Vogwell, J., and S. Culley, 1987, "Optimal Component Selection Using Engineering Databases," Proceedings of the International Conference on Engineering Design, Boston, MA, Aug. 17-20, 1987.

Voorhees, Ellen, 1994, "Query Expansion using Lexical-Sematic Relations", *SIGIR '94 : proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 61-69, Springer-Verlag, New York.

Voorhees, Ellen, 1993, "Using WordNet to Disambiguate Word Senses for Text Retrieval", *SIGIR '93 : proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ed. Robert Korfhage, pp. 171-180, ACM Press, Baltimore.

Vora, P., Helander, M., and Shalin, V., 1994. "Evaluating the Influence of Interface Styles and Multiple Access Paths in Hypertext", *Human Factors in Computing Systems: CHI '94*, pp. 323-329, Boston.

Voss, A., Coulon, C.-H., Grather, B., Linowski, B., Schaaf, J., Bartsch-Sporl, B., Borner, K., Tammer, E.C>, Durschke, H., and Knauff, M., 1994, "Retrieval of SImilar Layouts – About a Very Hybrid Approach in FABEL", in Gero, J., Sudweeks, F. (eds)*Artificial Intelligence in Design '94*I, Kluwer Academic Publishers, pp. 625-40.

Wang, J., and Howard, H.C., 1994, "Recording and Reuse of Design Strategies in an Integrated Case-based Design System", *Artificial INtelligence for Engineering Design, Analysis, and Manufacturing*, Vol. 8, No. 1, pp. 13-25, Cambridge University Press.

Ward, A., Liker, J., Sobek, D., and J. Cristiano, "Set-Based COncurrent Engineering and

Toyota", *Proceedings of Design Theory and Methodology - DTM '94*, DE-Vol. 68, pp. 79-90, ASME.

Wilensky, Robert, 1992, "Discourse versus Probability in the Theory of Natural Language Interpretation", *Probabilistic Approaches to Natural Language*, Technical Report FS-92-05, AAAI Press, Menlo Park, CA, pp. 128-135.

Will, P., 1991, "Simulation and Modeling in Early Concept Design: An Industrial Perspective". *Research in Engineering Design*, Vol. 3, No. 1, pp. 1-13, Springer-Verlag, New York.

Winston, P.H., 1977, Learning Simple descriptions", in *Artificial Intelligence*, Addison Wesley, Reading, MA.

Wood, W.H. and Agogino, A.M., 1994, "A Case-Based Conceptual Design Information Server for Concurrent Engineering,", to appear in *Computer Aided Design:CAD*, IPC Science and Technology Press, Guildford, Surrey, England.

Wood, W.H., and Agogino, A.M., 1996, "Engineering Courseware Content and Delivery: the NEEDS Infrastructure for Distance-Independent Education", to appear in *Journal of the American Association for Information Science.*

Zhao, F. and M.L. Maher, 1988, "Using Analogical Reasoning to Design Buildings", *Engineering with Computers*, Vol. 4, pp. 107 - 122.

Zook, I., 1994, "Development, Testing and Assessment of the Saturn Automobiles Multimedia Case Study," MS Project Report. Department of Mechanical Engineering, University of California at Berkeley 94720.

Appendix A

Performance of the CDIS Abstraction Modeling Method

Chapter 6 introduces the various methods that have been explored in this research for applying the IRTD method to the problem of deciding on the level of abstraction most appropriate to the current design decision. Here we go into further depth, dealing with the effects that changing the way that engineering models are formed and used have on the modeling and decision process. Specifically, these include: the application of constraint, the sensitivity of modeling and decision making to the smoothing parameter **c**, and the application of SOPNN for reducing computational complexity.

A.1 Application of Constraint

The modeling methods used here are both based on approximating a design space by adding Gaussian 'noise' to points which represent known catalog data. There are two methods used to apply constraints to the system models. The first is to include as a subset of the design space vector a constraint vector consisting of all numerical values that have been assigned in the design. Thus, the calculation of the 'distance' from any point in the unconstrained design space includes distances from the assigned constraints. Equation 6.4.1.1 is reformulated to calculate a conditional probability distribution in Equation A.1.1:

$$P(\mathbf{x}|\mathbf{y}) = \frac{1}{(2\pi)^{\frac{d}{2}}\sigma^d}\frac{1}{n}\sum_{i=1}^{n} e^{-\frac{(\mathbf{z}-\mathbf{z}_i)(\mathbf{z}-\mathbf{z}_i)^T}{2\sigma^2}} \qquad \text{(A.1.1)}$$

where:

$\mathbf{x}$ is the design space vector whose probability is being assessed
$\mathbf{y}$ is the constraint vector
$\mathbf{z} = [\mathbf{x}, \mathbf{y}]$
$\mathbf{z}_i$ is the i^{th} training pattern (i.e., catalog data sample)
n is the total number of training patterns
$d = \dim(\mathbf{z})$
σ is the distribution variance (smoothing parameter)

As before, in a [0,1] normalized design space:

$$\sigma = cn^{-\frac{1}{(d+4)}}$$
$$c = [0.01, 0.35]$$

Here, the joint distribution of Equation 6.4.1.1 becomes a conditional distribution where the probability density of the unconstrained design variables is conditioned on a (static) constraint vector, $\mathbf{y}$.

Two training data sets were used for a set of three experiments: the full catalog of motors and a catalog reduced by the screening query that motor power be in the range [30W,60W]. In the constrained experiments, the motor power was set to 45W. The following table represents the three experiments (in each experiment, the value of $\mathbf{c}$ has been set to 0.01 – more about the selection of $\mathbf{c}$ is found in Section A.2):

	Training Set		Constrained?		Figures
Experiment	Full	30 < P < 60	Yes	No	
1	X		X	X	A.1 - A.5
2	X	X	X		A.6 - A.10
3		X	X	X	A.11 - A.15

A.1.1 Applying Constraint to the Approximate Model

Figure A.1 - A.5 show the differences in modeling and decision making which result from applying a numerical constraint to the unconstrained design space adjoined by the constraint value. The models shown in Figure A.1 (probability distribution contour maps overlaid with the expected value of the objective with respect to the various design parameters) show a general similarity, behaving as one might expect constrained and unconstrained set might. The constrained distributions generally have more closely spaced contour lines, indicating less spread in the distributions. The exception here is the model for torque where the constraint removes a large number of lower torque motors, but does significantly change the spread in the torque distribution. Because torque and speed are both used in the calculation of power, the constraint variable, an increase in the spread of speed signals a requisite increase in the spread of torque.

Figures A.2 - A.5 depict the decision making recommendations of the system. Note that in each case, the top panel shows the expected value of the design variable, overlaid with a (not to scale) representation of the probability distribution of that variable. In all four cases, these distributions do not show marked change – applying a constraint has not significantly changed the range of any of the design variables. What has changed, though, is the expected value of the objective. In response to this change in objective, the lower panels reveal requisite alterations to the recommended design decisions. In A.2, the unconstrained model recommends that for the most likely torque range of the motor the designer first consider frameless motors; the constrained results reveal a strong preference for brushless motors as a first step in constraining motor type. Of course, these are not mutually exclusive decisions, just the ordering of importance has changed. More important is the relationships between expected values. Here, brushless motors of approximately 45W are generally *much* lighter than the general case. This holds (for the most part) throughout all of the design variable breakdowns in the remaining Figures, A.3 - A.5. In A.3, recommendations are reversed: ranging from low to high speed, the unconstrained model suggests frameless and then brushless; over the same range the constrained system reverses

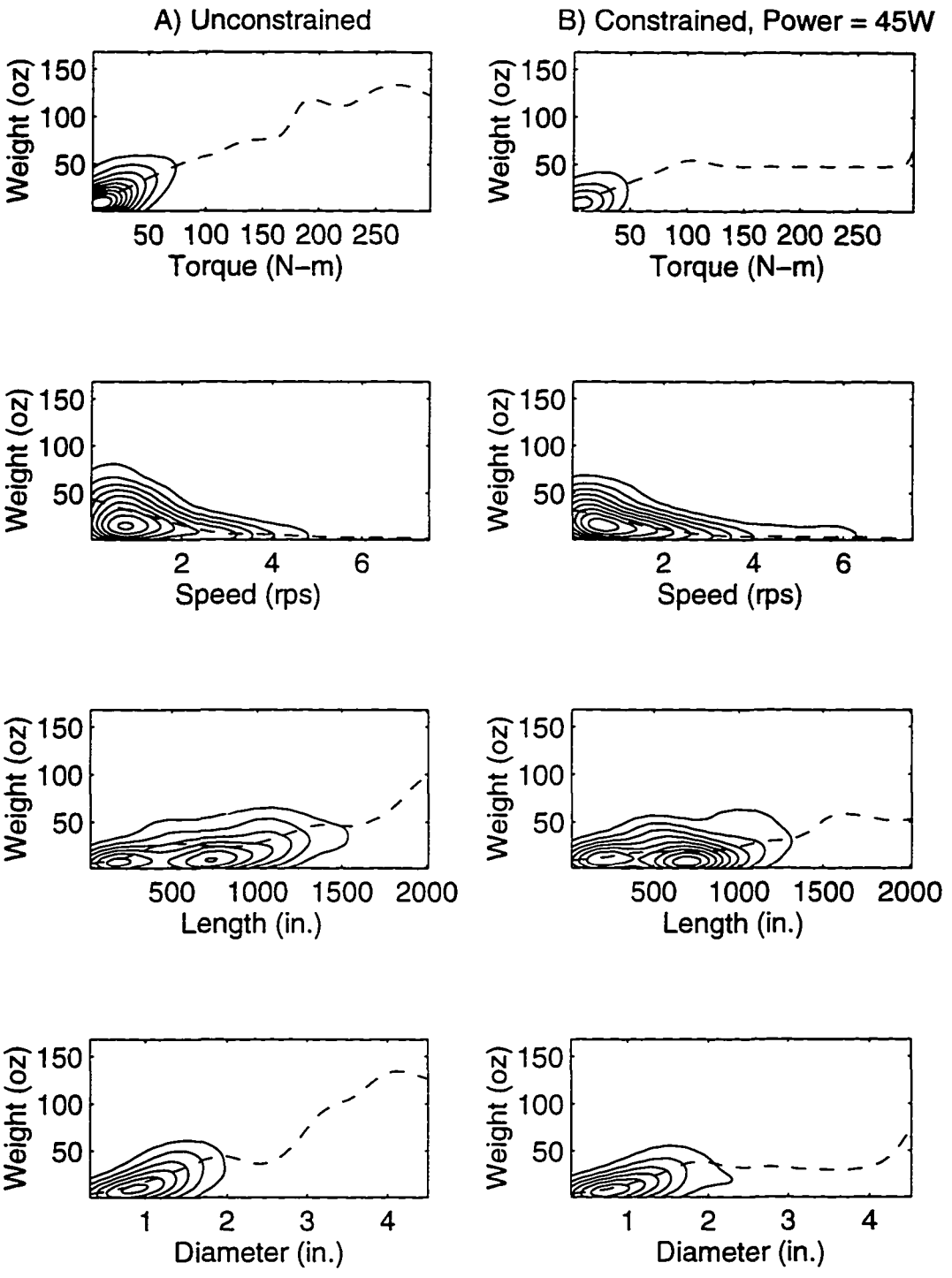


Figure A.1 Full Catalog Engineering Models. Column A is unconstrained, Column B constrained. Expected value of weight given the variable value is overlaid (dashed line). Note that the addition of a constraint causes the distributions to 'tighten' up considerably.

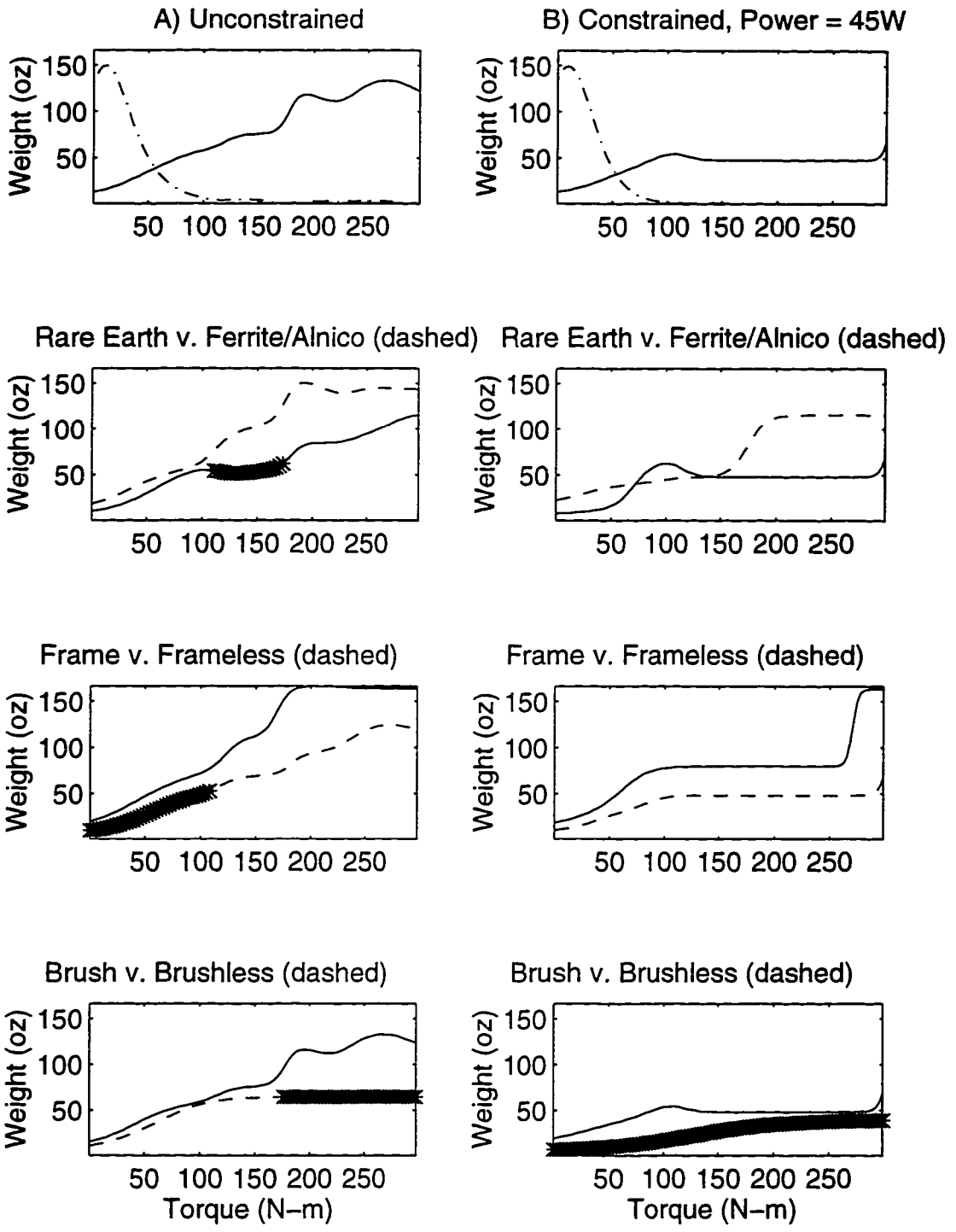


Figure A.2 Full Catalog Decision Making - Torque. Plot shows optimal decision ('*') for decreasing design abstraction based on the value of torque. Column A is unconstrained, Column B constrained.Top panel shows expected value of weight given torque overlaid with the distribution of torque (dot dash). Note how the addition of a constraint changes decision making and expected values.

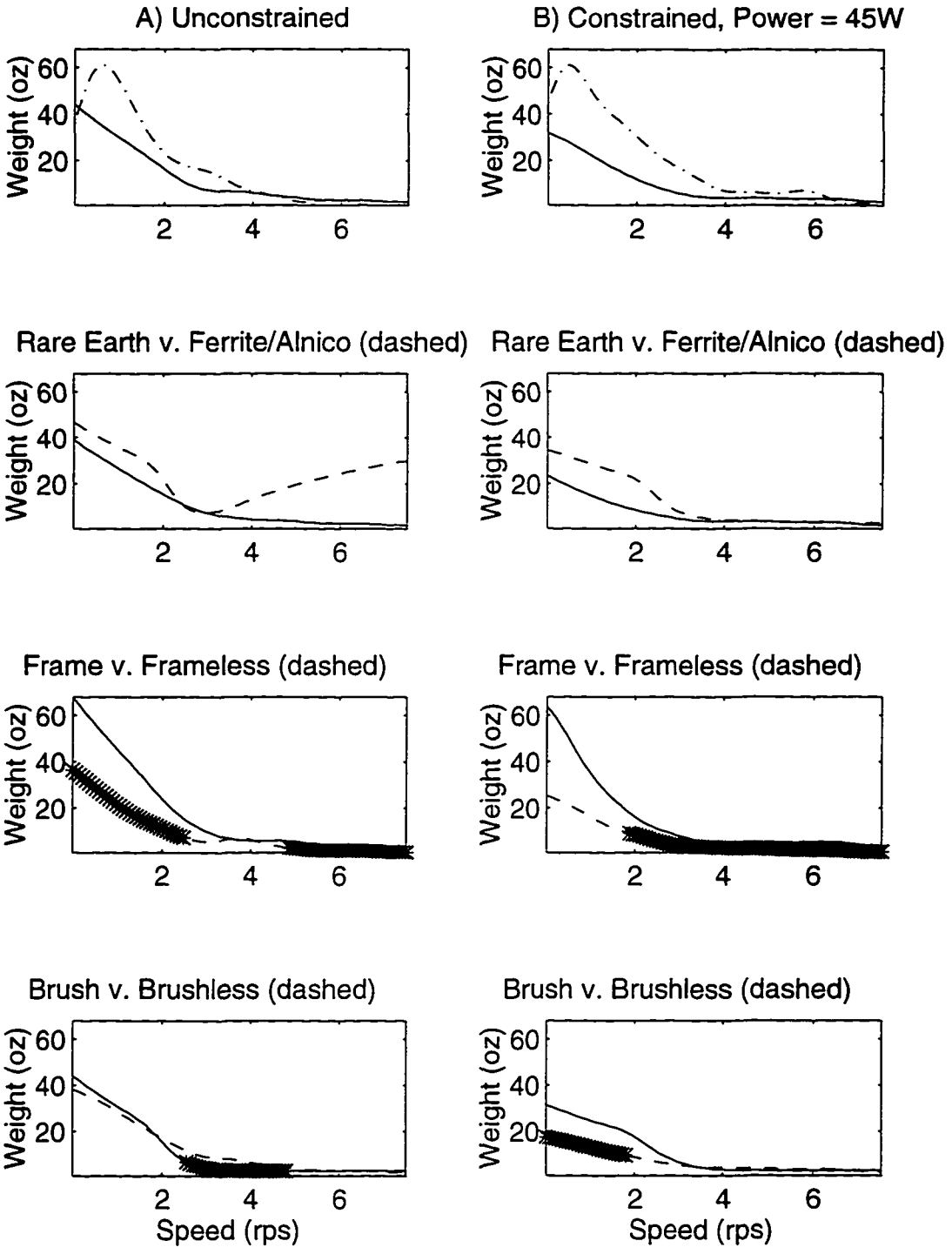


Figure A.3 Full Catalog Decision Making - Speed. Plot shows optimal decision ('*') for decreasing design abstraction based on the value of torque. Column A is unconstrained, Column B constrained.Top panel shows expected value of weight given speed overlaid with the distribution of speed (dot dash). Note how the addition of a constraint changes decision making and expected values.

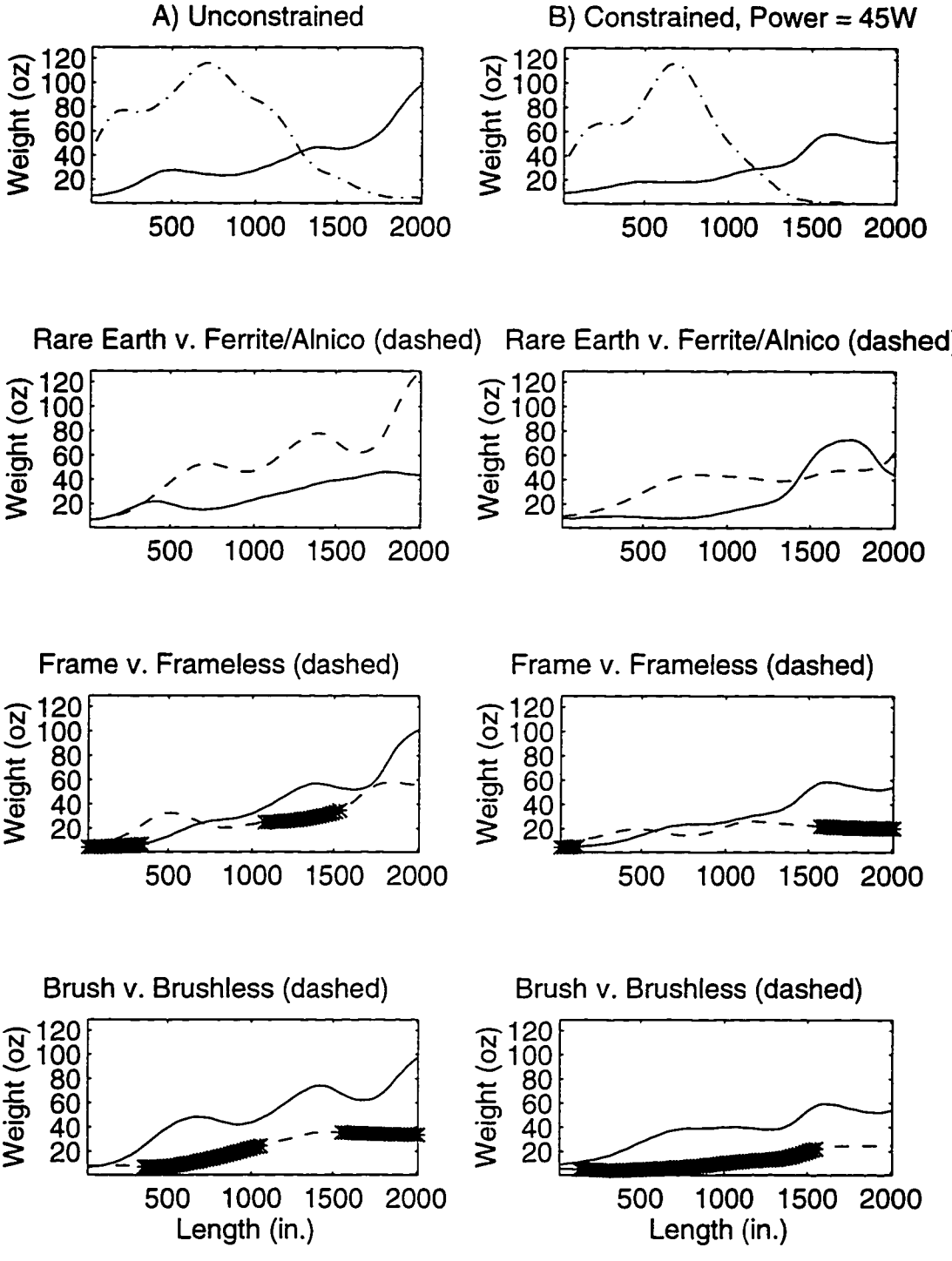


Figure A.4 Full Catalog Decision Making - Length. Plot shows optimal decision ('*') for decreasing design abstraction based on the value of torque. Column A is unconstrained, Column B constrained.Top panel shows expected value of weight given length overlaid with the distribution of length (dot dash). Note how the addition of a constraint changes decision making and expected values.

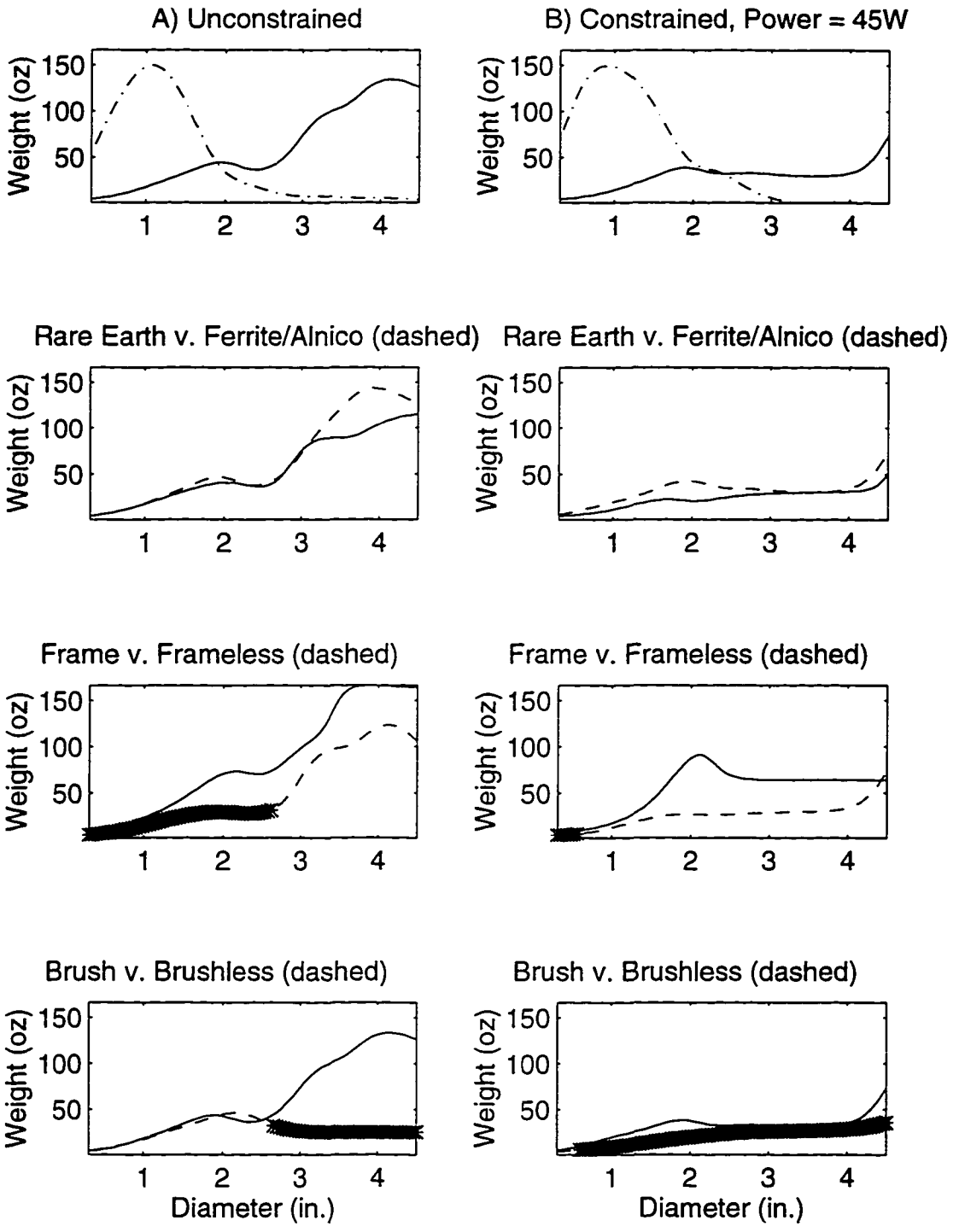


Figure A.5 Full Catalog Decision Making - Diameter. Plot shows optimal decision ("*") for decreasing design abstraction based on the value of diamater. Column A is unconstrained, Column B constrained.Top panel shows expected value of weight given diameter overlaid with the distribution of diameter (dot dash). Note how the addition of a constraint changes decision making and expected values.

the order. The main consistent point throughout is that magnet type has less of an impact on the weight objective than the other two classification variables.

A.1.2 Reducing the Design Space Near a Constraint

In this case, both design models are constrained to have a rated power of about 45W. The variation here is in the size of the sample space that is used in the calculation of the probability densities. For all of the *A* columns of plots in Figures A.6 - A.10, the entire catalog database was used for modeling; the *B* columns are based on results from the subset of motors between 30 and 60W (a loose range centered around the more formal 45W constraint). There are two main effects of this reduction in sampling space:

1. Fewer samples can mean a more uniform spread in normalized data. Regardless of the size of the initial sample set, the design space is normalized to fit in the range [0,1]. Figure A.6 shows models based on the full catalog compared to that generated from the subset (please note the shift in axes from columns *A* to column *B)*. Note that for torque and diameter, this represents much grater spread in the data. This greater spread puts more distance between samples which, in turn, lowers the amount of influence that 'neighbor' samples contribute. Compressing the space along only two axes in the full model produces a non-uniform calculation of 'distance' in the normalized space. Thus, one might expect that the constraint of 45W is less active along the compressed axes simply because most motors are tightly grouped.

2. The lack of contribution from more 'neighbor' samples produces a less smooth design domain. By placing a hard 'edge' on the design space, the filtering query produces much more fragmented distributions. More modes appear in the data of Figure A.6, these are amplified in the top panes of Figures A.7 - A.10 where use of consistent design variable axes highlights the direct correspondence between full and subset modeling. In all but Figure A.8 - A.10, both the expected value and

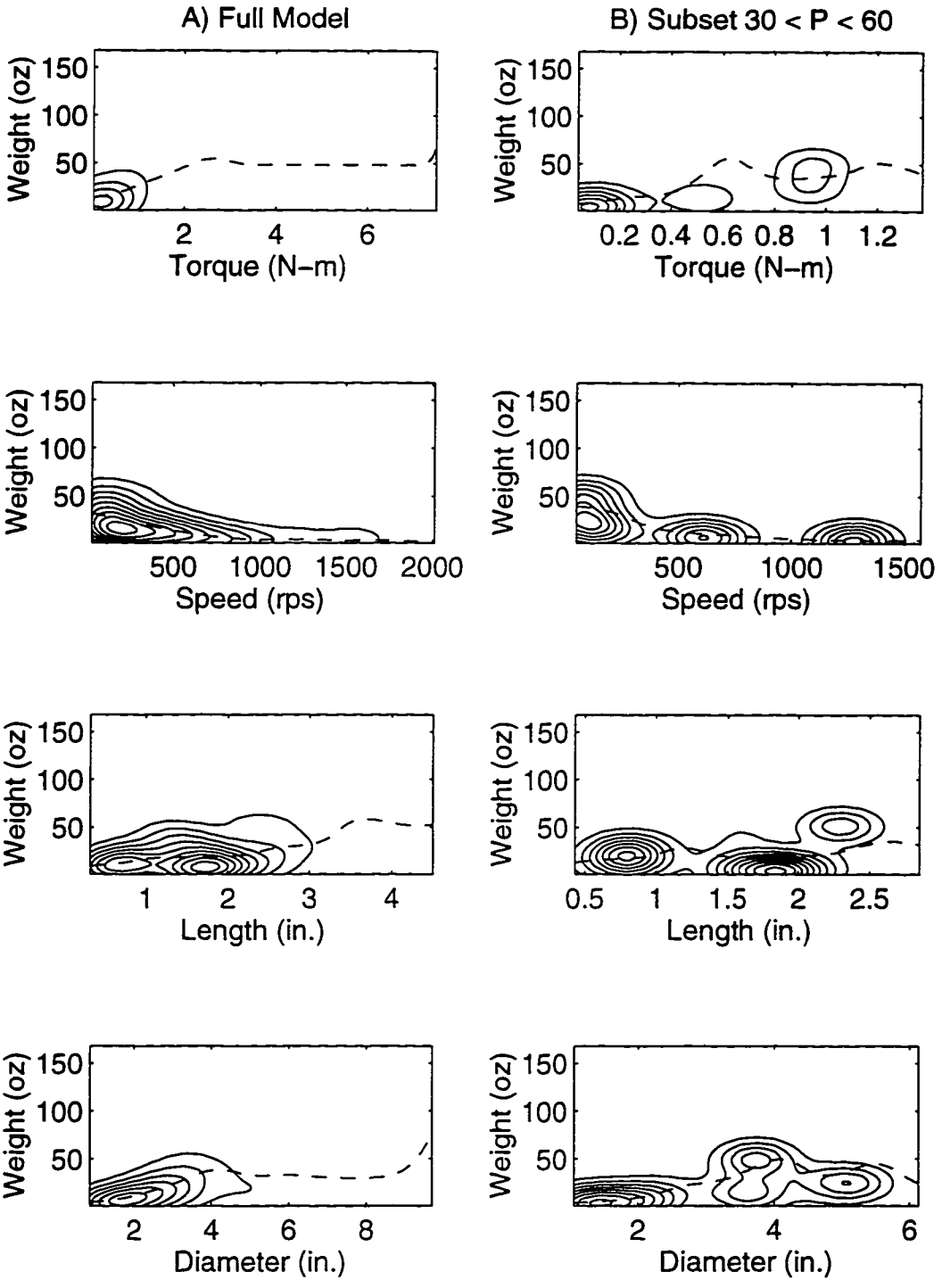


Figure A.6 Full/Subset Engineering Models. Plots show probability density contour maps relating key engineering variables to an objective – weight. Column A is based on the full catalog, Column B on a subset. Expected values are overlaid (dash). Note change in scale from column A to B, showing the range of the subset.

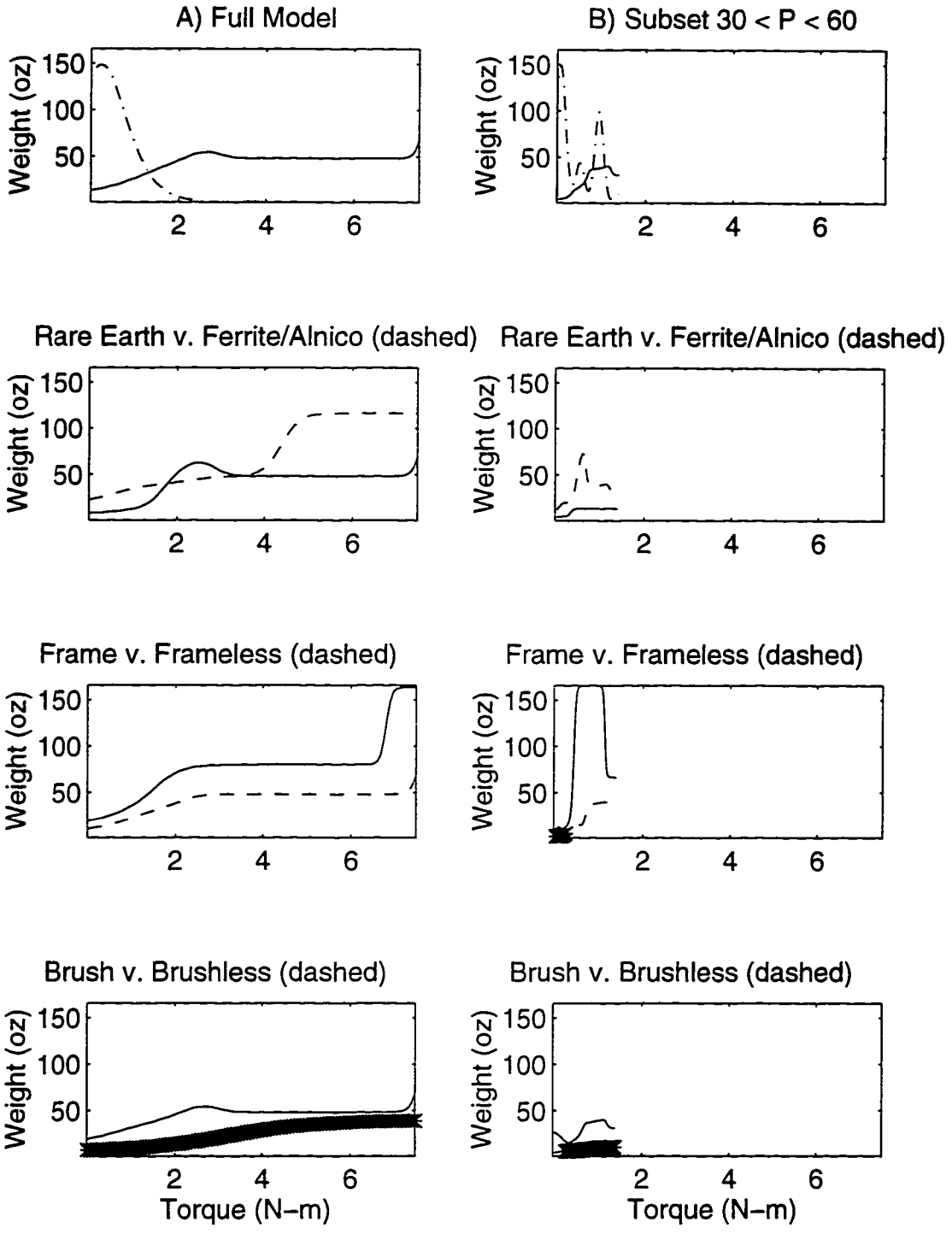


Figure A.7 Full/Subset Decision Making - Torque. Plot shows optimal decision ('*') for decreasing design abstraction based on the value of torque. Column A is the full model, Column B a subset.Top panel shows expected value of weight given torque overlaid with the distribution of torque (dot dash). Note the compression in range of torque values, some over-generalization in Column A is evident.

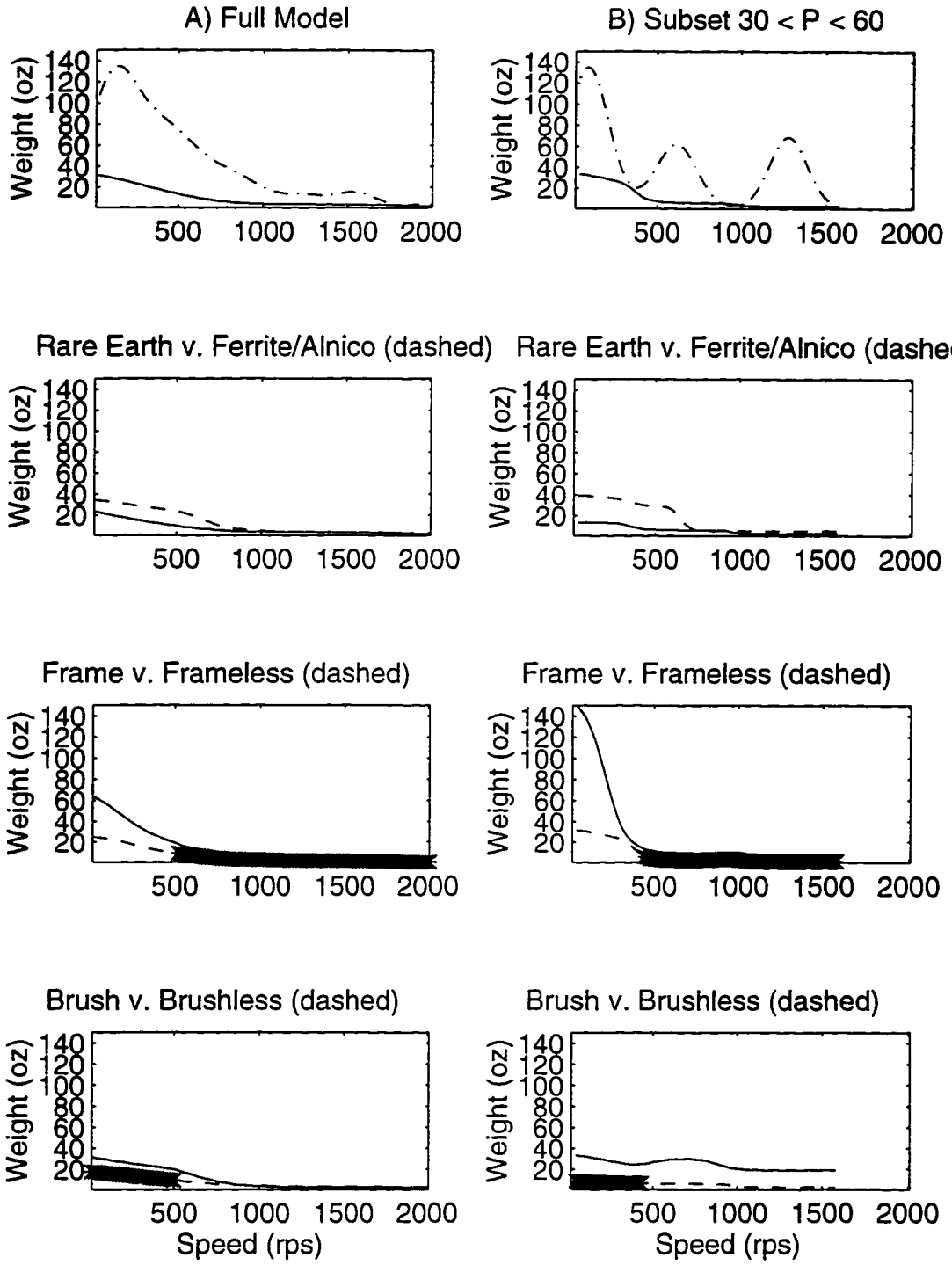


Figure A.8 Full/Subset Decision Making - Speed. Plot shows optimal decision ('*') for decreasing design abstraction based on the value of speed. Column A is the full model, Column B a subset.Top panel shows expected value of weight given speed overlaid with the distribution of speed (dot dash). Note the contrast between the smooth full catalog model and the 'bumpy' subset model.

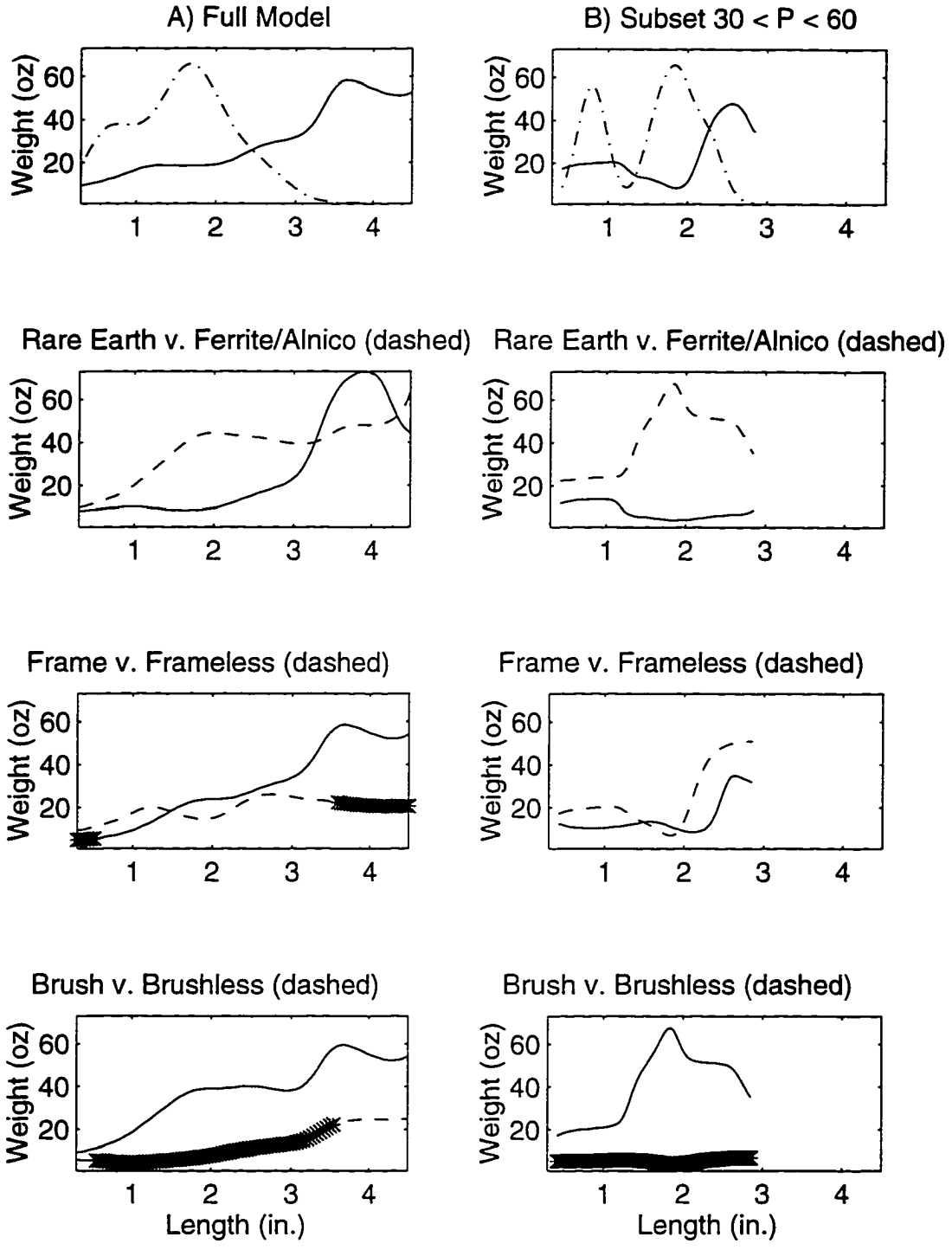


Figure A.9 Full/Subset Decision Making - Length. Plot shows optimal decision ('*') for decreasing design abstraction based on the value of length. Column A is the full model, Column B a subset.Top panel shows expected value of weight given length overlaid with the distribution of length (dot dash). Note the contrast in decision making between the two plots, and the exaggerated humps in the subset distribution.

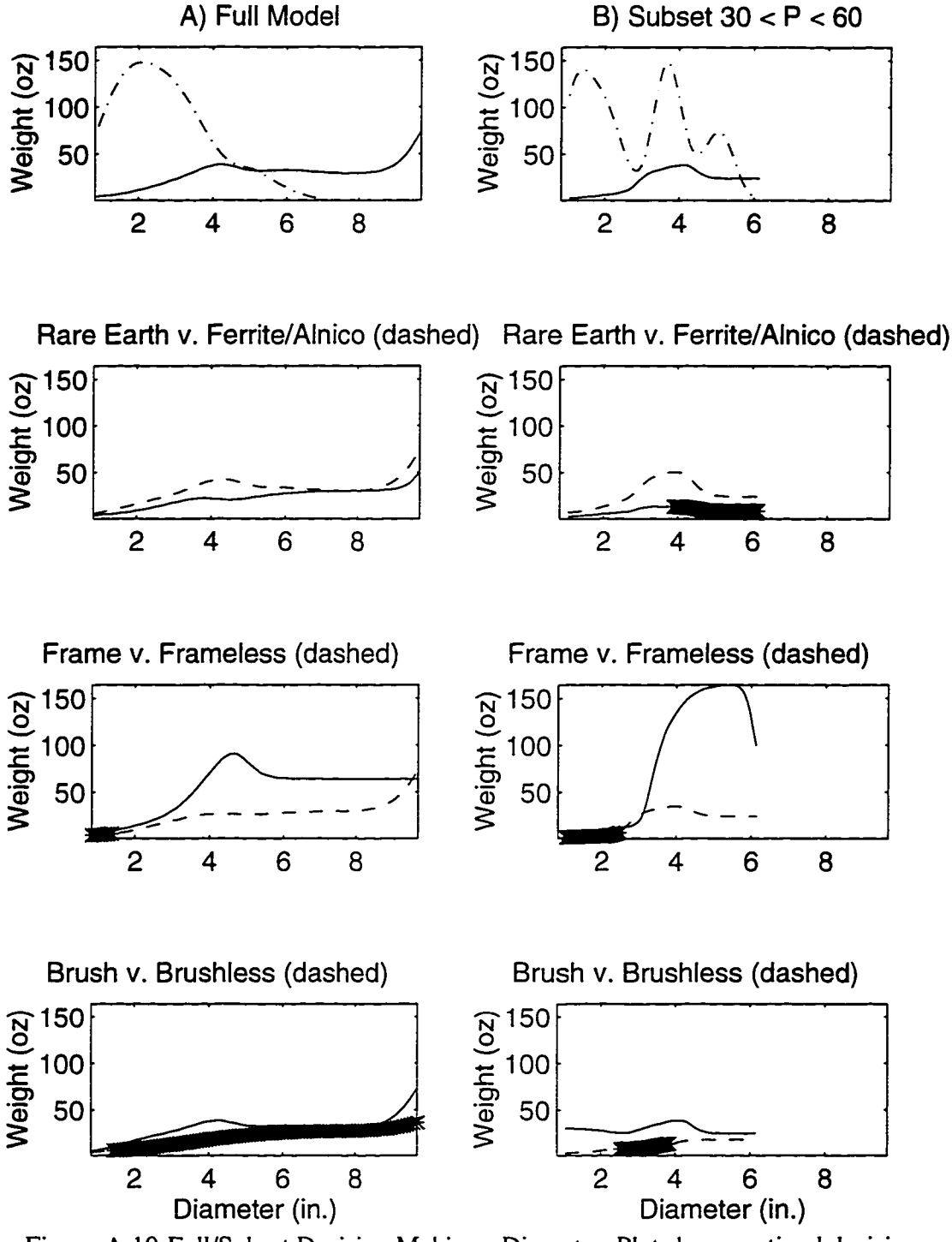


Figure A.10 Full/Subset Decision Making - Diameter. Plot shows optimal decision ('*') for decreasing design abstraction based on the value of diameter. Column A is the full model, Column B a subset.Top panel shows expected value of weight given diameter overlaid with the distribution of diameter (dot dash). Note the contrast in decision making between the two plots, and the exaggerated humps in the subset distribution.

probability densities show similar patterns, eccentricities apparent in the full model exaggerated in the reduced model. Figure A.7 shows the least correspondence, likely due to the design space compression along the torque axis for the full model. This brings in too many motors and smooths out both the expected value of the objective and the probability density of torque.

Overall, in Experiment 2, decision making is consistent from the full constrained model to the reduced model. The main deviations occur in the two design variables with the greatest difference in normalization: torque and diameter. Torque-based decision making is changed from brushless to frameless at extreme low values, otherwise decision making is consistent. Diameter-based decision making follows the same trend for moderate to low diameters, but the reduced model favors a switch from brushless motors to rare earth at larger diameters.

A.1.3 Comparing Nominal and Interval Constraints

In this experiment, the effectiveness of applying constraint the the design space using a screening query to reduce the design sample space is evaluated. Figure A.11 shows the typical differences in modeling; the constrained subset of column *A* shows tighter packing around modes reinforced by the constraint, the unconstrained subset of column *B* shows more gradual transitions in the probability distributions. Figures A.12 - A.15 bear this out. Expected value plots for the weight are fairly consistent from unconstrained to constrained distributions; probability densities of each design variable is smoother in all of the unconstrained runs. Despite this smoothing, decision making is remarkably consistent. As with an effect we will discuss in the next section, transitions among decisions remain consistent while transition threshold changes slightly. In general, applying constraint through a screening query produces very similar results to those obtained by applying a more 'forceful' constraint on the value of the query variable.

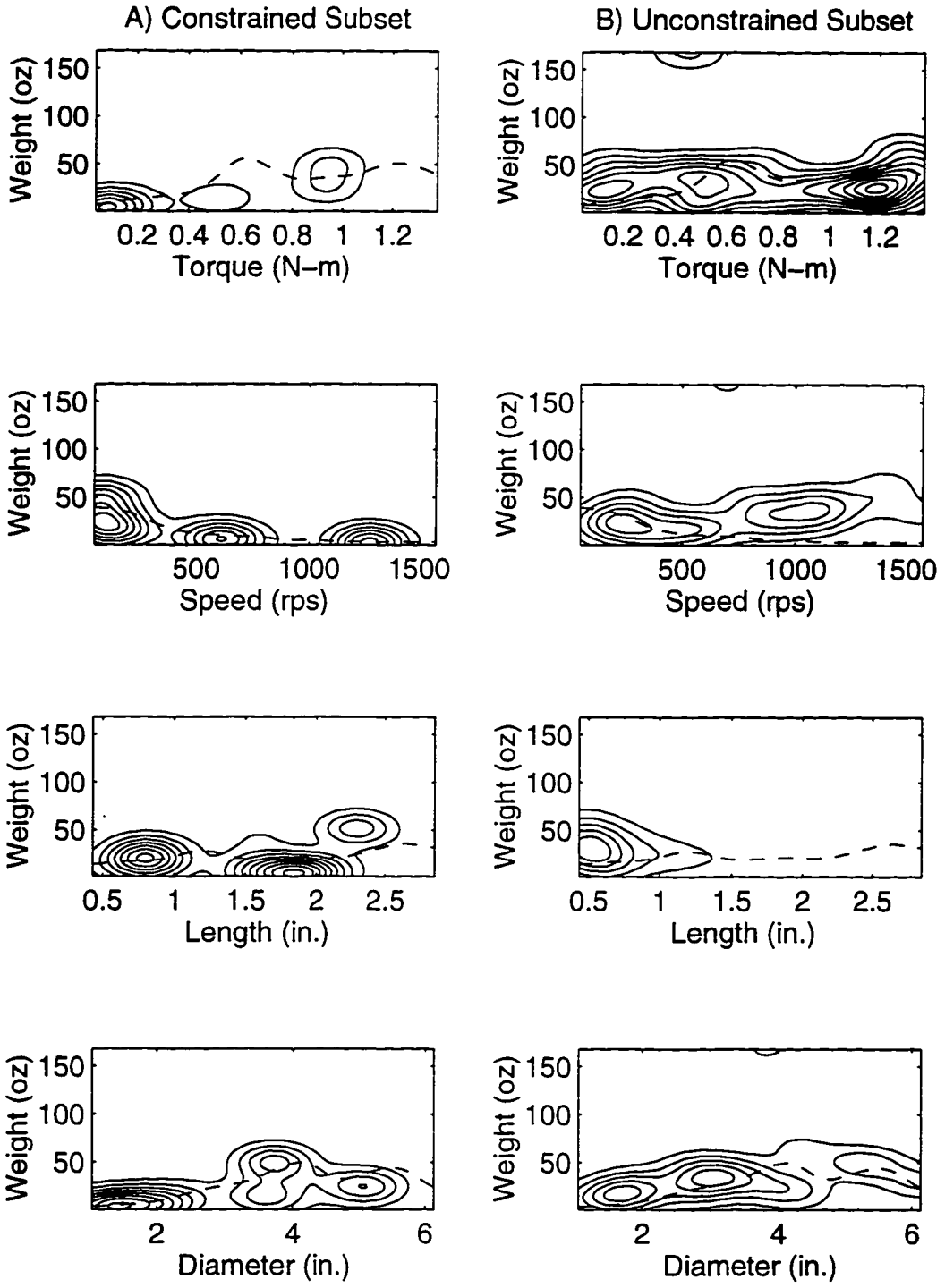


Figure A.11 Constrained/Unconstrained Subset Modeling. Contour plots of constrained model (A) and unconstrained model (B) generated from a subset of the design space, overlaid with expected value of weight (dashed line). Constrained plots show islands typical of under-generalization, unconstrained distributions are smoother.

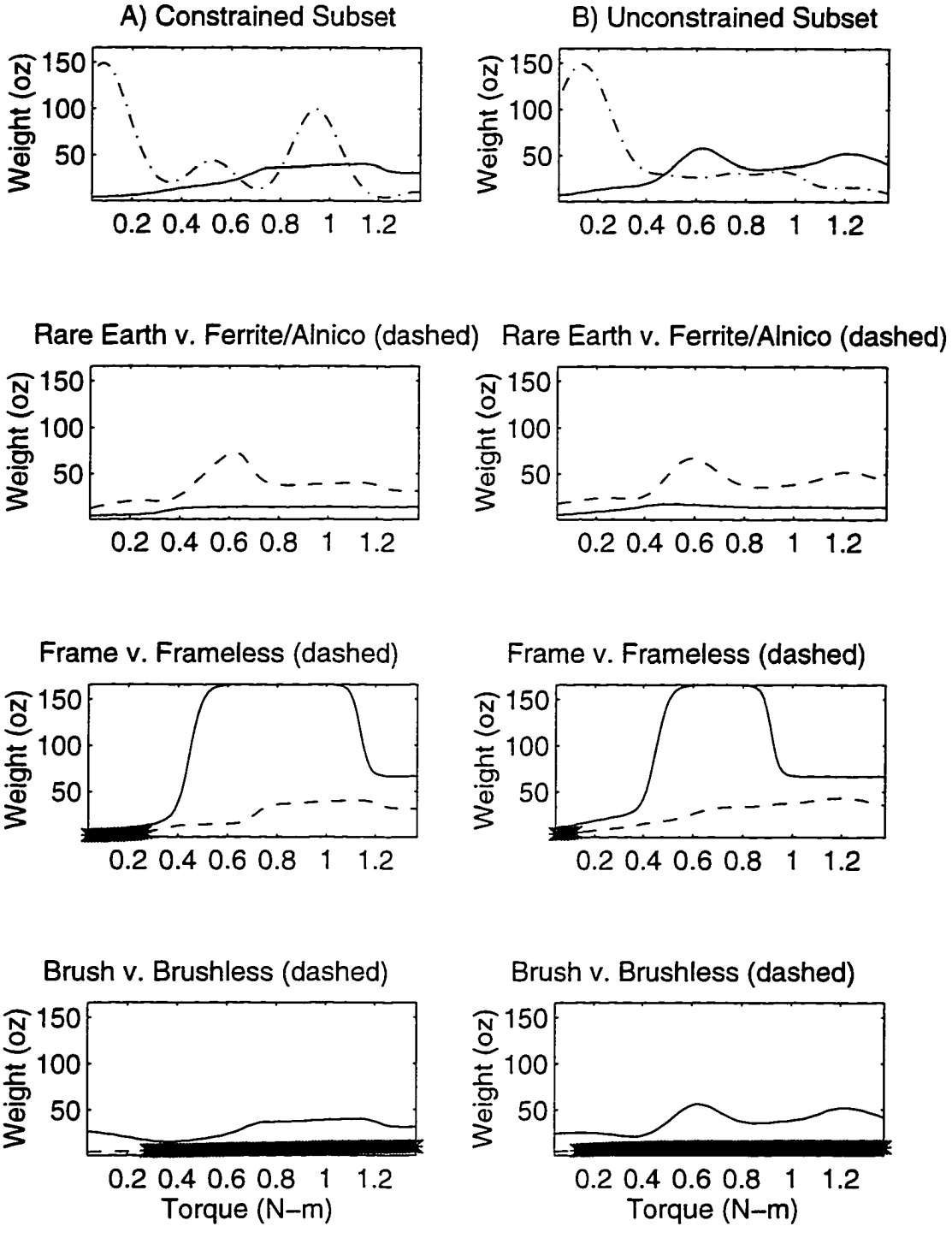


Figure A.12 Constrained/Unconstrained Subset Decision Making - Torque. Plot shows optimal decision ('*') for decreasing design abstraction based on the value of torque. Column A is constrained, Column B unconstrained.Top panel shows expected value of weight given torque overlaid with the distribution of torque (dot dash). Plots show consistent expected value and decision making.

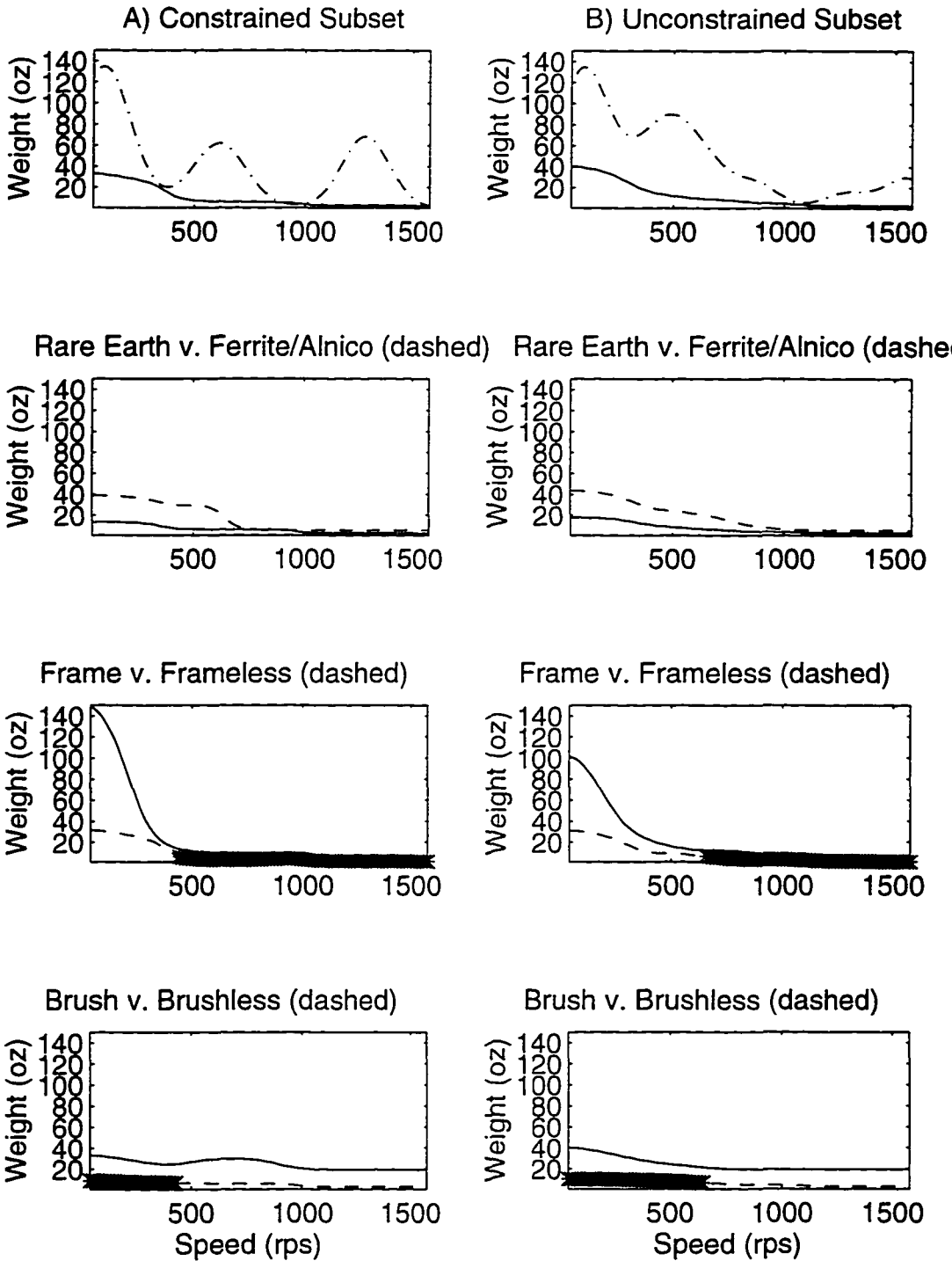


Figure A.13 Constrained/Unconstrained Subset Decision Making - Speed. Plot shows optimal decision ('*') for decreasing design abstraction based on the value of speed. Column A is constrained, Column B unconstrained.Top panel shows expected value of weight given speed overlaid with the distribution of speed (dot dash). Note how decision change threshold changes, under-generalization of constrained speed distribution.

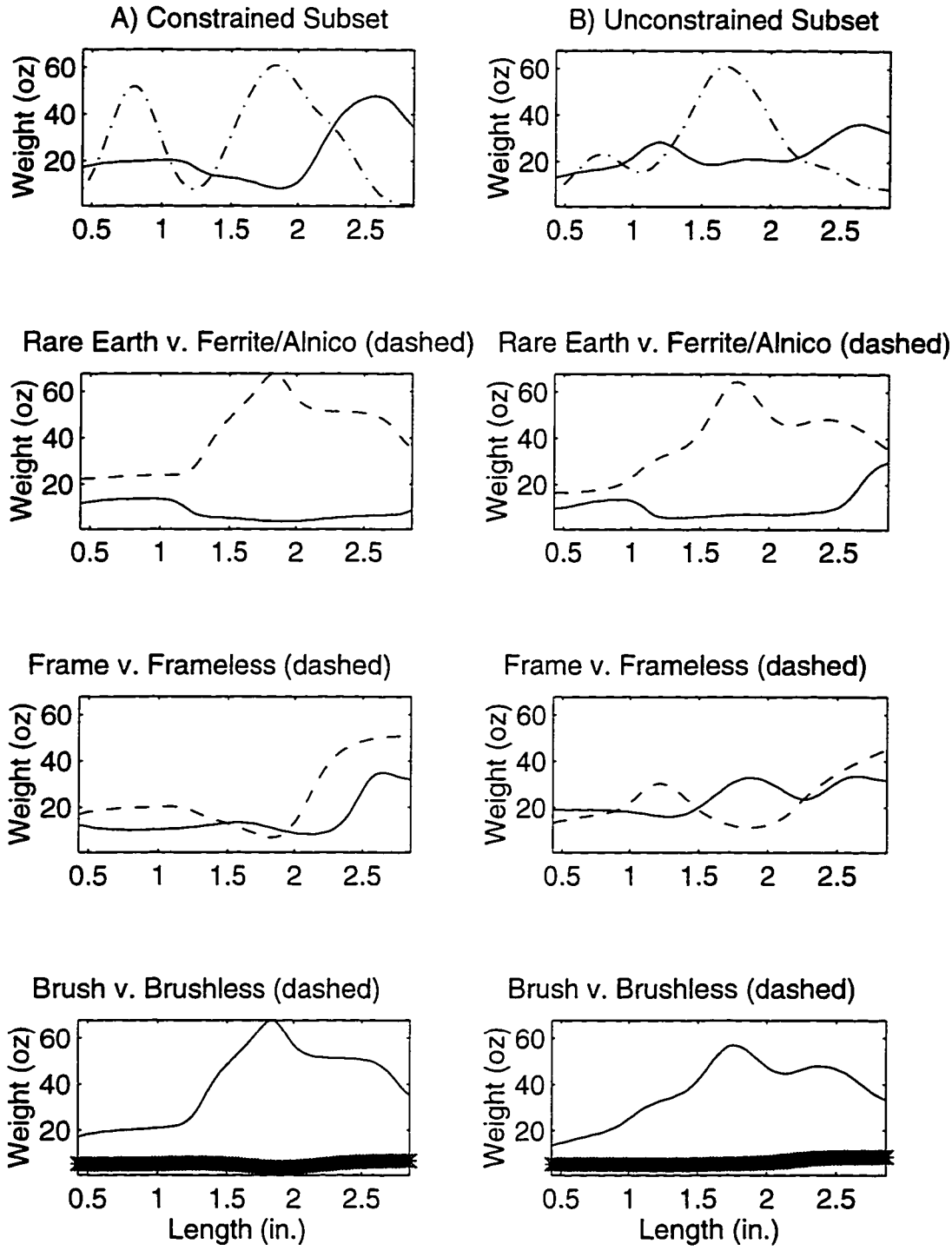


Figure A.14 Constrained/Unconstrained Subset Decision Making - Length. Plot shows optimal decision ('*') for decreasing design abstraction based on the value of length. Column A is constrained, Column B unconstrained.Top panel shows expected value of weight given length overlaid with the distribution of length (dot dash). Again, under-generalization of the cosntrained model appears as 'bumpy' distribution.

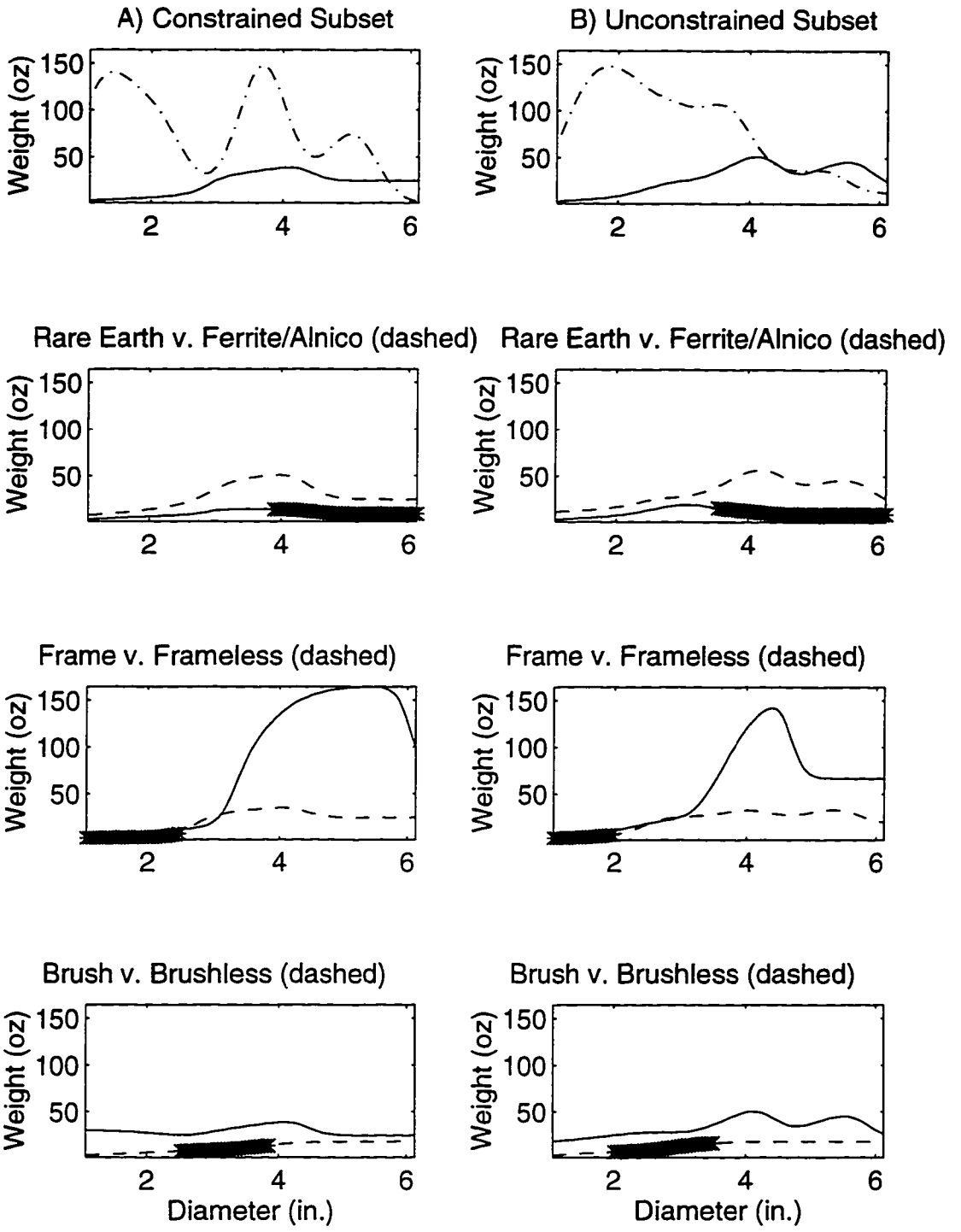


Figure A.15 Constrained/Unconstrained Subset Decision Making - Diameter. Plot shows optimal decision ('*') for decreasing design abstraction based on the value of diameter. Column A is constrained, Column B unconstrained.Top panel shows expected value of weight given diameter overlaid with the distribution of diameter (dot dash). Consistent decision making is seen.

A.1.4 Discussion

The three experiments discussed above show that application of constraint to the design space is possible using the design space approximations we have proposed. In addition, there are two valid means of accomplishing this: constraining a design variable to a nominal value or constraining it to a range of values. These two methods are shown to produce qualitatively similar results when used as the basis for design abstraction decision making as well as for expected value modeling.

In addition to this good result, one important aspect of modeling implementation has been discovered – normalization. The process of abstracting the real design space to one lying in the range [0,1] as specified by Specht [1988] cannot be taken lightly. Experiment 2 points out problems in the transition from a large to a small sample space. In the large space, the non-uniform distribution of design variables in the normalized space may cause different levels of system "generalization' along different design variable axes. In our case torque values tightly centered near the normalized origin in the full sample space are over generalized. More equitable normalization schemes must be found. Initial efforts using standard deviations produced results similar to the linear scaling results discussed here. Instead of tight packing near the origin, tight packing near the mid point caused the same problems.

A.2 Effect of the Smoothing Parameter on Modeling and Decision Making

Equation A.1.1 shows how samples (specific motor designs in the CDIS) of the design space are generalized to form a joint probability density approximating an underlying continuous design space. Once the design space is normalized to fall in the range [0,1], a single parameter can be used to represent the degree to which this generalization occurs – **c**. In experiments with generated distributions, Specht [1988] designates a range of values for **c** which express optimality with respect to integrated spared error. As is shown in Figure A.16, this range turns out to be a rather large one. The models shown here progress from a

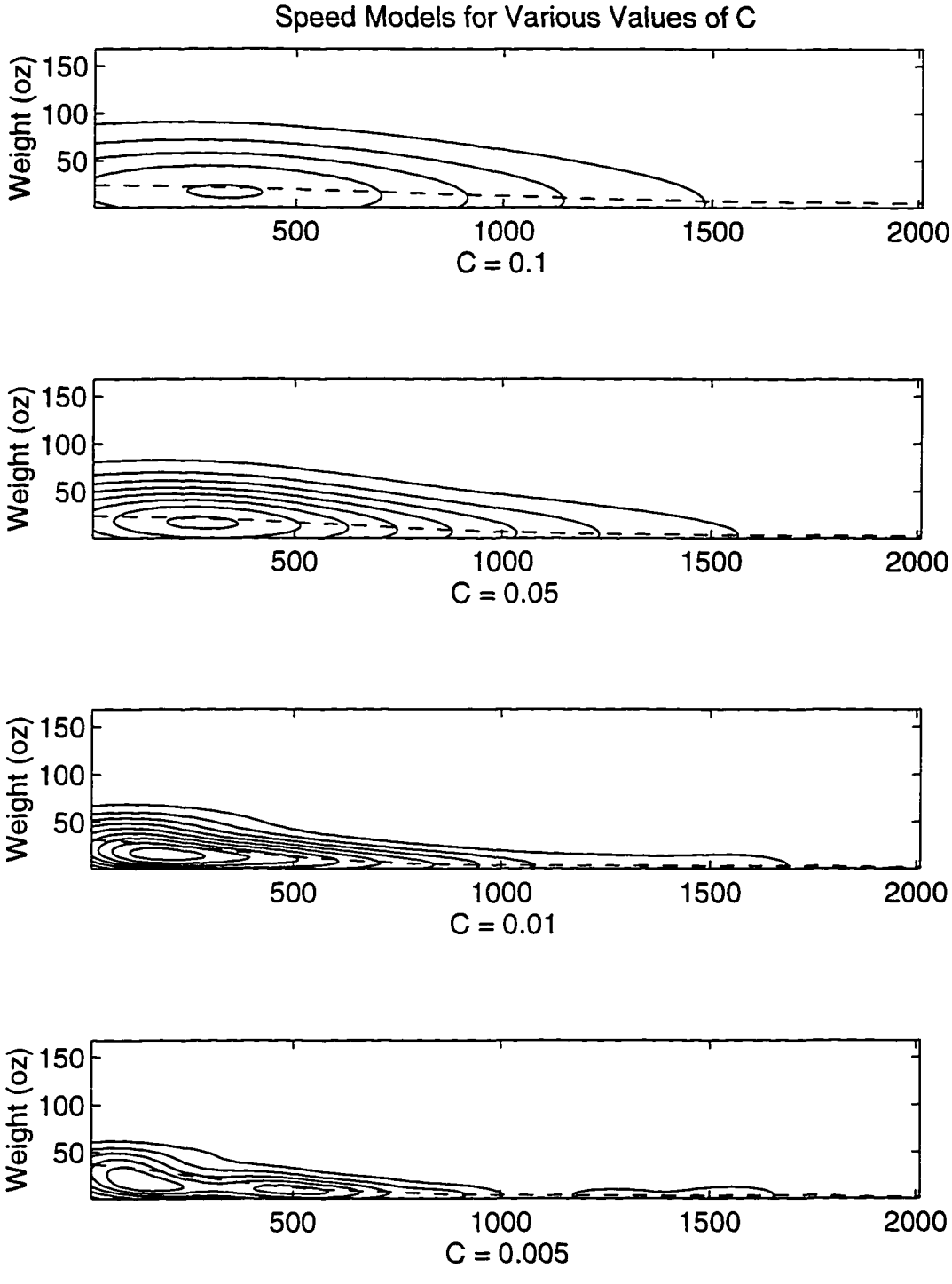


Figure A.16 Smoothing Parameter Variation in Engineering Models. Contour plots show the efect of changes in **c** in the model which relates speed to weight for motors. Note how decreasing **c** produces much tighter distributions, the lowest value yielding a multi modal distribution.

single mode distribution for **c** = 0.1 to the (larger than) trimodal distribution for **c** = 0.005. In the case of all of the examples shown here, the distributions are based on models that cannot be validated mathematically (motor weight is a complex relation of more variables than speed, torque, length, diameter, power, magnet type, frame type, and brush type). Figure A.17 shows the expected value of weight with respect to the rated speed of the motor (motor constrained to 45W, full catalog used to generate model). As **c** is decreased, the function approximating the relationship between speed and weight shows a gradual progression. With a large amount of generalization, the expected value is rather flat. As the local 'neighborhood' by which a point in the design space is influenced is decreased in size (i.e., lowering **c**), the function shows more features. At the lowest level of **c**, the curve begins to show distinct humps in which local clusters of designs disturb the generalization;

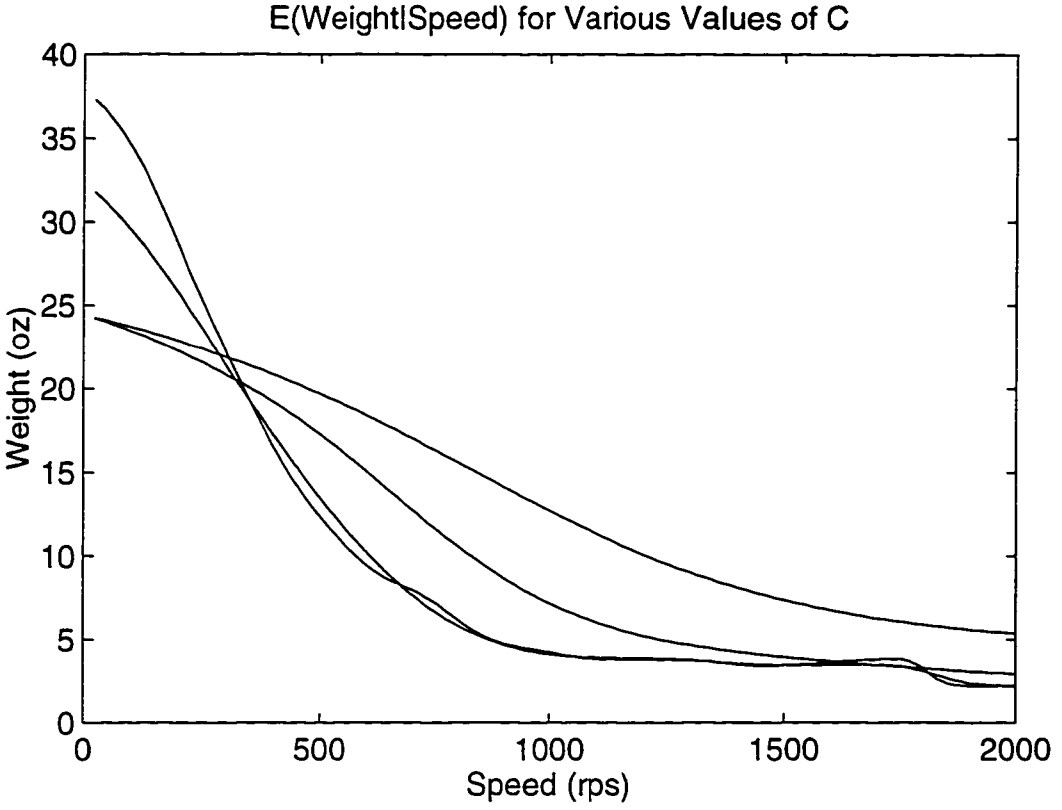


Figure A.17 Smoothing Parameter Variation in Expected Value. As **c** decreases, the plots of expected value of weight given speed change from relatively flat to steeper, and finally show bumps representative of the multimodal distribution of **c** = 0.005.

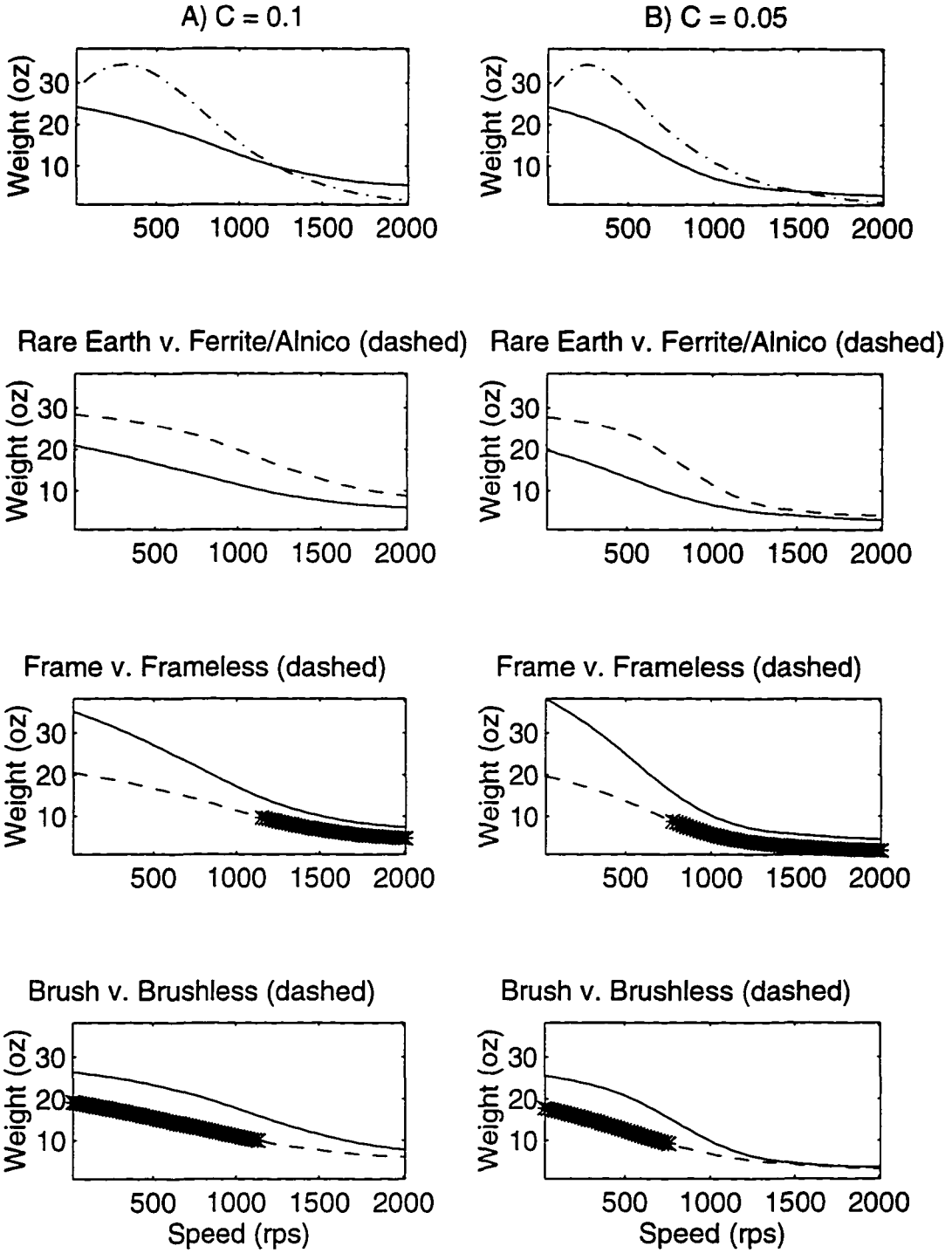


Figure A.18 Smoothing Parameter Variation in Decision Making. Plots show the optimal decision ('*') given speed. Top panes show expected value of weight given speed overlaid with the distribution of speed (dashed line). Note how the transition from one decision to another shifts with decreases in c.

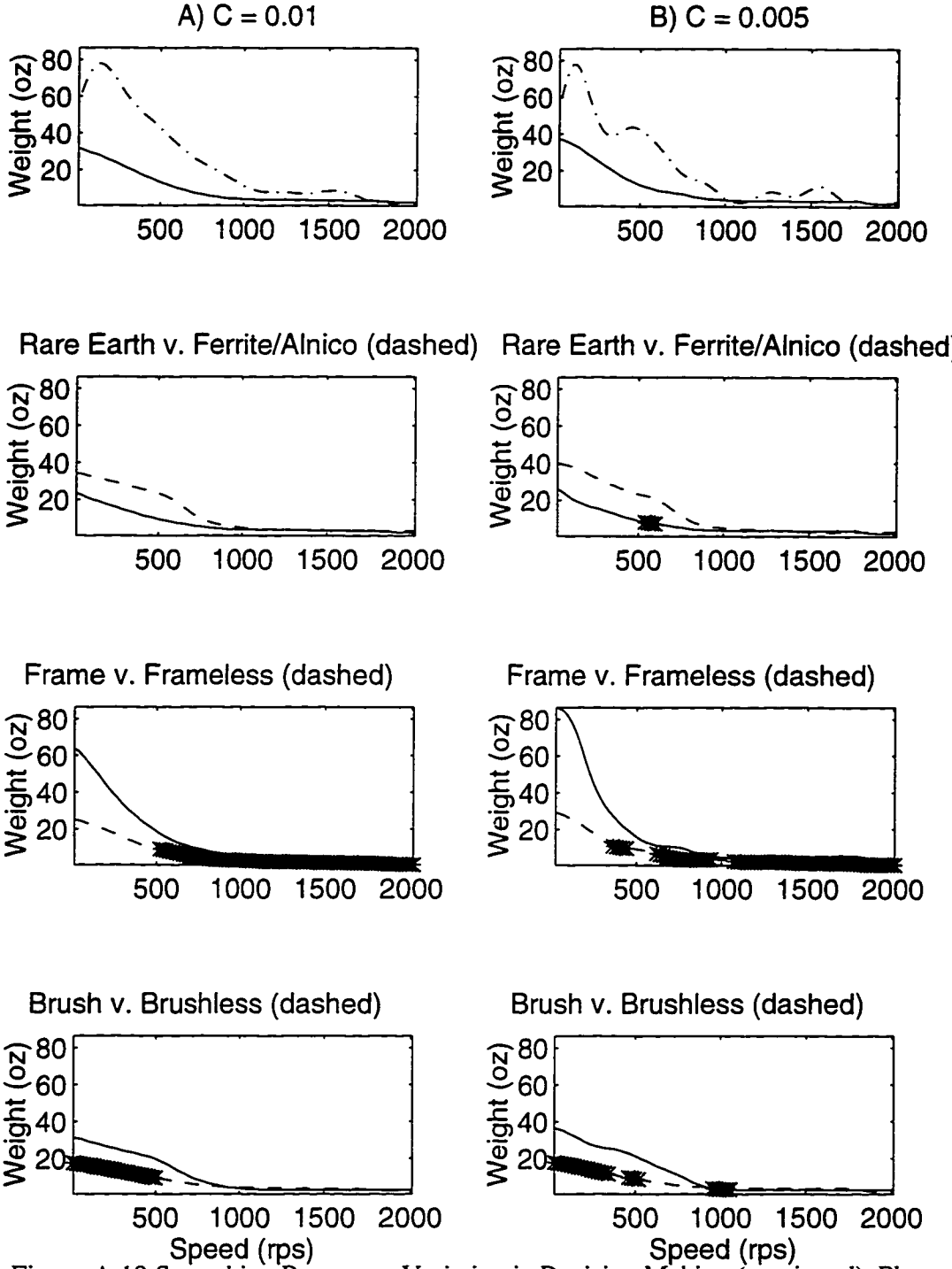


Figure A.19 Smoothing Parameter Variation in Decision Making (continued). Plots show the optimal decision ('*') given speed. Top panes show expected value of weight given speed overlaid with the distribution of speed (dashed line). Note that the transition from one decision to another still shifts with decreases in **c**. When **c** = 0.005, decision threshold becomes 'noisy', showing confusion from under-genrealization.

the multi modal nature shown in the bottom pane of Figure A.16 is revealed.

As far as decision making is concerned, an interesting effect is shown in Figures A.18 and A.19. Here, the design abstraction decision with respect to motor speed is shown for the four values of **c** used in Figures A.16 and A.17. Note that as **c** decreases, decision making transitions remain largely the same. The major change is in the threshold for these transitions. For the most general model, this transition occurs at about 1200 rps, moving steadily lower to 800 rps and then to 500 rps as **c** changes from 0.1 to 0.05 to 0.01. When **c** is then changed to 0.005, 'noise' around the 500 rps transition point appears in the form of an oscillation from one decision to another before the system finally 'settles' on the prior decision about an equal distance from the threshold from the point where the 'indecision' begins. Additional noise appears later in the decision range. All this is at a point which is slightly below that recommended by Specht.

The task remains to try to identify an optimal value for **c**. Because the results shown in Figures A.16 - A.19 are typical of those for all decision variables, it is an open question to determine how best to vary generalization. The fact that variation produces predictable results is significant, how to exploit this regularity is the question.

A.3 Reducing Computation using SOPNN

As discussed in Chapter 6, the SOPNN (Self Organizing Probabilistic Neural Network) proposed by Tseng [1991] uses clustering in the sample space to reduce the amount of computation necessary for estimating the joint probability distribution of the design space. Figures A.20 - A.24 provide a comparison of the performance of this clustered design space with that of the standard, full catalog design space model. In each case, the 45W power constraint is in place. The overriding impression one gets examining these figures is that the SOPNN clustering 'clumps' the design space into too few clusters. Generalizing from these looks similar to the effect of a smoothing parameter that is too small (discussed in the previous section). The probability density contour plots of column *A* of Figure A.20

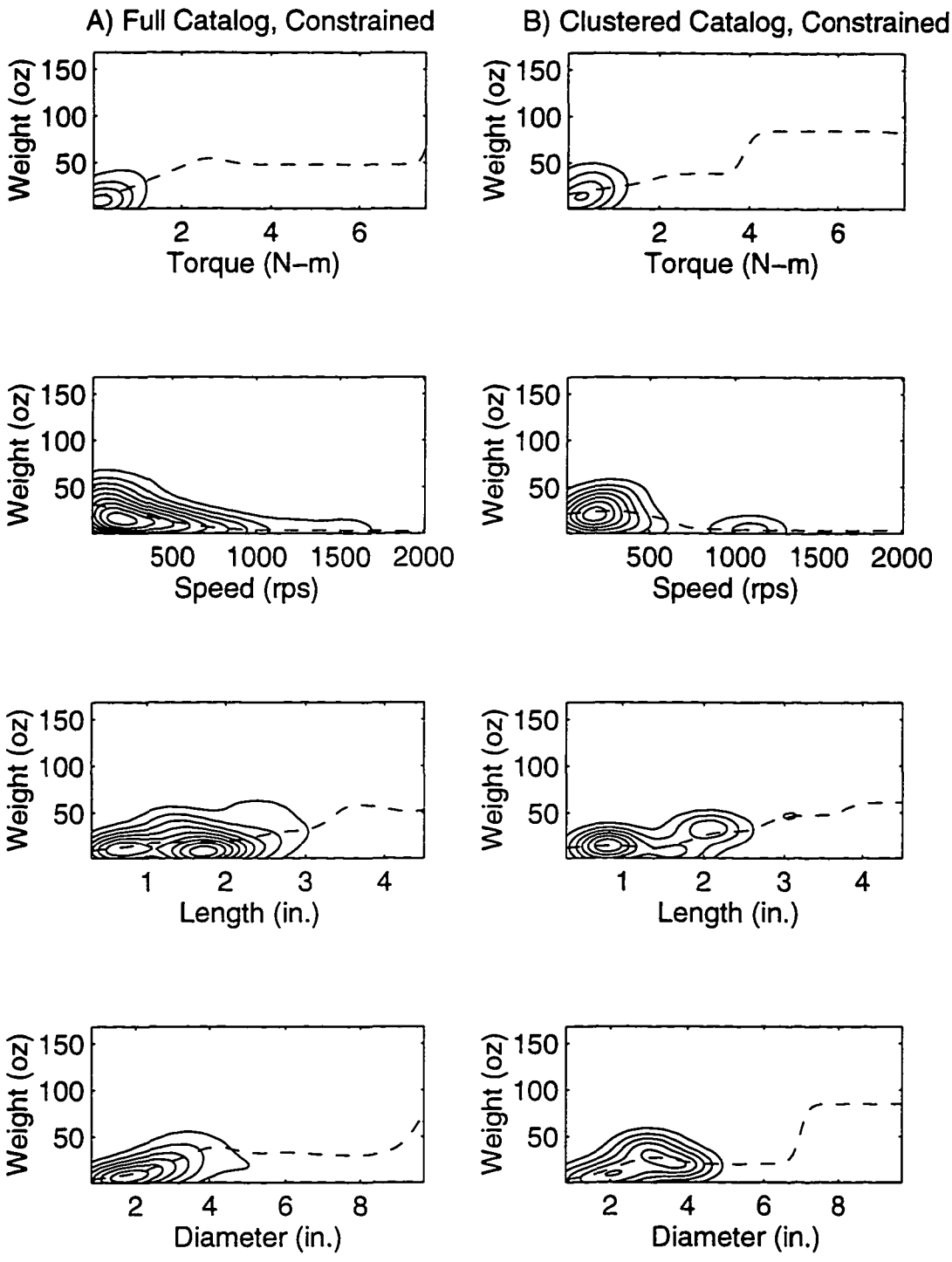


Figure A.20 Full/Clustered Constrained Modeling. Contour plots of full catalog model (A) and clustered (using SOPNN) model (B), overlaid with expected value of weight (dashed line). Clustered plot shows under-generalization as seen in constrined subset plots. If SOPNN were appropriate, plots would be identical.

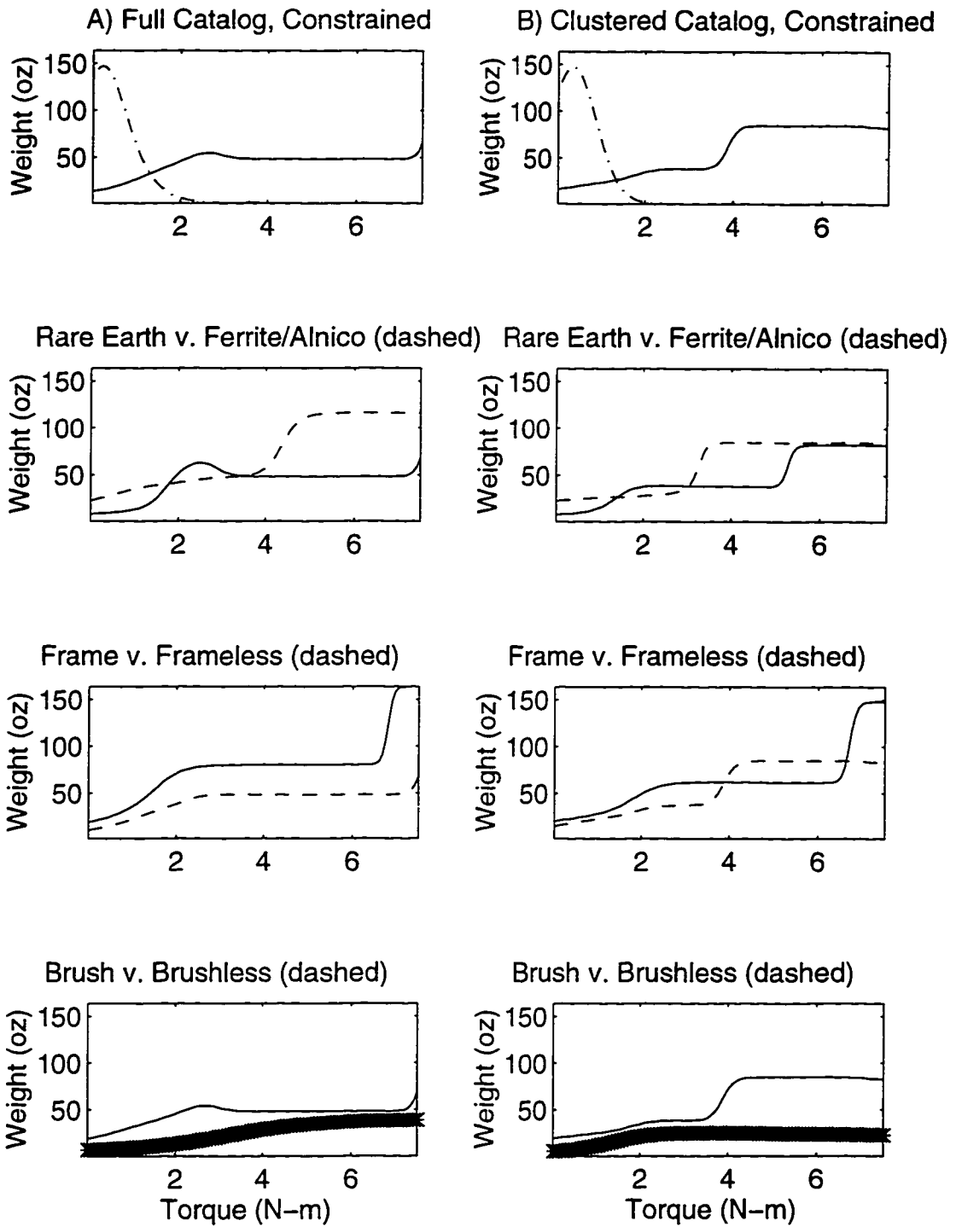


Figure A.21 Full/Clustered Constrained Decision Making - Torque. Plot shows optimal decision ('*') for decreasing design abstraction based on the value of torque. Column A is full catalog, Column B clustered.Top panel shows expected value of weight given torque overlaid with the distribution of torque (dot dash). Again, plots should be identical but clustering shows under-generalization.

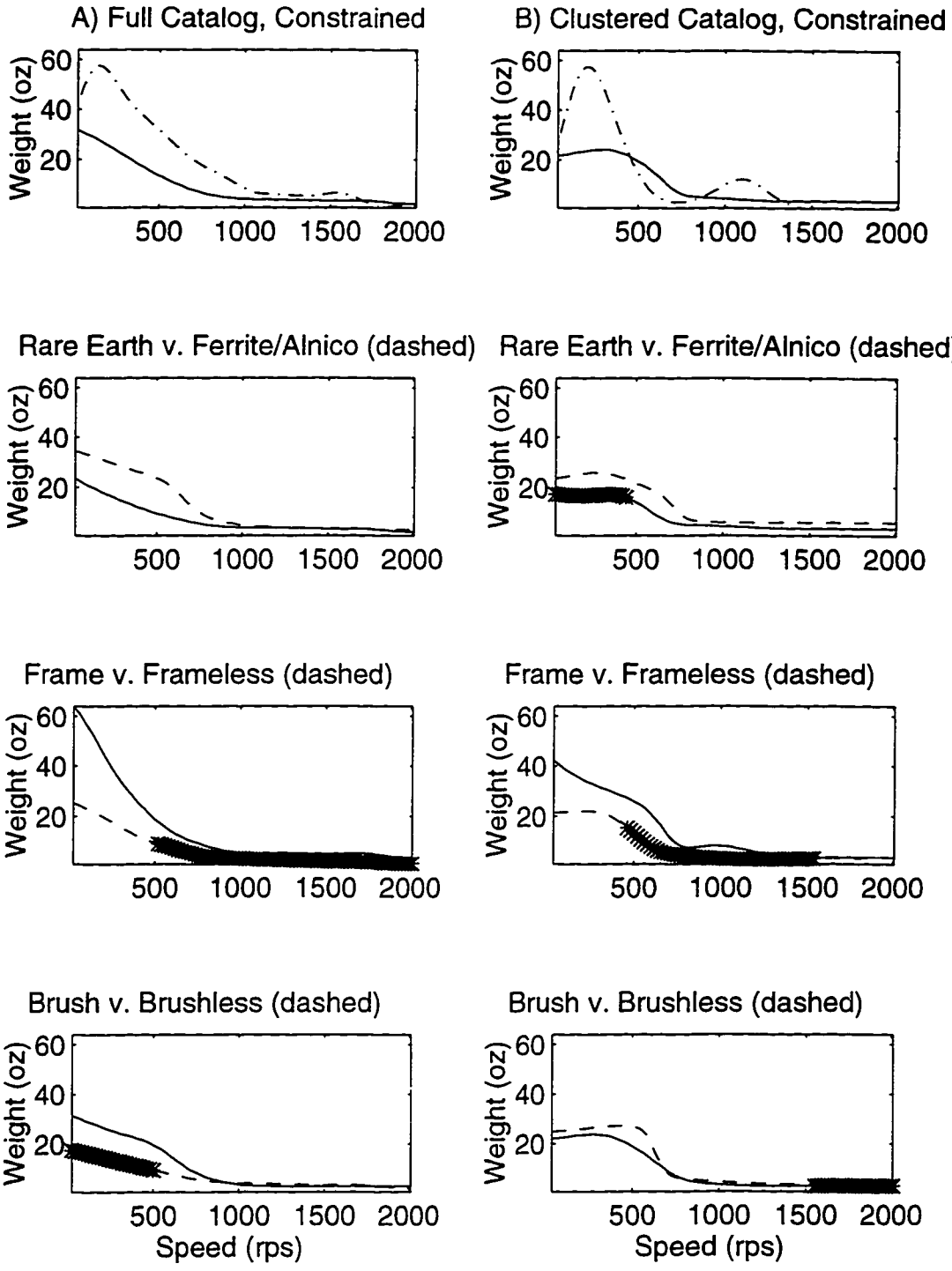


Figure A.22 Full/Clustered Constrained Decision Making - Speed. Plot shows optimal decision ('*') for decreasing design abstraction based on the value of speed. Column A is full catalog, Column B clustered.Top panel shows expected value of weight given speed overlaid with the distribution of speed (dot dash). Again, plots should be identical but clustering shows under-generalization, disagreement of expected value and decision making.

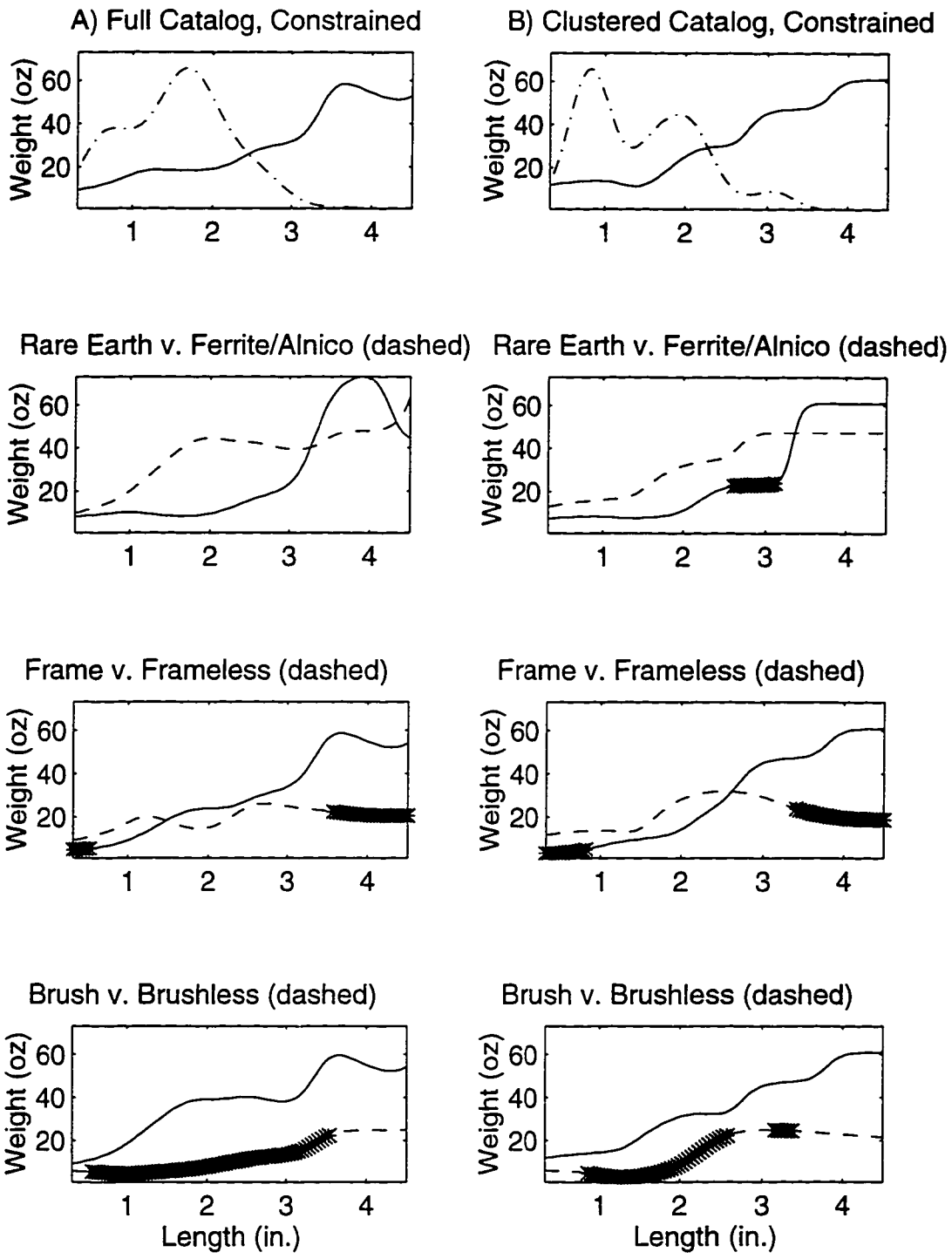


Figure A.23 Full/Clustered Constrained Decision Making - Length. Plot shows optimal decision ("*") for decreasing design abstraction based on the value of length. Column A is full catalog, Column B clustered.Top panel shows expected value of weight given length overlaid with the distribution of length (dot dash). Plots show better agreement than A.21 and A.22, but still disagree substantially with regard to expected value and decision making.

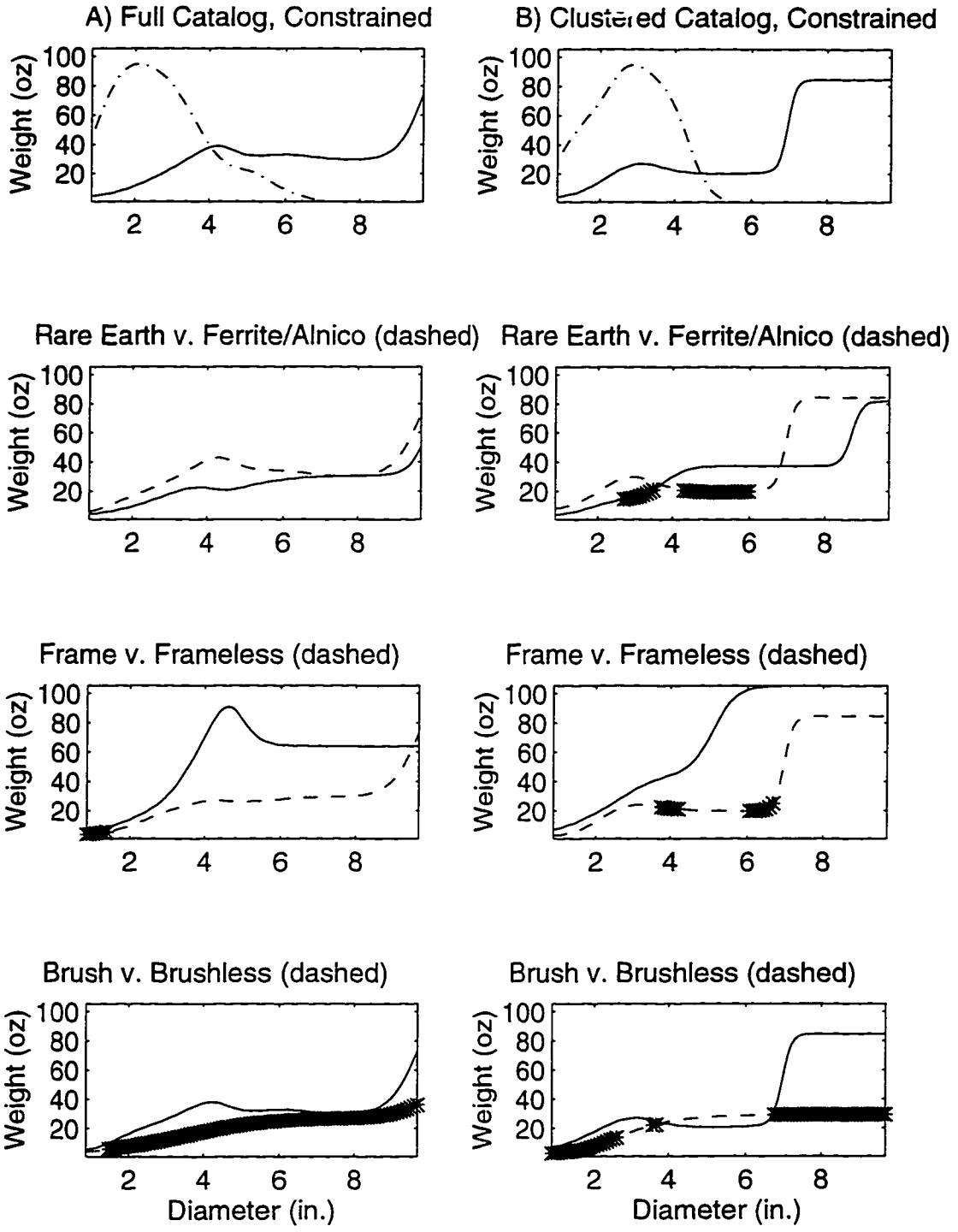


Figure A.24 Full/Clustered Constrained Decision Making - Diameter. Plot shows optimal decision ('*') for decreasing design abstraction based on the value of diameter. Column A is full catalog, Column B clustered.Top panel shows expected value of weight given diameter overlaid with the distribution of diameter (dot dash). Plots show significant disagreement in expected value.

show fairly smooth contours for the full catalog, mostly appearing to be of a single mode. In contrast, the plots of column *B* for the SOPNN clustered catalog show distinct multi modal behavior. The individual clusters are not being adequately 'smoothed'.

Figures A.21 - A.24 show the expected value and decision making plots for full vs. clustered design space. Of note is the much more 'blocky' character of the clustered results. Curves are not as smooth as those for the full catalog. In addition, unlike the slight 'bumps' that are apparent in the under-generalization case shown in section A.2, the aberrations for speed in Figure A.22 amplify the difference. Whereas under-generalization produced predictable changes in the expected value of the objective with respect to speed, there is a new effect in the SOPNN data. It looks to be the case that under-generalization is occurring in combination with an over-generalization of the underlying data. Observe the brush vs. brushless decision pane at the lower right of Figure A.22. The step function of weight in brushless motors appears to be the effect of a large distribution. Decision making in this figure also shows drastic changes from the full model recommending brushless for low speed motors to the clustered catalog recommending rare earth.

There is a relatively simple explanation for this effect. The electric motors in the catalog represent a design space in which tradeoffs among the varied performance parameters and the cost of producing motors has yielded an efficient set of design samples. Generalizing this set reduces coverage in the design space, resulting in the disjoint models of Figure A.20 and the discrepancies in decision making in Figures A.22 - A.24. The only consistent performance between clustered and unclustered catalogs is in the one design variable which was singled out in Section A.1 as having been over-generalized by a mismatch in the design space normalization process. It makes sense that this initial over-generalization washes out the difference between clustered and full catalogs. This initial poor result does not preclude ever using the SOPNN method within the CDIS. When the catalogs of several vendors are combined, SOPNN can be used to reduce computation by clustering together motors that compete against each other in the marketplace.

A.4 Summary

This appendix has discussed the various implementational concerns surrounding the use of Specht's method for approximating the probability density of the design space. Two methods have proven effective for applying constraint to the design space: applying nominal constraint using a conditional probability distribution and applying interval constraints through reducing the design space with screening queries. Other results with respect to the size of the sample space are necessary to understand more fully the boundaries of the size of the design sample space for which the approximation is valid.

Generalization through the manipulation of the approximation smoothing parameter, **c**, has also been discussed. While no conclusions as to the optimal value for this parameter have been made, a qualitative understanding of the effects of changing it has been gained. Further tests based on mathematical relationships defined within the database must be made to further narrow the range of acceptable generalization parameter values.

Finally, the impact of using the SOPNN method for generalizing the design sample space has been determined. In the case of a single manufacturer database such as the one used for testing, the SOPNN runs did serve to validate that manufacturer's product offerings as being well distributed in the design space. The use of SOPNN must be further explored in a multi-vendor database where more exact duplication of motor performance tradeoffs is more likely to occur.