

WL-TR-97-1165



**A KNOWLEDGE-BASED APPROACH TO
INTEGRATING AND QUERYING
DISTRIBUTED INFORMATION SYSTEMS
HETEROGENEOUS INTELLIGENT PROCESSING FOR
ENGINEERING DESIGN (HIPED)**

Professor Shamkant B. Navathe, P.I.

**Georgia Institute of Technology
College of Computing
Atlanta, Georgia 30332-0280**

AUGUST 1997

FINAL REPORT FOR PERIOD 30 September 1993 - 30 March 1997

Approved for public release; distribution unlimited

19980420 122

DTIC QUALITY INSPECTED 4

**AVIONICS DIRECTORATE
WRIGHT LABORATORY
AIR FORCE MATERIEL COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7623**

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government related procurement, the United States Government incurs no responsibility nor any obligation whatsoever. The fact that the Government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell patented invention that may in any way be related thereto.

This technical report has been reviewed and is approved for publication.

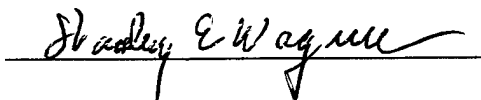
This report is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.



CHARLES P. SATTERTHWAITE, Project Engineer
Software Hardware Technology Branch
WL/AASH



JAMES S. WILLIAMSON, Acting Chief
Software Hardware Technology Branch
WL/AASH



STANLEY E. WAGNER, Chief
Systems Concepts & Simulation Division
WL/AAS

IF YOUR ADDRESS HAS CHANGED, IF YOU WISH TO BE REMOVED FROM OUR MAILING LIST, OR IF THE ADDRESSEE IS NO LONGER EMPLOYED BY YOUR ORGANIZATION, PLEASE NOTIFY WL/AASH, BLDG 620, 2241 AVIONICS CIRCLE, WRIGHT-PATTERSON AFB, OH 45433-7318 TO HELP US MAINTAIN A CURRENT MAILING LIST.

COPIES OF THIS REPORT SHOULD NOT BE RETURNED UNLESS RETURN IS REQUIRED BY SECURITY CONSIDERATIONS, CONTRACTUAL OBLIGATIONS, OR NOTICE ON A SPECIFIC DOCUMENT.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE August 1997	3. REPORT TYPE AND DATES COVERED Final 30 Sept 93 - 30 Mar 97		
4. TITLE AND SUBTITLE A Knowledge-Based Approach to Integrating and Querying Distributed Information Systems Heterogeneous Intelligent Processing for Engineering Design (HIPED)			5. FUNDING NUMBERS C: F33615-93-1-1338 PE: 62301E PR: A522 TA: 01 WU: 01	
6. AUTHOR(S) Prof. Shamkant B. Navathe, P.I.				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Georgia Institute of Technology College of Computing Atlanta, Georgia 30332-0280			8. PERFORMING ORGANIZATION REPORT NUMBER Grant #: F33615-93-1-1338	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AVIONICS DIRECTORATE WRIGHT LABORATORY AIR FORCE MATERIEL COMMAND WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7623 POC: Charles Satterthwaite, AFRL/IFTA, 937-255-6548 x3584			10. SPONSORING/MONITORING AGENCY REPORT NUMBER WL-TR-97-1165	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION AVAILABILITY STATEMENT Approved for public release; distribution unlimited			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) This program develops techniques which allows for the interactive access of dissimilar data sources; the informed processing of the resultant search; and the capture of the associated knowledge gained from the interactive usage of the system.				
14. SUBJECT TERMS Heterogeneous distributed Information, Intelligent Query Processing, Databases Knowledgebases			15. NUMBER OF PAGES 174	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT SAR	

TABLE OF CONTENTS

	<u>page(s)</u>
SECTION 1: Executive Summary	1-2
SECTION 2: Summaries of Research and Published Papers	3
PART I: HIPED: Heterogeneous Information Processing for Engineering Design	4-8
PUBLICATIONS (PART 1)	9
a. Towards Intelligent Integration of Heterogeneous Information Sources	10-18
b. Rule Based Database Integration in HIPED: Heterogeneous Intelligent Processing in Engineering Design	19-26
c. From Data to Knowledge: Method-Specific Transformations	27-35
d. Integrating Heterogeneous Databases for Engineering Design	36-50
e. Method-Specific Knowledge Compilation: Towards Practical Design Support Systems ...	51-69
PART II: Visualization And User Interface Techniques for Information Retrieval	70-73
PUBLICATIONS (PART 2)	74
f. Visual Interface for Textual Information Retrieval Systems	75-79
g. Querying, Navigating and Visualizing a Digital Library Catalog	80-83
h. Interactive TREC-4 at Georgia Tech	84-94
i. Evaluation of a Tool for Visualization of Information Retrieval Results	95-118
j. Effectiveness of a Graphical Display of Retrieval Results	119-128
PART III: Metadata Management for Intelligent Query Processing	129-131
PUBLICATIONS (PART 3)	132
k. Maintaining Instance-Based Constraints for Semantic Query Optimization	133-153
l. Maintaining Semantic and Structural Metadata in the View Graph Framework	154-173
PART IV: Development of Intelligent Interface Tool for Engineering Design	174-176
PUBLICATIONS (PART 4)	177
m. Explanatory Interface in Interactive Design Environments.....	178-197
n. Toward Design Learning Environments - I: Exploring How Devices Work	198-206
o. Meta-Cases: Explaining Case-Based Reasoning	207-220
p. Functional Explanations in Design	221-230
SECTION 3: Conclusions	231-234

Section 1: EXECUTIVE SUMMARY

This report is a compilation of a summary of all the research work and publications produced as a result of the project: "A Knowledge-Based Approach to Integrating and Querying Distributed Heterogeneous Information Systems", at the Georgia Institute of Technology, College of Computing. The project was also nicknamed "HIPED" for Heterogeneous Intelligent Processing for Engineering Design." The project was conducted during 1993-96 under DARPA's I3 (Intelligent Integration of Information) program by Professor Shamkant B. Navathe (P.I.), with Professors Edward Omiecinski and Ashok Goel as co-principal investigators. Several graduate students participated in the project including two who completed their Ph.D. dissertations - Dr. Aravindan Veerasamy and Dr. Jeff Pittges. Professor Leo Mark also participated as an advisor of Jeff Pittges.

The project was conducted keeping in mind the broad aim of the I3 program to create enabling mediator technology by which future large scale applications involving data from a variety of sources can be supported. The "intelligence" during the processing of data and knowledge was considered by incorporating meta data, rules and constraints as a part of the information bases. We used design of engineering devices as a sample application, partly because it provided an appropriate environment to study integration of data and for transforming the needs of an intelligent front-end tool already under development. This tool named KRITIK was enhanced so that its decision alternatives may be enriched by extracting information from database backends.

The project produced several prototype systems: (i) A HIPED testbed which incorporated a deductive database engine called CORAL, and worked with the KRITIK as a front end, (ii) a tool to support free-form queries against text database, incorporating user feedback and visualization of results, and (iii) an explanation interface in conjunction with the KRITIK3 interactive design tool.

There were four broad objectives of this research program which are described in the four parts of this report:

- 1. Integration:** Creation of a uniform way of accessing heterogeneous data sources by using a rule based approach that gives a flexibility to the user to express correspondences among existing databases. Integration of data and knowledge is achieved by linking front-end tools with reasoning capability to back-end databases with query processing capability.
- 2. Query Formulation and Refinement for Text Data:** Investigation and validation of an approach that allows users to put in free-form requests for

data from a document database and improves their retrieval productivity by appropriate mechanisms of feedback through visualization and user interface design.

3. Query Optimization based on Semantics: Improving the efficiency of query processing by exploring semantics of data, specifically in the form of constraints at the instance level. A secondary problem of the efficient management of such constraints was thoroughly investigated.

4. Improving the end user's understanding of query results: This is accomplished by providing appropriate explanations. This work was done so as to improve an existing engineering device design tool. - KRITIK 2.

The present research can be extended in several directions: determination of appropriateness of an existing information source, using thesauri and ontologies to capture domain knowledge during query processing, automated knowledge acquisition from existing sources, dealing with external knowledge sources during query processing, etc.

This work has developed many techniques and identified many open problems that are summarized in Section 3 and 4 of the report. By addressing several problems related to information integration, query formulation, processing and integration, we have contributed to the overall goals of the DARPA's I3 program.

SECTION 2.

SUMMARIES OF RESEARCH AND PUBLISHED PAPERS

PART I

**HIPED: HETEROGENEOUS INTELLIGENT PROCESSING FOR
ENGINEERING DESIGN**

PART II

**VISUALIZATION AND USER INTERFACE TECHNIQUES FOR
INFORMATION RETRIEVAL**

PART III

**METADATA MANAGEMENT FOR INTELLIGENT QUERY
PROCESSING**

PART IV

**DEVELOPMENT OF AN INTELLIGENT INTERACTIVE TOOL
FOR ENGINEERING DESIGN**

PART I

**HIPED: HETEROGENEOUS
INTELLIGENT PROCESSING
FOR ENGINEERING DESIGN**

PART I: HIPED - HETEROGENEOUS INTELLIGENT PROCESSING FOR ENGINEERING DESIGN.

Our main objective in this work was to develop an approach for supporting large scale engineering design activities that need access to heterogeneous database sources. We were particularly interested in developing a mediator which utilizes meta-knowledge of the underlying information sources to aid a user in browsing the underlying data or help a system or a user in retrieving specific relevant information.

OBJECTIVE 1 : Support Of Facilities For Accessing Heterogeneous Data And Knowledge Sources:

The mediator we have designed provides the following capabilities:

A. A uniform access method and view of any database/knowledge base system with relevant information regardless of the design of the individual information system. We have taken engineering design as the application domain.

The Engineering data is thought to be made up of various "Prototypes". Each Prototype has various "Properties". Each Property takes up some "Value" for every Prototype. We can compare the Values of various properties using the relations : =, <, >, <=, >=, <> etc. Thus any query can be represented as,

(Prototype <proto_name>) (Property <prop_name>)
(Value <value>) (Relation <rel>)

Paper [1.2] elaborates on this approach.

B. Metadata query facilities allowing the design system to determine relevant information about component parameters, previous design specifications, device function descriptions, etc. The data is organized at two levels. (1) the metadata repository: consisting of information about various databases and tables in them and (2) the actual data: which is distributed in various heterogeneous databases. This organization reduces the data to be dealt with at the first level to get to the appropriate database(s) and table(s). It also allows heterogeneity in the various databases involved. Our initial proposal for metadata management was described in paper [1-1]. It has been further refined in [1. 2]

The metadata is stored in the form of CORAL facts and rules. CORAL is a deductive database system which stores data as facts and rules, and allows for that data to be queried. It is public domain software developed at the

University of Wisconsin. By using CORAL the mediator can decide which database(s) and table(s) are useful in answering any given query. In particular, CORAL is used in deriving relationships like equivalence between attributes, between tables and databases. Any creation, deletion or modification of a table results in a change in the metadata repository. This dynamic behavior can be easily captured by CORAL. In essence, CORAL provides us with the facility for database integration through the facts and rules specified about tables and databases. However, this integration can be considered implicit rather than explicit since no global conceptual schema is explicitly formed.

C. Data querying facilities allowing the design system to retrieve the actual data regardless of location (local vs. remote) and data organization (relational, knowledge-base, rule system, etc). In addition, this process is transparent to the engineering design tool.

The actual data is distributed across various tables in various (possibly heterogeneous) databases. Each database provides with its access methods (e.g. an SQL engine) so that the data can be accessed by the query. In the prototype implementation, we used the Oracle relational database system to store the actual data.

As mentioned earlier, the metadata is stored in CORAL as facts and rules. For example, the tables belonging to a given database is represented by specific CORAL facts, Facts which specify the equivalence of attributes in different tables and databases are also included as well as other facts.

When a query is submitted to the system, all the tables that would generate a meaningful result for it are found. We use the C++ interface of CORAL for the purpose. The metadata (represented in CORAL facts and rules) is consulted in making these decisions. Once we identify the database(s) and table(s) that would satisfy query requirements, we construct and route a corresponding query (e.g., an SQL query) to each of them. Detailed examples are provided in [1.2] and [1.4].

In our current prototype, the CORAL/C++ program (which deals with the metadata and translates the original query) creates a file of SQL queries for the particular database. The file of SQL queries is processed by a pro*C program (i.e., a C language program which makes SQL calls to the Oracle relational database system). The pro*C program handles dynamic SQL queries (i.e., it parses the input query, connects to the Oracle database, submits the query for execution, receives the output a tuple at a time, formats the output for display and then disconnects from Oracle).

The result is given back to the user/HIPED front end.

OBJECTIVE 2: Knowledge Based System Integration:

A second objective in our HIPED work, which constituted a bulk of our effort as far as the faculty involvement was concerned - between Professors Navathe, Goel and Omiecinski was to use this as a vehicle for research in the development of large-scale high performance knowledge based systems. During the course of our project, we ended up addressing the following difficult issues:

- a) How to tie intelligent front ends to the (unintelligent) back end database systems which acts a sources of information.
- b) How to develop a dual approach that deals not only with data integration, but with method or process integration at the same time.

Papers [1.3] and [1.5] capture the essence of our results in this area. Figure 1 in [1.3] shows our dual approach to database and knowledge base integration where new data requests originate from the knowledge systems when there is a need to supply new information. These requests then go to the "global request broker" for further processing by consulting the meta-data repository.

In terms of our own work in this area in the context of HIPED, a team of students took a first pass at integrating KRITIK3, a knowledge based front end with a relational database. Tools like KQML and LIM and IDI were used for communication and access between the front end and the back end. This activity is documented in [1.1]. Two areas of our work related to "intelligent processing" in heterogeneous environments are outlined below:

A. Explanation in Heterogeneous Knowledge Based Systems

In the HIPED project, we investigated three issues in designing transparent knowledge systems: how to explain and illustrate the system's reasoning, how to explain and justify its results, and how to enable the user to navigate and browse its knowledge base. Our approach is to endow knowledge systems with meta-models of the system's knowledge and reasoning. An interactive design and learning environment called Interactive Kritik - KRITIK3 has been developed. Part IV of this report documents the work on generating explanations for the designer or a design student in a learning environment.

B. Knowledge Compilation In Query Answering

A major issue in querying large-scale heterogeneous distributed information sources is how to efficiently retrieve an answer to the query. During 1995, we developed a new case-based approach to this problem. In our approach, retrievals are compiled into meta-cases, where a meta-case is a triplet consisting of a [query, answer, trace]. The trace in a meta-case refers to the

trace of retrieving the answer to the past query. If the new query is identical to the old one, then, like with a meta-rule, the meta-case provides the needed answer. In addition, if the new query is similar to the old one, then the trace in the meta-case points to the neighborhood in a specific information source that needs to be searched for the answer.

PUBLICATIONS (PART1):

[1.1]." Towards Intelligent Integration of Heterogeneous Information Sources ," Shamkant B. Navathe and Michael J. Donahoo.. In *Proceedings of the 6th International Workshop on Database Re-engineering and Interoperability*, Computer Society of Hong Kong, March 1995.

[1.2]. " Rule Based Database Integration in HIPED : Heterogeneous Intelligent Processing in Engineering Design ", Shamkant B. Navathe, Sameer Mahajan, Edward Omiecinski . In *Proceedings of International Symposium on Cooperative Database Systems for Advanced Applications*, World Scientific Press, 1996.

[1.3]" From Data to Knowledge: Method-Specific Transformations," Michael J. Donahoo, J. William Murdock, Ashok K. Goel, Shamkant B. Navathe, Edward Omeicinski, *Proceedings of the 10th International Symposium on Methodologies for Intelligent Systems*, Charlotte, North Carolina, October 15-18, 1997, (Z. Ras, Ed.), Springer Verlag, 1997.

[1.4] "Integrating Heterogeneous Databases for Engineering Design," Sameer Mahajan and Shamkant B. Navathe, Working Paper, College of Computing, Georgia Institute of Technology, December 1996.

[1.5]"Method-Specific Knowledge Transformations Towards ," J. William Murdock, Michael J. Donahoo, Ashok K. Goel, Shamkant B. Navathe, to appear in *Proceedings of the International Conference on A.I. Applications in Design*, Lisbon, Portugal, August 1998.

Towards Intelligent Integration of Heterogeneous Information Sources*

Shamkant B. Navathe

Michael J. Donahoo

College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280
{sham,mjd} @ cc.gatech.edu

Abstract

Current methodologies for information integration are inadequate for solving the problem of integration of large scale, distributed information sources (e.g. databases, free-form text, simulation etc). The existing approaches are either too restrictive and complicated as in the "federated" (global model) approach or do not provide the necessary functionality as in the "multidatabase" approach. We propose a hybrid approach combining the advantages of both the federated and multidatabase techniques which we believe provides the most feasible avenue for large scale integration. Under our architecture, the individual data site administrators provide an *augmented export schema* specifying knowledge about the sources of data (where data exists), their structure (underlying data model or file structure), their content (what data exists), and their relationships (how the data relates to other information in its domain). The augmented export schema from each information source provides an intelligent agent, called the "mediator," knowledge which can be used to infer information on some of the existing inter-system relationships. This knowledge can then be used to generate a partially integrated, global view of the data.

1 Introduction

Much of the research in database interoperability has focused on two extremes: multidatabase and federated systems. Multidatabase [Lit90, Spe88] systems provide a uniform access language to a set of database systems. While this is a necessary first step in solving the problems of heterogeneity, it places most of the integration responsibility on the user which may be unacceptable. Federated systems [She90] propose to create a global view of the underlying systems making the heterogeneity completely transparent to the user. While this approach is enticing, the complexity of constructing a global schema for large scale integration makes this approach infeasible because it requires an administrator who understands the semantics of all underlying systems and can resolve all inter-system schematic conflicts [Bat86]. In addition, the maintenance of a global schema in the face of addition/deletion of systems is difficult.

A better approach to interoperability involves the combination of techniques of reasoning and learning with techniques of data modeling and access to provide a partially integrated, global view. To accomplish this, the administrator of each underlying system presents a semantic description (augmented export schema) of their information to the "mediator." This augmented export schema may be as simple as the typical export schema or as detailed as a knowledge-based data description of the data, its relationships, and the system's domain. A knowledge-base system, such as Loom [Bri94], provides the capability to represent knowledge about the underlying information repositories and to make inferences as to the relationships among the various autonomous systems and generalizations concerning the information in each system. We have previously demonstrated that classification hierarchies can be effectively used to carry out integration of schemas [Sav91]. In this paper, we

*To appear in Proceedings of 6th International Hong Kong Computer Society Database Workshop, Hong Kong, February 1995

review the goals and strategy of the project HIPED, Heterogeneous Information Processing for Engineering Design, which we are currently pursuing at the Georgia Institute of Technology.

2 Related Work

Earlier work in integration provides the motivation and framework for our efforts. Batini et al. [Bat86] detail the problems of schema integration and provide a methodology for comparison of proposed solutions. Unlike many earlier integration efforts, we do not limit ourselves strictly to integration of databases. Instead, we focus on the integration of *information sources* including databases, free-form text, hypertext, etc. One possible method of dealing with this wide variety of information is to use Stanford's Object Exchange Model (OEM)[Pap94] which allows information exchange via *self-described* objects[Mar85] between different types of information sources. We propose to adapt the mediator paradigm[Pap94, Wei92, Wei93, Are94] to perform integration of the augmented export schemas. Integration of heterogeneous information sources requires a semantically rich data model. Earlier work has shown that the CANDIDE[Bec89, Nav91] model provides unique integration capabilities not found in traditional models. One major feature of the CANDIDE model is its ability to compute class-subclass relationships even among classes from dissimilar systems by subsumption from class relationship information[Sav91, She93, Wha93, Bra85]. Work with classification in the object-oriented model has produced similar results[Nav95, Are]. A variety of such systems supporting description logics are surveyed in [Bor94].

3 Approach

Our main objective is to build and demonstrate an intelligent interface to a set of (possibly autonomous) information sources including structured databases, knowledge bases, and unstructured data. Figure 1 shows our proposed architecture. The parenthetical references are made to applications developed under the ARPA I3 Initiative. KQML (Knowledge Query and Manipulation Language)[Cha92] allows remote access to knowledge/data bases. LIM (Loom Interface Module)[Par93b] allows import of external database information into Loom data structures. IDI (Intelligent Database Interface)[Par93a] is a common access language to several commercial database systems.

The approach we have selected involves development of an Engineering Design Mediator (EDM) which utilizes meta-knowledge of the underlying information to aid a user in "browsing" the data for relevant information sources and to make informed decisions about a plan for retrieving the appropriate data. To demonstrate this technology, we intend to augment the capabilities of both an autonomous (KRITIK2) and an interactive (Canah-Chab[Goe93]) device design system by providing a mediated interface between the design system and a collection of data/knowledge based systems (D/KBS). The mediator will be responsible for processing queries from the device design systems by determining where relevant data is, sending the appropriate query to the information site, performing the appropriate translations on the data, and returning the data to the design system. The design of the mediator is predicated on the following design goals:

1. Autonomy of the remote systems. Additionally, the remote systems should not be required to perform any functions outside of those defined for the internetwork connecting the system to the mediator.
2. Meta-data query facilities which allow the design system to determine relevant information about component parameters, previous design specifications, device function descriptions, etc. The mediator may also take an active role in helping the design tool determine what information may be helpful (e.g. by use of a thesaurus, domain concept hierarchy, etc).
3. Separation of concerns of the device design system from the query system. This will facilitate reuse of the mediated query system for other intelligent tasks such as planning.

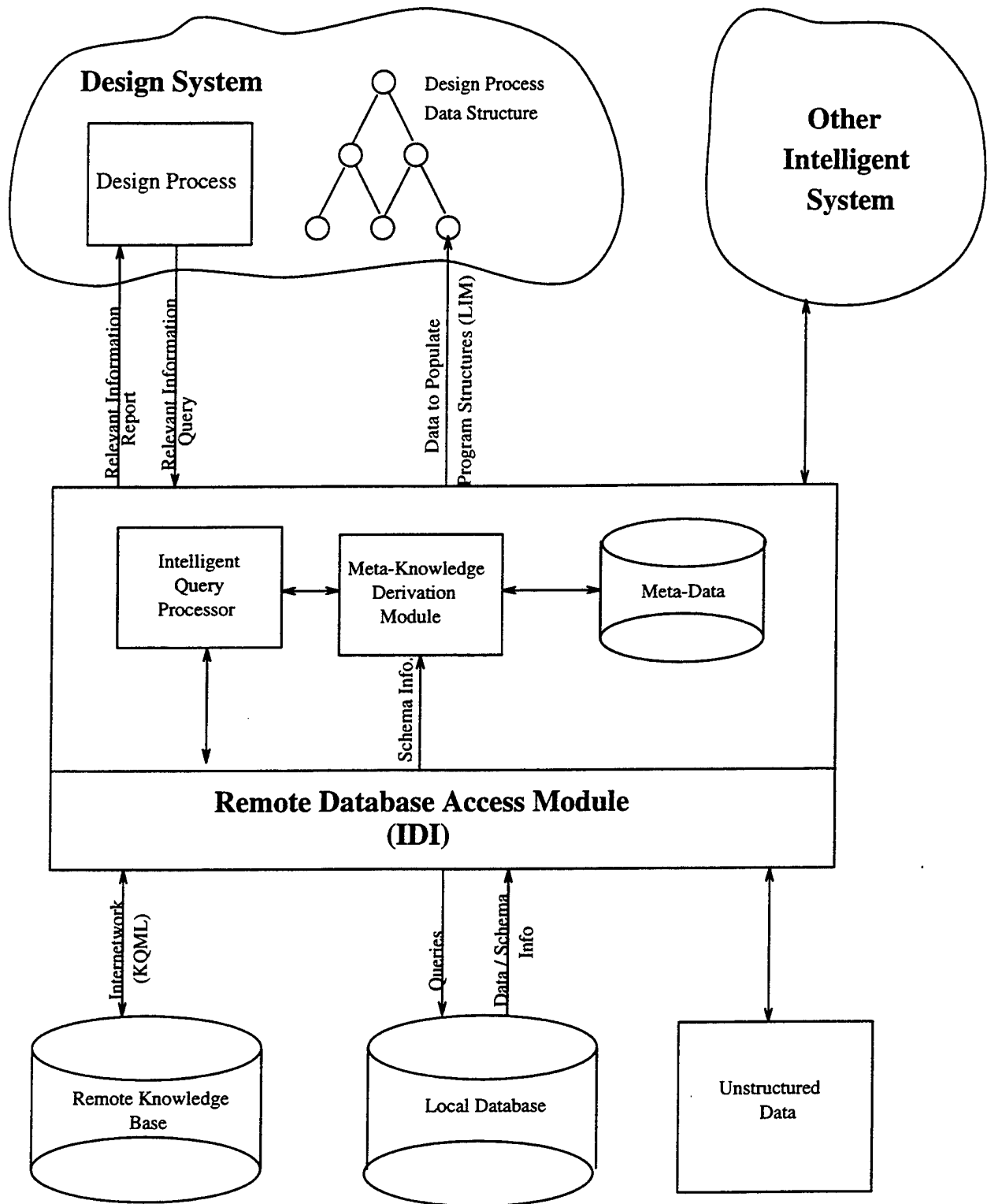


Fig. 1: Proposed Architecture for the Engineering Design Mediator (EDM)

4. Data location (remote vs. local) and data organization (relational, knowledge base, text, etc) transparency.
5. Easy import of external D/KBS information into existing design system data structures minimizing the required changes to the device design system.

These constraints are designed to facilitate reuse of the mediator and to make the use of the system as transparent to intelligent applications as possible. Figure 2 presents an example query processing scenario.

4 Ongoing Research

Research is currently under way in the following areas to facilitate construction of a prototype query system which can be integrated with the device design system:

- Selection and development of the appropriate export data model to represent the data stored at each information source.
- Construction of an export knowledge model whereby information source administrators can express the relationships between their data and real world domain concepts. This in combination with the export data model will define the *augmented export schema*.
- Development of techniques for providing integration of the schemas of information sources into a partially integrated, global schema.
- Determination of optimization techniques for querying the remote information sources. Since the information sources may be interconnected with a WAN, a query processing bottleneck may arise with frequent remote data transmission.
- Provision of a query interface which aids the user in deriving the best answer to a query. Since no completely integrated schema exists and the user does not know what information is available, a query processor is required to guide users to the desired information.
- Capability of inferencing intersource knowledge from the augmented export schemas specifically concerning the relationships between information source entities.
- Ability to learn new, relevant knowledge about information sources based on user interaction.

5 Future Direction

Our initial focus is on providing access of integrated information to intelligent device design systems, but many other applications of this technology exist. With the advent of internetworks which connect thousands of computers all over the world, an explosion has resulted of the available data, both unstructured (text, graphical documents, audio, video, program sources) and structured (under DBMS control), accessible to hundreds of thousands of users. It would be difficult, if not impossible, to integrate all these sites with the current heterogeneous database techniques especially since most sites will not all be willing to provide services beyond those defined by the internetwork. Many query applications already exist for the Internet. WAIS servers provide keyword access to documents; however these documents must be under the control of a WAIS server. Gopher allows sites to setup directories of information that users can browse, but the information can only be accessed in the organization defined by the site manager. Archie provides a keyword query interface to find source code, but the keywords only work on the name of the source file (the user cannot ask for a program that performs some function, X; instead they must find the name of a program that performs X and search for it by name. World Wide Web (WWW) provides a nice interface

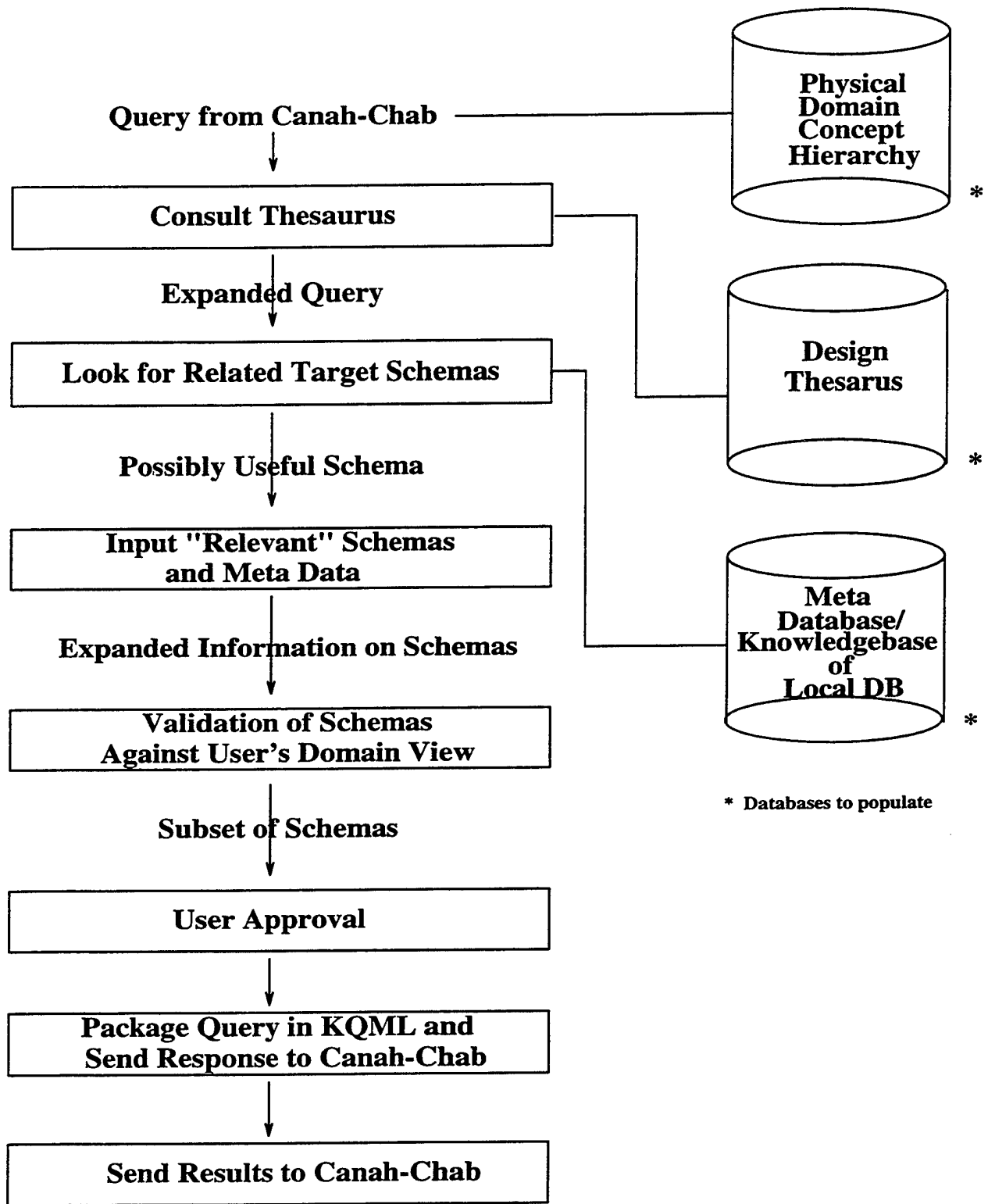


Fig. 2: Query Processing Scenario in the EDM

to information organized by site managers (similar to gopher), but users suffer from the "hypertext navigation problem" which creates difficulties in locating specific information and keeping track of where they are in the web of hypertext documents over time.

Several problems exist for the tools mentioned above. First, the tools access a particular type of data (e.g. Archie only finds source code). If a manual exists for a particular application whose source code is found by Archie, the user is not informed. Second, the tools lack relativism because the users must access the data in the manner dictated by the site manager (e.g. in WWW the data is explicitly organized by hyperlinks). Third, some of the applications require a particular site organization (e.g. Gopher requires a specific directory structure). If a site has information but no desire to organize it, a gopher search may not find the relevant information at that site. Fourth, the query processors provide little organization to the data (e.g. Archie does not organize its source code references by application type, instead all applications with a substring match on the query are returned). For these reasons, the Internet environment provides a true testbed for large scale, heterogeneous information source integration.

We propose a query processing application which, using the native internet capabilities, provides a single interface for accessing all types of data regardless of source or format. The following list proposes some of the necessary extensions to the EDM:

- The system should perform automated "net surfing" to create an intelligent index of each data store's information. The intelligence of the index lies in the ability to discern between types of data (audio, text, source, etc), utilize an indexing methodology tailored to the particular data type (e.g. organize keywords of a text document by the document section), and facilitate determination of an object's relevance for a query based on the knowledge of the user's interests and technical expertise. This should require no a priori knowledge of the individual data site organization. Work is being done at the Georgia Institute of Technology in intelligent text document processing and work has been done at IBM Almaden Research Center in file classification[Vee95a]. Extensive work has been done on parsers for the various document types (e.g. html, LaTeX) on the Internet.
- The problem of data overload may result from this large scale integration. Our query processor should utilize user profiles so that only data of specific relevance and technical difficulty will be derived. Unfortunately, the user profile method of data overload reduction may eliminate relevant documents. To deal with this problem, the user needs feedback from the query processor in the form of a description of what information is/is not being considered and an explanation of why. Work in explanation is part of the Canah-Chab System[Goe93].
- Keyword searches should not be limited by the vocabulary of the query; instead, a thesaurus should be used to consider synonyms. This may result in synonym overload so user profiles should also be used in pruning the list of synonyms.
- The user is assumed to be "browsing" the available information; therefore, the query interface should provide reformulation capabilities. Reformulation techniques include iterative query alteration and positive/negative feedback from the user[Vee95b].
- The system should attempt automated knowledge acquisition to provide a better understanding of indexed objects and to find other available data stores. The following list orders levels of object knowledge in ascending complexity:

ID Knowledge - System only knows site assigned ID of object (e.g. filename)

Content Knowledge - System knows information about object content (e.g. keywords for text)

Description Knowledge - System knows content knowledge and an external specification of the object.

Interrelational Knowledge - System knows all of the above and interobject relationships (e.g. papers about cancer research grouped together).

- The system should be extensible with respect to “plugging-in” different types of data indexing components and user profiles. Additionally, the system should transparently handle adding/subtracting participating sites. Utilities already exist for component indexing including parsers for various document types, image recognition utilities, etc.
- Different server systems should be able to exchange information and knowledge. Work in KQML at the University of Maryland facilitates knowledge interchange even with differing ontologies[Cha92].
- Objects must be described in terms of a nested model. For example, a document may be composed of sections which are composed of text, subsections, and graphics. Stanford’s Object Exchange Model (OEM) provides “self-describing,” nested objects[Pap94].
- The distributed control of the system leads to problems of object identity. For example, identical application source code may reside in multiple locations; therefore, the system should attempt to provide object identity to facilitate replicated object identification. Additionally, object versioning will allow the system to keep track of more recent versions of a retrieved object. A primitive form of object identification is supported in Stanford’s OEM project [Pap94].
- External knowledge sources should be used to learn about objects in the system. For example, the query processor could inspect newsgroups or look at the manner in which objects are used in WWW to acquire knowledge about the objects and their relationships. Primitive forms of natural language understanding and concept derivation techniques may be used.
- Use of existing query systems should be considered (e.g. use WAIS server to augment search).
- Special consideration should be given to optimization including reuse of retrieved data[Don93].

6 Conclusion

We have presented a framework for research in the area of intelligent, large scale integration of information sources. Clearly, much more work needs to be done before any of the detailed functionality can be implemented. We believe that much of the research into the necessary technology has begun, and the main task lies in tailoring these technologies to the needs of large scale integration and applying them in a prototype environment. We intend to further study the concepts presented above in order to develop a flexible and extensible scheme for integrating information from heterogeneous sources. Although we wish to experiment by applying our research in the area of augmenting intelligent device design in engineering, the applicability of this technology obviously extends beyond the engineering domain.

References

- [Are] Yigal Arens, Chin Chee, Chun-Nan Hsu, and Craig A. Knoblock. Retrieving and integration data from multiple information sources. To appear in International Journal on Intelligent and Cooperative Information Systems.
- [Are94] Yigal Arens, Chin Chee, Chun-Nan Hsu, Hoh In, and Craig A. Knoblock. Query processing in an information mediator. ISI Technical Report, 1994.

- [Bat86] C. Batini, M. Lenzenini, and S. B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):325–364, Dec. 1986.
- [Bec89] Howard W. Beck, Sunit K. Gala, and Shamkant B. Navathe. Classification as a query processing technique in the CANDIDE semantic data model. In *1989 IEEE Conference on Data Engineering*, pages 572–581. IEEE, 1989.
- [Bor94] Alexander Borgida. Description logics in data management. Technical report, Rutgers University, July 1994.
- [Bra85] R. Brachman and G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.
- [Bri94] David Brill. *Loom Reference Manual (Version 2.0)*. ISX Corp, October 1994.
- [Cha92] Hans Chalupsky, Tim Finin, Rich Fritzson, Don McKay, Stu Shapiro, and Gio Weiderhold. An overview of KQML: A knowledge query and manipulation language. Technical report, KQML Advisory Group, April 1992.
- [Don93] Michael J. Donahoo. *Integration of Information in Heterogeneous Library Information Systems*. Master's thesis, Baylor University, May 1993.
- [Goe93] Ashok K. Goel, Andres Garza, Nathalie Grue, M. Recker, and T. Govindaraj. Beyond domain knowledge: Towards a computing environment for the learning of design strategies and skills. Technical report, College of Computing, Georgia Tech, 1993.
- [Lit90] Witold Litwin, Leo Mark, and Nick Roussopoulos. Interoperability of multiple autonomous databases. *ACM Computing Surveys*, 22(3):267–293, September 1990.
- [Mar85] Leo Mark. *Self-Describing Database Systems - Formalization and Realization*. PhD thesis, Computer Science Department, University of Maryland, 1985.
- [Nav91] Shamkant Navathe, Sunit K. Gala, and Seong Geum. Application of the CANDIDE semantic data model for federations of information bases. In *Invited paper, COMAD '91*, Bombay, India, December 1991.
- [Nav95] Shamkant B. Navathe and Ashoka N. Savasere. A practical schema integration facility using an object-oriented model. To be published in *Object Oriented Multidatabase Systems: A Solution for Advanced Applications* (O. Bukhres and A. Elmagarmid, eds), Prentice-Hall, January 1995.
- [Pap94] Yannis Papakonstantinou, Hector Garcia-Molina, and Jennifer Widom. Object exchange across heterogeneous information sources. Stanford University, Department of Computer Science, Technical Report, 1994.
- [Par93a] Paramax System Corporation. *Computer System Operator's Manual for the Cache-Based Intelligent Data Interface of the Intelligent Database Interface*, revision 2.3 edition, Feb. 1993.
- [Par93b] Paramax Systems Corporation. *Software Design Document for the Loom Interface Module (LIM) of the Cache-Based Intelligent Database Interface*, revision 2.0 edition, Jan. 1993.
- [Sav91] Ashoka Savasere, Amit Sheth, Sunit Gala, Shamkant Navathe, and Howard Marcus. On applying classification to schema integration. In *First International Workshop on Interoperability in Multidatabase Systems*, pages 258–261. IEEE Computer Society, IEEE Computer Society Press, April 1991.

- [She90] Amit P. Sheth and James A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183-236, September 1990.
- [She93] Amit P. Sheth, Sunit K. Gala, and Shamkant B. Navathe. On automatic reasoning for schema integration. *International Journal of Intelligent and Cooperative Information Systems*, 2(1):23-50, 1993.
- [Spe88] R. Speth, editor. *Global View Definition and Multidatabase Languages - Two Approaches to Database Integration*. Amsterdam: Holland, April 1988.
- [Vee95a] Aravindan Veerasamy, Scott Hudson, and Shamkant Navathe. Visual interface for textual information retrieval systems. To appear in Proceedings of IFIP 2.6 Third Working Conference on Visual Database Systems, Lausanne, Switzerland, Springer Verlag, March 1995.
- [Vee95b] Aravindan Veerasamy and Shamkant Navathe. Querying, navigating and visualizing an online library catalog. Submitted for Publication, January 1995.
- [Wei92] Gio Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, pages 38-49, March 1992.
- [Wei93] Gio Wiederhold. Intelligent integration of information. In Arie Segev, editor, *ACM SIGMOD International Conference*, volume 22, pages 434-437. ACM, ACM Press, June 1993.
- [Wha93] Whan-Kyu Whang, Sharma Chakravathy, and Shamkant B. Navathe. Heterogeneous databases: Toward merging and querying component schema. *Computing Systems*, 6(3), August 1993. (a Univ. of California Press publication).

Rule Based Database Integration in HIPED : Heterogeneous Intelligent Processing in Engineering Design

Shamkant B. Navathe

Sameer Mahajan

Edward Omiecinski

College of Computing,
Georgia Institute of Technology,
Atlanta, GA 30332-0280, USA.
{sham,sameer,edwardo}@cc.gatech.edu

Abstract

In this paper¹ we describe one aspect of our research in the project called HIPED, which addressed the problem of performing design of engineering devices by accessing heterogeneous databases. The front end of the HIPED system consisted of interactive KRITIK, a multimodal reasoning system that combined case based and model based reasoning to solve a design problem. This paper focuses on the backend processing where five types of queries received from the front end are evaluated by mapping them appropriately using the "facts" about the schemas of the underlying databases and "rules" that establish the correspondance among the data in these databases in terms of relationships such as equivalence, overlap and set containment. The uniqueness of our approach stems from the fact that the mapping process is very forgiving in that the query received from the front end is evaluated with respect to a large number of possibilities. These possibilities are encoded in the form of rules that consider various ways in which the tokens in the given query may match relation names, attribute names, or values in the underlying tables. The approach has been implemented using CORAL deductive database system as the rule processing engine.

1 Introduction

Heterogeneity of databases is becoming a necessary factor to contend with in the design of new applications because of the proliferation of database management systems that used diverse data models over the last three decades. Among widely implemented data models we have the hierarchical, network, relational and object oriented data models. A large body of work exists that deals with the mapping of these models among one another (e.g. see the mapping of models using the entity relationship model as an intermediate model in [1] [3]. While vendors are also providing middleware solutions to draw data from these legacy systems, the semantic problems of resolving, naming, scale, structure etc. that were pointed out several years ago [5] [6] still remain. The purpose of the present research was to develop a technique to

dealing with the semantic differences in data by taking a flexible rule based approach. Another goal of the project was to tie a set of heterogeneous databases to an "intelligent front end application" which would make requests for data without any knowledge of the schemas of the target databases. To limit the degree of difficulty we assume that we are dealing with data in relational databases only. This assumption is reasonable in the sense that the data is coming from a hierarchical or a network DBMS, we can first convert the schema to a relational one before treating it for purposes of integration.

The database integration problem we discuss here is couched in the context of engineering design which, like any other design application, relies on extracting data from existing databases containing material data, components, existing designs etc. The exact context and the application scenario will be explained in the next section.

We assume that relevant data for the design application is stored in relations (tables) whose schemas are available at "design time" to construct a rule-base. It is conceivable that to support large scale engineering designs, data from a variety of databases, i.e., from multiple schemas would be required. To facilitate integration of data among these databases we assume that the "correspondances", i.e., the similarities and differences among the (meaning of) attributes is encoded in the form of rules. Furthermore, for our application context, the front end of HIPED issues certain queries looking for relevant design information. We show in this paper how a query may have several interpretations, each one of which is encoded in the form of rules again.

Because of these two kinds of rules involved in the integration approach we have termed our approach a rule based approach to database integration. The present approach is an improvement over previous approaches where we handled integration by using the correspondance information to derive the process [2] [6] [7] [8].

2 Application Context

In this section we will provide the overall architecture of the HIPED system and point out the need for heterogeneous database processing which will be de-

¹To appear in the Proceedings of International Symposium on Cooperative Database Systems for Advanced Applications, Heian Shrine, Kyoto, Japan, World Scientific Press, 1996.

scribed and illustrated in the next two sections.

2.1 Overall Architecture of HIPED

Our main objective in the HIPED project is to build and demonstrate an intelligent interface to a set of (possibly autonomous) information sources including structured databases, knowledge bases, and unstructured data. The approach we have selected involves the development of a mediator which utilizes meta-knowledge of the underlying information stores to aid a user in browsing data or to enable an application front-end to retrieve specific relevant information for problem solving.

The overall architecture of HIPED is described in Figure 1. We look at only the "Database Backend" in this paper. The data is organized at two levels namely, (1) the metadata repository : consisting of information about various databases and tables in them and (2) the actual data which is distributed in various heterogeneous databases. This organization reduces the data to be dealt with at the first level to get to the appropriate database(s) and table(s). It also allows heterogeneity in the various databases involved. The Querying Interface is as described in section 3.1. The "data" together with its "wrapper" forms a database system. "Wrapper" simply defines the access methods to the data for reading purposes. A wrapper can be designed for each target database management system. A user query would be translated into the corresponding query, as understood by the corresponding "wrapper", for each of the relevant tables. This query would then be routed to the corresponding database, that contains this table. The metadata repository is consulted in determining these relevant tables and finding the corresponding database. The user would get the result, obtained after running the query against the table through the concerned "Output Data" channel(s).

2.2 Interactive KRITIK Front End

We developed the HIPED architecture by assuming a frontend system called Interactive Kritik [4]. This system is a multimode reasoning system which works like a design assistant for the design of devices such as acid coolers, electrical devices. In its current form the system uses "hard-wired" knowledge in the form of LISP data structures. The goal was to extend the capability of interactive Kritik to make it scalable to real-life design problems by incorporating databases of relevant design data as the back end. We therefore abstracted different forms of generic query types which would be used as requests to the back end. By coupling an intelligent front end application to a set of heterogeneous databases, we can thus extend the scope of problem solving by a large measure. For engineering device design, the above front end generates a number of requests for data from the underlying design databases such as design prototypes, properties of devices and components, material data, design specifications and tolerances etc. For illustrative purposes we have chosen five generic types of queries that are most commonly presented by the front end. They will be explained in detail in the following section.

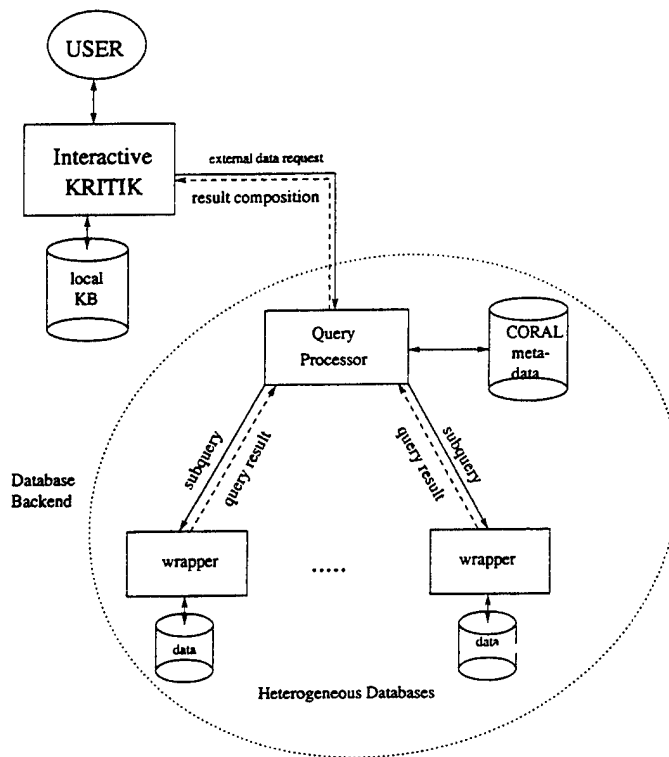


Figure 1: The High Level View of HIPED

3 Rule Based Approach to Database Integration

As explained earlier the main contribution of this research is the use of the two types of rules to accomplish access to the underlying heterogeneous information sources. The first set of rules deals with establishing various types of relationships among relation names and among attribute names across databases. The second set deals with the interpretation of queries from the front end so that various possible mappings to the interface of underlying target databases may be considered. We will explain both these types of rules when we discuss the generic queries and their mappings.

3.1 Five generic types of queries

The user is assumed to use this system as an Engineering Database for device design. Let us limit the application domain for illustrative purposes. We assume that during the design process, he would typically like to find components that satisfy his requirements (e.g. batteries with voltage rating higher than 10V and cheaper than \$10). Keeping this user's perspective in mind, the Engineering data is thought to be made up of various "Prototypes". Each Prototype has various "Properties". Each Property takes up some "Value" for every Prototype. We can compare the Values of various properties using the relations : ==, <, >, <=, >=, <> etc. The queries can be classified into the following five generic types,

1. (Prototype <proto_name>) : here the user is looking for all the prototypes identified by "proto_name". It is implicit that the user wants to see the various values for various properties (attributes) of these prototypes.
2. (Property <prop_name>) : the user is interested in all the prototypes having the specific Property identified by "prop_name". It is implicit that the user wants to see the values taken by this property for the various prototypes, that would be listed.
3. (Prototype <proto_name>)
(Property <prop_name>) : the user wants to see all the prototypes identified by "proto_name" and having property identified by "prop_name". It is implicit that the user also wants to see the corresponding value that the property takes for the particular prototype.
4. (Prototype <proto_name>)
(Property <prop_name>)
(Value <value>) (Rel-op <op>) : the user is interested in prototypes identified by "proto_name" having a property identified by "prop_name". In addition to this he wants only those prototypes for which the property takes a value which is related to the given "value" or a constant in the query by the operator "op" (i.e. it is equal to "value" or greater than "value" etc.)
5. (Property <prop_name>) (Value <value>)
(Rel-op <op>) : the user is interested in all the prototypes for which the property identified by "prop_name" takes a value which is related to the given "value" by the operator "op".

Data is distributed among various databases and various tables in each of those databases. The only assumption that we make about any database system is that it has an SQL access method. It is a reasonable assumption and is made to contain the complexity of the problem.

The system needs to find out which databases and which tables in these databases have the relevant data to answer a particular query. It then translates the query into a corresponding SQL query for every table. This SQL query is run against that table to get an answer. As we made an assumption of a uniform SQL interface to all the databases, we can simply translate a request for data into a set of SQL queries in each of these cases.

3.2 Rules for Interpretation of Queries

For better understanding of the following discussion, let us take up an example query. Let the four components of the query be,

(Prototype Battery) (Property Voltage)
(Value 10) (Relation ==).

As there can be various tables with different schema, we need to run this query with only those tables that might give meaningful results for the query. We can easily observe that any of "Prototype", "Battery", "Property" and "Voltage" can be a table or a column of a table. The "Battery" and "Voltage" can also be

values in the columns (e.g. those labeled as "Prototype" and "Property" respectively). Of course there are a lot of dependencies amongst these components - e.g. if "Prototype" is a table then "Battery" has to be a column of this table. On the other hand if there is a table called the "Battery", then we are looking for values in the column "voltage" or "volts" - so that the query would generate meaningful results with the table. Now we take up an example query for each of the five types listed above. For every query we list the possible interpretations according to our scheme.

1. (Prototype Battery). The user typically means that he wants all the batteries with their properties and their corresponding values. Hence we will have to run this query against all the tables which,
 - are equivalent to "Prototype Table" and have a column equivalent to "Battery" or
 - are equivalent to "Battery Table"
 - have a column equivalent to "Prototype" (and only the tuples with Prototype as "Battery" would be considered).

if and only if these tables have columns equivalent to "Property" and "Value" each.

2. (Property Voltage). The user is interested in listing all the Prototypes having "Voltage" as their one of the Properties. The Values of these Properties would also be significant from his standpoint. Hence we consider all the tables which,
 - are equivalent to "Property Table" and have a column equivalent to "Voltage" or
 - are equivalent to "Voltage Table"
 - have a column equivalent to "Property" (and only the tuples with Property as "Voltage" would be considered).

if and only if they have "Prototype" equivalent column.

3. (Prototype Battery) (Property Voltage). The user wants all the batteries with special interest in their voltages. Hence we will run the query against all the tables which,
 - are equivalent to "Prototype Table" and have "Battery", "Property" and "Value" equivalent columns and we would be interested only in the tuples having an entry of "Voltage" in the "Property" equivalent column or
 - are equivalent to "Prototype Table" and have "Battery", "Voltage" equivalent columns or
 - are equivalent to "Battery Table" and have "Property" and "Value" equivalent columns. We would be interested only in those tuples having Property "Voltage" or

- are equivalent to "Battery Table" and have a column equivalent to "Voltage" or
 - are equivalent to "Property Table" and have columns equivalent to "Voltage", "Prototype" and "Value". We would be interested in tuples with Prototype as "Battery".
 - are equivalent to "Property Table" and have "Voltage" and "Battery" equivalent columns.
 - are equivalent to "Voltage Tables" and have "Prototype" and "Value" equivalent columns. We would look for only tuples with Prototype as "Battery".
 - are equivalent to "Voltage Table" with "Battery" and "Value" equivalent columns.
 - have "Prototype" and "Property" equivalent columns as far as they have "Value" equivalent column. Only the tuples with Prototype as "Battery" and Property as "Voltage" would be considered.
4. (Prototype Battery) (Property Voltage) (Value 10) (Relation ==). Here the interest is indicated in all the batteries having Voltage as "10". The query can be run with all the tables as indicated as above with an added constraint that only those tuples which have an entry of "10" in the "Voltage" or "Value" column - whichever is applicable - (Note the table can have only one of these columns at a time) will be considered.
5. (Property Voltage) (Value 10) (Relation ==). All the Prototypes having voltage of "10" are being considered. Thus all the tables that,
- are equivalent to "Property Table" and have a column equivalent to "Voltage"
 - are equivalent to "Voltage Table" and have a column equivalent to "Value"
 - have "Property" and "Value" equivalent columns along with "Prototype" column. (only tuples with Property "Voltage" and Value "10" would be taken into consideration).

would be considered if and only if they have a column equivalent to "Prototype". All the tuples with "Voltage" or "Value" being 10 would be taken into account.

3.3 Rules to establish Data Correspondance

We need to relate various attributes and tables, within and across databases. The relationship could be of equivalence, subsumption, overlap, disjointness or containment. The relationship between attributes needs to be supplied by the schema developer. e.g. Attributes called "volt" and "voltage" in different tables are actually equivalent. The relationship between tables can either be supplied or can be deduced by the relationships of their individual attributes. A simple deduction rule can be that two tables are equivalent if all their attributes are equivalent.

4 Use of CORAL for rule representation and query processing

The metadata is stored in the form of CORAL [10] [11] facts and rules. CORAL is a deductive database system which stores data as facts and rules, and allows for that data to be queried. By using CORAL the mediator can decide which database(s) and table(s) are useful in answering any given query. In particular, CORAL is used in deriving relationships like equivalence; between attributes, tables and databases. Any creation, deletion or modification of a table results in a change in the metadata repository. This dynamic behavior can be easily captured by CORAL. In essence, CORAL provides us with the facility for database integration through the facts and rules specified about tables and databases. However, this integration can be considered implicit rather than explicit since no global conceptual schema is explicitly formed. Also the C++ interface provided by CORAL makes writing general purpose programs easy.

We explain the implementation with the help of an example. One more sample system for a single database environment is given in Table 5. Some sample input queries and the corresponding output SQL queries are shown in Tables 6 and 7 respectively.

4.1 A Simple Example

Consider the query,

(Prototype Battery) (Property Voltage).

Let us assume that there are two databases - db1 and db2. Let db1 have tables : Table 1 and Table 2. and

CompNo	Prototype	Property	Value
B101	Battery	Voltage	10
M101	Motor	Voltage	10
B110	Battery	Voltage	100
B111	Battery	Current	100

Table 1: "Components" Table in db1

let db2 have Table 3 and Table 4.

We observe that according to the discussion in section 3.2 only the tables in db1 would produce meaningful results with the query under consideration.

BatteryNo	Voltage
B101	15
B102	30

Table 2: "Battery" Table in db1

BatteryNo	Current
B101	15
B102	30
:	

Table 3: "Battery" Table in db2

BatteryNo	Supplier No
B101	4567
B102	4568
:	

Table 4: "Supplier" Table in db2

4.2 Schema Representation

It is stored as CORAL facts and rules. The advantage of such a storage is that we can utilize the strong deductive power of CORAL (e.g. deducing equivalence of attributes, equivalence of tables etc.). The various components of the repository are described below.

- First we list all the tables in all the databases as CORAL facts :

```
% For the first database, db1.
belongsTo(components,db1).
belongsTo(battery,db1).
```

```
% For the second database, db2.
belongsTo(battery,db2).
belongsTo(supplier,db2).
```

- Then we list attributes of individual tables as CORAL facts. The first argument of these predicates is the database name. It is so because the same table may have different attributes in different databases. e.g. the "battery" table in the two databases "db1" and "db2" as shown below.

```
% for db1
hasAttribs(db1,components,
  [compName,prototype,property,value]).
hasAttribs(db1,battery,[bName,voltage]).
```

```
% for db2
hasAttribs(db2,battery,[bName,current]).
hasAttribs(db2,supplier,[bName,sName]).
```

- We also need facts to list what attributes are equivalent. The equivalence of tables can be either given by facts or can be deduced by the rules

(e.g. two tables with equivalent attributes are equivalent). But we do not need them in this particular example.

- To find whether a table has a particular attribute in a given database we define a CORAL rule as,

```
module isAttrib.
export isAttrib(bff).
isAttrib (Db,Table,Attri) :-
    hasAttribs(Db,Table,Attribs),
    iselem (Attri,Attribs).
end_module.
```

% Module "iselem" is defined for the % sake of completeness.

```
module iselem.
export iselem(bb).
@pipelining+. % Solve in a
              % top-down fashion
```

```
iselem(X, [X|_]).
iselem(X, [_|Z]) :- iselem(X,Z).
end_module.
```

4.3 Sample Query Mapping Algorithm

The mapping of input requests into SQL queries is done according to the scheme suggested in section 3.2. We use the C++ interface of CORAL for this matter. In fact, an imperative interface (e.g. in C) would have been enough for the purpose. We check for the various conditions given in the scheme and generate the appropriate SQL queries for the existing tables. We run through the algorithm for the example query under consideration,

```
begin
For every "table" equivalent to
"prototype table"
  for every attribute equivalent
  to "battery", say attrib1
    for every attribute equivalent
    to "property", say attrib2
      if "table" has "attrib1"
      as well as "attrib2"
        for every attribute
        equivalent to "value",
        say attrib3
          if "table" has attrib3
            select the corresponding
            database and fire SQL query,
            SELECT * FROM table
            WHERE attrib2 == voltage or
            some equivalent value.
            goto next table
        for every attribute equivalent
        to "voltage", say attrib4
          if "table" has attrib4
            select the corresponding
            database and fire SQL query,
            SELECT * FROM table
```

```

For every 'table' equivalent to
'battery table'
  for every attribute equivalent
  to 'voltage', say attrib1
    if 'table' has attrib1
      select the corresponding
      database and fire SQL query,
      SELECT * FROM table
      goto next table
  for every attribute equivalent
  to 'property', say attrib2
    for every attribute equivalent
    to 'value', say attrib3
      if 'table' has attrib2
      and attrib3
        select the corresponding
        database and fire SQL query,
        SELECT * FROM table
        WHERE attrib2 == voltage or
        some equivalent value.

```

```

For every 'table' equivalent
to 'property table'
  for every attribute equivalent
  to 'voltage', say attrib1
    for every attribute equivalent
    to 'prototype', say attrib2
      if 'table' has 'attrib1'
      as well as 'attrib2'
        for every attribute
        equivalent to 'value',
        say attrib3
          if 'table' has attrib3
            select the corresponding
            database and fire SQL query,
            SELECT * FROM table
            WHERE attrib2 == battery or
            some equivalent value.
            goto next table
        for every attribute equivalent
        to 'battery', say attrib4
          if 'table' has attrib4
            select the corresponding
            database and fire SQL query,
            SELECT * FROM table

```

```

For every table equivalent to
'voltage table'
  for every attribute equivalent
  to 'battery', say attrib1
    if 'table' has attrib1
      select the corresponding
      database and fire SQL query,
      SELECT * FROM table
      goto next table
  for every attribute equivalent
  to 'prototype', say attrib2
    for every attribute equivalent
    to 'value', say attrib3
      if 'table' has attrib2
      and attrib3
        select the corresponding
        database and fire SQL query,

```

```

SELECT * FROM table
WHERE attrib2 == battery or
some equivalent value.

```

```

For every table having columns
equivalent to each of
prototype, property and value
  select the corresponding
  database and fire SQL query
  SELECT * FROM table
  WHERE prototype equivalent column
  == battery equivalent value
  AND
  property equivalent column
  == voltage equivalent value
end

```

4.4 The Result

Let us say that the wrapper of db1 can handle SQL queries. In that case we first select that database and then simply run a query,

```

SELECT *
FROM components
WHERE prototype == 'battery'
AND property == 'voltage'

```

against the first ("components") table in the database. We take similar actions for the other table in (possibly various) databases. The other query in this case would be,

```

SELECT *
FROM battery

```

again with the same database namely, db1. The result is presented to the user as displayed by the corresponding "wrapper".

5 Conclusions and Future Work

In this paper we illustrated the implementation of a rule-based database integration scheme by considering two types of rules: (1) Rules to establish the "correspondence" among underlying component databases and (2) Rules to interpret data requests in an "open-ended" fashion where no knowledge of the component database schemas is expected from the application front end. We also described an interface to heterogeneous databases in which a user may directly access the back end data by making use of the rules of data correspondance and an SQL-like syntax for the queries.

The system makes an assumption that all the databases involved provide an SQL interface. This condition can be relaxed. In this case we need to generate different queries, as understood by each of the databases involved. This work was predicated on the assumption that the data relevant to our application was stored in relational tables. An extension of the present work involves relaxing this assumption and illustrating the utility of the approach by actually providing wrappers for hierarchical and network databases and sequential files. That would establish

```

% CORAL facts
isTable(battery).
hasAttribs(battery,
  [bname,voltage,current,life]).

isTable(compTable).
hasAttribs(compTable,
  [no,prototype,property,value]).

isTable(dummy).
hasAttrib(dummy,[prototype,property]).

isTable(prototype).
hasAttribs(prototype,
  [motor,property,value]).

isTable(motor).
hasAttribs(motor,[property,value]).

isTable(property).
hasAttribs(property,
  [rps,prototype,value]).

isTable(rps).
hasAttribs(rps,[prototype,value]).

isTable(voltage).
hasAttribs(voltage,[battery,value]).

% CORAL rules
module isAttrib.
export isAttrib(ff).
isAttrib (X,Y) :- hasAttribs(X,Z),
                 iselem (Y,Z).
end_module.

```

Table 5: A Single Database System

```

prototype battery property voltage
prototype battery property current
prototype motor property rps
prototype sheet property size

```

Table 6: Sample Input Queries

```

+++++++ for the first data request ++++++
SELECT * FROM battery;
SELECT * FROM voltage;
SELECT * FROM compTable
WHERE prototype == battery
      AND property == voltage;
+++++++ for the second data request ++++++
SELECT * FROM battery;
SELECT * FROM compTable
WHERE prototype == battery
      AND property == current;
+++++++ for the third data request ++++++
SELECT * FROM prototype
WHERE property == rps;
SELECT * FROM motor
WHERE property == rps;
SELECT * FROM property
WHERE prototype == motor;
SELECT * FROM rps
WHERE prototype == motor;
SELECT * FROM compTable
WHERE prototype == motor
      AND property == rps;
+++++++ for the fourth data request ++++++
SELECT * FROM compTable
WHERE prototype == sheet
      AND property == size;

```

Table 7: The corresponding SQL queries

the practical utility of the approach in a significant way. The next step would be to work on a query optimization by introducing a stage after the query interpretation phase to evaluate possible orderings of sub queries and cross subquery reduction of redundant processing.

From the engineering design standpoint, the problem horizon can be extended to include additional types of design problems. The current implementation can be initially enhanced by considering additional types of design queries.

Currently only the individual tables are checked to see whether they provide satisfactory data to answer a particular query. But it is possible that two or more tables taken separately do not have enough information to answer a query. At the same time, when taken together (e.g. their join), they provide data to answer the query. Consider that there are two tables - which might be in the same database or in different databases - one with columns "Component Number" and "Prototype". The other with columns "Component Number" and "Voltage". Then neither of them provides enough information for the query,

(Prototype Battery) (Property Voltage)
 But their equijoin with the additional condition of "Prototype == Battery" for the tuples is of interest to us. The extended solution can exhaustively take care of all such cases.

In essence, the overall rule based approach appears promising in the context of Navathe's long standing

investigation of the database integration problem [5] [6] [7] [8] [9].

Acknowledgement

We would like to thank Ashok Goel and William Murdock for their work on Interactive KRITIK. The work crystallized the HIPED front end to our system. We are also grateful to Jeff Donahoo, Ashok Savasere and Byong-soo Jeong for initializing the implementation work. The idea of using CORAL as a deductive database system was generated by Ed Omiecinski. The implementation was completed by Siddharth Bajaj. Support from ARPA grant no. F33615-93-1-1338 is greatly appreciated. Prof. Navathe's work is also partially supported by Center of Excellence in Information Science, Clark Atlanta University, Contract No. OSP-93-09-400-002.

References

- [1] C. Batini, S. Ceri, and S.B. Navathe, **Conceptual Database Design: An Entity Relationship Approach**, Benjamin Cummings, August 1991, 470 pp.
- [2] C. Batini, M. Lenzerini and S.B. Navathe, A Comparative Analysis of Methodologies for Database Schema Integration, *ACM Computing Surveys*, 18, 4, December 1986, pp. 323-364.
- [3] R. Elmasri, S. Navathe, **Fundamentals of Database Systems**, Addison Wesley Computer Science, 2nd Edition, 1994.
- [4] Ashok Goel, Andres Gomez, Nathalie Grue, William Murdock, Margaret Recker, and T. Govindaraj. Design Explanations in Interactive Design Environments. In *Proc. Fourth International Conference on AI in Design*, Palo Alto, June 1996.
- [5] J. Larson, S. B. Navathe, and R. Elmasri A Theory of Attribute Equivalence in Databases with Application to Schema Integration, *IEEE Transactions on Software Engineering*, Vol. 15, No. 4, April 1989.
- [6] S.B. Navathe, R. Elmasri and J.A. Larson, Integrating User Views in Database Design, *IEEE Computer*, Vol. 19, No. 1, January 1986, pp. 50-62.
- [7] S. B. Navathe and A. Savasere, "A Practical Schema Integration Facility using an Object Oriented Approach," *Multidatabase Systems (A. Elmagarmid and O. Bukhres, Eds.)*, Prentice Hall, 1996.
- [8] S. Prabhakar, J. Srivastava, S.B. Navathe, et al., Federated Autonomous Databases: Project Overview, *Proceedings of the International Conference on Interoperability in Multidatabase Systems (IMS'93)*, Vienna, Austria, April 19-20, 1993.
- [9] A. Sheth, S.K. Gala, S.B. Navathe, *On Automatic Reasoning for Schema Integration*, *Int. Journal of Intelligent Co-operative Information Systems*, Vol. 2, No.1, March 1993.
- [10] R. Ramakrishnan, D. Srivastava and S. Sudarshan, CORAL: Control, Relations and Logic, *Proceedings of the International Conference on Very Large Databases*, 1992.
- [11] R. Ramakrishnan, D. Srivastava, S. Sudarshan and P. Seshadri, Implementation of the CORAL deductive database system, *Proceedings of the ACM SIGMOD Conference on Management of Data*, 1993.

From Data to Knowledge: Method-Specific Transformations *

Michael J. Donahoo, J. William Murdock, Ashok K. Goel, Shamkant Navathe,
and Edward Omiecinski

Georgia Institute of Technology
College of Computing
801 Atlantic Drive
Atlanta, Georgia 30332-0280, USA

Source: Proceedings of the 1997 International Symposium Symposium on Methodologies for Intelligent Systems.

Abstract. Generality and scale are important but difficult issues in knowledge engineering. At the root of the difficulty lie two hard questions: how to accumulate huge volumes of knowledge, and how to support heterogeneous knowledge and processing? One answer to the first question is to reuse legacy knowledge systems, integrate knowledge systems with legacy databases, and enable sharing of the databases by multiple knowledge systems. We present an architecture called HIPED for realizing this answer. HIPED converts the second question above into a new form: how to convert data accessed from a legacy database into a form appropriate to the processing method used in a legacy knowledge system? One answer to this reformed question is to use method-specific transformation of data into knowledge. We describe an experiment in which a legacy knowledge system called INTERACTIVE KRITIK is integrated with an ORACLE database using IDI as the communication tool. The experiment indicates the computational feasibility of method-specific data-to-knowledge transformations.

1 Motivations, Background, and Goals

Generality and scale have been important issues in knowledge systems research ever since the development of the first expert systems in the mid sixties. Yet, some thirty years later, the two issues remain largely unresolved. Consider, for example, current knowledge systems for engineering design. The scale of these systems is quite small both in the amount and variety of knowledge they contain, and the size and complexity of problems they solve. In addition, these systems are both domain-specific in that their knowledge is relevant only to a limited class of domains, and task-specific in that their processing is appropriate only to a limited class of tasks.

At the root of the difficulty lie two critical questions. Both generality and scale demand huge volumes of knowledge. Consider, for example, knowledge systems for

* This work was funded by a DARPA grant monitored by WPAFB, contract #F33615-93-1-1338, and has benefited from feedback from Chuck Sutterwaite of WPAFB. We appreciate the support.

a specific phase and a particular kind of engineering design, namely, the conceptual phase of functional design of mechanical devices. A robust knowledge system for even this very limited task may require knowledge of millions of design parts. Thus the first hard question is this: How might we accumulate huge volumes of knowledge? Generality also implies heterogeneity in both knowledge and processing. Consider again knowledge systems for the conceptual phase of functional design of mechanical devices. A robust knowledge system may use a number of processing methods such as problem/object decomposition, prototype/plan instantiation, case-based reuse, model-based diagnosis and model-based simulation. Each of these methods uses different kinds of knowledge. Thus the second hard question is this: How might we support heterogeneous knowledge and processing?

Recent work on these questions may be categorized into two families of research strategies: (i) *ontological engineering*, and (ii) *reuse, integration and sharing* of information sources. The well known CYC project [?] that seeks to provide a global ontology for constructing knowledge systems exemplifies the strategy of ontological engineering. This bottom-up strategy focuses on the first question of accumulation of knowledge. The second research strategy has three elements: reuse of information sources such as knowledge systems and databases, integration of information sources, and sharing of information in one source by other systems. This top-down strategy emphasizes the second question of heterogeneity of knowledge and processing and appears especially attractive with the advent of the world-wide-web which provides access to huge numbers of heterogeneous information sources such as knowledge systems, electronic databases and digital libraries. Our work falls under the second category.

[?] have pointed out that a key question pertaining to this topic is how to convert data in a database into knowledge useful to a knowledge system. The answer to this question depends in part on the processing method used by the knowledge system. The current generation of knowledge systems are heterogeneous both in their domain knowledge and control of processing. They not only use multiple methods, each of which uses a specific kind of knowledge and control of processing, but they also enable dynamic method selection. Our work focuses on the interface between legacy databases and legacy knowledge systems of the current generation.

The issue then becomes: given a legacy database, and given a legacy knowledge system in which a specific processing method poses a particular knowledge goal (or query), how might the data in the database be converted into a form appropriate to the processing method? The form of this question indicates a possible answer: *method-specific transformation* (or MST), which would transform the data into a form appropriate to the processing strategy. The goal of this paper is to outline a conceptual framework for the MST technique. Portions of this framework are instantiated in an operational computer system called HIPED (for Heterogeneous Intelligent Processing for Engineering Design). HIPED integrates a knowledge system for engineering design called INTERACTIVE KRITIK [?, ?] with an external database represented in Oracle [?]. The knowledge system and the database communicate through IDI [?].

2 HIPED Architecture

Figure 1 illustrates the general scheme. We describe this architecture in the following subsection by decomposing it into database, knowledge system, and user components.

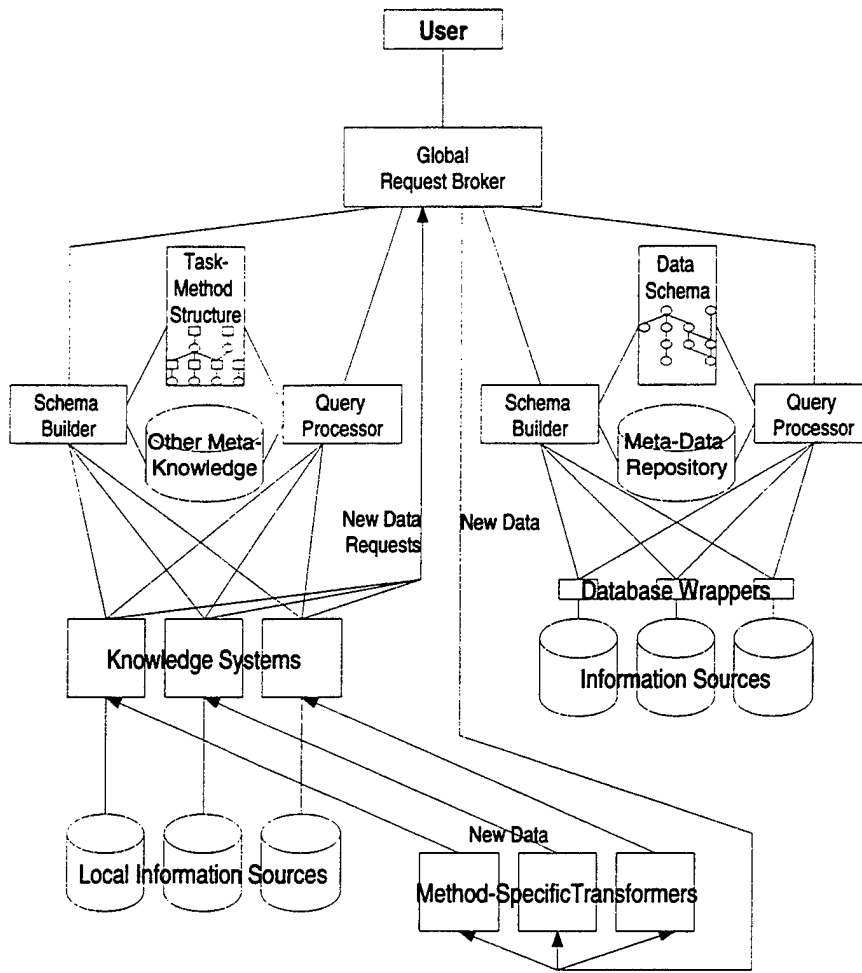


Fig.1. The HIPED architecture (Arrowed lines indicate unidirectional flow of information; all other lines indicate bidirectional flow. Annotations on lines describe the nature of the information which flows through that line. Rectangular boxes indicate functional units and cylinders represent collections of data).

2.1 Database Integration

An enormous amount of data is housed in various database systems; unfortunately, the meaning of this data is not encoded within the databases themselves. This lack of metadata about the schema and a myriad of interfaces to various database systems creates significant difficulties in accessing data from various legacy database systems. Both of these problems can be alleviated by creating a single, global representation of all of the legacy data, which can be accessed through a single interface.

Common practice for integration of legacy systems involves manual integration of each legacy schema into a global schema [?]. Clearly, this approach does not work for integration of a large number of database systems. We propose (see the right side of Figure 1) to allow the database designers to develop a metadata description, called an augmented export schema, of their database system. A collection of augmented export schemas can then be automatically processed by a schema builder to create a partially integrated global schema² which can be as simple as the actual database schema, allowing any database to easily participate, or as complicated as the schema builder can understand (See [?] for details on possible components of an augmented export schema). A user can then submit queries on the partially integrated global schema to a query processor which fragments the query into queries on the local databases. Queries on the local databases can be expressed in a single query language which is coerced to the local databases query language by a database wrapper.

2.2 Knowledge System Integration

As with databases, a considerable number knowledge systems exist. Most knowledge systems do not provide an externally accessible description of the tasks and methods they address. We propose (see the left side of Figure 1) to allow knowledge system designers to develop a description, called a "task-method schema," of the tasks each local knowledge system can perform [?]. In this approach, a set of knowledge systems, defined at the level of tasks and methods, is organized into a coherent whole by a query processor or central control agent. The query processor uses a hierarchically organized schema of tasks and methods as well as a collection of miscellaneous knowledge about processing and control (i.e. other meta-knowledge). Both the task-method structure and the other meta-knowledge may be constructed by the system designer at design time or built up by an automated schema builder.

2.3 Integrated Access

Transparent access to data and knowledge is important. We propose the provision of a global request broker which takes a query from a user, submits the query to both knowledge and database systems and returns an integrated result. Knowledge systems needing data not available in their local repositories may act as users themselves.

2.4 Method-Specific Transformation

In this paper, we are concerned with the transformation of knowledge from external sources into a form suitable for use by a knowledge system method. A naive approach

² A mechanism for complete, automated integration is unlikely.

involves writing a transformation function for every permutation of knowledge system and database. Clearly, this limits the overall scalability of the system.

We propose to leverage the partially integrated global representation of the knowledge and database systems by creating a method-specific transformation for each knowledge system which transforms knowledge from the partially integrated global schema into a knowledge system specific representation. The number of necessary method-specific transformations is linear with respect to the number of knowledge systems, increasing the scalability of our approach.

2.5 Information Flow

Consider a knowledge system which spawns a task for finding a design part such as a battery with a certain voltage. In addition to continuing its own internal processing, the knowledge system also submits a query to the Global Request Broker. The broker sends the query to the query processors for both integrated knowledge and database systems. The database query processor fragments the query into subqueries for the individual databases. The data derived is merged, converted to the global representation, and returned to the Global Request Broker. Meanwhile, the knowledge query processor, using its task-method schema, selects knowledge systems with appropriate capabilities and submits tasks to each. Solutions are converted to a common representation and sent to the Global Request Broker. It then passes the output from both the knowledge and database system query processors through a method-specific transformer which coerces the data into a form which is usable by the requesting knowledge system. The resulting battery may be an actual battery which satisfies the voltage specification from a knowledge or database system information source or it may be a battery constructed from a set of lower voltage batteries by a knowledge system.

3 An Experiment with HIPED

We have been conducting a series of experiments in the form of actual system implementations. Figure 2 presents an architectural view of one such experiment, in which a legacy knowledge system requests and receives information from a general-purpose database system. Since this experiment deals with only one knowledge system and only one database, we are able to abstract away a great many issues and focus on a specific question: method-specific transformation.

3.1 General Method

The overall algorithm developed in this experiment breaks down into four steps which correspond to the four architectural components shown in Figure 2:

- Step 1 The *knowledge system* issues a request when needed information is not available in its *local information source*.
- Step 2 The *query processor* translates the request into a query in the language of the *information source*.
- Step 3 The *information source* processes the query and returns data to the *query processor* which sends the data to the *method-specific transformer*.
- Step 4 The *method-specific transformer* converts the data into a knowledge representation format which can be understood by the *knowledge system*.

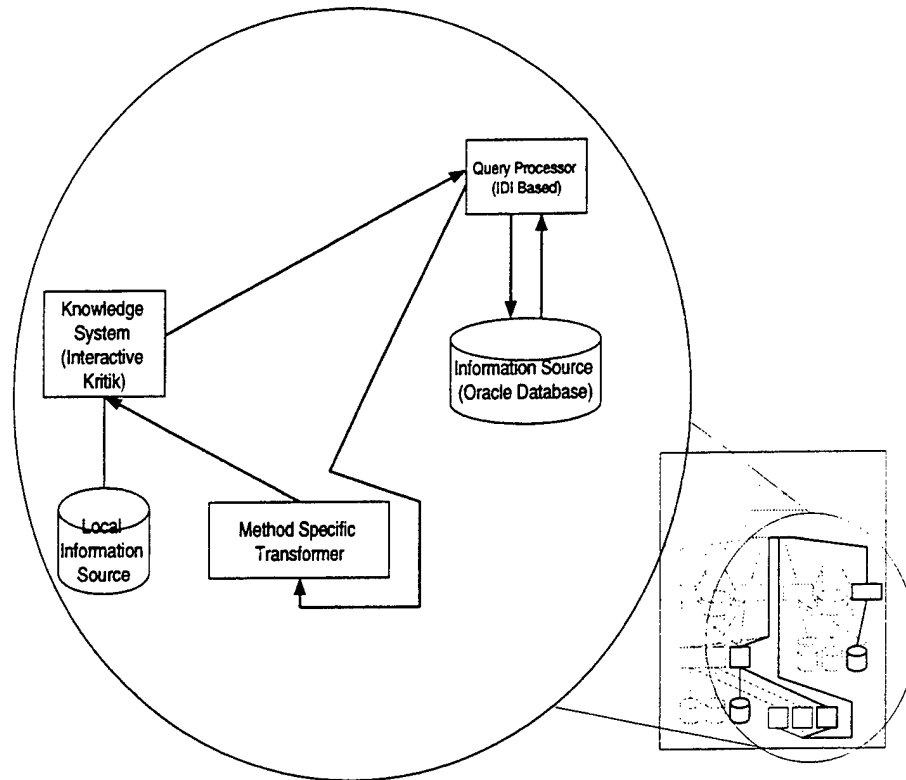


Fig. 2. The portion of the architecture relating to the proposed solution

All four of these steps pose complex problems. Executing step one requires that a knowledge system recognize that some element is missing from its knowledge and that this element would help it to solve the current problem. Performing step two requires a mechanism for constructing queries and providing communication to and from the external system. Step three is the fundamental problem of databases: given a query produce a data item. Lastly, step four is a challenging problem because the differences between the form of the data in the information source and the form required by the knowledge system may be arbitrarily complex. We focus on the fourth step: method-specific transformation. The algorithm for the method-specific transformer implemented in our experimental system is as follows:

- Substep 4.1 Database data types are coerced into to knowledge system data types.
- Substep 4.2 Knowledge attributes are constructed from fields in the data item.
- Substep 4.3 Knowledge attributes are synthesized into a knowledge element.

3.2 Integration

The particular legacy systems which we combined in our implementation were INTERACTIVE KRITIK and a relational database system [?] developed under Oracle. INTERACTIVE KRITIK is a knowledge system which performs conceptual design of simple

physical devices and provides visual explanations of both the reasoning processes it goes through and the design products it produces. It is an inherently multi-strategy knowledge system. It uses case-based reasoning as a general process for performing design and it also uses an assortment of model-based methods for doing specific design tasks such as diagnosis and repair.

The experimental system which we have written serves as an interface between INTERACTIVE KRITIK and our Oracle database. It is used when INTERACTIVE KRITIK is attempting to redesign a device by component substitution, one redesign strategy in its library of strategies. As a simple example, consider the situation in which the system has determined that a flashlight is not producing enough light and has decided that a more powerful bulb is needed. When the system identifies a single component whose replacement could potentially solve the design problem, it consults its library of components to see if such a replacement exists; in the example, it would check to see if it knows about a more powerful bulb and would make a substitution only if it did. In earlier implementations, the library of components was stored entirely within INTERACTIVE KRITIK itself in the form of data structures in memory. In our experiment, these data structures are not present in memory and the request for an appropriate component takes place through our partial implementation of the pieces of the HIPED architecture illustrated in Figure 2.

INTERACTIVE KRITIK sends its request to the query processor. The request is made as a LISP function call to a function named lookup-database-by-attribute which takes three arguments: a prototype, an attribute, and a value for that attribute. An example of such a call from the system is a request for a more powerful light bulb for which the prototype is the symbol 'L-BULB which refers to the general class of light bulbs, the attribute is the symbol 'CAPACITY, and the value is the string "capacity-more" which is internally mapped within INTERACTIVE KRITIK to a value, 18 lumens. The query processor uses IDI to generate an SQL query as follows:

```
SELECT DISTINCT RV1.inst_name
FROM   PROTO_INST RV1, INSTANCE RV2
WHERE  RV1.proto_name = 'l-bulb'
AND    RV1.inst_name = RV2.name
AND    RV2.att_val = 'capacity-more'
```

IDI sends this query to Oracle running on a remote server. Oracle searches through the database tables illustrated in Table 1. The first of these tables, INSTANCE, holds the components themselves. The second table, PROTO_INST is a cross-reference table which provides a mapping from components to prototypes.

Table 1. The tables for the Oracle database

Table INSTANCE			Table PROTO_INST	
NAME	ATTRIBUTE	ATT_VAL	INST_NAME	PROTO_NAME
littlebulb	lumens	capacity-less	littlebulb	l-bulb
bigmotor	watts	power-more	bigmotor	motor
bigbulb	lumens	capacity-more	bigbulb	l-bulb

If Oracle finds a result, as it does in this example, it returns it via the method-specific transformer. In this case, the query generates the string "bigbulb" as the result. The prototype name and the value are also part of the result, but they are not explicitly returned by the database since they are the values used to select the database entry in the first place. The method-specific transformer converts the raw data from the database to a form comprehensible to INTERACTIVE KRITIK by using the algorithm described in Section 3.1. In Substep 4.1, the string "bigbulb" is converted from a fixed length, blank padded string, as returned by Oracle, to a variable length string, as expected by INTERACTIVE KRITIK. In Substep 4.2, the attributes of the new bulb are generated. The values "bigbulb" and 'L-BULB are used as the knowledge attributes name and prototype-comp; the values 'CAPACITY, 'LUMENS, and "capacity-more" are combined into a CLOS object of a class named parameter and a list containing this one object is created and used as the parameters attribute of the component being constructed. Finally, in Substep 4.3 these three attribute values are synthesized into a single CLOS object of the component class. The end result of this process is an object equivalent to the one defined by the following statement:

```
(clos:make-instance 'component
  :init-name      "bigbulb"
  :prototype-comp 'L-BULB
  :parameters     (list (clos:make-instance 'parameter
                                           :init-name      'CAPACITY
                                           :parm-units    'LUMENS
                                           :parm-value    "capacity-more"))))
```

These commands generate a CLOS object of the component class with three slots. The first slot contains the component name, the second contains the prototype of the component, and the third is a list of parameters. The list of parameters contains a single item which is, itself, a CLOS object. This object is a member of the parameter class and has a parameter name, the units which this parameter is in, and a value for the parameter. This object is then returned to INTERACTIVE KRITIK which is now able to continue with its processing.

4 Discussion

The complexity involved in constructing a knowledge system makes reuse an attractive option for true scalability. However, the reuse of legacy systems is non-trivial because we must accommodate the heterogeneity of systems. The scalability of the HIPED architecture comes from the easy integration of legacy systems and transparent access to the resulting pool of legacy knowledge. Sharing data simply requires that a legacy system designer augment the existing local schema with metadata that allows a global coordinator to relate data from one system to another, providing a general solution to large scale integration.

The specific experiment described in Section 3 models only a small portion of the general architecture described in Section 2. In a related experiment, we have worked with another portion of the architecture [?]. Here, five types of queries that INTERACTIVE KRITIK may create are expressed in an SQL-like syntax. The queries are evaluated by mapping them using facts about the databases and rules that establish correspondences among data in the databases in terms of relationships such as equivalence,

overlap, and set containment. The rules enable query evaluation in multiple ways in which the tokens in a given query may match relation names, attribute names, or values in the underlying databases tables. The query processing is implemented using the CORAL deductive database system [?].

While the experiment described in this paper demonstrates method-specific transformation of data into knowledge usable by INTERACTIVE KRITIK, the other experiment shows how queries from INTERACTIVE KRITIK can be flexibly evaluated in multiple ways. We expect an integration of the two to provide a seamless and flexible technique for integration of knowledge systems with databases through method-specific transformation of data into useful knowledge.

Integrating Heterogeneous Databases for Engineering Design *

Sameer Mahajan

Shamkant Navathe

College of Computing,
Georgia Institute of Technology,
(sameer,sham)@cc.gatech.edu

Abstract

A number of applications access data residing in heterogeneous databases, based on various data models, having differing schemas, consisting of different internal representations etc. In this paper we pick up a generic application of Engineering Design and assume a predetermined intelligent user interface. We concentrate mainly on relational databases with the SQL interface for the purpose of an illustrative implementation. We demonstrate the use of the CORAL deductive database management system for the representation and maintenance of the metadata repository; and for the generation of multiple possible interpretations of the user queries. CORAL facts store information about the various schemas in the system. CORAL rules establish various relationships amongst different databases, tables, attributes and values. The C++ interface of CORAL (also termed as CORAL++) along with its deductive power is used for arriving at the multiple interpretations of the user queries.

1 Introduction

Heterogeneity of databases is becoming a necessary factor to contend with in the design of new applications because of the proliferation of database management systems that used diverse data models over the last three decades. Among widely implemented data models we have the hierarchical, network, relational and object oriented data models. A large body of work exists that deals with the mapping of these models among one another (e.g. see the mapping of models using the entity relationship model as an intermediate model in [1] [2]). While vendors are also providing middleware solutions to draw data from these legacy systems, the semantic problems of resolving conflicts regarding naming, scale, structure etc. that were pointed out several years ago [3] [4] still remain. The purpose of the project was to tie a set of heterogeneous databases to an "intelligent front end application" which would make requests for data without any knowledge of the schemas of the target databases. To limit the degree of difficulty we assume that we are dealing with data in relational databases only. This assumption is reasonable in the sense that if the data is coming from a hierarchical or a network DBMS, we can first convert the schema to a relational one before treating it for purposes of integration. The database integration problem we discuss here is couched in the context of engineering design which, like any other design application, relies on extracting data from existing databases containing material data, components, existing designs etc. The exact context and the application scenario will be explained in the next section.

We assume that relevant data for the design application is stored in relations (tables) whose schemas are available at "design time". It is conceivable that to support large scale engineering designs, data from a variety of databases, i.e., from multiple schemas would be required. To facilitate integration of data among these databases we assume that the "correspondences", i.e., the similarities and differences among the (meaning of) attributes is encoded in the form of rules. Furthermore, for our application context, the user issues certain queries looking for relevant design information. We

show in this paper how a query may have several interpretations. A deductive database system like CORAL makes it easy to represent the schema information and the interrelationships in a natural way. The C++ interface embedded in CORAL++ makes it easy to write general purpose programs to access and update this information.

2 Problem Definition

Our main objective is to build and demonstrate an intelligent interface to a set of (possibly autonomous) information sources including structured databases, knowledge bases, and unstructured data. The approach we have selected involves the development of a mediator which utilizes meta-knowledge of the underlying information stores to aid a user in browsing data or a system in retrieving specific relevant information.

2.1 Predetermined Querying Interface

The user is assumed to use this system as an Engineering Database for designing purposes. So he would typically like to find components that satisfy his requirements (e.g. batteries with voltage rating higher than 10V and cheaper than \$10). Keeping this user's perspective in mind, the Engineering data is thought to be made up of various "Prototypes". Each Prototype has various "Properties". Each Property takes up some "Value" for every Prototype. We can compare the Values of various properties using the relations : $=$, $<$, $>$, \leq , \geq , $<>$ etc. For the purpose of our implementation, the queries can be classified into the following five types,

1. (Prototype \langle proto_name \rangle) : here the user is looking for all the prototypes identified by "proto_name". It is implicit that the user wants to see the various values for various properties of these prototypes.
2. (Property \langle prop_name \rangle) : the user is interested in all the prototypes having the Property identified by "prop_name". It is implicit that the user wants to see the values taken by this property for the various prototypes, that would be listed.
3. (Prototype \langle proto_name \rangle) (Property \langle prop_name \rangle) : the user wants to see all the prototypes identified by "proto_name" and having property identified by "prop_name". It is implicit that the user also wants to see the corresponding value that the property takes for the particular prototype.
4. (Prototype \langle proto_name \rangle) (Property \langle prop_name \rangle) (Value \langle value \rangle) (Relation \langle rel \rangle) : the user is interested in prototypes identified by "proto_name" having a property identified by "prop_name". In addition to this he wants only those prototypes for which the property takes a value which is related to the given "value" in the query by the relation "rel" (i.e. it is equal to "value" or greater than "value" etc.)
5. (Property \langle prop_name \rangle) (Value \langle value \rangle) (Relation \langle rel \rangle) : the user is interested in all the prototypes for which the property identified by "prop_name" takes a value which is related to the given "value" by the relation "rel".

2.2 Heterogeneous Databases as the Backend

Data is scattered in various databases and various tables in each of those databases. The databases are assumed to be relational with the SQL interface for the illustrative purpose. It is a reasonable assumption and is made to contain the complexity of the problem. The system needs to find out which databases and which tables in these databases have the relevant data to answer a particular query. It then translates the query into a corresponding SQL query for every table. This SQL query is run against that table to get an answer. As we made an assumption of a uniform SQL interface to all the databases, we can simply translate a query into one in SQL against a target database in each of these cases.

3 Overall Operations in the System

The data is organized at two levels namely, (1) the metadata repository : consisting of information about various databases and tables in them and (2) the actual data : which is distributed in various heterogeneous databases. This organization reduces the data to be dealt with at the first level to get to the appropriate database(s) and table(s). It also allows heterogeneity in the various databases involved. The Querying Interface is as described in section 2.1. The “data” together with its “wrapper” forms a database system. “Wrapper” simply defines the access methods to the data for reading and updating purposes. A user query, which is of the form described above, would be translated into the corresponding query, as understood by the corresponding “wrapper”, for each of the relevant tables. This query would then be routed to the corresponding database, that contains this table. The metadata repository is consulted in determining these relevant tables and finding the corresponding database. The user would get the result, obtained after running the query against all the applicable databases through the “Result Composer”. The overall system architecture is given in figure 1.

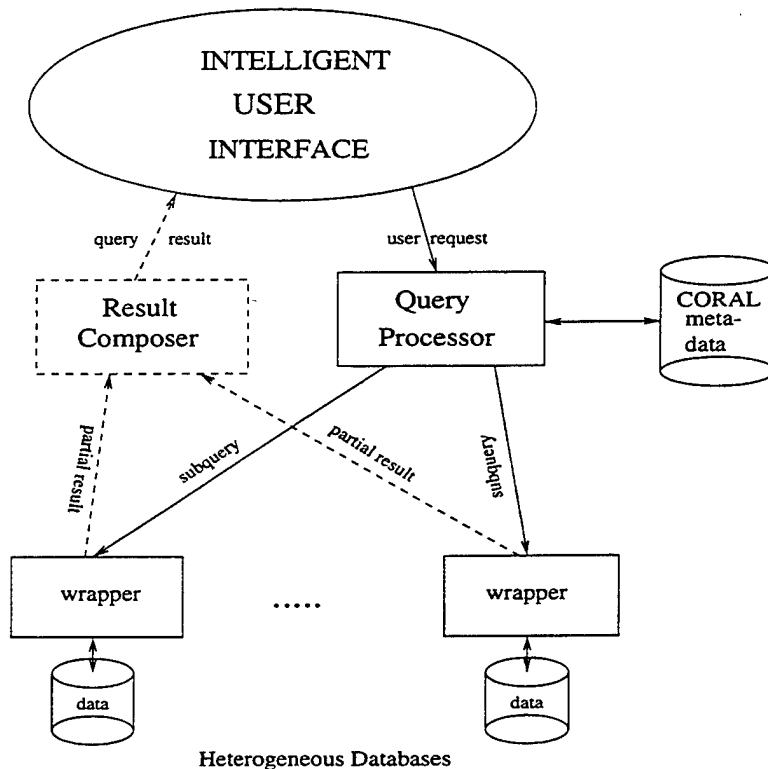


Figure 1: The High Level System View

3.1 Multiple Interpretations of the various types of queries

For better understanding of the following discussion, let us take up an example query. Let the four components of the query be,

(Prototype Battery) (Property Voltage) (Value 10) (Relation ==).

As there can be various tables with different schema, we need to run this query with only those tables that might give meaningful results for the query. We can easily observe that any of “Prototype”, “Battery”, “Property” and “Voltage” can be a table or a column of a table. The “Battery” and

"Voltage" can also be values in the columns (e.g. those labeled as "Prototype" and "Property" respectively). Of course there are a lot of dependencies amongst these components - e.g., if "Prototype" is a table, then "Battery" has to be a column of this table. On the other hand, if there is a table called the "Battery", then we are looking for values in the column "voltage" or "volts" - so that the query would generate meaningful results with the table. Now we take up an example query for each of the five types listed above. For every query we list the possible tables according to our scheme.

1. (Prototype Battery). The user typically means that he wants all the batteries with their properties and their corresponding values. Hence we will have to run this query against all the tables which,

- are equivalent to "Prototype Table" and have a column equivalent to "Battery" or
- are equivalent to "Battery Table"
- have a column equivalent to "Prototype" (and only the tuples with Prototype as "Battery" would be considered).

if and only if these tables have columns equivalent to "Property" and "Value" each.

2. (Property Voltage). The user is interested in listing all the Prototypes having "Voltage" as one of their Properties. The Values of these Properties would also be significant from his standpoint. Hence we consider all the tables which,

- are equivalent to "Property Table" and have a column equivalent to "Voltage" or
- are equivalent to "Voltage Table"
- have a column equivalent to "Property" (and only the tuples with Property as "Voltage" would be considered).

if and only if they have "Prototype" equivalent column.

3. (Prototype Battery) (Property Voltage). The user wants all the batteries with special interest in their voltages. Hence we will run the query against all the tables which,

- are equivalent to "Prototype Table" and have "Battery", "Property" and "Value" equivalent columns and we would be interested only in the tuples having an entry of "Voltage" in the "Property" equivalent column or
- are equivalent to "Prototype Table" and have "Battery", "Voltage" equivalent columns or
- are equivalent to "Battery Table" and have "Property" and "Value" equivalent columns. We would be interested only in those tuples having Property "Voltage" or
- are equivalent to "Battery Table" and have a column equivalent to "Voltage" Or
- are equivalent to "Property Table" and have columns equivalent to "Voltage", "Prototype" and "Value". We would be interested in tuples with Prototype as "Battery".
- are equivalent to "Property Table" and have "Voltage" and "Battery" equivalent columns.
- are equivalent to "Voltage Tables" and have "Prototype" and "Value" equivalent columns. We would look for only tuples with Prototype as "Battery".
- are equivalent to "Voltage Table" with "Battery" and "Value" equivalent columns.
- have "Prototype" and "Property" equivalent columns as far as they have "Value" equivalent column. Only the tuples with Prototype as "Battery" and Property as "Voltage" would be considered.

4. (Prototype Battery) (Property Voltage) (Value 10) (Relation ==). Here the interest is indicated in all the batteries having Voltage as "10". The query can be run with all the tables as indicated as above with an added constraint that only those tuples which have an entry of "10" in the "Voltage" or "Value" column - whichever is applicable - (Note the table can have only one of these columns at a time) will be considered.

5. (Property Voltage) (Value 10) (Relation ==). All the Prototypes having voltage of "10" are being considered. Thus all the tables that,

- are equivalent to "Property Table" and have a column equivalent to "Voltage"
- are equivalent to "Voltage Table" and have a column equivalent to "Value"
- have "Property" and "Value" equivalent columns along with "Prototype" column. (only tuples with Property "Voltage" and Value "10" would be taken into consideration).

would be considered if and only if they have a column equivalent to "Prototype". All the tuples with "Voltage" or "Value" being 10 would be taken into account.

4 Effective Use of CORAL for the Integration

The metadata is stored in the form of CORAL [5] [6] facts and rules. CORAL is a deductive database system which stores data as facts and rules, and allows for that data to be queried. By using CORAL the mediator can decide which database(s) and table(s) are useful in answering any given query. In particular, CORAL is used in deriving relationships like equivalence; between attributes, tables and databases. Any creation, deletion or modification of a table results in a change in the metadata repository. This dynamic behavior can be easily captured by CORAL. In essence, CORAL provides us with the facility for database integration through the facts and rules specified about tables and databases. However, this integration can be considered implicit rather than explicit since no global conceptual schema is explicitly formed. Also the C++ interface provided by CORAL makes writing general purpose programs easy. We explain the implementation by taking help of an example. The integration of CORAL deductive engine with the C++ interface makes CORAL++ rich and well-suited for our system.

4.1 A Simple Example

Consider a query,

(Prototype Battery) (Property Voltage).

This query is asking for information in all the databases that has something do with battery as a prototype and voltage as a property.

Let us assume a system (as illustrated in figure 2) that has two databases - db1 and db2. Let db1 have the populated tables as shown in Table 1 and Table 2.

CompNo	Prototype	Property	Value
B101	Battery	Voltage	10
M101	Motor	Voltage	10
B110	Battery	Voltage	100
B111	Battery	Current	100
	⋮		

Table 1: "Components" Table in db1

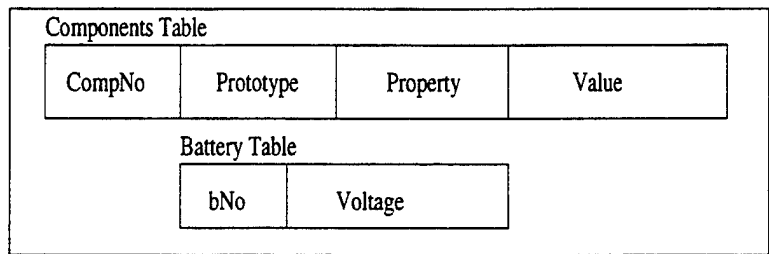
and let db2 have populated tables as shown in Table 3 and Table 4.

We observe that according to the discussion in section 3 only the tables in db1 would produce meaningful results with the query under consideration.

4.2 Representation of the Metadata Repository

It is stored as CORAL facts and rules. The advantage of such a storage is that we can utilize the strong deductive power of CORAL (e.g. deducing equivalence of attributes, equivalence of tables etc.). The various components of the repository are described below.

db1



db2

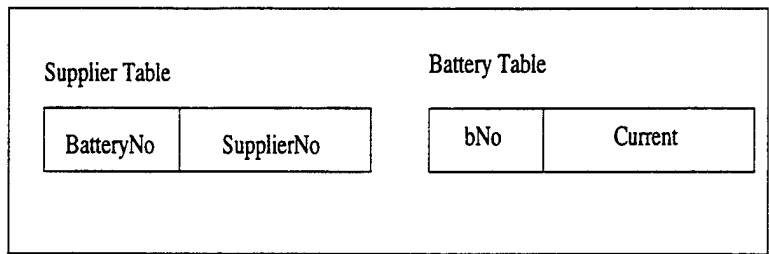


Figure 2: An Example Database System

BatteryNo	Voltage
B101	15
B102	30
	⋮

Table 2: "Battery" Table in db1

BatteryNo	Current
B101	15
B102	30
	⋮

Table 3: "Battery" Table in db2

BatteryNo	Supplier No
B101	4567
B102	4568
	⋮

Table 4: "Supplier" Table in db2

- First we list all the tables in all the databases as CORAL facts like,

```
% For the first database, db1.
belongsTo(component, db1).
belongsTo(battery, db1).
```

```
% For the second database, db2.
belongsTo(battery, db2).
belongsTo(supplier, db2).
```

- Then we list attributes of individual tables as CORAL facts. The first argument of these predicates is the database name. It is so because the same table may have different attributes in different databases. e.g. the “battery” table in the two databases “db1” and “db2” as shown below.

```
% for db1
hasAttribs(db1, component, [compName, prototype, property, value]).
hasAttribs(db1, battery, [bName, voltage]).
```

```
% for db2
hasAttribs(db2, battery, [bName, current]).
hasAttribs(db2, supplier, [bName, sName]).
```

- We also need facts to list what attributes are equivalent. The equivalence of tables can be either given by facts or can be deduced by the rules (e.g. two tables with equivalent attributes are equivalent). But we do not need them in this particular example.
- To find whether a table has a particular attribute in a given database we define a CORAL rule as,

```
module isAttrib.
export isAttrib(bff).
isAttrib(Db, Table, Attri) :- hasAttribs(Db, Table, Attris),
                             iselem(Attri, Attris).
end_module.
```

```
% Module ‘‘iselem’’ is defined for the sake of completeness.
module iselem.
export iselem(bb).
@pipelining+.    % Solve in a top-down fashion

iselem(X, [X|_]).
iselem(X, [_|Z]) :- iselem(X,Z).
end_module.
```

4.3 Deducing the Appropriate Tables

It is done according to the scheme suggested in section 3. We use the C++ interface of CORAL for this matter. In fact, an imperative interface (e.g. in C) would have been enough for the purpose. We check for the various conditions given in the scheme and generate the appropriate SQL queries for the existing tables. We run through the algorithm for the example query under consideration,

```
begin
```

```

For every 'table' equivalent to 'prototype table'
  for every attribute equivalent to 'battery', say attrib1
    for every attribute equivalent to 'property', say attrib2
      if 'table' has 'attrib1' as well as 'attrib2'
        for every attribute equivalent to 'value', say attrib3
          if 'table' has attrib3
            select the corresponding database and fire SQL query,
            SELECT * FROM table
            WHERE attrib2 == voltage equivalent value.
            goto next table
          for every attribute equivalent to 'voltage', say attrib4
            if 'table' has attrib4
              select the corresponding database and fire SQL query,
              SELECT * FROM table

```

```

For every 'table' equivalent to 'battery table'
  for every attribute equivalent to 'voltage', attrib1
    if 'table' has attrib1
      select the corresponding database and fire SQL query,
      SELECT * FROM table
      goto next table
  for every attribute equivalent to 'property', attrib2
    for every attribute equivalent to 'value', attrib3
      if 'table' has attrib2 and attrib3
        select the corresponding database and fire SQL query,
        SELECT * FROM table
        WHERE attrib2 == voltage equivalent value.

```

```

For every table equivalent to 'property table'
  for every attribute equivalent to 'voltage', say attrib1
    for every attribute equivalent to 'prototype', say attrib2
      if 'table' has 'attrib1' as well as 'attrib2'
        for every attribute equivalent to 'value', say attrib3
          if 'table' has attrib3
            select the corresponding database and fire SQL query,
            SELECT * FROM table
            WHERE attrib2 == battery equivalent value.
            goto next table
          for every attribute equivalent to 'battery', say attrib4
            if 'table' has attrib4
              select the corresponding database and fire SQL query,
              SELECT * FROM table

```

```

For every table equivalent to 'voltage table'
  for every attribute equivalent to 'battery', attrib1
    if 'table' has attrib1
      select the corresponding database and fire SQL query,
      SELECT * FROM table
      goto next table
  for every attribute equivalent to 'prototype', attrib2
    for every attribute equivalent to 'value', attrib3
      if 'table' has attrib2 and attrib3

```

```
select the corresponding database and fire SQL query,  
SELECT * FROM table  
WHERE attrib2 == battery equivalent value.
```

```
For every table having columns equivalent to each of  
    prototype, property and value  
select the corresponding database and fire SQL query  
SELECT * FROM table  
WHERE prototype equivalent column) == battery equivalent value  
    AND property equivalent column == voltage eqv. value  
end
```

4.4 The Result of Sample Queries

Let us say that the wrapper of db1 can handle SQL queries. In that case we first select that database and then simply run a query,

```
SELECT *  
FROM component  
WHERE prototype = 'battery' AND property = 'voltage'
```

against the first ("component") table in the database. We take similar actions for the other table in (possibly various) databases. The other query in this case would be,

```
SELECT *  
FROM battery
```

again with the same database viz. db1. The result is presented to the user as displayed by the corresponding "wrapper". The task of Result Composer is trivial in this case. It needs to simply display the two tables with appropriate header information (e.g. table names etc.). In general it might be required to merge tables coming from multiple databases. It might also be required to take into consideration the interrelationships amongst various various tables, attributes and values in the same database.

5 Exploiting the Deductive Power of CORAL

The equivalence relationships amongst the various attributes are stored as CORAL facts. The deduced equivalences between tables can be added (and modified as and when required) dynamically for performance benefits. Additional relationships, including semantic ones, between attributes and tables can also be easily captured under this scheme. Sometimes it might be necessary to account for these relationships. They might improve the efficiency of arriving at the result in other cases. A sample system with definition of some additional relationships is given in Appendix A. This example is given for illustrative purposes only. It does not address completeness or efficiency. It means that neither all the facts and rules required are given nor the rules are written to ensure optimal search results.

6 Conclusions and Future Work.

Integration of heterogeneous databases is achieved with respect to an Engineering Data Processing Application. The effective use of deductive database engine integrated with the C++ like interface is illustrated with the help of CORAL++ implementation. CORAL++ makes it easy to represent the information and deduce the outcomes in a natural way.

The system makes an assumption that all the databases involved provide an SQL interface. This condition can be relaxed. In this case we need to generate different queries, as understood by each of the databases involved. Currently only the individual tables are checked to see whether they provide satisfactory data to answer a particular query. But it is possible that two or more tables taken separately do not have enough information to answer a query. At the same time, when taken together (e.g. their join), they provide data to answer the query. Consider that there are two tables - which might be in the same database or in different databases - one with columns "Component Number" and "Prototype". The other with columns "Component Number" and "Voltage". Then neither of them provides enough information for the query

(Prototype Battery) (Property Voltage)

But their equijoin with the additional condition of "Prototype == Battery" for the tuples is of interest to us. The extended solution can exhaustively take care of all such cases. The five types of queries, given in section 2.1, were decided intuitively. They can be modified and / or extended based on a systematic treatment of the user needs.

References

- [1] C. Batini, S. Ceri, and S.B. Navathe, **Conceptual Database Design: An Entity Relationship Approach**, Benjamin Cummings, August 1991, 470 pp.
- [2] R. Elmasri, S Navathe, **Fundamentals of Database Systems**, 2nd Edition, 1994.
- [3] J. Larson, S. B. Navathe, and R. Elmasri A Theory of Attribute Equivalence in Databases with Application to Schema Integration, *IEEE Transactions on Software Engineering*, Vol. 15, No. 4, April 1989.
- [4] S.B. Navathe, R. Elmasri and J.A. Larson, Integrating User Views in Database Design, *IEEE Computer*, Vol. 19, No. 1, January 1986, pp. 50-62.
- [5] R. Ramakrishnan, D. Srivastava and S. Sudarshan, CORAL: Control, Relations and Logic, Proceedings of the International Conference on Very Large Databases, 1992.
- [6] R. Ramakrishnan, D. Srivastava, S. Sudarshan and P. Seshadri, Implementation of the CORAL deductive database system, Proceedings of the ACM SIGMOD Conference on Management of Data, 1993.

Appendix A

```
db(db2).
db(db1).

%
% Attributes in LDB 1 (University Registration)
%
attrib(db1, professor, profname).
attrib(db1, professor, desc).

attrib(db1, student, name).
attrib(db1, student, id).

attrib(db1, project, id).
attrib(db1, project, projname).
attrib(db1, project, desc).

attrib(db1, course, coursename).
attrib(db1, course, desc).

attrib(db1, manage, id).
attrib(db1, manage, profname).
attrib(db1, manage, projname).

attrib(db1, participate, id).
attrib(db1, participate, projname).
attrib(db1, participate, stdid).

attrib(db1, assigned, id).
attrib(db1, assigned, coursename).
attrib(db1, assigned, profname).

attrib(db1, enroll, id).
attrib(db1, enroll, coursename).
attrib(db1, enroll, stdid).

%
% Attributes in LDB 2 (University Payroll)
%
attrib(db2, project, id).
attrib(db2, project, fund).

attrib(db2, professor, id).
attrib(db2, professor, name).

attrib(db2, student, name).
attrib(db2, student, id).

attrib(db2, department, id).
attrib(db2, department, name).
```

```

attrib(db2, grapay, code).
attrib(db2, grapay, stdid).
attrib(db2, grapay, projid).

attrib(db2, gtapay, code).
attrib(db2, gtapay, stdid).
attrib(db2, gtapay, deptid).

attrib(db2, deptpay, code).
attrib(db2, deptpay, profname).
attrib(db2, deptpay, deptid).

attrib(db2, projpay, code).
attrib(db2, projpay, profname).
attrib(db2, projpay, projid).
%
% Entities (classes) in Two Databases
%
% Entities in LDB 1 (University Registration)

class(db1, professor).
class(db1, student).
class(db1, project).
class(db1, course).
class(db1, manage).
class(db1, participate).
class(db1, assigned).
class(db1, enroll).

% Entities in LDB 2 (University Payroll)

class(db2, professor).
class(db2, student).
class(db2, project).
class(db2, department).
class(db2, grapay).
class(db2, gtapay).
class(db2, projpay).
class(db2, deptpay).

%
% Attributes of an individual class listed
%

has_attribs(db2, professor, [id, name, address, officeno]).
has_attribs(db1, student, [id, name, address]).
has_attribs(db1, project, [id, courseno, desc]).
has_attribs(db1, course, [no, name, desc]).
has_attribs(db1, manage, [id, projid, profno]).
has_attribs(db1, participate, [id, stdid, projid]).
has_attribs(db1, assigned, [id, courseno, profno]).

```

```
has_attribs(db1, enroll, [id, stdid, courseno]).
```

```
has_attribs(db2, professor, [id, name]).
```

```
has_attribs(db2, student, [id name]).
```

```
has_attribs(db2, project, [id, fund]).
```

```
has_attribs(db2, department, [id, name]).
```

```
has_attribs(db2, grapay, [code, projid, stdid]).
```

```
has_attribs(db2, gtapay, [code, deptid, stdid]).
```

```
has_attribs(db2, projpay, [code, projid, profid]).
```

```
has_attribs(db2, deptpay, [code, deptid, profid]).
```

```
% Information from Schema Analysis
```

```
%
```

```
% 1. Attributes Relationship
```

```
keqv(db1, project, id, db2, project, id).
```

```
keqv(db1, student, id, db2, student, id).
```

```
keqv(db1, student, name, db2, student, name).
```

```
keqv(db1, professor, name, db2, professor, name).
```

```
kcontain(db2, student, id, db2, grapay, stdid).
```

```
kcontain(db2, student, id, db2, gtapay, stdid).
```

```
kcontain(db1, student, id, db1, participate, stdid).
```

```
kcontain(db2, professor, name, db2, project, name).
```

```
kcontain(db2, professor, name, db2, department, name).
```

```
kcontain(db1, professor, name, db1, assigned, name).
```

```
kcontain(db1, professor, name, db1, manage, name).
```

```
kcontain(db1, course, name, db1, assigned, name).
```

```
kcontain(db1, course, name, db1, enroll, coursename).
```

```
% Information from Schema Analysis
```

```
%
```

```
% 1. Class Relationship
```

```
% The following relationships would be added dynamically as they are
```

```
% deduced by the system.
```

```
ksubsume(db1, professor, db2, professor).
```

```
ksubsume(db1, project, db2, project).
```

```
ksubsume(db1, student, db2, student).
```

```
%
```

```
% equivalent attributes
```

```
%
```

```
eqv(X, X) :- attrib(Db, T, X).
```

```
eqv(X, Y) :- keqv(Db1, T1, X, Db2, T2, Y).
```

```
eqv(X, Y) :- keqv(Db1, T1, Y, Db2, T2, X).
```

```
eqv(X, Z) :- eqv(X, Y), eqv(Y, Z).
```

```

%
% Attribute Containment Relationship
%

contain(X, Z) :- eqv(X, Z).
contain(X, Y) :- kcontain(Db1, T1, X, Db2, T2, Y).
contain(X, Z) :- contain(X, Y), contain(Y, Z).
contain(X, Z) :- eqv(X, Y), contain(Y, Z).

%
% Overlapping Attributes
%

overlap(X, Y) :- contain(X, Y).
overlap(X, Y) :- contain(Y, X).
overlap(X, Y) :- eqv(X, Y).

%
% Disjoint Attributes
%

disjoint(X, Y) :- attrib(Db1, T1, X), attrib(Db2, T2, Y), not overlap(X, Y).

%
% equivalent classes
%

eqclass(X, Y) :- keqclass(Db1, X, Db2, Y).
eqclass(X, X) :- class(Db, X).
eqclass(X, Y) :- eqclass(Y, X).
% class equivalence derived from attributes.
eqclass(X, Y) :- has_attribs(X, Z), has_attribs(Y, W), eqlist(Z,W).
eqclass(X, Z) :- eqclass(X, Y), eqclass(Y, Z).

%
% Equivalent Lists of Attributes
%

eqlist([], []).
eqlist([X|Y], [W|Z]) :- eqv(X,W), !, eqlist(Y,Z).
eqlist([X|Y], [W|Z]) :- iseqv(X,Z), eqlist(Y, [W|Z]).

%
% Testing for being a member of a list.
%

iseqv(X, []) :- !, fail.
iseqv(X, [Y|_]) :- eqv(X,Y).
iseqv(X, [_|Y]) :- iseqv(X,Y).

%
% class X subsumes class Y
%

```

```
subsume(X, Y) :- ksubsume(Db1, X, Db2, Y).
subsume(X, Z) :- class(Db1, X), class(Db2, Y), class(Db3, Z),
subsume(X, Y), subsume(Y, Z).
subsume(X, Z) :- eqclass(X, Z).
subsume(X, Z) :- eqclass(X, Y), subsume(Y, Z).

%
% overlapping classes
%

overclass(X, Y) :- subsume(X, Y).
overclass(X, Y) :- subsume(Y, X).
overclass(X, Y) :- eqclass(X, Y).

%
% disjoint classes
%

disclass(X, Y) :- class(Db1, X), class(Db2, Y), not overclass(X, Y).
```

METHOD-SPECIFIC KNOWLEDGE COMPILATION:
TOWARDS PRACTICAL DESIGN SUPPORT SYSTEMS

J. WILLIAM MURDOCK AND ASHOK K. GOEL

Intelligent Systems Group

MICHAEL J. DONAHOO

Networking and Telecommunications Group

AND

SHAMKANT NAVATHE

Database Systems Group

College of Computing

Georgia Institute of Technology

Atlanta, Georgia 30332-0280, USA

[To appear in: Proceedings of the 5th International Conference on Artificial Intelligence and Design (AID'98).

Lisbon, Portugal, July 20-23, 1998.]

Abstract.

Modern knowledge systems for design typically employ multiple problem-solving methods which in turn use different kinds of knowledge. The construction of a heterogeneous knowledge system that can support practical design thus raises two fundamental questions: how to accumulate huge volumes of design information, and how to support heterogeneous design processing? Fortunately, partial answers to both questions exist separately. Legacy databases already contain huge amounts of general-purpose design information. In addition, modern knowledge systems typically characterize the kinds of knowledge needed by specific problem-solving methods quite precisely. This leads us to hypothesize method-specific data-to-knowledge compilation as a potential mechanism for integrating heterogeneous knowledge systems and legacy databases for design. In this paper, first we outline a general computational architecture called HIPED for this integration. Then, we focus on the specific issue of how to convert data accessed from a legacy database into a form appropriate to the problem-solving method used in a heterogeneous knowledge system. We describe an experiment in which a legacy knowledge system called INTERACTIVE KRITIK is integrated with an ORACLE database using IDI as the communication tool. The limited experiment indicates the computational feasibility of method-specific data-to-knowledge compilation, but also raises additional research issues.

1. Motivations, Background, And Goals

Knowledge systems for design developed thus far appear incapable of supporting practical design. This critique surely is valid for all laboratory knowledge systems such as AIR-CYL (Brown and Chandrasekaran, 1989), COMADE (Lenz et al., 1996), and our own KRITIK series of systems described at earlier AI in Design conferences (Stroulia et al., 1992; Bhatta et al., 1994; Goel et al., 1996b). But it is also applicable to systems that have directly led to real applications such as R1 (McDermott, 1982), PRIDE (Mittal et al., 1986), VT (Marcus et al., 1988), CLAVIER (Hennessy and Hinkle, 1992), and ASK-JEF (Barber et al., 1992). The problem is the limited scale and scope of knowledge systems for design. The scale of these systems is quite small both in the size and complexity of problems they solve and the amount and variety of knowledge they contain. In addition, these systems are both domain-specific in that their knowledge is relevant only to a very narrow class of domains, and task-specific in that their processing is appropriate only to a very narrow class of tasks.

At the root of the above problem lie two critical questions. Firstly, both scope and scale imply heterogeneity in knowledge and processing. Consider, for example, KRITIK, which integrates case-based reasoning and model-based reasoning for addressing the conceptual phase of functional design of mechanical devices. In KRITIK, even this limited task requires many different kinds of knowledge ranging from design cases to device models to repair plans, and many different processing strategies ranging from problem-driven case retrieval to model-based adaptation to goal-directed plan instantiation. Thus the first hard question is this: How might we support heterogeneous knowledge and processing? Secondly, both scale and scope demand huge volumes of knowledge. A robust version of KRITIK capable of supporting practical design may require knowledge of millions of design cases, primitive components, device models, physical processes, engineering mechanisms, repair plans, etc. Thus the second hard question is this: How might we accumulate huge volumes of knowledge?

1.1. PAST RESEARCH

The above two questions are among the core issues in AI research on knowledge systems in general. AI strategies for answering them may be divided into two general categories: (i) *ontological engineering*, and (ii) *information integration*. The well known CYC project (Lenat and Guha, 1990), which seeks to provide a global ontology for constructing knowledge systems, exemplifies the strategy of ontological engineering. Ontolingua (Gruber, 1993) provides another, domain-specific example of this strategy. The bottom-up strategy focuses on the second question of accumulation of knowledge:

domain-specific and domain-independent ontologies may one day enable interactive knowledge acquisition from external sources and autonomous acquisition of knowledge through learning from experience. But the strategy requires the building of new systems based on a common ontology.

In contrast, the strategy of information integration emphasizes the reuse of *legacy* information sources such as databases, integration of the information sources, and sharing of information in one source by other systems. This top-down strategy focuses on one part of the first question above, namely, heterogeneity of knowledge. The strategy appears especially attractive with the advent of the World Wide Web which provides potential access to huge numbers of heterogeneous information sources such as electronic databases and digital libraries.

Various projects on information integration have focused on different aspects of information integration. For example, KQML (Finin and Wiederhold, 1991) provides a protocol language for communication among database systems, and KIF (Genesreth and Fikes, 1991) provides a meta-language for enabling translation between knowledge systems. In contrast, (Brodie, 1988) has emphasized the need for integrating knowledge systems and databases. (McKay et al., 1990) in particular have pointed out that a key question is how to convert data in a database into knowledge useful to a knowledge system. The answer to this question clearly depends in part on the problem-solving method used by the knowledge system.

1.2. THIS WORK

Modern knowledge systems for design may be considered as belonging to the third generation. The first generation, such as R1, used a single problem-solving method, e.g., rule-based reasoning, characterized by a single kind of knowledge, e.g., production rules, and a single control of processing, e.g., forward chaining. The second generation of systems, such as AIR-CYL, PRIDE and VT, not only used multiple problem-solving methods, but they also explicitly represented the control of processing of each method. AIR-CYL, for example, used top-down plan instantiation and expansion for the task of generating preliminary designs and bottom-up pattern matching for the subtask of selecting appropriate design plans. The third generation of knowledge systems for design, such as COMADE and KRITIK, not only use a richer array of problem-solving methods, but also allow for flexible strategic control and enable dynamic method selection (Punch et al., 1996).

Given the variety and complexity of design, the above evolution is only natural. But it introduces a new element in the AI strategy of information integration: how to accommodate heterogeneity of *processing*. For heterogeneous knowledge systems that employ multiple problem-solving methods,

we may translate the question posed by McKay et al. as follows: given a legacy database, and given a legacy knowledge system in which a specific problem-solving method poses a particular goal (or query), how might the data in the database be converted into a form appropriate to the method? The form of this question itself suggests a possible answer: *method-specific knowledge compilation*, which would transform the data into a form appropriate to the processing method.

This above issue is hard and complex. The modest goal of this paper is simply to examine the AI strategy of information integration from the perspective of heterogeneous knowledge systems for design. We outline a computational architecture for supporting method-specific data to knowledge compilations. We also report on an experiment with a particular instantiation of one portion of the architecture in an operational computer system called HIPED (for Heterogeneous Intelligent Processing for Engineering Design). HIPED integrates INTERACTIVE KRITIK (Goel et al., 1996b; Goel et al., 1996c), the current version of KRITIK, with an external database represented in Oracle (Koch and Loney, 1995). The knowledge system and the database communicate through IDI (Paramax, 1993).

2. The HIPED Architecture

To avoid the enormous cost of constructing knowledge systems for design, HIPED proposes the reuse of legacy knowledge and database systems, so that we can quickly and inexpensively construct large scale systems with capabilities and knowledge drawn from existing systems. To facilitate easy integration which, in effect, increases overall scalability, we restrict ourselves to making few, if any, changes to the participating legacy systems. The long-term goal is to allow a system to easily access the capabilities of a pool of legacy systems. The architecture of Figure 1 illustrates the general scheme. The architecture presented in this figure is a projected reference architecture and not a description of a specific existing system. In this section, we describe the entire reference architecture. In the following sections we will further elaborate on a particular piece of work which instantiates a portion of this architecture.

2.1. DATABASE INTEGRATION

An enormous amount of engineering design data is housed in various database systems. Unfortunately, the meaning of this data is not encoded within the databases themselves. At best, the database schema has meaningful names for individual data elements, but often it is difficult to infer all, if any, of the meaning of the data from the schema. This lack of metadata about the schema and a myriad of interfaces to various database systems creates

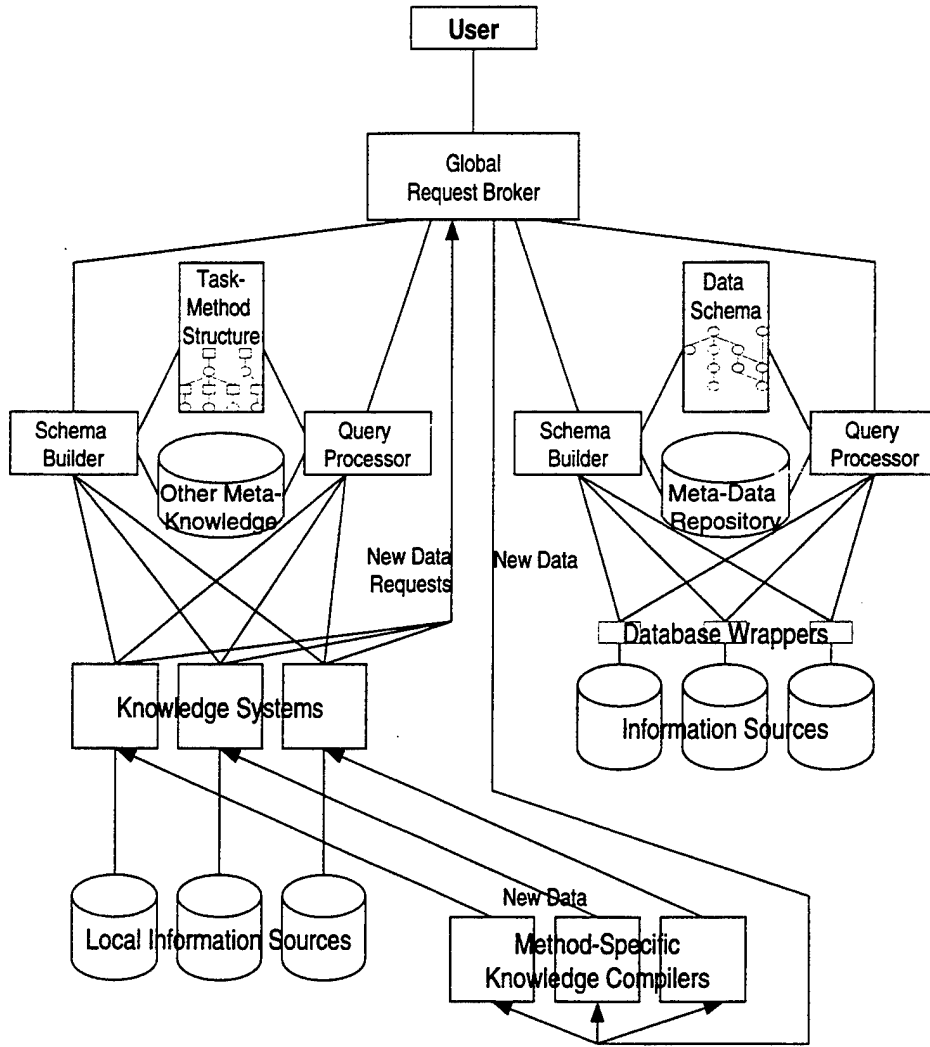


Figure 1. The HIPED architecture (Arrowed lines indicate unidirectional flow of information; all other lines indicate bidirectional flow. Annotations on lines describe the nature of the information which flows through that line. Rectangular boxes indicate functional units and cylinders represent collections of data).

significant difficulties in accessing data from various legacy database systems. Both of these problems can be alleviated by creating a single, global representation of all of the legacy data, which can be accessed through a single interface.

Common practice for integration of legacy systems involves manual integration of each legacy schema into a global schema. That is, database designers of the various legacy systems create a global schema capable of representing the collection of data in the legacy databases and provide a mapping between the legacy system schemas and this global schema (Batini et al., 1986). Clearly, this approach does not work for integration of a large number of database systems. We propose (see the right side of Figure 1) to allow the database designers to develop a metadata description, called an **augmented export schema**, of their database system. A collection of augmented export schemas can then be automatically processed by a **schema builder** to create a **partially integrated global schema**¹ which can be as simple as the actual database schema, allowing any database to easily participate, or as complicated as the schema builder can understand (See (Navathe and Donahoo, 1995) for details on possible components of an augmented export schema). A user can then submit queries on the partially integrated global schema to a **query processor** which fragments the query into sub-queries on the local databases. Queries on the local databases can be expressed in a single query language which is coerced to the local database's query language by a **database wrapper**.

2.2. KNOWLEDGE SYSTEM INTEGRATION

As with databases, a considerable number of knowledge systems exist for design (e.g. R1, AIR-CYL, PRIDE, COMADE) each with their own abilities to perform certain tasks with various methods. Users wishing to access the capabilities of a collection of such systems encounter problems of different interfaces and knowledge representations. Most knowledge systems do not provide an externally accessible description of the tasks and methods they address. As with the database system, one way to integrate legacy knowledge systems is to gather together the designers and construct an ad hoc interface which combines the capabilities of the underlying systems. Once again, this approach does not work for integration of a large number of knowledge systems.

We propose (see the left side of Figure 1) to allow knowledge system designers to develop a description, called a "task-method schema," of the tasks each local knowledge system can perform (Stroulia and Goel, 1995). In this approach, a set of **knowledge systems**, defined at the level of tasks and

¹A mechanism for complete, automated integration is unlikely.

methods, are organized into a coherent whole by a **query processor** or central control agent. The query processor uses a hierarchically organized schema of tasks and methods as well as a collection of miscellaneous knowledge about processing and control (i.e. **other meta-knowledge**). Both the task-method structure and the other meta-knowledge may be constructed by the system designer at design time or built up by an automated **schema builder**.

2.3. INTEGRATED ACCESS

The dichotomy of knowledge systems and database systems is irrelevant to global users. Users simply want answers and are not concerned with whether the answer was provided directly from a database or derived by a process in a knowledge system. We propose the provision of a **global request broker** which takes a query from a user, submits the query to both knowledge and database systems and returns an integrated result. It is completely transparent to a global user how or from where an answer was derived.

Furthermore, the individual knowledge systems may, themselves, act as users of the integrated access mechanism. The knowledge systems each have their own local repositories of data but may also find that they need information from a database or another knowledge system. When they need external knowledge, they simply contact the global request broker which can either recursively call the general collection of knowledge systems to generate a response or contact the system of databases. When either a database or a knowledge system generates a response to a request from a knowledge system, the resulting answer is then sent through a **method-specific knowledge compiler** which does whatever specific translation is needed for the particular system.

2.4. METHOD-SPECIFIC KNOWLEDGE COMPILATION

In this paper, we are concerned with the compilation of knowledge from external sources into a form suitable for use by a knowledge system method. Recall that we do not want to alter the knowledge system, so the form of the knowledge may be very specific to the particular method which executed the query; consequently, we call this a "method-specific knowledge compilation." We will examine the mechanisms behind this component of the reference architecture in more detail in later sections.

2.5. INFORMATION FLOW

Consider a design knowledge system which spawns a task for finding a design part such as a battery with a certain voltage. In addition to con-

tinuing its own internal processing, the knowledge system also submits a query to the global request broker. The broker sends the query to the query processors for both integrated knowledge and database systems. The database query processor fragments the query into subqueries for the individual databases. The data derived is merged, converted to the global representation, and returned to the global request broker. Meanwhile, the knowledge query processor, using its task-method schema, selects knowledge systems with appropriate capabilities and submits tasks to each. Solutions are converted to a common representation and sent back to the global request broker. It then passes the output from both the knowledge and database system query processors through a method-specific knowledge compiler which coerces the data into a form which is usable by the requesting knowledge system. The resulting battery may be an existing battery which satisfies the voltage specification from a knowledge or database system information source or it may be a battery constructed from a set of lower voltage batteries by a knowledge system.

3. Knowledge Compilation

The architecture described in the previous section raises an enormous variety of issues. The one which we want to focus on more closely here is that of knowledge compilation, i.e. the principled transformation of knowledge from one form to another. Large volumes of information can be found in existing databases of components, design schematics, etc. An intelligent design system can make use of this existing data by compiling it into a form which is suitable to its use. There are several closely interrelated perspectives on knowledge compilation as presented in "Knowledge Compilation: A Symposium" (Goel (ed.), 1991). A few of these perspectives relevant to this context are:

1. Knowledge implemented in one representational paradigm may be transformed into another. For example, tuples in a relational database may be transformed into objects in an object-oriented programming language.
2. Knowledge in a generally useful organization may be transformed into a different organization which is more efficient for a specific application. For example, a model of a device may be transformed into a set of rules for diagnosing faults of that device, as per (Keller, 1991).
3. Declarative knowledge may be transformed into procedural knowledge, i.e. a specification of a task may be compiled into a procedure for accomplishing that task. This is really an extreme form of the former approach; the result is a knowledge element which only supports one application, its execution, but presumably does so in the most effi-

cient way that the compiler can generate. (Tong, 1991) embodies this approach.

4. Knowledge of patterns or categories can be inductively inferred from elements. This can be also be seen as an extension of point 2, above; knowledge of instances may be voluminous and difficult to apply to new situations and thus this knowledge is compiled into descriptions or rules which directly enable recognition, classification, etc. Virtually all work done in the field of machine learning can be viewed as knowledge compilation from this perspective.

In this work, we have limited our attention to the first of these topics. We believe that all of these approaches to knowledge compilation are interesting and important. We intend to address all of these concerns in our future research. However, for the purposes of supporting large scale, heterogenous processing, it is clear that the first issue, that of transforming the structural details of the representation, is inherently fundamental; without a framework for such basic transformations, any of the more complex, sophisticated approaches to knowledge compilation are useless because they simply cannot access any knowledge to compile.

4. INTERACTIVE KRITIK

INTERACTIVE KRITIK is a legacy knowledge system which we have integrated into the HIPED architecture. INTERACTIVE KRITIK is a computer-based design environment. A major component of this system is KRITIK3, an autonomous knowledge-based design system. When completed, INTERACTIVE KRITIK is intended to serve as an interactive constructive design environment. At present, when asked by a human user, INTERACTIVE KRITIK can invoke KRITIK3 to address specific kinds of design problems. In addition, INTERACTIVE KRITIK can provide explanations and justifications of KRITIK3's design reasoning and results, and enable a human user to explore the system's design knowledge.

KRITIK3 evolves from KRITIK, an early multi-strategy case-based design system. Since KRITIK is described in detail elsewhere (see, for example, (Goel and Chandrasekaran, 1989; Goel et al., 1996a)), in this paper we only sketch the outlines of KRITIK3. One of the major contributions of KRITIK is its device modeling formalism: the Structure-Behavior-Function (SBF) language. The remarkable characteristics of SBF models are: (i) they are functional, i.e. they describe both basic components and complex devices in terms of the function they achieve; (ii) they are causal, i.e. they describe sequences of interactions which constitute the internal behavior of the device; and (iii) they are compositional, i.e. they describe how the function of the device emerges from the functions of the components.

KRITIK3 is a multi-strategy process model of design in two senses. First, while the high-level design process in KRITIK3 is case-based, the reasoning about individual subtasks in the case-based process is model-based; KRITIK3 uses device models described in the SBF language for adapting a past design and for evaluating a candidate design. Second, design adaptation in KRITIK3 involves multiple modification methods. While all modification methods make use of SBF device models, different methods are applicable to different kinds of adaptation tasks.

The primary task addressed by KRITIK3 is the extremely common functions-to-structure design task in the domain of simple physical devices. The functions-to-structure design task takes as input the functional specification of the desired design. For example, the functions-to-structure design of a flashlight may take as an input the specification of its function of creating light when a force is applied on a switch. This task has the goal of giving as output the specification of a structure that satisfies the given functional specification, i.e., a structure that results in the given functions.

KRITIK3's primary method for accomplishing this task is case-based reasoning. Its case-based method sets up four subtasks of the design task: *problem elaboration*, *case retrieval*, *design adaptation*, and *case storage*.

The task of problem elaboration takes as input the specification of the desired function of the new design. It has the goal of generating a probe to be used by case retrieval for deciding on a new case to use. KRITIK3 uses domain-specific heuristics to generate probes based on the surface features of the problem specification. The task of case retrieval takes as input the probes generated by the problem elaboration component. It has the goal of accessing a design case, including the associated SBF model whose functional specification is similar to the specification of desired design. KRITIK3's case memory is organized in a discrimination tree, with features in the functional specifications of the design cases acting as the discriminants. Its retrieval method searches through this discrimination tree to find the case that most closely matches the probe.

The task of design adaptation takes as input (i) the specification of the constraints on the desired design, and (ii) the specifications of the constraints on and the structure of the candidate design. It has the goal of giving as output a modified design structure that satisfies the specified constraints. KRITIK3 uses a model-based method of design adaptation which divides the design task into three subtasks: *computation of functional differences*, *diagnosis*, and *repair*. The idea here is that the candidate design can be viewed as a failed attempt to accomplish the desired specifications. The old design is first checked to see how its functionality differs from the desired functionality. The model of the design is then analyzed in detail to determine one or more possible causes for the observed difference. Lastly,

KRITIK3 makes modifications to the device with the intent of inducing the desired functionality.

The method of repair used by KRITIK3 is *generate and test*. This method sets up two subtasks of the repair task: *model revision* and *model verification*. The task of model revision takes as input (i) the specification of the constraints on the desired design, and (ii) the model of the candidate design. It has the goal of giving as output a modified model that is expected to satisfy the constraints on the desired design. KRITIK3 knows of several model revision methods such as component replication or component replacement. KRITIK3 dynamically chooses a method for model revision at run time based on the results of the diagnosis task. Depending on the modification goals set up by the diagnosis task, the system may also use more than one model-revision method.

The task of model verification takes as input (i) the specification of the constraints on the desired design, and (ii) the specification of the structure of the modified design. It has the goal of giving as output an evaluation of whether the modified structure satisfies the specified constraints. KRITIK3 qualitatively simulates the revised SBF model to verify whether it delivers the functions desired of it.

The task of case storage takes as input (i) a specification of the case memory, and (ii) a specification of a new case. It has the goal of giving as output a specification of the new case memory with the new case appropriately indexed and organized in it. Recall that KRITIK3's case memory is organized in a discrimination tree. The system uses a model-based method for the task of storing a new case in the tree. This method sets up the subtasks of *indexing learning* and *case placement*. The SBF model of the new design case enables the learning of the appropriate index to the new case. This directly enables the task of case placement.

5. An Experiment with HIPED

We have been conducting a series of experiments in the form of actual system implementations. Some of these experiments have focused on issues most closely related to the data end of the data to knowledge compilation process; these issues include data organization, access, transfer, etc. The experiment we focus on here, however, is more closely connected to the knowledge end of the process. This experiment examines the use of knowledge compiled at run-time in the context of the operation of INTERACTIVE KRITIK.

Figure 2 presents an architectural view of the experiment, in which a legacy knowledge system for design requests and receives information from a general-purpose database system. Since this experiment deals with

only one knowledge system and only one database, we are able to abstract away a great many issues and focus on a specific question: method-specific knowledge compilation.

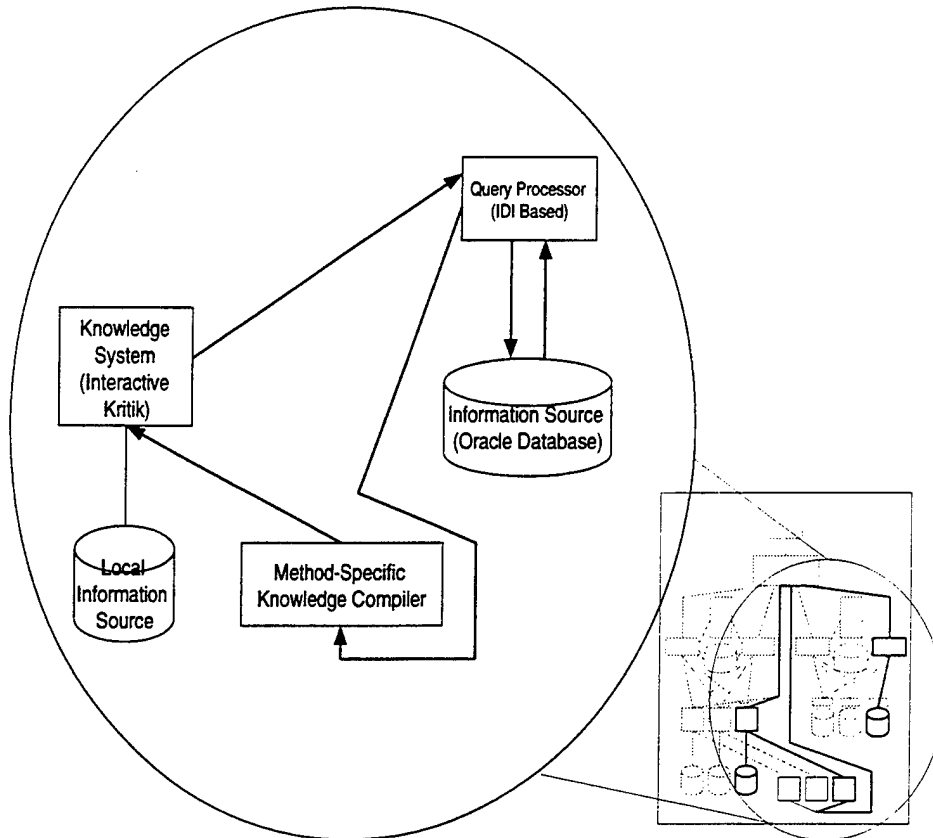


Figure 2. The portion of the architecture relating to the proposed solution

5.1. GENERAL METHOD

The overall algorithm developed in this experiment breaks down into four steps which correspond to the four architectural components shown in Figure 2:

- Step 1** The *knowledge system* issues a request when needed information is not available in its *local information source*.
- Step 2** The *query processor* translates the request into a query in the language of the *information source*.

Step 3 The *information source* processes the query and returns data to the *query processor* which sends the data to the *method-specific knowledge compiler*.

Step 4 The *method-specific knowledge compiler* converts the data into a knowledge representation format which can be understood by the *knowledge system*.

All four of these steps pose complex problems. Executing step one requires that a knowledge system recognize that some element is missing from its knowledge and that this element would help it to solve the current problem. Performing step two requires a mechanism for constructing queries and providing communication to and from the external system. Step three is the fundamental problem of databases: given a query produce a data item. Lastly, step four poses a challenging problem because the differences between the form of the data in the information source and the form required by the knowledge system may be arbitrarily complex. We focus on the fourth step: method-specific knowledge compilation. The algorithm for the method-specific knowledge compiler implemented in our experimental system is as follows:

Substep 4.1 Database data types are coerced into to knowledge system data types.

Substep 4.2 Knowledge attributes are constructed from fields in the data item.

Substep 4.3 Knowledge attributes are synthesized into a knowledge element.

The particular legacy systems for which we have implemented these algorithms are INTERACTIVE KRITIK and a relational database system (Codd, 1970) developed under Oracle. Thus the experimental system serves as an interface between INTERACTIVE KRITIK and our Oracle database.

5.2. AN ILLUSTRATIVE EXAMPLE

Our experiment takes place during a session in which INTERACTIVE KRITIK is designing an electric light circuit. It has retrieved from its case-memory a model of a circuit which produces light. However, in comparing the functional specification of the retrieved case with the desired functional specification, INTERACTIVE KRITIK determines that the retrieved case does not produce enough light. Consequently, it applies its diagnosis methods to determine components which might be responsible for the amount of light produced. One of the results generated by the diagnosis mechanism is that a higher capacity bulb will lead to the production of more light. Consequently, INTERACTIVE KRITIK may be able to apply the *component*

replacement method of model revision. However, in order to apply this method, it must have knowledge of a light bulb of sufficient capacity. No such bulb is available in its local knowledge base. In earlier versions of this system, it would conclude that replacement of the bulb is impossible and thus either attempt to replace a different component or attempt a different model revision method altogether. However, in this version, INTERACTIVE KRITIK has access to an external source of knowledge via the HIPED architecture.

INTERACTIVE KRITIK sends a request for the desired light bulb to the query processor. The request is made as a LISP function call to a function named `lookup-database-by-attribute` which takes three arguments: a prototype, an attribute, and a value for that attribute. An example of such a call from the system is a request for a more powerful light bulb for which the prototype is the symbol 'L-BULB which refers to the general class of light bulbs, the attribute is the symbol 'CAPACITY, and the value is the string "capacity-more" which is internally mapped within INTERACTIVE KRITIK to a value, 18 lumens.² The query processor uses IDI to generate an SQL query as follows:

```
SELECT DISTINCT RV1.inst_name
FROM   PROTO_INST RV1, INSTANCE RV2
WHERE  RV1.proto_name = 'l-bulb'
AND    RV1.inst_name = RV2.name
AND    RV2.att_val = 'capacity-more'
```

IDI sends this query to Oracle running on a remote server. Oracle searches through the database tables illustrated in Table 1. The first of these tables, `INSTANCE`, holds the components themselves. The second table, `PROTO_INST`, is a cross-reference table which provides a mapping from components to prototypes.

If Oracle finds a result, as it does in this example, it returns it via the method-specific knowledge compiler. In this case, the query generates the string "bigbulb" as the result. The prototype name and the value are also part of the result, but they are not explicitly returned by the database since they are the values used to select the database entry in the first place. The method-specific knowledge compiler converts the raw data from the database to a form comprehensible to INTERACTIVE KRITIK by using

²INTERACTIVE KRITIK makes use of both quantitative and qualitative values in its reasoning methods. The details of the interactions between these two kinds of information within the system are moderately complex and beyond the scope of this paper. Obviously, the experiment would be more realistic if the external database used quantitative values. This would add another step to the method-specific knowledge compilation process (mapping quantitative to qualitative values) but would not significantly affect the process as a whole.

TABLE 1. The tables for the Oracle database

Table INSTANCE			Table PROTO_INST	
NAME	ATTRIBUTE	ATT_VAL	INST_NAME	PROTO_NAME
littlebulb	lumens	capacity-less	littlebulb	l-bulb
bigmotor	watts	power-more	bigmotor	motor
bigbulb	lumens	capacity-more	bigbulb	l-bulb

the algorithm described in Section 5.1. In Substep 4.1, the string "bigbulb" is converted from a fixed length, blank padded string, as returned by Oracle, to a variable length string, as expected by INTERACTIVE KRITIK. In Substep 4.2, the attributes of the new bulb are generated. The values "bigbulb" and 'L-BULB' are used as the knowledge attributes name and prototype-comp; the values 'CAPACITY', 'LUMENS, and "capacity-more" are combined into a CLOS³ object of a class named parameter and a list containing this one object is created and used as the parameters attribute of the component being constructed. Finally, in Substep 4.3 these three attribute values are synthesized into a single CLOS object of the component class. The end result of this process is an object equivalent to the one defined by the following statement:

```
(clos:make-instance 'component
  :init-name      "bigbulb"
  :prototype-comp 'L-BULB
  :parameters     (list (clos:make-instance 'parameter
                                           :init-name      'CAPACITY
                                           :parm-units     'LUMENS
                                           :parm-value     "capacity-more"))))
```

These commands generate a CLOS object of the component class with three slots. The first slot contains the component name, the second contains the prototype of the component, and the third is a list of parameters. The list of parameters contains a single item which is, itself, a CLOS object. This object is a member of the parameter class and has a parameter name, the units which this parameter is in, and a value for the parameter. This object is then returned to INTERACTIVE KRITIK.

³CLOS stands for Common LISP Object System. CLOS is a mechanism within Common LISP which can be used to represent information in an object oriented framework.

Once INTERACTIVE KRITIK has received the description of the bulb, it is consequently able to apply the component replacement method. It replaces the bulb in the case retrieved earlier with the new bulb returned by the query processor. The effects of this substitution are propagated through the model and INTERACTIVE KRITIK verifies that the adapted model does accomplish the requirements which were initially specified. Finally, INTERACTIVE KRITIK presents the revised model to the user and stores it into the case-memory for further reuse in a later problem-solving session.

6. Discussion

Building knowledge systems for practical design requires careful analysis of many issues such as potential usefulness of the system to designers, usability and learnability of the system, accuracy and precision of the knowledge representations, and scope and scale of the system. Some recent research in design education seems to suggest that KRITIK-like SBF models are useful for enabling design students to organize, comprehend, and articulate design knowledge (Hmelo, 1997; Puntambekar and Hubscher, 1997). This educational work, however, does not use KRITIK or any other computer-based knowledge system. In parallel, we have been incrementally converting KRITIK from a laboratory system to a prototype tool for potential introduction in a design classroom. INTERACTIVE KRITIK, which we described in the last AI in Design conference (Goel et al., 1996b), provides a graphical explanatory "front-end" to KRITIK. HIPED attempts to provide a database "back-end" to INTERACTIVE KRITIK.

While the experiment described in Section 5 shows how data in a general-purpose design database can be compiled into a specific kind of knowledge required by a particular problem-solving method, it raises an additional issue. If the number of the problem-solving methods is large, and each method requires a knowledge compiler specific to it, then the HIPED architecture would require the construction of a large number of method-specific data to knowledge compilers. In the case of knowledge systems for design, which typically contain many problem-solving methods, this itself would make for significant knowledge engineering.

The issue then becomes whether we can identify primitive building blocks from which we can rapidly construct individual method-specific knowledge compilers. In the example discussed in Section 5, it appears that the three steps of the specific method for converting data into knowledge described can all reasonably be considered to be relatively generic units of functionality. Consider Substep 4.1 in the example, coercion of database types into knowledge system types: it is not unreasonable to expect that a wide variety of methods might have the same data coercion

requirements and thus be able to use the same data coercion routines in their method-specific knowledge compilers. Further, many knowledge systems for design use representations which are characterized as knowledge elements composed of a set of attribute-value pairs. The general framework for Substeps 4.2 and 4.3 of the algorithm (building attribute-value pairs and then combining them to form a knowledge element) probably can be applied to a wide variety of knowledge-based methods. Furthermore, to the extent that some methods have similar forms and mechanisms for constructing these elements, they might be able to share specific routines. Our experiment suggests that it may be possible to abstract generic components of method-specific compilations. Doing so may partially mitigate the problem of constructing large numbers of method-specific knowledge compilers as individual knowledge compilers might be built from a small and parsimonious set of components. But our experiments with HIPED have not yet demonstrated this; more research is required to fully explore this hypothesis.

The experiment described in Section 5 models only a small portion of the general architecture described in Section 2 and addresses only one of the applications of knowledge compilation presented in Section 3. In a related experiment, we have worked with another portion of the architecture (Navathe et al., 1996). Here, five types of queries that INTERACTIVE KRITIK may create are expressed in an SQL-like syntax. The queries are evaluated by mapping them into data using facts about the databases and rules that establish correspondences among data in the databases in terms of relationships such as equivalence, overlap, and set containment. The rules enable query evaluation in multiple ways in which the tokens in a given query may match relation names, attribute names, or values in the underlying databases' tables. The query processing is implemented using the CORAL deductive database system (Ramakrishnan et al., 1992). While the experiment described in this paper shows method-specific compilation of data into knowledge usable by INTERACTIVE KRITIK, the other experiment shows how queries from INTERACTIVE KRITIK can be flexibly evaluated in multiple ways.

The complexity involved in constructing knowledge systems for practical design makes integration of legacy knowledge systems and legacy databases an attractive option. But, insofar as we know, past research on information integration has almost exclusively focused on heterogeneity of information, not on heterogeneity of processing. However, third-generation knowledge systems for design are heterogeneous in that they use multiple problem-solving methods, each of which uses a specific kind of knowledge and control of processing. Thus the goal of constructing third-generation knowledge systems for practical design requires support for heterogeneity

of processing in addition to that of information. This itself is a hard and complex goal that requires long-term research. Building on earlier work on integrating knowledge systems and databases systems (Brodie, 1988; McKay et al., 1990), HIPED identifies some issues in supporting heterogeneous information processing and takes a first step towards achieving the goal.

Acknowledgements

This paper has benefited from many discussions with Edward Omiecinski. This work was funded by a DARPA grant monitored by WPAFB, contract #F33615-93-1-1338, and has benefited from feedback from Chuck Sutterwaite of WPAFB.

References

- Barber, J., Jacobson, M., Penberthy, L., Simpson, R., Bhatta, S., Goel, A. K., Pearce, M., Shankar, M., and Stroulia, E. (1992). Integrating artificial intelligence and multimedia technologies for interface design advising. *NCR Journal of Research and Development*, 6(1):75-85.
- Batini, C., Lenzerini, M., and Navathe, S. B. (1986). A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):325-364.
- Bhatta, S., Goel, A. K., and Prabhakar, S. (1994). Analogical design: A model-based approach. In *Proceedings of the Third International Conference on Artificial Intelligence in Design*, Lausanne, Switzerland.
- Brodie, M. (1988). Future intelligent systems: AI and database technologies working together. In Mylopoulos and Brodie, editors, *Reading in Artificial Intelligence and Databases*, pages 623-641. Morgan Kaufman.
- Brown, D. and Chandrasekaran, B. (1989). *Design Problem Solving: Knowledge Structures and Control Strategies*. Pitman, London, UK.
- Codd, E. (1970). A relational model for large shared data banks. *CACM*, 13(6).
- Finin, T. and Wiederhold, G. (1991). An overview of KQML: A knowledge query and manipulation language. Available through the Stanford University Computer Science Department.
- Genesreth, M. R. and Fikes, R. (1991). *Knowledge Interchange Format Version 2 Reference Manual*. Stanford University Logic Group.
- Goel, A. K., Bhatta, S., and Stroulia, E. (1996a). Kritik: An early case-based design system. In Maher, M. L. and Pu, P., editors, *Issues and Applications of Case-Based Reasoning to Design*. Lawrence Erlbaum Associates.
- Goel, A. K. and Chandrasekaran, B. (1989). Functional representation of designs and redesign problem solving. In *Proc. Eleventh International Joint Conference on Artificial Intelligence*, pages 1388-1394. Morgan Kaufmann Publishers.
- Goel, A. K., Gomez, A., Grue, N., Murdock, J. W., Recker, M., and Govindaraj, T. (1996b). Explanatory interface in interactive design environments. In Gero, J. S. and Sudweeks, F., editors, *Proceedings of the Fourth International Conference on Artificial Intelligence in Design*, Stanford, California. Kluwer Academic Publishers.
- Goel, A. K., Gomez, A., Grue, N., Murdock, J. W., Recker, M., and Govindaraj, T. (1996c). Towards design learning environments - I: Exploring how devices work. In Frasson, C., Gauthier, G., and Lesgold, A., editors, *Proceedings of the Third International Conference on Intelligent Tutoring Systems*, number 1086 in Lecture Notes in Computer Science, Montreal, Canada. Springer.
- Goel (ed.), A. K. (1991). Knowledge compilation: A symposium. *IEEE Expert*, 6(2).

- Gruber, T. R. (1993). A translation approach to portable ontology specification. *Knowledge Acquisition*, 5(2):199-220.
- Hennessy, D. and Hinkle, D. (1992). Applying case-based reasoning to autoclave loading. *IEEE Expert*, pages 21-26.
- Hmelo, C. (1997). Using structure-behavior-function models in design problems for science learning. Presented to the NSF Design Education Workshop, Georgia Institute of Technology, Atlanta.
- Keller, R. M. (1991). Applying knowledge compilation techniques to model-based reasoning. *IEEE Expert*, 6(2).
- Koch, G. and Loney, K. (1995). *Oracle: The Complete Reference*. Osborne/McGraw Hill/Oracle, 3rd edition.
- Lenat, D. and Guha, R. (1990). *Building Large Knowledge Based Systems: Representation and Inference in the CYC Project*. Addison-Wesley.
- Lenz, T., McDowell, J., Kamel, A., Sticklen, J., and Hawley, M. C. (1996). The evolution of a decision support architecture for polymer composites design. *IEEE Expert*, 11(5):77-83.
- Marcus, S., Stout, J., and McDermott, J. (1988). VT: An expert elevator designer that uses knowledge-based backtracking. *AI Magazine*, 9(1):95-112.
- McDermott, J. (1982). R1: A rule-based configurer of computer systems. *Artificial Intelligence*, 19:39-88.
- McKay, D., Finin, T., and O'Hare, A. (1990). The intelligent database interface. In *Proceedings of the Eight National Conference on Artificial Intelligence*, pages 677-684, Menlo Park, CA. AAAI.
- Mittal, S., Dym, C., and Morjaria, M. (1986). PRIDE: An expert system for the design of paper handling systems. *Computer*, 19(7):102-114.
- Navathe, S. B. and Donahoo, M. J. (1995). Towards intelligent integration of heterogeneous information sources. In *Proceedings of the 6th International Workshop on Database Re-engineering and Interoperability*.
- Navathe, S. B., Mahajan, S., and Omiecinski, E. (1996). Rule based database integration in HIPED: Heterogeneous intelligent processing in engineering design. In *Proceedings of the International Symposium on Coopering Database Systems for Advanced Applications*. World Scientific Press.
- Paramax (1993). *Software User's Manual for the Cache-Based Intelligent Database Interface of the Intelligent Database Interface*. Paramax Systems Organization, 70 East Swedesford Road, Paoll, PA, 19301. Rev. 2.3.
- Punch, W., Goel, A. K., and Brown, D. (1996). A knowledge-based selection mechanism for strategic control with application in design, diagnosis and planning. *International Journal of Artificial Intelligence Tools*, 4(3):323-348.
- Puntambekar, S. and Hubscher, R. (1997). A structure-behavior-function analysis of the design process. Presented to the NSF Design Education Workshop, Georgia Institute of Technology, Atlanta.
- Ramakrishnan, R., Srivastava, D., and Sudarshan, S. (1992). CORAL: Control, relations, and logic. In *Proceedings of the International Conference of the International Conference on Very Large Databases*.
- Stroulia, E. and Goel, A. K. (1995). Functional representation and reasoning in reflective systems. *Journal of Applied Intelligence*, 9(1). Special Issue on Functional Reasoning.
- Stroulia, E., Shankar, M., Goel, A. K., and Penberthy, L. (1992). A model-based approach to blame assignment in design. In Gero, J. S., editor, *Proceedings of the Second International Conference on Artificial Intelligence in Design*.
- Tong, C. (1991). The nature and significance of knowledge compilation. *IEEE Expert*, 6(2).

PART II

VISUALIZATION AND USER INTERFACE TECHNIQUES FOR INFORMATION RETRIEVAL

PART II: VISUALIZATION AND USER INTERFACE TECHNIQUES FOR INFORMATION RETRIEVAL

Our Objectives in this part of the project are multifold:

- To investigate the possibility of "free form" textual queries for the purposes of processing large amounts of textual information.
- To develop visualization and user interface techniques for users to improve their performance in large scale information retrieval tasks.
- To evaluate our approach experimentally to determine its merit.

All of the above goals were accomplished in this research. Additionally, a Ph.D. dissertation was completed [Veerasamy 97] under the supervision of Professors Sham Navathe and Scott Hudson.

A. Free form textual queries for text databases.

A large number of queries in information systems for day-to-day applications, industrial and government operations, as well as in military environments are made against text data in document databases. A majority of users are uninitiated in computer languages and need user interfaces for formulating queries and getting them answered by the system. They must ideally have interfaces available to them with free form text capabilities. In our research we decided to investigate this problem further and designed an approach to document retrieval/text retrieval that is unique in the following sense: a) it is not keyword based. b) it makes use of visualization and a thesaurus to improve upon the user's ability to formulate correct queries and c) it makes a provision of informing the user why a certain set of documents were retrieved and d) it provides the user with a means of influencing the system so that relevance of documents for subsequent iterative retrievals will be enhanced. As repository of text we used the data provided by the Text Retrieval Conference. [TREC] which is also similar to that from Linguistic Data Consortium of DARPA TIPSTER program. It contains news articles from AP newswire, Wall Street Journal, Department of Energy releases, etc. [see 2.3]. The queries considered used truly free form text : e.g., "How has affirmative action affected the public works projects undertaken by the construction industry?" The text retrieval engine called INQUERY was obtained from the University of Massachusetts and used in the experiments. The Wordnet thesaurus, available in the public domain was also employed in the design of the user interface.

- a. Efficiency and effectiveness of discovering relevant documents.
- b. Effectiveness in supporting query reformulation.

The experiments were performed by using 24 of the 25 TREC-4 topics on 36 subjects drawn from an undergraduate non-computer science major class at Georgia Tech. Because of the variability of topics, subject differences in the different groups, and subject-topic interaction, hard conclusions could not be drawn regarding the usefulness of visualization in contributing to both the above goals.

The details of Experiments 1 and 2 are given in [2.4]. The non-conclusive nature of the first two experiments led us to the design of experiments that were narrower in scope but that tested the effectiveness of the visualization tool that displays the results graphically as a first stage on the final outcome of the querying and on the query reformulation process. In [2.5] it is shown that the visualization tool helps users in identifying document relevance quicker by about 20%. This is made possible by set-at-a-time perusal of graphical displays rather than document-at-a-time perusal of textual displays. The experiments also showed that users with the visualization tool did significantly better in accurate identification of document relevance. The relevance judgement measure was broken into two measures for a better understanding: interactive precision and interactive recall.

While the effect of the visualization tool was marginally significant for interactive precision, it was highly significant for interactive recall. Thus, in [2.5] it is shown that the visualization tool helps users in identifying more relevant documents out of the displayed documents; it also helps users in identifying them more quickly.

Additional References:

A. Veerasamy, "Visualization and User Interface Techniques for Interactive Information Retrieval Systems," Ph.D. Dissertation, Georgia Institute of Technology, March 1997.

B. Visualization and User Interface Techniques

As a part of this work we developed a prototype that incorporated new techniques of visualization and user interfaces for textual information retrieval. [see 2.1, 2.2]. A ranked output information retrieval system is used as the basis - in our case it is the INQUERY system from the University of Massachusetts. We support querying, navigation and browsing on top of the retrieval engine in a seamless fashion [2.2]. There are two overall goals of this work. The first is to enable the user to understand why the top ranked documents are retrieved and ranked in that fashion. The second goal is to allow the user to influence the subsequent retrieval process by giving feedback to the retrieval engine to adjust the weighting of query words. The interface developed has potential applications in retrievals from digital libraries, any large corpuses of text, and on the world wide web. Following are the highlights of the user interface and visualization scheme described in [2.1] and [2.2]:

- The user interface facilitates a user in constructing complex structured queries by simple drag-and-drop operations.
- The visual feedback to the user in the form of a histogram (see examples in the paper) of each of the non-noise words in the query against the top 100 or 150 documents retrieved illustrates how the presence/frequency of different query words influences the ranking of the documents.
- An intuitive model where the user classifies a part of the information retrieved into positive and negative aids the user by supplying a rich feedback regarding his or her relevance criteria.
- To suit the continuously evolving and somewhat uncertain information needs of the user, the interface provides for navigational features such as browsing documents by specific authors, or browsing the table of contents of publications.

We see the utility of the above techniques not only in actual text retrieval but also in the retrieval of meta data or catalog information.

C. Experimental Evaluation and Validation

The prototype we developed was subjected to a thorough analysis at the TREC-4, (Text Retrieval Conference) [See 2.3] where we competed against about two dozen other prototypes for answering queries (topics) on the text databases provided by TREC. We then conducted four different experiments to test a variety of hypothesis with respect to our unique approach to user interface and visualization.

The first two experiments were designed to test the usefulness of the visualization tool to address two problems -

PUBLICATIONS (PART2):

- [2.1]. Visual Interface for Textual Information Retrieval Systems", Aravindan Veerasamy, Scott Hudson, Shamkant Navathe. In *Proceedings of IFIP 2.6 3rd Working Conference on Visual Database Systems* 1995, Elsevier, North Holland, pp. 333-345.
- [2.2]. "Querying, Navigating and Visualizing a Digital Library Catalog", Aravindan Veerasamy, Shamkant Navathe. In *Second International Conference on the Theory and Practice of Digital Libraries*, June 11-13, 1995, Austin, TX
- [2.3]. "Interactive TREC-4 at Georgia Tech", Aravindan Veerasamy. In *Fourth Text REtrieval Conference*, Oct, 1995, Gaithersberg, MD
- [2.4]. "Evaluation of a tool for visualization of information retrieval results", Aravindan Veerasamy and Nick Belkin. In *Proceedings of the SIGIR 1996, the 19th Annual International Conference on Research and Development in Information Retrieval.*, ACM, New York.
- [2.5]. "Effectiveness of a graphical display of retrieval results", Aravindan Veerasamy and Russell Heikes. In *Proceedings of the SIGIR 1997, the 20th Annual International Conference on Research and Development in Information Retrieval.* ACM, New York.

Visual Interface for Textual Information Retrieval Systems* †

A. Veerasamy, S. Hudson and S. Navathe

College of Computing, 801, Atlantic Drive, Georgia Institute of Technology, Atlanta, Georgia 30332-0280, USA.

Email: {veerasam, hudson, sham}@cc.gatech.edu

Abstract

A prototype user interface implementation for text information retrieval system is described. Using a visualization scheme, the interface provides visual feedback to the user about how the query words influence the ranking of retrieved documents. The interface also helps the user in constructing complex structured queries by simple drag-and-drop operations. An intuitive model where the user classifies the information provided to him/her as being positive and negative aids him/her in supplying rich relevance feedback information to the system. Our prototype interface has been built on top of INQUERY [CCH92]. Preliminary experience with the interface shows it to be a valuable tool in aiding the interactive search process between the user and the system. To test the effectiveness of the interface, we plan to conduct studies on users with real information need searching a large corpus of articles.

Keywords

Visualization of results, visual query languages, query processing, information retrieval

1 User Interface issues for Information Retrieval systems

User Interface issues and interaction techniques for full text information retrieval systems have in general received much less attention than system issues like document representation and retrieval algorithms. We have developed an interface that facilitates the user in visually constructing powerful queries for ranked output retrieval systems. The interface

*This work was supported in part by ARPA Grant No. F33615-93-1-1338 under the Intelligent Integration of Information Program

†Appeared in the Proceedings of the Third IFIP 2.6 Working Conference on Visual Database Systems, 1995

includes a scheme for visualizing the query results in a form that enables the user to see the relationships between the query results and the query. While a majority of online library catalog systems use a boolean model of retrieval, a vast majority of existing experimental information retrieval systems retrieve a ranked set of documents in decreasing order of relevance in response to a free-form textual query. In ranked output systems, the documents and the queries are modeled by a set of weighted index terms. The index term weighting function for the documents primarily takes into consideration

- the frequency of occurrence of the index term in the document,
- the number of documents in the corpus containing that index term.

The effectiveness of a retrieval system is measured by two metrics: recall (the ratio of the number of relevant documents retrieved to the total number of relevant documents in the corpus) and precision (the ratio of the number of relevant documents retrieved to the total number of documents retrieved). The reader is referred to [BC87, Rij79, SM83] for a comprehensive description of evaluation metrics of information retrieval systems, document representation and retrieval techniques.

While processing a free-form textual query, most ranked output Information Retrieval systems automatically extract index terms from the query and weight them. The weighted query index terms are then matched against the weighted index terms of documents to retrieve a ranked set of documents in decreasing order of relevance. Each document is weighted, the higher the weight of a document, the more likely it is to be relevant to the query. Most of the existing library information systems (On-line Public Access Catalogs, OPAC) follow a boolean retrieval model. In this model, the documents retrieved in response to a boolean query are not ranked. If a document satisfies the boolean query specification, it is retrieved. Compared to boolean systems, ranked output systems are a significant improvement since the query can be in a free-form text as opposed to a strict boolean syntax. Also, the retrieved documents are ranked, thereby placing the more useful documents at the top of the list. This is a particularly useful feature since it has been shown that users of boolean systems spend a considerable effort in reducing the size of the result set [Spi93]. On the other hand, ranked output systems introduce a new problem: For a naive user, the logic behind the ranking of documents in response to a query is not as apparent and straightforward as a boolean system. The interface we have developed is aimed at alleviating this problem. It helps the user in understanding how the system computed the ranking of retrieved documents by visualizing the relationship between query terms and the results of the query.

The interface also aids the user in formulating complex structured queries by graphically manipulating objects on the screen. A simple mechanism of classifying any information on the screen into positive and negative instances lends itself to easy formulation of structured queries. The interface is built using Tcl/Tk [Ous94] on top of INQUERY [CCH92], a ranked output retrieval system based on Bayesian inference networks. The interface supports two types of feedback:

- feedback from the user to the system and

- feedback from the system to the user.

It is interesting to note that the term “feedback” in the field of Information Retrieval typically refers to user’s feedback to the system, while in the field of Human Computer Interfaces, “feedback” usually refers to the system’s feedback to the user. The user’s feedback to the system and the different levels of granularity at which the feedback can be provided is discussed in section 4. The system’s feedback to the user and the visualization technique is discussed in section 5.

2 Related Work

Numerous studies on user interaction with online library access catalog systems with a boolean retrieval model have been conducted [Spi93, SS92, Dal90, Fid91a, Fid91b, Fid91c]. Spink [Spi93] studies the different forms of user feedback during a retrieval session. Of the total number of feedback actions by the user, 45% were aimed at adjusting the size of the retrieved set of documents, and about 40% were related to relevancy of documents. Fidel [Fid91a, Fid91b, Fid91c] discusses the issue of user interaction by studying the process of search term selection and searching styles in online library access catalogs. Dalrymple [Dal90] looks at the feedback process from a user-centered perspective. Bates [Bat90] describes a boolean retrieval system which integrates an online thesaurus. None of the above studies involve a ranked output system supporting free-form textual queries. All of the systems deal with boolean retrieval model only. We believe that there is a significant difference in the way users interact with a boolean system and a ranked output system. The reader is referred to [Har92] and [HB92] for a comparative discussion of boolean systems and ranked output systems. While building our interface we have borrowed valuable ideas from the studies mentioned above. In particular, the need to integrate an on-line thesaurus with the search interface in an easy-to-use fashion and a simple interaction scheme to include words from documents into the query have been influenced by the results of above-mentioned studies.

Walker and Beaulieu [Wal87, HB92] describe their OKAPI system which is a ranked output retrieval system for library catalogs. Similarly, Fox [FFS⁺93] describes their MARIAN system which is also a ranked output system for library catalogs based on the vector-space model. While OKAPI has facilities for relevance feedback and query expansion using a thesaurus, it largely lacks any means of providing system feedback to the user about how the ranking was computed. The interface we have developed integrates relevance feedback information from the user as well as feedback from the system illustrating the relationship between query results and query words.

A number of visualization schemes for information retrieval systems have also been proposed. The perspective wall [CRM91] describes a visualization scheme which supports browsing of documents. While such a system will not handle qualitative document classifications such as library subject catalogs, it is very useful for visualizing documents based on data which is linear in nature (like date of publication). Other visualization schemes

such as [Kor91, Spo94, HKW94] have facilities for viewing a large document space. But visualizing the document space along more than 3 - 4 dimensions simultaneously becomes very cumbersome using the above systems. Also, most of them do not provide support for querying with relevance feedback and none of them provide support for query expansion using a thesaurus. The visualization scheme in our interface can gracefully handle much higher number of query word dimensions.

2.1 Novelty of our approach

The novelty of our system is in integrating a diverse set of interaction features in a seamless fashion into a single system thereby facilitating the interactive and iterative nature of the information seeking process. The following features are integrated in our system:

- Using a visualization scheme, the interface provides visual feedback to the user about how the query words influence the ranking of retrieved documents.
- By simple drag-and-drop operations of objects on the screen, the interface facilitates a naive end-user in constructing complex structured queries and in providing relevance feedback. This feedback is utilized by the system in a manner described later.
- The interface integrates an online thesaurus which provides words related to the query that can be used by the user to expand the original query.

Belkin and his group's work [BMC93, BMA⁺91, HB94] on user interfaces for information retrieval systems elucidates the issues in user interface and interaction techniques for full text retrieval systems. Belkin [BMA⁺91] mentions that "This type of analysis led to another important conclusion, namely that information systems for end users must support a variety of goals and tasks, but through some common interface or seamless access mechanism to a variety of relevant information sources and system functionalities". Our interface takes a step in that direction by integrating different pieces of information with a visualization scheme and simple interaction techniques.

3 Interactive Construction of Queries

Searching a database for information is a highly interactive process with the user constantly refining the query after examining the results of previous iteration until he/she is either satisfied with the results or is frustrated with the process and gives up. In existing information retrieval systems, the interaction proceeds by the user providing feedback on which of the retrieved documents are relevant to his/her information need. The system uses this information to modify the original query resulting in an improved ranking of retrieved documents. It has also been shown by Spink [SS92] that during iterative query reformulation, users tend to expand the query using search terms from various sources

such as a thesaurus, previously retrieved documents and user's background knowledge. Expanding the query with terms from such sources can contribute to retrieval of more relevant documents in the next iteration.

Our interface encourages the interaction between the user and the system by providing the user with simple interaction technique to let him/her supply relevance feedback at different levels of granularity: whole documents, document portions, phrases and individual words. Almost any information appearing on the screen can be used for feedback. This is achieved by simple "drag-and-drop"ping the feedback object into either a "Positive Objects" window colored green or a "Negative Objects" window colored red. This scheme provides a simple abstraction to the user for classifying any type of information without having to worry about what action to take for what type of information. A typical user session along with the response of the interface for every user action is described below using an example (please refer to Figure 1). The database being queried contains a collection of titles, authors and abstracts of thousands of CACM articles.

- The user types in his free form textual query in the query window. In the example shown in figure 1, the query is "image audio and text data compression".
- As every query word is typed in, the system consults an on-line thesaurus and displays words and phrases related to the query word in an adjacent window.
- At any point during the session the user can drag-and-drop any of the related words/phrases into the positive and negative windows. Internally the system expands the query by treating the positive words/phrases as synonyms of the corresponding query word. The negative words/phrases are included in the query with a NOT operator. For example, if for a query word "bank", the phrase "financial institution" is classified as positive and "river bed" is classified as negative, the corresponding internal query would be "#SYNONYM(bank #2[†](financial institution)) #NOT(#2(river bed))". The end-result of this classification is a possible improvement in the precision measure since documents containing the phrase "river bed" will be weighted lower than other documents, and a possible improvement in the recall measure since documents containing the phrase "financial institution" are also retrieved. The interface facilitates construction of such structured queries by simple drag-and-drop operations. In the example in figure 1, three words related to the query word "compression", namely, "compaction", "shortening" and "condensation" have been classified as positive. Internally the systems treats these three words as synonyms of "compression".
- After the user types in the query, the system evaluates the query and displays the titles of top-ranked documents in the "Query Results" window.
- The user examines the query result. Double-clicking any title with the mouse will bring up the full document.

[†]#2 is the proximity operator in INQUERY specifying that the words should appear within a distance of 2 within each other

Querying, Navigating and Visualizing a Digital Library Catalog

Aravindan Veerasamy, Shamkant Navathe
College of Computing
801, Atlantic Drive
Georgia Institute of Technology
Atlanta, Georgia 30332-0280, USA.
Phone: 1-404-894-8791
E-mail: {veerasam, sham}@cc.gatech.edu

ABSTRACT

We describe the design of a User Interface for a ranked output Information Retrieval system that integrates querying, navigation and visualization in a seamless fashion. Highlights of the system include the following:

- Using a visualization scheme, the interface provides visual feedback to the user about how the query words influence the ranking of retrieved documents.
- By simple drag-and-drop operations of objects on the screen, the interface facilitates a naive end-user in constructing complex structured queries and in providing relevance feedback.
- To suit the evolving information needs of the user, the interface supports navigational features such as browsing documents by specific authors and browsing the Table of Contents of publications.
- The interface integrates an online thesaurus which provides words related to the query that can be used by the user to expand the original query.

By providing a rich set of features, the interface coherently supports a wide spectrum of information gathering tactics for different classes of users.

KEYWORDS: Visualization of results, visual query languages, query processing, information retrieval

WALK-THROUGH OF A TYPICAL USER SESSION

A typical user session along with the response of the interface for every user action is described below using an example (refer to Figure 1).

- The user types in his/her free form textual query in the query window. In the example shown in figure 1, the query is "ozone depletion and melanoma"
- As every query word is typed in, the system consults an online thesaurus and displays words and phrases related to the query word in an adjacent window.
- At any point during the session the user can "drag-and-drop" (using the mouse) any of the related words/phrases into the positive and negative windows. Internally the system expands the query by treating the positive words/phrases as synonyms of the corresponding query word. The negative words/phrases are included in the query with a NOT operator. For example, if for a query word "bank", the phrase "financial institution" is classified as positive and "river bed" is classified as negative, the corresponding internal query would be "#SYNONYM(bank #2¹(financial institution)) #NOT(#2(river bed))". The interface facilitates construction of such structured queries by simple "drag-and-drop" operations of the mouse. In the example in figure 1, a phrase, namely "skin cancer" that is related to the query word "melanoma" has been classified as positive. Internally the systems treats the phrase as a synonym of "melanoma".
- After the user types in the query, the system evaluates the query and displays the titles of top-ranked documents in the "Query Results" window.
- The user examines the query result. Clicking any title with the mouse will bring up the full document.
- Figure 2 is a visualization of the query results for the base query "ozone depletion and melanoma". The leftmost column of bars corresponds to the top-ranked document, with the columns progressing to the right representing progressively lesser ranked documents. We can see that almost all of the 150 documents were retrieved because they contained the query words "ozone" and "depletion". Only 15 of the top 150 documents have anything to do with melanoma. Further, of those

¹#2() is the proximity operator in INQUERY specifying that the words inside braces should appear within a distance of 2 of each other in the document.

15 documents, only one discusses ozone (the top-ranked document – leftmost column in Figure 2.) Thus we can clearly see that either there are not many documents dealing with melanoma and ozone or the ozone-layer concept drowns out melanoma during retrieval.

- The user can classify any document as being relevant or non-relevant by “drag-and-drop”ping the document into positive and negative windows. In the example in figure 1, the user has classified two documents titled “CFC-free integral skin foams for steering wheels.” and “Video comparator system for early detection of cutaneous malignant melanoma” as positive. The document titled “Symposium on chemistry of the Atmosphere” has been classified as negative.
- The user can also highlight a portion of a document and “drag-and-drop” that portion into the positive and negative windows. The words in the highlighted document portion are used to expand the query in the next iteration.
- During the next iteration, the reformulated query with the feedback information is processed by the system resulting in an improved ranking of documents.
- Figure 3 is a visualization of the results of the revised query (i.e., the query with relevance feedback information). The figure shows that there are four documents dealing with melanoma and ozone. (Note that the documents which deal with melanoma and it’s synonym skin cancer are displayed in the same histogram titled “melanoma”, since melanoma and skin cancer represent the same query concept). Thus there are three additional documents retrieved due to the effect of classifying the phrase “skin cancer” as a synonym of “melanoma”. But still there are not many documents about melanoma compared to ozone depletion. Our experience with this visualization scheme has shown it to be a useful tool for identifying different facets of the query, as in this case, the facets are melanoma and ozone.
- Using any document as a starting point, the user can browse through the list of other articles in the same journal issue or conference proceedings with a help of a Table-of-Contents which is generated automatically. This is useful in many cases such as when the user comes across a special-issue of a journal devoted to the search topic.
- The user can also browse through the list of articles written by the same author. For example, an author who has written an article about the effects of ozone layer depletion on skin cancer has probably authored more articles along the same lines, and the user might want to see them.

CONCLUSION & FUTURE WORK

A prototype interface [4] written in Tcl/Tk [3] using a ranked output information retrieval system, INQUERY [1] for a library catalog, Compendex containing about 300,000 documents has been implemented. The interface facilitates the inherently interactive nature of the information seeking process. “Drag-and-drop” operations (using the mouse) form the basis of interaction encouraging the user to provide feedback information to the system and helps in the dialog between the user

and the system. Almost any information on the screen can be used by the user to provide feedback information. An on-line thesaurus, WordNet [2], is integrated with the interface to form a single system.

The interface also supports a visualization scheme which illustrates how the query results are related to the query words. Visualizing the results of the query keeps the user more informed on how the system computed the ranking of documents. With this information, the user is better equipped to reformulate the query for the next iteration. The interface also has facilities to browse the Table of Contents of publications and to browse the list of articles written by a specific author. It is our opinion that integrating all of the above features in a seamless interface leads to an interplay between different items that is much more beneficial than the sum of the individual items in isolation.

We are in the final stages of implementation, and in future, we intend to test the effectiveness of the interface by conducting studies on how library users, experts looking for detailed information as well as naive users, interact with the interface and how they react to ranked output systems as opposed to existing boolean systems. We plan to include a domain-specific thesaurus for the engineering domain from Compendex and a collection-specific word-association thesaurus if possible.

ACKNOWLEDGEMENTS

We are thankful to Dr. Bruce Croft for letting us use the INQUERY retrieval system. We are indebted to the Dean of Georgia Tech Library Ms. Miriam Drake and Engineering Information Inc without whom it would have been impossible to use Compendex data for the experiment. Many thanks to Dr. Marti Hearst whose Tcl/Tk code for the SMART system was helpful as a spring board for us to write the interface. Support in part by ARPA contract No. F33615-93-1-1338 is also appreciated.

REFERENCES

1. J.P. Callan, W.B. Croft, and S.M. Harding. The inquiry retrieval system. In *Third International Conference on Database and Expert Systems Applications*, September 1992.
2. George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. Introduction to WordNet: An on-line lexical database. *Journal of Lexicography*, 3(4):235–244, 1990.
3. John K. Ousterhout. *Tcl and the Tk Toolkit*. Addison-Wesley, 1994.
4. A. Veerasamy, S. Navathe, and S. Hudson. Visual interface for textual information retrieval systems. In *To appear in Proceedings of the Third Conference on Visual Database Systems*. IFIP 2.6, 1995.

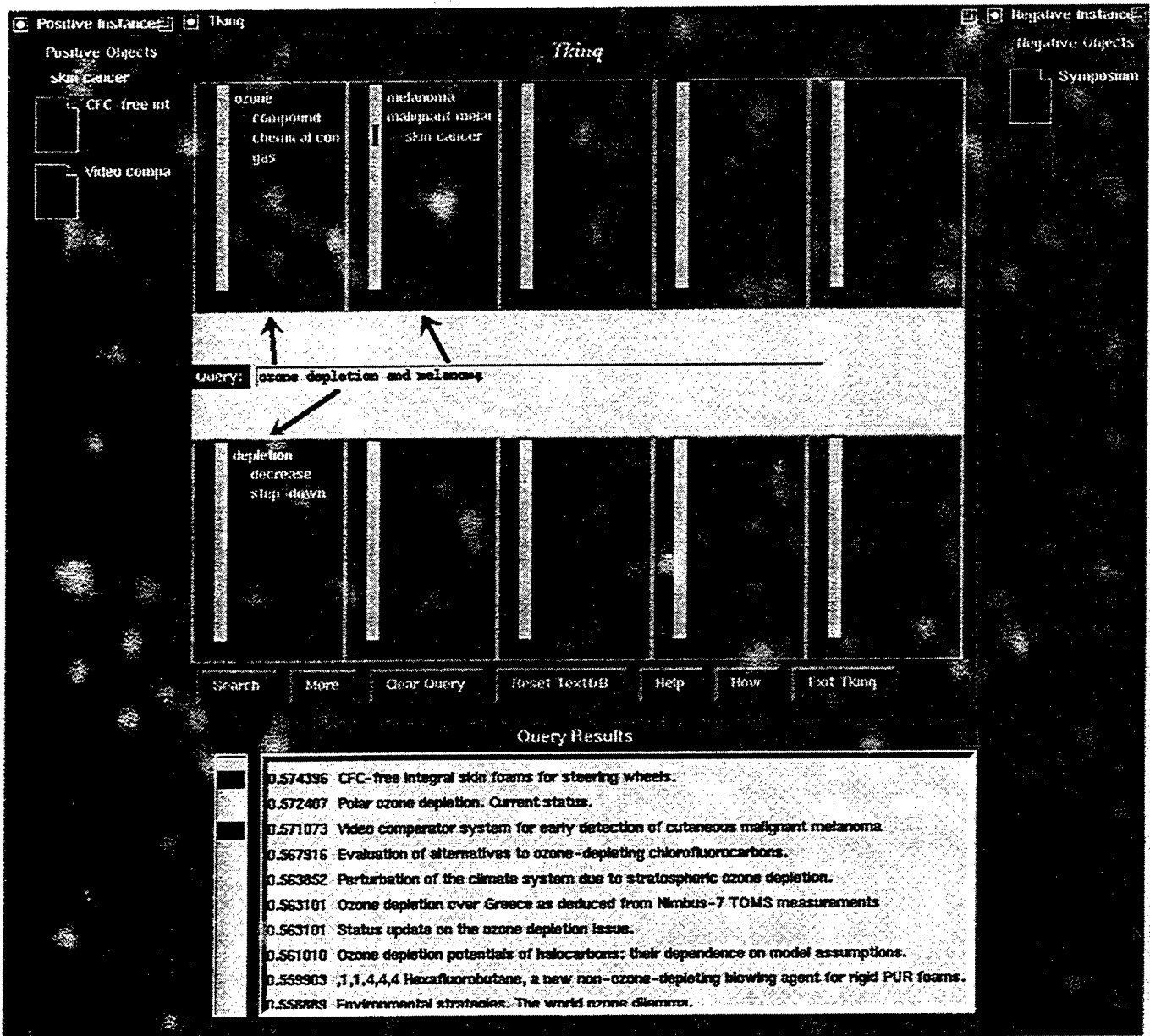


Figure 1: Sample querying session. The window titled "Positive Objects" is colored green and the window titled "Negative Objects" is colored red. All "incantations" of an object in the display are colored green/red whenever it is classified as positive/negative.

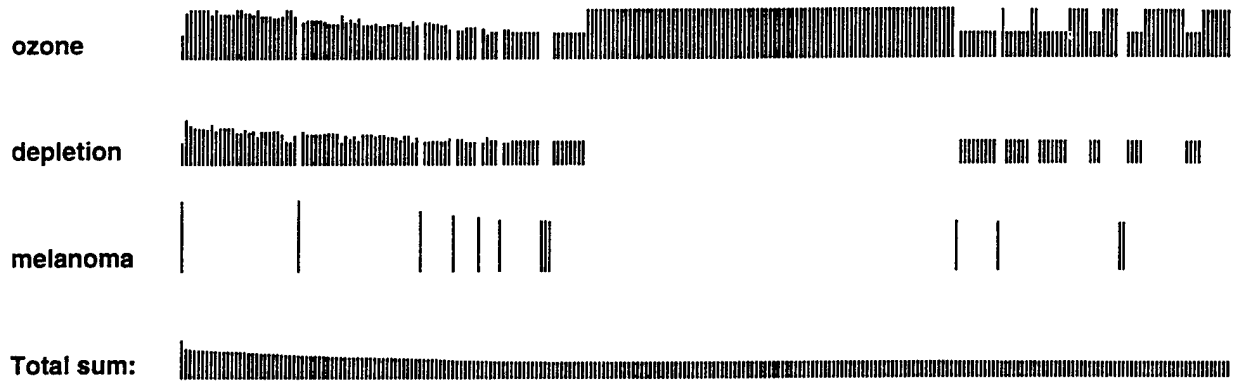


Figure 2: Visualization of results for the base query.

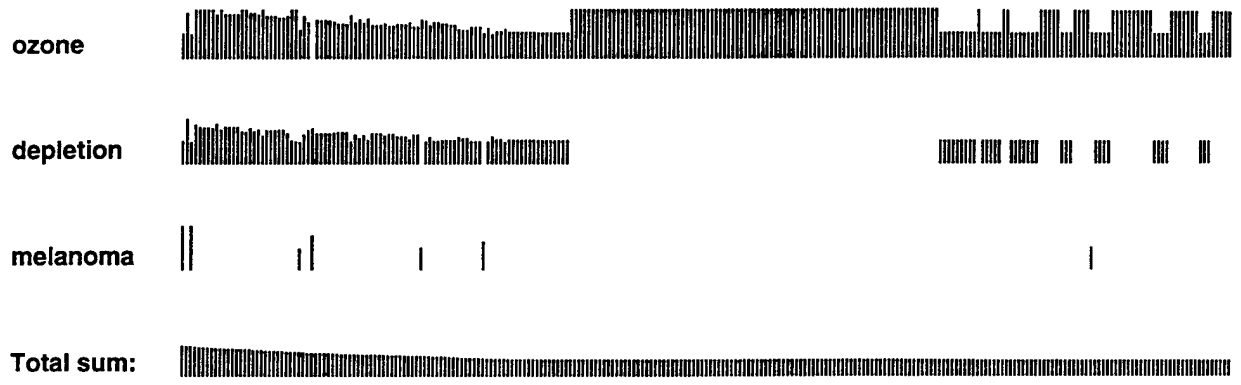


Figure 3: Visualization of results for query with feedback information.

Interactive TREC-4 at Georgia Tech

Aravindan Veerasamy
veerasam@cc.gatech.edu
College of Computing
801, Atlantic Drive
Georgia Institute of Technology
Atlanta, Georgia 30332-0280
Phone: 404-894-8791
Fax: 404-894-9442

Abstract

At Georgia Tech, we investigated the effectiveness of a visualization scheme for Information Retrieval systems. Displayed like a bar-graph, the visualization tool shows the distribution of query words in the set of documents retrieved in response to a query. We found that end-users use the visualization for two purposes:

- to gain specific information about individual documents – such as the distribution of different query words in that document.
- to gain aggregate information about the query result in general – such as getting a sense of the direction of the query results.

In general they used the visualization tool as much as the title and full text in the process of deciding if a document addresses the given search topic. In structured post-session interviews with searchers, we also obtained information about what the searcher liked, what was frustrating to them, and what they wanted in the system.

1 Introduction

At the TREC-4 interactive experiments at Georgia Tech, we were interested in investigating the effectiveness of a visualization scheme for IR systems that we have developed. The visualization scheme, as given in Figure 2, is intended to provide more information to the user about the query results in addition to just the title and full text. In ranked output systems, the naive end-user has little knowledge about why the system retrieved and ranked the documents in a given way in response to a free-form text query. This problem does not arise in boolean systems since there is no element of surprise in why a particular document was retrieved. The above-mentioned lack of knowledge in ranked output systems can be quite disturbing when a user is not able to get the set of documents he/she needs and does not know enough about the system

to modify the query to get the documents he/she needs. It is with this in mind that we have developed a visualization scheme that shows the distribution of query words in the retrieved documents. This visual display of distribution information provides a good overview of the retrieved set of documents with respect to the free-form user query.

For TREC-4 we were interested in investigating how end-users used the visualization scheme. We were also interested in finding what aspects of the system were frustrating, what aspects they liked and what they wanted in the system. We have yet to do a thorough statistical analysis of the trace data to quantitatively determine the ways in which users with visualization tool acted different from the users without the visualization tool. What we report here is our observations of user interactions, information from structured interviews, and questionnaires.

In the next section we give a brief description of our system. Then we describe our experimental design followed by our observations as it relates to the visualization tool. Then we discuss user's frustrations, likes and wants.

2 System Description

For our study, we used the INQUERY retrieval engine from University of Massachusetts, Amherst [CCH92]. We built a simple graphical user interface on top of INQUERY using Tcl/Tk [Ous94]. There are two versions of our system – one with the visualization, and one without. In our base system, as shown in Figure 1, there are three windows: the top left window is for entering and editing the query. The titles of retrieved documents are displayed immediately below that window. Thirty titles can be displayed in one screen. One can scroll down to a maximum of 150 document titles. Mouse-clicking a title brings up the full text of that document in the window at the bottom right. By clicking the “Next Query Word” button in the full text window, one can position the full text display such that the next occurrence of query word in the document is at the top of the window.

One can save documents and mark documents for relevance feedback by clicking the “Save?” and “Rel?” buttons immediately to the left of the title in the title display window. The only operator that is allowed is the adjacency operator: A hyphen between two words specifies that the two words must appear right next to each other in the same order in a document in order for the word-combination to contribute to the retrieval of that document. There is no negation operator. Automatic stemming and stopping are performed.

The visualization tool is displayed in another window as shown in Figure 2. It

consists of a series of vertical column of bars. There is one column of bars for each document. The leftmost vertical column of bars corresponds to the document ranked 1 and the rightmost vertical column corresponds to the document ranked 150 with all the intermediate ranks lying in between. In each vertical column there are multiple bars – one each for each query word. The height of the bar at the intersection of query word row and a document column corresponds to the weight of that query word in that document. Thus if there are a handful of query words that convey the crux of the query and is very important for a document to contain these query words, one can quickly see from the visualization which retrieved documents have those important words. One can also see how many of the retrieved documents have those words in combination to get a feel for the overall goodness of query results. The effects of modifying the query, like adding a query word, would clearly be shown in the visualization. One can quickly take stock of how useful the query modification turned out. Moving the mouse cursor over the vertical columns would highlight the column directly beneath the mouse cursor and simultaneously highlight the title corresponding to that document in the title display window.

Apart from the query words typed in by the user, the visualization also shows the distribution information for words added by the system due to relevance feedback. In summary, all the words internally used by the system in computing the query results are shown in the visualization. The words in the visualization are also stopped and stemmed.

3 Experimental Setup

The searchers for our study were undergraduate student volunteers from a course on library searching at Georgia Tech. All the searchers had prior computer experience – a majority of them more than 4 years. All the students were majoring in an engineering discipline. They had differing levels of experience with the Georgia Tech Electronic Library catalog – a boolean online public access catalog.

All the users were asked to fill out a background questionnaire. They were given a tutorial on how to use the system. They were then asked to do a practice search on topic 224 for 15 minutes. Following that they were asked to find as many documents as they can that address the given information problem without too much rubbish (as specified by the interactive track guidelines). This was followed by another intermediate tutorial and then a search for a second topic. Immediately after each of the two real searches, they filled out a search evaluation questionnaire. Finally, there was a structured interview.

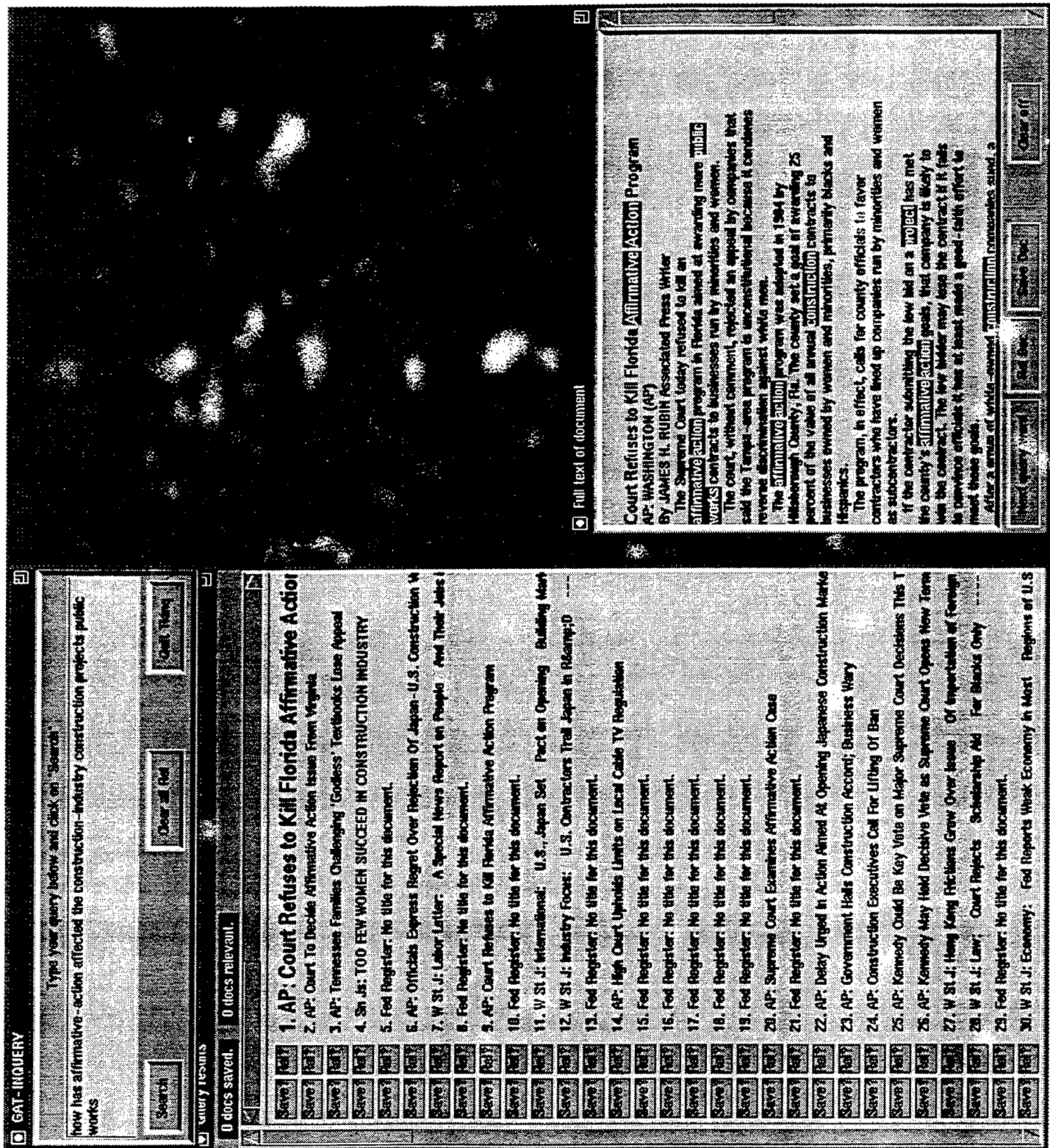


Figure 1: Sample querying session. The window in the top-left corner is the query entry window. Immediately below that is another window where the titles of retrieved documents are displayed. To the bottom right is another window where the full text of documents are displayed.

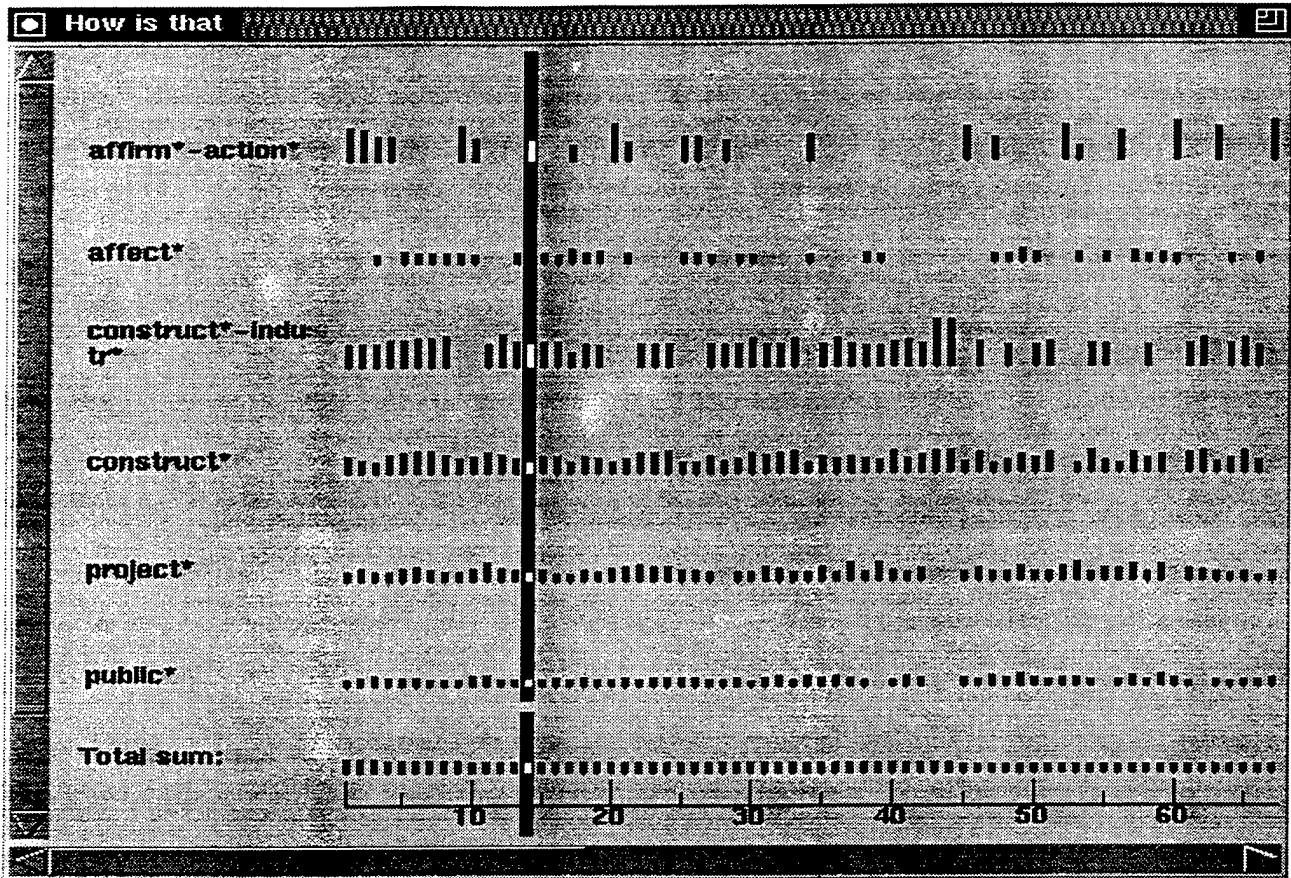


Figure 2: Visualization of results. The highlighted vertical column corresponds to document ranked 14. The title of document ranked 14 document will also be highlighted in the title display window. Clicking the highlighted vertical column brings up the full text of that document.

The searchers were divided into three groups. In each group there were 12 searchers. In the first group (hereafter named "w:w", since both first and search topics are searched WITH visualization), the searchers used the visualization tool for all the searches and the tutorial. In the second group (hereafter named "wo:w", since the first search topic is searched WITHOUT and second topic WITH visualization), the initial tutorial, the practice search and the first search was done without the visualization tool. The intermediate tutorial introduced the visualization tool and the search for the second topic was done with the visualization tool. In the third group (hereafter named "wo:wo", since both the search topics are searched WITHOUT the visualization), all the tutorials and searches were done without the visualization tool. The intermediate tutorial for the w:w and wo:wo groups was a dummy tutorial to compensate for the intermediate tutorial of the wo:w group.

Since each searcher searched for two topics and there were 12 searchers in each group, all the 24 topics were covered by each of the three groups. The 24 topics were randomly divided into 12 pairs and each pair was searched by 3 searchers, one each from the w:w, wo:w and wo:wo groups. The idea was to compare the performance among the three groups to find out the effects of the visualization scheme. Only 24 of the 25 topics for the interactive track were given to end-users in the study. The remaining one topic (topic 223) was searched by the author using the visualization tool.

The searchers were asked to think aloud as they used the system. For the most part, there was an observer in the same room using a different computer and simultaneously observing the searcher. Based on such observations while the user session was in progress, we felt that huge searcher differences in interpreting the query combined with huge differences in the nature of the search topics will greatly confound the effects of the visualization tool. As a result, we decided to run a second study. In the second study, we picked topic 242 for the practice search and the practice search was extended to 30 minutes. The intermediate tutorial was removed. We picked topics 203 and 236 for all the searchers. There were two groups of searchers for the second study - the first group had the visualization tool and second group did not have the visualization tool. There were 18 searchers in each group. By keeping the two search topics constant for all these searchers, we expected to eliminate the effects of search topic difference. It turns out that the searcher variability in interpreting the search topic is so huge among searchers that it is not fair to compare different searchers using different systems unless the search topic is extremely clear and specific.

4 End-users view of the visualization tool

A vast majority of the users mentioned visualization as one of the aspects of the system that they liked. They mentioned using the visualization tools in the following ways.

- Some searchers mentioned using it to see the importance of query words in the retrieved documents – as given by the height of the bar. They mentioned that they were more likely to look at the full text of a document if it has a higher concentration of the important query words.
- Most of the searchers mentioned using it most frequently to see the co-occurrence of important query words in the retrieved documents. They mentioned it being easier to use the visualization tool to look for the co-occurrence information than going through the full text of documents in search of occurrences of the important query words.
- Many searchers felt that the visualization in conjunction with the document title gives a fairly good idea of what the document is about. If the title looks promising and the visualization shows that the document has the right combination of query words, one is tempted to look at the full text of the document.
- They mentioned using it to get a quick overview of the number of retrieved documents a query words appears in. They mentioned using it as a checkpoint to see if a query has turned out the way they had expected it to. If not, they were tempted to readjust the query to get a better result. This happens often when some of the crucial query words are not well represented in the retrieved documents. In that case, one is tempted to add synonyms or words related to those crucial query concepts.
- Some of the searchers mentioned that the visual nature of the distribution information was much easier to identify things than reading text information. This suggests that the mental effort of reading textual information as being much higher than interpreting a simpler visual pattern, and given a choice, the users are more likely to choose the latter.
- **Disadvantages:** A few searchers mentioned that relying heavily on the visualization can also hurt as follows: They mentioned that using the bar-graph to pick out a document containing certain query words may not be indicative of the content of the document – just as the title may not be a good indicator of content. An exemplar case is searcher 35 on the topic of “status of nuclear proliferation treaties”. Since almost all of the retrieved documents had something

to do with “nuclear proliferation”, the searcher mentioned using the visualization tool to pick those documents containing the query word “status” – only to see that the usage of “status” in the document was not in the context of nuclear proliferation treaties. Then the searcher started paying little emphasis on the presence of “status” in documents. Although relying on that information was initially detrimental, one tends to learn when and how to rely on the visualization. We believe that the presence in retrieved documents of adjectives, adverbs and verbs from the query may not be good content indicators especially when they have a high collection frequency. And relying on the visualization to select documents that have these adjectives, adverbs and verbs from the query may not help.

In summary, the visualization tool seems to help in the following ways:

- to gain more information about specific documents in addition to the title before looking at the full text. Higher concentration of important query words in a document suggests a closer look at the document.
- to gain aggregate information about the query result. The absence of important query words in a vast majority of the retrieved documents suggests query reformulation by adding synonyms and other related concepts.

5 Likes, Frustrations and Wants of users

Apart from the visualization, we were also interested in finding if there are any specific facilities that the users wanted, what features they liked, and what aspects were frustrating. While interpreting the following, we wish to reiterate that all the searchers had some amount of experience with the Georgia Tech Electronic Library catalog which is a character-based-command-driven interface to a boolean system. Some of the features they liked may arise out of the fact that they have had little experience with ranked output systems and the only other major information retrieval system they know is a character based interface to a boolean system.

5.1 Likes

- A vast majority of the searchers with the visualization mentioned that the visualization tool and relevance feedback as the two major aspects of the system they liked. Searchers without the visualization mentioned relevance feedback as the most important feature they liked.

- A number of searchers found the fact that all the information (like the user query, titles of documents and the document full text) is displayed simultaneously in one screen to be very useful. In the Georgia Tech library system, one has to switch between screens to get different types of information. There seems to be a significant mental overload in the context switch between screens. Having simultaneous access to all information seems to bring about a rich interplay between the different sources of information.
- many searchers mentioned that the mouse-based graphical nature of the interface is a significant improvement over a command line based interface.
- many searchers also mentioned that the free-form textual queries without having to worry about any syntax leads to a free flow of thought. "I like the fact that I can type in whatever comes to my mind ... knowing that it will ignore all the junk words like a, an, the, etc...".
- The "Next Query Word" feature was also liked by many searchers. They liked it because they did not have to scroll through a long document looking for occurrences of query words. (All the occurrences of query words in a document are highlighted by the system).

5.2 Frustrations

- A number of searchers mentioned that it was frustrating when the system takes a long time to get the full text of a large document. Similarly, they were also frustrated when it takes a long time to evaluate a query with a large number of relevance feedback documents. The longest delay for evaluating a query was about 2 minutes (when there are about 30 relevant documents). Most of the query evaluations took less than 20 seconds. They said that they understand that the system has to process a lot information (when there a number of relevant documents), but it was frustrating nevertheless.
- Some searchers said that it was frustrating to spend some time reading through the full text of a document and when they are halfway, realizing that they had already seen the same/similar document.
- While some searchers seemed to like having access to 150 retrieved documents, some others mentioned that 150 documents is too much especially when most of the 150 are not relevant. They seem to have the opinion that if some documents are definitely not relevant to the query, then they should not be shown. Thus, this problem is not alleviated even if one reduces the number of documents

displayed. They seem to be quite sensitive about precision. They are not as sensitive about recall – since they are usually satisfied if they get a few documents concerning the topic. Based on our observations, we believe that when the non-relevant documents consistently come from a particular subject area, and when the user is not in a position to remove those documents, they tend to get more frustrated. Using subject classification schemes (where available) to negate disinteresting subject areas would help in this regard.

- In our system, when the title for a document is not available, the message “No title for this document” is displayed instead of the title in the title display window. Many of the federal register documents do not have a title and this is quite annoying to some searchers since they do not have any idea about the document content. This makes it difficult to decide whether to request the full text or not. In cases where the full text is requested, the document happens to be large and hence takes a lot of time to retrieve, thereby adding to the frustration.
- Some federal register documents do not have anything worthwhile – they consist of a listing of subject areas or table of contents. Some searchers wondered why these documents were in the database in the first place.
- Some searchers mentioned a general dislike towards federal register documents partly because they felt that many of them did not have any important piece of information, partly because in general they have no title, partly because it took too long to retrieve them.
- Some searchers were frustrated when a document that they know as non-relevant keeps coming up in the query result. The fact that they were not able to delete the document from the display seemed to add to the frustration.

5.3 Wants

Many of the frustrations mentioned above seemed to directly translate into wants for removing the causes of frustration. In addition to those wants, we observed the following:

- Many searchers expressed a desire to remove certain query words that were added by the system from relevance feedback documents – especially when they are proper names and when they are not necessarily what they are looking for.

- A number of searchers wanted a keyboard equivalent of mouse actions. This is not to say that they did not want mouse actions. It seems to be a significant effort for these searchers to move the right hand out of the keyboard, reach over to the mouse, look at the screen to position the mouse cursor, click the mouse button and move back to the keyboard.
- When asked if they felt a need to have access to an online thesaurus, some searchers expressed a desire for it and some did not. Some of those who did not want a thesaurus mentioned that relevance feedback seemed to alleviate the need for a thesaurus.
- Many searchers wanted to be able to specify that the system should definitely avoid retrieving certain documents in subsequent query iterations. They wanted to have a negative relevance feedback where the system avoids all documents like a particular nonrelevant document.

6 Acknowledgments

The tremendous help from Prof. Nick Belkin regarding the experimental setup and questionnaire design is highly appreciated. Many thanks to Prof. Scott Hudson and Prof. Shamkant Navathe who were instrumental at every stage of the interface development. We appreciate the help of Prof. Jan Crowe who was very cooperative in letting the students of her class participate in the experiments. Special thanks to Prof. Bruce Croft for letting us use the INQUERY retrieval system. Support from the ARPA contract No. F33615-93-1-1338 is appreciated.

References

- [CCH92] J.P. Callan, W.B. Croft, and S.M. Harding. The inquiry retrieval system. In *Third International Conference on Database and Expert Systems Applications*, September 1992.
- [Ous94] John K. Ousterhout. *Tcl and the Tk Toolkit*. Addison-Wesley, 1994.

Evaluation of a Tool for Visualization of Information Retrieval Results

Abstract

We report on the design and evaluation of a visualization tool for Information Retrieval (IR) systems that aims to help the end user in the following respects:

- As an indicator of document relevance, the tool graphically provides specific query related information about individual documents.
- As a diagnosis tool, it graphically provides aggregate information about the query results that could help in identifying how the different query terms influence the retrieval and ranking of documents.

Two different experiments using TREC-4 data were conducted to evaluate the effectiveness of this tool. Results, while mixed, indicate that visualization of this sort may provide useful support for judging the relevance of documents, in particular by enabling user to make more accurate decision about which documents to inspect in detail. Problems in evaluation of such tools in interactive environments are discussed.

1 Introduction

The disadvantages of Boolean IR systems, especially in terms of end-user query formulation, are well known. Best-match (i.e. ranked output) systems address several of these problems by allowing users to submit unstructured queries, and by ranking the retrieved documents in (presumed) order of relevance. However, such systems also

introduce new problems, or exacerbate problems that are not so severe in Boolean systems.

For instance, understanding why a document (or set of documents) was retrieved is relatively straightforward in exact-match systems, since all members of the set are required to contain exactly the query specification. Furthermore, the ordering within the set of retrieved documents is typically based on relatively well-understood formal characteristics of the documents, such as date of publication or alphabetical order by title or author. In best-match systems, on the other hand, neither the matching rule nor the ranking rule is necessarily easily understandable. The former is usually based on characteristics and algorithms which don't have simple relationships with the unstructured query; the latter are intended to reflect complex conceptual relationships between the query and the individual documents, and between the documents themselves.

Furthermore, query reformulation may be more difficult in Boolean than in best-match systems. Obtaining a manageable output set size in Boolean systems (the most typical re-formulation task) may be less demanding than attempting to rearrange the list of retrieved documents in a best-match system by manipulating an unstructured query. This is of course especially difficult when the rules for ordering and matching are not well understood.

Current best-match IR systems take relatively little account of these issues. In response to a user's query, most systems display surrogates (title, source, author ...) of the top 'n' retrieved documents, in a list, with some number(s) indicating the rank, or reason for being in that rank, i.e. a retrieval status value (RSV). Some systems display by default more information about the first retrieved document; most require

the user to request such information (e.g. the full text of the document) explicitly. The only explanation of *why* the documents are ranked the way they are is typically the RSV, about which there is no further information than the number itself. More explanation may not be necessary in situations where the top retrieved documents are all clearly relevant. But when this is not the case, and the user needs to modify the query in order to get better results, then understanding the causal relationship between query and document ranking becomes very important. Having an accurate idea of *why* a list of documents was retrieved, of *how* they were ranked, and of *what* is sub-optimal about the ranking could be useful in effective query reformulation.

Of course, knowledge about the relationships between query and ranking of retrieved documents is not of itself sufficient for effective query reformulation. It is also necessary that the user be able to effectively manipulate the query after the problem has been identified. For instance, just knowing that an important query concept is missing in most of the retrieved documents is not sufficient for effective query reformulation. One must then be able to come up with the right words (or other techniques) for increasing the importance of the concept in the query. Without the ability to take corrective action once the problem is diagnosed, the diagnostic information is of little value.

A possible means for addressing problems of this sort is to display to the user something about the documents which relates them directly to characteristics of the query, and which relates them to one-another. Highlighting query terms in the text display of retrieved documents attempts to accomplish the former, and the indication of RSV is an attempt to accomplish the latter. However, neither of these techniques appears to give sufficient information to guide effective query reformulation. More

information of each type needs to be displayed in order to provide effective support for this task. Graphical displays of the characteristics of retrieved documents (*visualizations*) which are relevant to their retrieval and ranking is one obvious approach to this problem.

A further problem in IR systems in general has to do with the multi-stage nature of presentation of results. The initially-presented surrogates are meant to provide a concise picture of what a document is about. Based on these surrogates, the user may request more detailed information about particular documents which look promising (or for which the surrogate information is equivocal). In some cases, this might be the "full" bibliographic information about the item, in others an abstract, and in many systems now, it could be the full text of the item. Thus, as the user progresses through the stages of display, that which is displayed is more complete and informative, allowing increasingly accurate relevance judgments. But, the information in the later stages of display is also more time-consuming to peruse. Therefore, it is useful for the searcher to be reasonably certain that it is worthwhile doing this inspection. The information displayed in the earlier stages thus serves as a filter which supports the user in deciding which documents do not need further inspection (either because they are obviously good or obviously bad), and which documents do justify the further effort.

It seems, then, that displaying a great deal of information at the surrogate stage of display would be a useful device. In this case, the user has more information on which to judge the relevance or usefulness of the document. The advantage is that when the user requests the second-stage display, it is more likely that that document will be relevant to the user than if there were less information in the first stage. The

disadvantages to this strategy, of course, are that since there is more text to display in the first stage, fewer items can be presented, and more time must be spent in perusing the first-stage display. Thus, the total number of documents seen by the user may well be fewer, although the quality of the decision-making may be higher.

If, on the other hand, one chooses to display less information at the initial surrogate stage, then there is of course less information on the basis of which one can make a decision about whether to look at the more complete display. Hence, the proportion of second-stage documents which turn out to be relevant is likely to be low. The advantage of seeing more documents, more quickly, in the first stage is thus offset by the additional time that is spent perusing non-relevant documents in the second stage.

A possible means to addressing this problem is to display information about the document in the first stage in some form that does not require as much perusal time and screen space as text. Graphical displays of the characteristics of documents which are significant in supporting the decision to peruse or not (*visualizations*), could support set-at-a-time perusal of documents, rather than document-at-a-time perusal of text displays.

It will not escape the reader that the suggested solutions to the two classes of problems that we have raised here are rather similar, and could, indeed, be instantiated by the same sort of display. We present here a visualization tool which is intended to address these problems in IR systems, and a preliminary evaluation of this tool. The remainder of this paper is organized as follows. We first present a description of the visualization tool, and a rationale for the features of this tool with respect to the problems in IR interaction that we have discussed above. We then discuss

some related work in IR visualization that addresses this type of problem, and draw some comparisons between that work and ours. We follow with a description of the experiments we conducted to evaluate the visualization tool, and the results of those experiments. We conclude with some comments on the implications of our results, on future work, and on the implications of our evaluation experience for evaluation of interactive IR in general.

2 Visualization tool

In this section, we briefly describe the visualization tool and then discuss how its features are intended to help the end user in selecting relevant documents and in formulating better queries (resulting in more optimal document ranking).

2.1 Description

The visualization tool is an adjunct to a basic interface for IR. This interface is structured as indicated in Figure 1, with a query window, a display of titles retrieved, and the full text of a document. This serves as the baseline interface interaction with which is compared to the visualization tool. A screen snapshot of the visualization tool is shown in Figure 2. The visualization corresponds to the query "how has affirmative-action affected the construction-industry construction projects and public works".

The visualization consists of a series of vertical columns of bars. There is one column of bars for each document. The leftmost vertical column of bars corresponds to the document ranked 1 and the rightmost vertical column corresponds to the

document ranked 150 with all the intermediate ranks lying in between. In each vertical column there are multiple bars – one each for each query word. The height of the bar at the intersection of query word row and a document column corresponds to the weight of that query word in that document. Thus if there are a handful of query words that convey the crux of the query and it is very important for a document to contain these query words, one can quickly see from the visualization which of the retrieved documents have those important words. One can also see how many of the retrieved documents have the multiple words in combination (in this example, the number of documents containing the term “affirmative action” in combination with “construction industry” or any of its related terms) to get a feel for the overall goodness of query results. In figure 2, we can see that “affirmative action” co-occurs with “construction industry” or “construction” or “public” or “project” only in about 20 documents. “construction industry” and its related terms appear in almost all the documents whereas “affirmative action” appears in only about 20 documents. The effects of modifying the query, such as adding a query word (for example, a synonym of “affirmative action”), are clearly shown in the visualization. One can quickly take stock of how useful the query modification turned out. Moving the mouse cursor over the vertical columns highlights the column directly beneath the mouse cursor and simultaneously highlights the title corresponding to that document in a title display window.

Apart from the query words typed in by the user, the visualization also shows the distribution information for words added by the system due to relevance feedback. Thus, all the words internally used by the system in computing the query results are shown in the visualization. The words in the visualization are also stopped and

stemmed. The basic interface, and the visualization tool, utilize the INQUERY retrieval engine, version 2.1p3 [CCH92]. We use all of the default features of that system, including their relevance feedback, stemming and stoplist algorithms, but do not use any of the structured query facilities.

2.2 Response to problems of IR interaction

In support of query reformulation, the visualization makes the connection between the query and retrieved documents explicit by graphically displaying the contribution of each of the query words to the retrieval of each document. The higher the contribution of a particular query word to the retrieval of a document, the taller the bar at the intersection of the corresponding query word and document. The absence of a bar at the intersection illustrates the absence of the term in the document. Absence of an important query concept in a number of retrieved documents points to a problem situation which the user needs to work on. The visualization also makes relations between the documents themselves explicit, since the characteristics which have led to their rank (the number and contribution of matching terms) are explicitly displayed.

In support of informative first-stage display, the visualization provides a great deal of information useful for deciding whether to view the full text of a document in a highly condensed way, and allows many document surrogates to be displayed at one time. The presence or absence of specific significant words in any document can be quickly seen, and it is possible to identify sequences of documents which do, or do not have important contributions from (implicitly discussions of) specific query words.

For the example search topic ("How has affirmative action affected the construc-

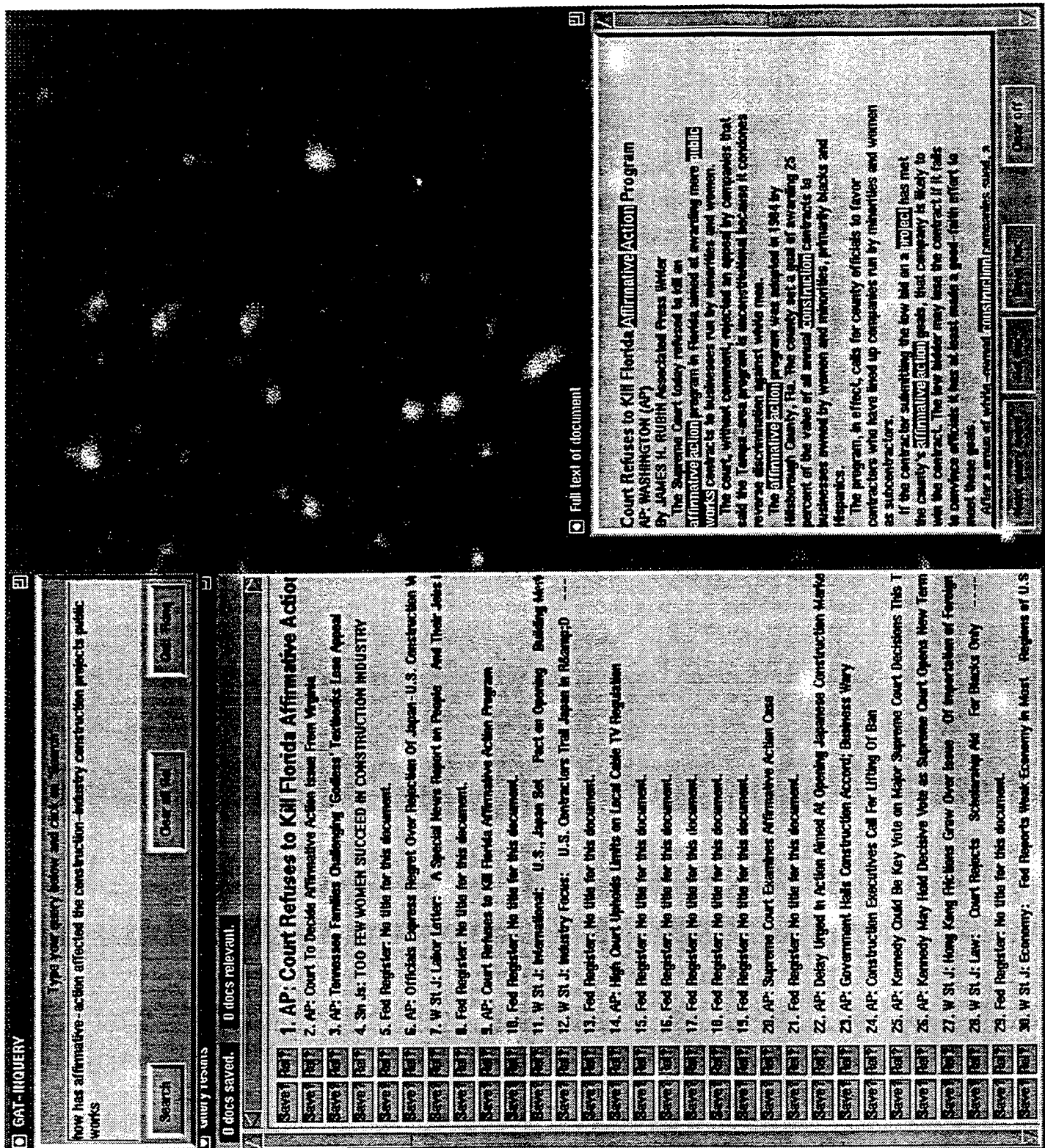


Figure 1: Sample querying session. The window in the top-left corner is the query entry window. Immediately below that is another window where the titles of retrieved documents are displayed. To the bottom right is another window where the full text of documents are displayed.

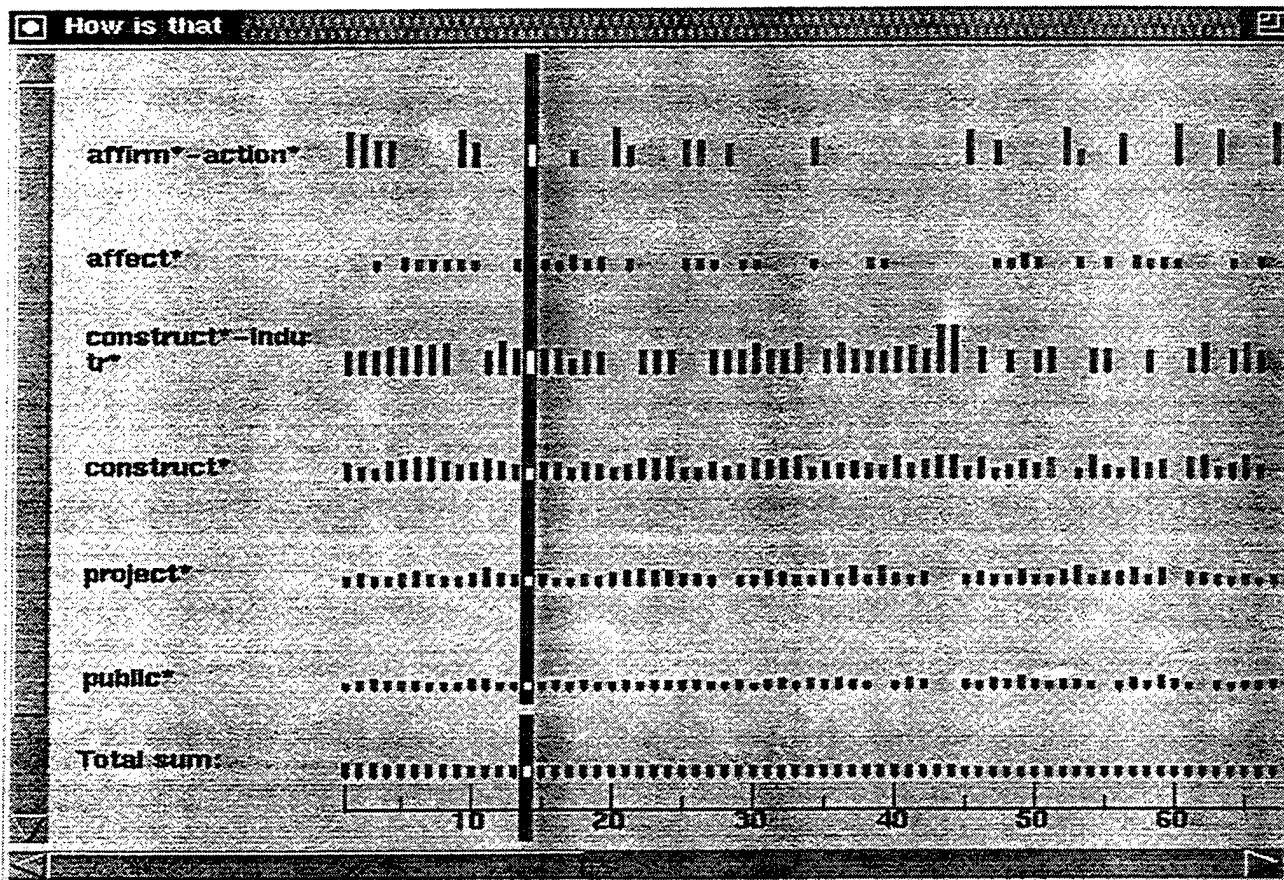


Figure 2: Visualization of results. The highlighted vertical column corresponds to document ranked 14. The title of document ranked 14 document will also be highlighted in the title display window. Clicking the highlighted vertical column brings up the full text of that document.

tion industry?"), there are two facets that are central: "affirmative action" and "construction industry". From the visualization tool, we can immediately see that most of the documents are concerned with the "construction industry" and only a portion of the documents have the term "affirmative action". We can also see that the "affirmative action" concept is spread sparsely throughout the top 70 documents. The visualization tool provides a *wider coverage* of documents because now the user may be willing to look at a document that is ranked at the bottom if it has both the central concepts of the query. By guiding the user to promising documents that contain all the important query concepts, the visualization tool acts as an efficient filter that indicates document relevance. Note that the visualization tool not only provides new information (about the presence of query concepts in documents), but also provides that information in a graphical format. The graphical format of presentation has some important advantages in that it does not take as much time for a user to identify and interpret information as it would for an equivalent text display. Thus the visualization tool not only guides the user to relevant information, it also indicates the non-relevant documents that the user can skip over.

From the visualization, one gets an immediate idea of how the different query words influence the document ranking (as given by the height of the bars). One can see that the concept "affirmative action" is not well represented in the retrieved documents. This suggests that synonyms and words related to that concept must be added to the query to reinforce that query concept in subsequent search iterations. From the visualization tool, one can infer that the system interprets "public" and "project" as two separate words and that the contribution of those two words to the retrieval of almost all documents is uniformly low (as given by the height of the

bars). One can probably improve the situation by making "public projects" a phrase, thereby retrieving documents that have these two words in close proximity. Gaining such overall information about the query results by reading the document text is at best cumbersome if at all possible.

The visualization scheme graphically displays the significance of the query words in each of the retrieved documents thereby providing document surrogate information that is directly related to the query. By displaying the importance of the different query words in the retrieved documents, the visualization provides surrogate information in addition to other surrogates in the database. Therefore, a user has more relevance indicators to judge which of the documents are relevant and which are not – thereby *increasing interactive precision*. This is not achieved at a cost of heavy perusal time since the time taken to interpret the visualization is only a fraction of the time it takes to peruse equivalent text information. The visualization may thus be used as a first filter to disregard those documents that do not have the important query concepts and hence clearly non-relevant. The user examines the titles and other text surrogates of the potentially relevant documents that have the important query concepts. Based on the text surrogates, the user may decide to request the second stage of display for promising documents. With the visualization tool, when the user is looking at the documents ranked in the 20's, he can also *simultaneously* identify that the documents ranked in the 70's (say) contain all the important query concepts. Simultaneously identifying a whole set of documents as potentially relevant (or non-relevant) is not possible with text displays. This simultaneous relevance judgement of sets of documents *increases document coverage greatly without demanding perusal time*.

3 Related visualization work

A number of visualization schemes for information retrieval systems have been proposed. The perspective wall [CRM91] describes a visualization scheme which supports browsing of documents. While such a system can not handle qualitative document classifications such as library subject catalogs, it is very useful for visualizing documents based on data which is linear in nature (like date of publication). A nice way of integrating different visualization schemes for efficient navigation through the hypermedia space has been proposed by Sougata [MFH95]. These schemes are primarily useful for navigational tasks. Other visualization schemes such as [Kor91, Spo94, HKW94] have facilities for viewing a large document space. But visualizing the document space along more than 3 - 4 dimensions simultaneously becomes very cumbersome using their systems. The visualization scheme in our tool can gracefully handle much higher number of query word dimensions. Also, most of them do not support querying with relevance feedback. Many systems are tailored towards easy construction of queries [Spo94, ACRS93, AB93] but do not pay much attention to the display of query results.

The TileBars work by Marti Heart [Hea95] visually shows the query term distribution and overlap in retrieved documents. The term distribution in retrieved documents is shown right besides the title of the document. In a number of respects, the reasons and motivations for her work are similar to those of our visualization work [VNH95, VN95, Vee95]. There are some important ways in which TileBars differs from the visualization that we propose.

- TileBars provides more document surrogate information than just the impor-

tance of query words in documents. Tilebars provide information on how the different query facets overlap in different sections of a long document. Our visualization scheme does not provide information at that fine levels of granularity.

- However such additional information comes at a cost to usability: To obtain maximum benefits from tilebars, the information need should be decomposed into more-or-less orthogonal facets. It remains to be seen how cumbersome it is for a naive end user to decompose her/his information need. There are advantages in letting the user specify the information need as free-form text.
- In TileBars, it is not as easy (as the visualization we propose) to gain an overall picture of the query word distribution for a whole set of documents in one glance. Being able to identify the general pattern and knowing which query concepts are not well represented in the retrieved documents as a whole is extremely beneficial in query reformulation. With TileBars, obtaining such aggregate information about a whole set of documents is tedious at best.
- While our visualization scheme is equally effective for both long and short documents, TileBars seems to be best suited for long documents.

4 Experiments to test the effectiveness of visualization

Below, we discuss two experiments to test the effectiveness of the visualization scheme. The basic scheme of both experiments is to test the effectiveness, usability and acceptability of the visualization tool by comparing searching with an interface using the visualization, versus searching with the same interface, but without the visualiza-

tion tool. The underlying retrieval engine used in these experiments was INQUERY version 2.1p3, from the University of Massachusetts, Amherst, generously made available to us by Prof. Bruce Croft [CCH92]. We developed the graphical user interface using Tcl/Tk on top of INQUERY.

The experiments were conducted as part of the TREC-4 interactive track [Har96]. Thus, the task for the searchers in the experiment was the TREC-4 interactive track task:

- Find as many documents as you can which address the given information problem, but without too much rubbish. You should complete the task in about 30 minutes or less.

The “information problems” were chosen from the 25 adhoc topics used for the TREC-4 interactive track, and the database was the TREC Disks 1 and 2 database of the full texts of about 550,000 documents.

Both experiments were designed to test the usefulness of the visualization tool for addressing the two problems that we have discussed and that motivated the design of the tool:

- efficiency and effectiveness in discovering relevant documents; and,
- effectiveness in supporting query reformulation.

In order to test the former, we predict that searchers using the visualization tool will make better decisions about which documents to look at (or not look at) than those without visualization. We operationalize this difference with the following dependent variables:

- the number of documents saved per search (*s-p-s*). Since search times are more-or-less constant (about 30 minutes) across searchers, this measure reflects efficiency in being able to see more documents.
- the proportion of documents whose full text was viewed that were judged relevant by TREC evaluators (interactive trec precision or *i-t-p*). This measure indicates the quality of the documents which were chosen for viewing.
- the proportion of documents whose full text was viewed that were saved by the searcher (interactive user precision or *i-u-p*). This measure also indicates quality of documents which were chosen for viewing, but is indicative of the relationship of the display to the searcher's own concept of relevance to the problem, rather than being dependent upon the external relevance judgments.
- precision of the search, measured in the required manner for the TREC-4 interactive track; that is, as the proportion of documents saved by the searcher that were judged relevant by the external judges. This measure is indicative of the effectiveness of retrieval performance, and is the only variable we used to gauge the effect of visualization on query reformulation.

For all of these measures, higher numbers mean better performance.

The subjects for both experiments were undergraduate student volunteers who were registered in a one-credit hour course on library searching in the College of Computer Science at Georgia Tech. All subjects had prior computer experience, the majority with more than four years. All subjects were majoring in an engineering discipline, and had varying levels of experience with the Georgia Tech Electronic

Library Catalog. They had no other IR experience than that offered by the class. Two different groups of subjects were used in the two different experiments.

All the subjects in both experiments followed the same general introductory and tutorial procedure. They were asked to fill out a background questionnaire about their computer and IR experience, major, and so on.(takes about 5 minutes). They then had a hands-on tutorial (about 1 hour) on how to use the version of the system which they would be using for the first experimental search. They were then asked to do a practice search on TREC topic 224 (“What can be done to lower blood pressure for people diagnosed with high blood pressure? Include benefits and side effects.”) for 15 minutes. They then did the assigned searching tasks (details differ between the two experiments), during which they were instructed to “think aloud”, which was recorded on audio tape. All the user interaction with the system was logged. After each search, they completed a search evaluation questionnaire. At the end of the session, a structured interview on their use of the system was administered. For the most part, there was an observer in the same room using a different computer and simultaneously observing the searcher.

4.1 Experiment 1

Thirty-six subjects were randomly divided into three groups of twelve each. Twenty-four of the 25 TREC-4 interactive track topics were randomly divided into twelve pairs. Each of the twelve pairs of search topics was randomly assigned to one of the searchers in each group, one to be searched in the “first” condition, the other to be searched in the “second” condition for the group of which the searcher was a member. The topic pairs were searched in the same order in all groups. Thus, the same twelve

of the 24 topics were searched in the first condition for all three groups, and the other twelve were searched in the second condition for all of them.

The three groups were defined according to the combination of conditions or treatments. Group wo:w (for WithOut:With) did their In the first group (hereafter named “w:w”, since both first and second search topics are searched With visualization), the initial tutorial, the practice search and the first search was done without the visualization tool. An intermediate tutorial introduced the visualization tool and the search for the second topic was done with the visualization tool. Group “w:w” (for With:With) used the visualization tool for all the searches and the introductory tutorial. Group “wo:wo” (for WithOut:Without) did all the searches and the introductory tutorial without the visualization tool. In both the w:w and wo:wo groups, an intermediate tutorial on the interface with which they were working was introduced between the two searches to compensate for the intermediate tutorial of the wo:w group.

This “within subjects” design was used in order to control for user differences, and to account for any possible learning effects from search 1 to search 2. It was predicted that performance on the various measures would improve in the wo:w group, more than in either the wo:wo or w:w groups.

4.2 Experiment 2

In this experiment, 36 subjects were randomly divided into two groups, one with the visualization tool (“viz”), the other without (“noviz”). Three search topics were chosen for searching by all eighteen searchers in each of the two groups, always in the same order. The searchers in the two different groups followed the same pattern of

participation as those in experiment 1, but without any intermediary tutorial, and with the practice search time extended to 30 minutes. We picked topic 242 (“How has affirmative action affected the construction industry?”) for the practice search. The first “experimental” search was on topic 236 (“Are current laws of the sea uniform? If not, what are some of the areas of disagreement?”), and the second was topic 203 (“What is the economic impact of recycling tires?”).

This “between-subjects” design was used to control for the effects of search topic difference, and to have larger numbers of subjects in the two conditions. It was predicted that performance in the viz group would be better than performance in the noviz group for each topic.

5 Results

In this paper, we report only on results with respect to the performance measures we have defined. Results from the questionnaires with respect to use and usability of the two systems, and with respect to interaction measures and “thinking aloud” will be reported in subsequent publications.

5.1 Experiment 1

The results of experiment 1, displayed in Table 1, are something of a disappointment. There are no significant differences (using the Wilcoxon Matched-Pairs Signed-Ranks Test, one-tailed at $p_i = .05$) between any of our four measures between the without- and with-visualization treatments in the wo:w group. Furthermore, there are no significant differences between any of the matched without-visualization/visualization

groups (i.e. between the second searches of wo:w and wo:wo groups, the first searches of wo:w and w:w groups, and both searches of the wo:wo and w:w groups). There is no consistent pattern on any of these measures from first to second search (i.e. there appears to be no learning effect, nor does it appear that one of the sets of twelve topics is in general more difficult than the other, nor is it the case that any of the groups does consistently better or worse for either search). These very mixed results lead us to think that our experimental design in this case suffers from two significant problems. The first is that the results suffer from great inter-subject and inter-topic variability; the second is that we have too few subjects for each condition to adequately test significance of any differences that may exist.

The results of experiment 1 led to the design of experiment 2, whose results are displayed in Table 2. The rows in Table 2 are in the order that the topics were searched. In order to test the significance of these results, it is necessary to compare them topic-by-topic, without cumulation, to maintain the assumption of independence, since each searcher did three searches (including the practice search, 242) in the same condition. To test for significance of results, we used the Mann-Whitney U test with $p = .05$, one-tailed. For precision, there is no significant difference between nonviz and viz for any of the three topics. For s-p-s, the trend is in favour of viz in all three cases, but significantly so at the chosen level only for topic 242 (although for topic 236 it only just misses). For i-t-p, again the trend is nominally in favor of viz, but is again significant only for topic 242. For i-u-p, the same trend holds, and again viz is significantly better than noviz only for topic 242.

For three of the four measures we can see that there are obvious topic differences which cannot be accounted for by a learning effect, since the direction is wrong. Two

points are important to note here. First, it appears that topic 242 was "easier" than the other two topics, and that topic 242 benefited most from visualization. Second, it is clear that differences in topics are likely to affect results averaged over topics significantly, unless there are also quite large numbers of searchers for each topic. Although for these three topics, three of the measures follow a consistent pattern between noviz and viz, the differences are really very small.

Interpreting these results is somewhat difficult, although they are a bit more promising than those of experiment 1, with respect to the potential of the visualization tool investigated here. It is of some interest that only topic 242 showed significant differences between the noviz and viz groups. This might be explained by that topic's being for some reason more suited to visualization than the other two. Although the numbers of relevant documents for the three queries are rather similar (242: 38, 236: 43; 203: 33), on the basis of median precision reported by all of the TREC-4 interactive track participants, topic 242 is "easier" than topics 203 and 236 (0.2368 vs 0.1515 vs 0.0465, respectively). This of course follows the pattern of precision results by the searchers in experiment 2, but it is not clear how this would explain the apparently beneficial effect of visualization for this topic. An alternative explanation might be that visualization of this sort is helpful for naive searchers, but loses its effect as they become more experienced with the IR system. On the basis of the data we have available, there is no way to decide between these alternatives.

In any event, it seems reasonable to accept, on the basis of the results of experiment 2, that there could indeed be some value to visualization of the sort we have tested here. However, this statement certainly must be very tentative, and subject to much more testing. The results of experiment 1 do not lead to any such conclusion. It must

be said, however, that the very mixed nature of these results may well be an effect of the experimental design, and in particular of the inability to take proper account of what may be very large topic differences and searcher differences. Of course, another possible reason for the seeming lack of effect of visualization is the implementation that we chose. This issue needs further investigation.

6 Conclusions

The study reported here intended to demonstrate the potential of visualization to support particular kinds of interactions in IR, and to test one implementation of such visualization. Although the results of our experiments are mixed, at best, it appears to us that some of them are positive enough to justify further such experiments. But there are some other serious implications of our results.

We are not aware of other work reporting comparisons of visualization tools for IR with equivalent non-visualization interfaces. Our experience suggests that it is important to conduct more such studies, in particular to move beyond assuming the efficacy of visualization to demonstrating it in experimental environments. But our study also demonstrates the severe problems that arise in conducting interactive IR experiments. These include the problems of finding enough subjects to account for inter-subject differences, and of being able to account for inter-topic differences. Balancing these two demands is an exceedingly difficult problem, which is currently severely exercising the TREC-5 interactive track participants. Another evaluation problem which raised by our study is how we are to measure the effectiveness of visualization tools. The problems with using precision as a measure for evaluating

interactive IR are now well-known, especially if precision is decided according to relevance judgments from experts, rather than the searchers. It is also the case that for certain functions of visualization, precision is an inappropriate measure. But we do not have available a suite of accepted alternative measures for evaluating the effectiveness of systems with respect to these functions. So, in our case, it was necessary for us to invent some new measures which appear appropriate to the IR tasks that we wished to support. Whether these were good choices also needs to be further investigated.

In conclusion, we find that this study has given some support for the general idea of visualization as a tool for enhancing user interaction with search results, and for the specific tool with which we implemented this idea. We also find that the level of support for these statements from this study is not high, and that it is necessary to conduct further studies, with better designs, before we can become confident in the value of visualization for these purposes, as opposed to other tools for interaction. Finally, we find that our study has shown, again, the necessity of developing better measures and methods for the evaluation of interactive IR systems, and the necessity of rigorous comparative evaluation of visualization in IR.

Acknowledgments

Support from the ARPA contract No. F33615-93-1-1338 to the first author is appreciated. The work of the second author was in part supported by NIST Cooperative Agreement No. 70NANB5H0050.

References

[AB93] H.C. Arents and W.F.L. Bogaerts. Concept-based retrieval of hypermedia

information - from term indexing to semantic hyperindexing. *Information Processing Management*, 29:387-396, 1993.

- [ACRS93] M. Aboud, C. Chrisment, R. Razouk, and F. Sedes. Querying a hypertext information retrieval system by the use of classification. *Information Processing Management*, 29:387-396, 1993.
- [CCH92] J.P. Callan, W.B. Croft, and S.M. Harding. The INQUERY retrieval system. In *Third International Conference on Database and Expert Systems Applications*, September 1992.
- [CRM91] S. Card, G. Robertson, and J. Mackinlay. The information visualizer, an information workspace. In *Proceedings of CHI 91 Human Factors in Computer Systems.*, 1991.
- [Har96] Donna Harman. *TREC-4, Proceedings of the fourth Text REtrieval Conference*. GPO, 1996.
- [Hea95] Marti A. Hearst. TileBars: Visualization of term distribution information in full text information access. In *Proceedings of CHI 95, Denver, Colorado.*, 1995.
- [HKW94] Matthias Hemmje, Clemens Kunkel, and Alexander Willet. LyberWorld - A visualization user interface supporting full text retrieval. In *Proceedings of the 17th Annual International Conference on Research and Development in Information Retrieval*, pages 249-259, 1994.
- [Kor91] Robert Korfhage. To see, or not to see - is that the query? In *Proceedings of the 14th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 134-141, 1991.
- [MFH95] Sougata Mukherjea, James. Foley, and Scott Hudson. Visualizing complex hypermedia networks through multiple hierarchical views. In *ACM SIGCHI*, 1995.
- [Spo94] Anslem Spoerri. InfoCrystal: A visual tool for information retrieval and management. In *Human Factors in Computing Systems CHI 94 Conference Companion*, pages 11-12, 1994.
- [Vee95] A. Veerasamy. Interactive TREC-4 at Georgia Tech. In *The Fourth Text REtrieval Conference (TREC-4)*, 1995.
- [VN95] A. Veerasamy and S. Navathe. Querying, navigating and visualizing a digital library catalog. In *Proceedings of the Second International Conference on the Theory and Practice of Digital Libraries*, 1995.
- [VNH95] A. Veerasamy, S. Navathe, and S. Hudson. Visual interface for textual information retrieval systems. In *Proceedings of the Third Conference on Visual Database Systems. IFIP 2.6*, pages 333-345, 1995.

Effectiveness of a graphical display of retrieval results

Aravindan Veerasamy

College of Computing, 801, Atlantic Drive

Georgia Institute of Technology

Atlanta, Georgia 30332-0280

Email: veerasam@cc.gatech.edu

Russell Heikes

Statistics Center

School of Industrial Systems and Engineering

Georgia Institute of Technology

Atlanta, Georgia 30332

Email: russell.heikes@isye.gatech.edu

Abstract

We present the design of a visualization tool that graphically displays the strength of query concepts in the retrieved documents. Graphically displaying document surrogate information enables set-at-a-time perusal of documents, rather than document-at-a-time perusal of textual displays. By providing additional relevance information about the retrieved documents, the tool aids the user in accurately identifying relevant documents. Results of an experiment evaluating the tool shows that when users have the tool they are able to identify relevant documents in a shorter period of time than without the tool, and with increased accuracy. We have evidence to believe that appropriately designed graphical displays can enable users to better interact with the system.

1 Introduction

The overall concern of all components of an IR system is to present the user as much relevant information as possible. While there has been a lot of work on effective algorithms for retrieving and ranking relevant documents, not much attention has been paid to study the effectiveness of user interface components of IR systems. Apart from retrieval mechanisms, interactive IR systems must also be concerned with the design of appropriate display mechanisms that present the retrieved information in the "best possible manner". We discuss what constitutes "best possible" display by examining a typical user interaction with an IR system. A typical interaction with current IR systems proceeds as follows:

- User in an Anomalous State of Knowledge [BOB82] expresses his information need as a query that is interpretable by the system.
- The system matches the query with the stored documents and retrieves a set of documents. In the case of ranked output systems, the result is ranked in the decreasing order of relevance. Boolean systems may rank the documents in a chronological order.
- At the *first stage* of display, a set of document surrogates for the retrieved documents are displayed to

the user. These surrogates typically consist of a combination of titles, author, source, date of publication, etc.

- The user inspects the document surrogates and requests more information (such as the full text if available) about those that look relevant. This leads to a *second stage* of display that provides as much information about the document (in many cases, the complete document itself) as is available in the system.
- After going through a sufficient number of documents, the user quits the session or reformulates the query to retrieve a better set of documents.

In this scheme, the first stage display of document surrogates is meant to provide a concise and accurate indication of document content. The second stage display of documents provides more information about the document. In cases where the document full text may not be available for the second stage (such as a typical online library catalog), users proceed to a third stage where they examine a paper-copy in library bookshelves where the complete document may be available.

Thus as the user progresses from the initial to the later stages of display, that which is displayed is more complete and informative, allowing increasingly accurate relevance judgments. However, since more information is displayed about a document in later stages of display, they are also more time-consuming to peruse. Furthermore, requesting second stage of display may be more costly since some systems charge a certain fee to deliver the full text of documents. Apart from the human frustration of waiting for the delivery of full text, one may have to pay for it monetarily since certain systems charge the user based on connect-time and the volume of downloaded data. Therefore, it is advantageous for the searcher to be reasonably certain about the relevance of a document before requesting a second stage of display.

For the user to make accurate relevance judgments based on the first stage display, the *form* and *content* of first stage of display should provide good indication of what document is about. The *form* of the first stage display should be such that it is quickly perusable – the purpose of the first stage display (of providing a quick and concise indication of document content) is lost otherwise. The *content* of the first stage display should be such that users can make accurate judgments about document relevance.

We can expect an improvement in the accuracy of relevance judgment if more content from the documents are dis-

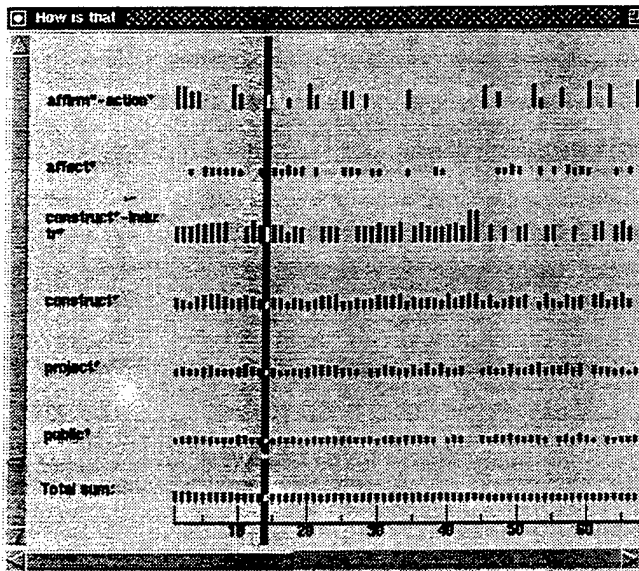


Figure 1: Visualization of results. The highlighted vertical column corresponds to document ranked 14. The title of document ranked 14 document will also be highlighted in the title display window. Clicking the highlighted vertical column brings up the full text of that document.

played. For example, we can expect greater accuracy with a first stage display that shows document titles, authors and subject keywords compared to one that shows just the document titles. When this additional document content is displayed in textual form, the increased accuracy may however bring along a negative effect on perusal time (increase in perusal time). This is because more time is consumed perusing the additional content.

A possible means to addressing this problem of displaying more information in the first stage without increasing perusal effort and perusal time is to display information in some form that does not require as much perusal time and screen space as text. Graphical displays (*visualizations*) of the characteristics of documents which are significant in supporting the decision to peruse or not, could enable set-at-a-time perusal of documents, rather than document-at-a-time perusal of text displays.

In the remainder of this paper, we describe a visualization tool meant to address this issue; describe and present the results of an experiment evaluating the tool; and draw some conclusions about its effectiveness as a first stage display.

2 Visualization tool

The visualization tool is an add-on to a basic interface for an IR system. There is a query window. The titles and ranks of retrieved documents (first stage of display) is shown below the query window. Figure 1 shows the visualization tool corresponding to the query "How has affirmative-action affected the construction-industry, construction projects and public works".

The visualization consists of a series of vertical columns

of bars. There is one column of bars for each document. The left-most vertical column corresponds to the document ranked 1 and the right-most vertical column corresponds to the document ranked 150. In each vertical column there are multiple bars – one each for each query word. The height of the bar at the intersection of a query-word-row and a document-column corresponds to the weight of that query word in that document. Moving the mouse cursor over the vertical columns highlights the column directly beneath the mouse cursor and simultaneously highlights the title corresponding to that document in the title-display window. The visualization window is scrollable, in case the number of query words exceeds the available vertical space. The words in the visualization are also stemmed and stemmed. Thus the combination of the visualization tool and the title display forms the first stage of display in our system. The basic interface, and the visualization tool utilize the INQUERY retrieval engine, version 2.1p3 [CCH92].

2.1 Response to the need for a concise display of document content

In the Introduction, we discussed the need for a concise first stage display which can also be perused quickly. We believe this visualization scheme to qualify for such a first stage display. It provides information valuable in deciding the relevance of document such as the weight of query concepts in the retrieved documents. The information is also displayed in a highly condensed way, and allows many document surrogates to be perused at one time. Textual display of document surrogates force the user to peruse them a document-at-a-time. However, with this visualization one can infer global patterns such as the following. Suppose we are faced with a search topic where a query term 'q' is so important that all relevant documents will have that query word. We would then ask the following questions: To identify relevant documents, we might ask "Which documents have the important query word 'q'?" To evaluate the goodness of the query, we might ask "Does the important query word 'q' appear in most of the retrieved documents?". When comparing the contribution of two query words, one might ask questions such as "What is the contribution of query word q2 compared to q5?". Answers for such questions seem to emerge from the visualization quickly. Such global perception of data is not possible with text displays that emphasize the *parts* rather than the *whole*. We refer to this kind of global perception as "set-at-a-time perusal", since the information gained is about a set of documents.

The presence or absence of specific significant words can be quickly seen, and it is possible, in one glance, to identify sequences of documents which do, or do not have important contributions from specific query words. For the example search topic ("How has affirmative action affected the construction industry?"), there are two facets that are central: "affirmative action" and "construction industry". From the visualization tool, we can immediately see that most of the documents are concerned with the "construction industry" and only a portion of them have the term "affirmative action". We can also see that the "affirmative action" concept is spread sparsely throughout the top 70 documents. The graphical format of presentation has some important advantages in that it is more condensed and can be more easily and quickly perused than an equivalent text display.

3 Related work

A number of visualization schemes for information retrieval have been proposed [CRM91, MFH95, Kor91, Spo94, HKW94, ACRS93, AB93] But most of these do not address either the display of query results or the problem of support of relevance assessment. An exception is TileBars [Hea95], but there are some important ways in which TileBars differs from the visualization proposed here.

- TileBars provide information on how the different query facets overlap in different sections of a long document. Our visualization scheme does not provide information at that fine levels of granularity.
- To make the best use of such additional information in TileBars, the user has to decompose the information need into more-or-less orthogonal facets of a query. However, in our visualization, the user can type in the information need as a free-form textual query.
- TileBars presents the document surrogates in a list, making it more difficult than in our tool to gain an overall picture of the query word distribution for a whole set of documents in one glance.
- TileBars seems best suited for long documents, while our visualization scheme seems to be equally effective for short and long documents.

There are a handful of studies that have investigated the effectiveness of document surrogates as content-indicators to enable human relevance judgments [Jan91, Sar69, RRS61, Tho73, MKB78]. None of them studied the effectiveness of graphical displays (visualizations) of document surrogates as content indicators. A result common to all of these studies is that "accuracy" in relevance judgments increases with increasing information (e.g. Title < Abstract < Full text). On the whole, we find that there has been a lack of studies to evaluate the effectiveness of graphical displays of document surrogates as indicators of relevance. This is mainly due to the fact that only recently has it been technologically and economically feasible to render such displays in real-time by the computer. Our study is an attempt to fill that gap.

4 Experimental Setup

In this section, we discuss an experiment to test the effectiveness of the visualization tool as a first stage display, and as a tool to aid effective query reformulation. The part on query reformulation will be discussed in a subsequent paper. We used a portion of the TREC [Har96] database consisting of all of disk1 and disk2 except the "Federal Register" documents. We did not use the Federal Register documents because a high proportion of them did not have a title. We used INQUERY 2.1p3 as the search engine [CCH92]. The retrieval mechanism of the search engine is based on bayesian inference networks using the word occurrence statistics in documents. All of the TREC information topics that we used were very detailed in their description of information need. We picked ten information topics for this study. The criterion used to pick the topics will be discussed below.

A slightly modified version of the *Description* field (mainly removing the introductory words such as "Document will report") was submitted to the retrieval system. 120 documents from the top 150 retrieved documents were obtained and split into two groups as follows: High precision group consisting of 60 documents ranked 1 through 60 and a low

precision group consisting of 60 documents ranked 91 through 150. We controlled for precision¹ as a factor in the experiment since we felt that precision might impact the perusal time: Users might more quickly identify non-relevant documents, than the relevant documents. Earlier studies [Sar69, RRS61, MKB78] indicate that precision also influences the ability to judge non-relevance.

Each of the two precision groups were further split into two groups: documents with odd ranks and the documents with even ranks. Thus, there were 4 groups of 30 documents for each information topic: *High_precision_even_ranks*, *High_precision_odd_ranks*, *Low_precision_even_ranks* and *Low_precision_odd_ranks*. The criterion used to pick the information topics for this study was that the "description" field when used as the query statement must retrieve a set of documents that had a distinct split in the precision values between the high precision group (ranks 1 through 60) and the low precision group (ranks 90 through 150). Since we did not want any overlap in precision values between the high precision group and the low precision group for all the ten chosen topics, we discarded the documents ranked 61 through 90. The precision values in the high precision group for all the chosen topics ranged from 0.43 to 0.6 while those of the low precision group ranged from 0.03 to 0.23.

The experiment we describe was aimed at investigating the effect of visualization on two problems for users:

- accurately identifying relevant documents
- effectively reformulating queries

In this paper, we report on results relevant to only the first of these, but because both problems were addressed in the same experimental design, we describe the entire experiment.

In the experiment, users were given two different types of tasks:

- Task of judging relevance: The users were given the information topic and the search statement used to retrieve documents. They were asked to judge the relevance of each of the 30 documents that were displayed to them as one of
 - relevant to the information topic.
 - non-relevant to the information topic.
 - Unsure.

For the purposes of the current experiment, clicking the left mouse-button over a document title in the title-display window or over a vertical column in the visualization window marks the document as relevant. Clicking the right mouse button over the title (or the column in the visualization window) marks the document as non-relevant. Middle-clicking it marks the document as "Unsure". Also, left-clicking a query word in the visualization window marks all documents containing that query word as relevant. Right-clicking a query word marks all documents that do not contain that word as non-relevant. Full text or any other information about the documents was not made available to users.

- Query reformulation task: Here the users were asked to "modify the preconstructed query into a form that will retrieve more relevant documents". For half of

¹ Precision is the density of relevant documents

the topics, users had the visualization tool and for the other half users did not have the visualization tool - making it a within-subjects, between-topics study.

For the "relevance judgment" task, precision (two levels: high and low) and visualization (two levels: with or without) were controlled in this within-subjects, within-topics study. The even ranked document group was shown with the visualization tool and the odd ranked document group was shown without the visualization tool. The users were not told that the 4 different document groups had two different precision levels. Instead, they were told that the query was issued against 4 different databases and the top 30 documents from each database was presented to them as 4 separate tasks - two with and the other two without visualization. For a given topic, the first task was always a "relevance judgment" task with a high-precision group. The next task was a query reformulation task. The third, fourth and fifth tasks were relevance judgment tasks for the other three groups of 30 documents. The first task was always a relevance judgment task because we wanted the users to have a good feel for the retrieved set of documents before they embarked on the query reformulation task. The first task of relevance judgment was always done with a high-precision document group because, in the real-world the users almost always inspect the top-ranked high-precision document range before they go down the ranks to inspect the low-precision range. Each user did the 5 tasks (4 relevance judgment tasks for the 4 document groups, and one query reformulation task) for 6 information topics, and finally did the search reformulation task for 4 more topics. The 6 topics for which the users did both the relevance judgment and query reformulation were:

- Topic 77: Document will report a poaching method used against a certain type of wildlife.
- Topic 115: Document will report specific consequence(s) of the U.S.'s Immigration Reform and Control Act of 1986.
- Topic 134: Document will report on the objectives, processes, and organization of the human genome project.
- Topic 136: Document will report on attempts by Pacific Telesis to diversify beyond its basic business of providing local telephone service.
- Topic 145: Document will describe how, and how effectively, the so-called "pro-Israel lobby" operates in the United States.
- Topic 197: Document will discuss legal tort reform (a civil wrong for which the injured party seeks a judgment) with regard to placing limitations on monetary compensation to plaintiffs.

The order in which the six topics were presented were balanced across the 37 subjects. The order in which the two visualization conditions appeared for a given topic were also balanced. The order in which the two precision groups appeared in a given topic was not balanced due to the constraint that a high precision group is always the first condition.

The human subjects in this experiment were Georgia Tech undergraduate students enrolled in a one-credit hour class on library searching. Students who participated in the study got full scores in two homework assignments. The complete experiment was split over two days. Subjects were asked to sign a consent form upon arrival. They were then

given a demo of the system by the experimenter. They then had a hands-on tutorial where they practiced both the "relevance judgment" task and the "query reformulation" task. Then, they did the 5 tasks for each of the three information topics marking the end of the experiment for the first day. On the second day, they did the 5 tasks for each of the other 3 topics, followed by the "query reformulation" task for 4 other topics.

The subjects were given monetary incentive to do well in the experiment. They were evaluated as follows: We knew a-priori, the relevance of all the documents as given by the TREC assessors. For the relevance judgment task, for each document the user obtained a +1 point if their relevance judgment matches the TREC assessor's judgment, a -1 point if their judgment does not match, and 0 points if they are "Unsure". The user has to judge all of the 30 displayed documents. Thus, for the 4 groups of 30 documents, for the 6 topics, each subject made a total of $4 \times 30 \times 6 = 720$ judgments.

		TREC judgment	
		Rel	Not_rel
User judgment	Rel	RuRt	RuNt
	Not_rel	NuRt	NuNt
	Unsure	UuRt	UuNt

The time taken by the subject to complete a task was also noted down. The top 10 quickest subjects with the most points were given monetary awards as follows: All participants were ranked on increasing order of time and decreasing order of points scored. Each participant's rank on both the categories (time and points) were added to get the sum-rank. The participant with the lowest sum rank was considered the best performer. Hence, to do well, one must be both accurate and quick. The top performer was given \$50, the second and third performers were given \$30 each, the fourth through sixth performers were given \$20 each and the seventh through the tenth performers were given \$10 each. The participants were told of the rating scheme, so we can assume that they optimized for time and accuracy equally.

Since we claim that graphical display of additional document surrogates does not increase perusal time significantly (due to the set-at-a-time perusal of documents), we predict that the time taken to complete the task for the visualization group will not be significantly higher than the non-visualization group. We also predict an increase in accuracy of relevance judgments for the visualization group, because we claim that very pertinent document surrogate information (i.e., the weight of query words in the retrieved documents) is being displayed in addition to the standard text surrogates such as title and source.

Effectiveness of the visualization tool was measured by what the subjects optimized upon: time, accuracy and the combined time-accuracy rank, where accuracy is the number of correct judgments minus the number of incorrect judgments after discarding the Unsure judgments, i.e., $Accuracy = RuRt + NuNt - RuNt - NuRt$. However, since the accuracy measure includes the correct judgments, Type I errors and Type II errors all in one score, we split the accuracy measure into distinct components. Here we borrow the analogs of two traditional IR measures "recall" and "precision" and extend them to the interactive situation. In the traditional recall and precision measures, the number of documents that the system judges to be relevant is artificially determined by a cut-off point of top 'X' documents. Let RsRt be the number of documents judged relevant by the system and relevant by the TREC assessor (the user with the original information

need). Let R_sN_t be the number of documents judged relevant by the system and non-relevant by the TREC assessor. Let N_sR_t be the number of documents judged non-relevant by the system and relevant by the TREC assessor and. Let N_sN_t be the number of documents judged non-relevant by the system and non-relevant by the TREC assessor.

While traditional "Recall" refers to the ratio of truly relevant documents that the system judged as relevant (i.e., $R_sR_t/(R_sR_t + N_sR_t)$), we define "Interactive Recall" as the ratio of the truly relevant documents that were judged as relevant by the user (i.e., $R_uR_t/(R_uR_t + N_uR_t + U_uR_t)$). While traditional "Precision" refers to the ratio of documents judged as relevant by the system that were truly relevant (i.e., $R_sR_t/(R_sR_t + R_sN_t)$), we define "Interactive Precision" as the ratio of the documents judged as relevant by the user that were truly relevant (Interactive precision = $R_uR_t/(R_uR_t + R_uN_t)$). Here, a "truly relevant" document is a document that was judged relevant by the TREC assessor. Thus, if we are trying to build an effective first stage display mechanism, we would strive for a display mechanism which would enable a user to pick (and read the full-text of) all of the relevant documents and only the relevant documents displayed. When a user picks a non-relevant document as relevant, it would be time and money wasted perusing a non-relevant document. As a corollary, not being able to pick a relevant document, would be a missing out on relevant information.

However, "Unsure" documents pose a problem. It can be handled in two ways: If we assume that a user always reads the full text of an Unsure document, we should treat the Unsure documents as being judged relevant by the user. Conversely, if a user always skips over an Unsure document, we should treat the Unsure document as being judged non-relevant by the user. Below, we present the analysis with both the interpretations. Thus, if we assume the user to inspect the Unsure documents, we treat the Unsure documents as relevant.

Interactive Recall = $(R_uR_t + U_uR_t) / (R_uR_t + N_uR_t + U_uR_t)$

Interactive Precision = $(R_uR_t + U_uR_t) / (R_uR_t + U_uR_t + R_uN_t + U_uN_t)$

If we assume the user to not inspect the Unsure documents, we treat the Unsure documents as not-relevant,

Interactive Recall = $R_uR_t / (R_uR_t + N_uR_t + U_uR_t)$

Interactive Precision = $R_uR_t / (R_uR_t + R_uN_t)$

In summary, our hypotheses are:

- Visualization will not increase the time taken to complete the relevance judgment task.
- Visualization will improve the Accuracy of relevance judgments.
- Visualization will improve Interactive Recall.
- Visualization will improve Interactive Precision.

5 Results

Statistical analysis of the experimental data empirically shows that our hypotheses about the relevance judgment task are valid. Since there were 37 subjects, and all subjects did 6 topics with 4 tasks (for each of the 4 groups within the topic) per topic, there were a total of $37 \times 6 \times 4 = 888$ observations. The approach used in all analyses was to construct a least squares, linear additive model of each performance

measure as a function of the main effects and interactions of the manipulated experimental variables.

The need for consideration of possible learning/ordering effects, due to the same subjects providing multiple responses at various experimental conditions, is minimized by the balancing of the order in which different experimental conditions are presented to the subjects. However, due to the requirement that within a topic, the high precision condition always be presented first, this balance could not be achieved for this factor. To account for this, the model included a term representing the observation order within subject/topic combination. The design thus allows for independent estimation of all effects except precision and observation order. The analysis presented will focus on the statistical significance of each term assuming the presence of the the other term in the model (i.e on the adjusted sums of squares in the Analysis of Variance (ANOVA) tables), as this provides evaluation of the marginal effect.

The residuals of the models constructed were analyzed to assure reasonable compliance with the normality, independence and constant variance assumptions required for validity of ANOVA,

For the dependent variable "time", the residuals indicated a higher variance for conditions resulting in larger values of time, and hence we transformed time values into $\log_{10}(\text{time in seconds})$ to check for statistical significance. The ANOVA tables for $\log_{10}(\text{time})$, accuracy and final score are shown in Tables 1, 2 and 3 respectively. The means and standard errors are shown in table 4. As can be seen from the tables, viz is significantly better than noviz for logtime, accuracy and final score. It is also clear that low precision condition does significantly better than high precision for logtime, accuracy and final score. The interaction effects of precision and visualization are shown in figures 2, 3 and 4 with a 95% confidence interval around the means. When precision is high, visualization does not significantly affect logtime, but when precision is low, there is a decrease in logtime of 0.08. This corresponds to a reduction of 17.2 seconds, nearly a 20% decrease in average time required. Thus we can conclude that the visualization tool helps users in identifying document relevance *more quickly*. It is also interesting to note (from Table 1) that the interaction effect of topic with visualization was not statistically significant, although the main effect of topic was significant. Thus, visualization helps improve speed of judgment irrespective of topic.

For the accuracy measure, there is no significant interaction between precision and visualization as shown by the almost-parallel lines in figure 3. Precision has a huge impact on accuracy, again consistent with previous studies [Sar69, MKB78]. While the effect of visualization on accuracy is significant, it is not as huge as the effect of precision. Users can identify document relevance *more accurately* with the visualization tool than without. The ability of users to identify non-relevant documents as non-relevant is much higher than their ability to identify relevant documents as relevant. This is reflected in the significantly very high accuracy value for low precision than for high precision. It is also interesting to note that (from Table 2) the interaction between topic and visualization was statistically significant. However, the main effect of visualization was much greater than the topic*viz interaction effect.

Final score is a rank measure, which reflects the users ability to *accurately and quickly* identify document relevance. It is plotted in figure 4. Lower values are better for final score. As with accuracy, precision has a much higher impact

than visualization, but both variables have a significant effect. Visualization tool improves Final Score and so does low precision. There is a higher proportion of non-relevant documents in the low precision condition. This implies that users can more quickly and accurately judge a non-relevant document as non-relevant compared to judging a relevant document. It is also interesting to note (from Table 3) that the interaction between topic and visualization was statistically significant. However, the main effect of visualization was much greater than the topic*viz interaction effect.

Table 1: ANOVA for log10(time in seconds).

Source	DF	Adj SS	Adj MS	F	P
topic	5	1.59993	0.31999	21.28	0.000
precis	1	0.45954	0.45954	30.56	0.000
viz	1	0.36761	0.36761	24.44	0.000
precis*viz	1	0.29215	0.29215	19.43	0.000
topic*viz	5	0.15612	0.03122	2.08	0.067

Table 2: ANOVA for Accuracy.

Source	DF	Adj SS	Adj MS	F	P
topic	5	10566.13	2113.23	95.04	0.000
precis	1	11842.00	11842.00	532.55	0.000
viz	1	490.54	490.54	22.06	0.000
precis*viz	1	24.67	24.67	1.11	0.293
topic*viz	5	1248.65	249.73	11.23	0.000

Table 3: ANOVA for Final Score.

Source	DF	Adj SS	Adj MS	F	P
topic	5	7187688	1437538	90.98	0.000
precis	1	6789177	6789177	429.68	0.000
viz	1	362841	362841	22.96	0.000
precis*viz	1	133133	133133	8.43	0.004
topic*viz	5	352669	70534	4.46	0.001

As discussed before, accuracy combines the following four items into one: ability to judge relevant and non-relevant documents ($RuRt + NuNt$), type I error, i.e., wrongly rejecting relevant documents, and type II error, i.e., wrongly accepting non-relevant documents. We feel that identifying non-relevant documents ($NuNt$) in and of itself is not as important as the other 3 items. For, it is important

- to minimize Type I errors, or else one runs the risk of missing out too many relevant documents.
- to minimize type II errors, or else one runs the risk of wasting too much money and effort in examining non-relevant documents.

We can capture all the interesting data with interactive recall and interactive precision as described in the previous section. In our tables, when users are assumed to treat unsure documents as relevant, the interactive precision and interactive recall are denoted by "iprecwu" and "irecwu" respectively. Correspondingly, when unsure documents are assumed to be treated as non-relevant, interactive precision

Table 4: Least Square Means and Standard errors for Logtime, Accuracy and Final score

Precis	Viz	Logtime	Accur	FinScore
Low	Without	2.01	15.72	353.2
Low	With	1.93	17.54	288.4
High	Without	2.04	5.72	576.1
High	With	2.04	6.87	560.2
STD ERR OF EST		0.009	0.35	9.4

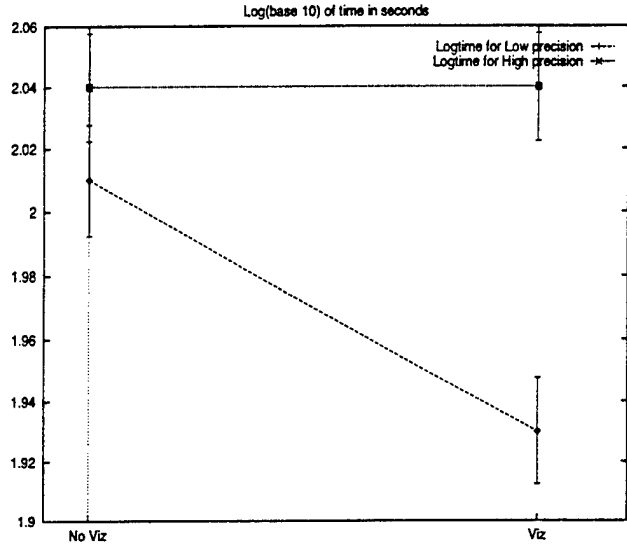


Figure 2: Interaction effects of precision and visualization on logtime.

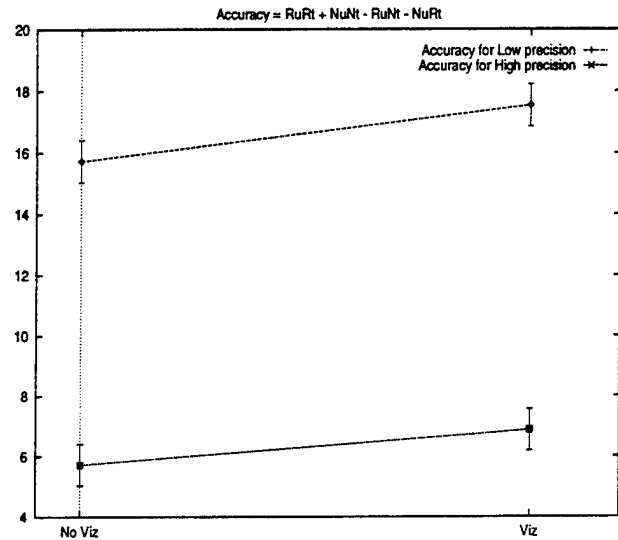


Figure 3: Interaction effects of precision and visualization on accuracy.

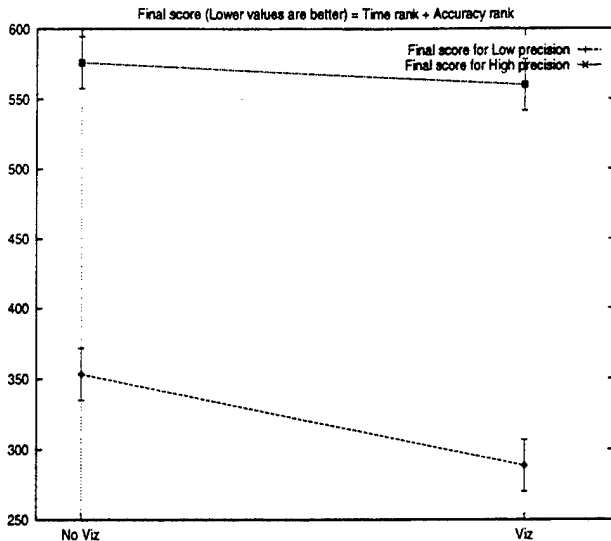


Figure 4: Interaction effects of precision and visualization on Final Score.

and interactive recall are denoted by the mnemonics “iprecwu” and “irecwou” respectively.

In considering the interactive precision measure there are a large number of cases where the values result in responses of zero divided by zero when users did not pick any of the displayed documents as relevant. Rather than eliminate these cases, the raw data (i.e., RuRt, RuNt, NuRt, NuNt, UuRt, UuNt) was aggregated over high and low precision levels for the same viz condition and the interactive precision and interactive recall measures then computed. Thus, for example, for topic 77, the RuRt values for the high_precision_viz case for subject 1 was added to the RuRt value of the low_precision_viz case of the same subject 1 and same topic 77. Now we end up with 444 observations instead of the original 888 observations. This eliminated the need for the “precision” term in the model, although the variability due to this factor is included in the error term. One of the terms is labeled “topic+ord” because the “topic” term also includes some “condition order” effects since for different topics, the four conditions appeared in different orders. The design is now orthogonal to the remaining factors. However for interactive precision when unsure documents are considered non-relevant (iprecwu), there remain 2 cases where the response variable is still zero divided by zero. The result is a design where estimated effects are minimally dependent. Also, there are some quantization errors introduced in the interactive precision measure due to the denominator value being too close to zero². The statistical significance of visualization for Interactive precision and interactive recall (with unsure documents treated as relevant and non-relevant) are shown in tables 5, 6, 7 and 8, and table 9 shows the estimated means.

Visualization had no significant effect on interactive pre-

²For interactive precision when unsure documents are considered non-relevant (iprecwu), there were 2 cases where the denominator had a value of 1, 5 cases of value 2, 6 cases of value 3. For interactive precision when unsure documents are considered relevant (iprecwu), there were 0 cases of denominator values 0 and 3, 1 case of values 1 and 2. Given that there were 444 observation points, these quantization errors are not expected to distort the results much.

cision when Unsure documents were treated as non-relevant (iprecwu) at the 0.05 level, however, it was significant when Unsure documents were treated as relevant (iprecwu) (See figure 5). Although statistically significant, the absolute increase in interactive precision is very minimal (about 0.015). However, visualization had a significant effect on interactive recall (both when unsure documents were treated as non-relevant (irecwou) and when unsure documents were treated as relevant (irecwu)). Also, in the absolute sense, the improvement in interactive recall due to visualization is approximately 0.07 +/- 0.02 (about a 15% increase). Clearly this is of sufficient magnitude to be of practical importance.

Table 5: ANOVA for Interactive Precision “iprecwu” (Unsure documents treated as non-relevant)

Source	DF	Adj SS	Adj MS	F	P
topic+ord	5	8.15469	1.63094	163.81	0.000
viz	1	0.03065	0.03065	3.08	0.081
topic+ord*viz	5	1.70775	0.34155	34.31	0.000

Table 6: ANOVA for Interactive Precision “iprecwu” (Unsure documents treated as relevant)

Source	DF	Adj SS	Adj MS	F	P
topic+ord	5	6.90892	1.38178	180.62	0.000
viz	1	0.04166	0.04166	5.45	0.021
topic+ord*viz	5	1.02194	0.20439	26.72	0.000

Table 7: ANOVA for Interactive Recall “irecwou” (Unsure documents treated as non-relevant)

Source	DF	Adj SS	Adj MS	F	P
topic+ord	5	3.04486	0.60897	34.92	0.000
viz	1	0.62601	0.62601	35.89	0.000
topic+ord*viz	5	0.72200	0.14440	8.28	0.000

6 Conclusions

We have presented a visualization tool designed to be an effective first stage display of retrieved documents. The results about the query reformulation task and a detailed analysis of all the experimental factors can be found in the thesis by Veerasamy [Vee97]. User experiments empirically show that when precision is low, the visualization tool helps users in identifying document relevance quicker by about 20%. Our hypothesis was that the time taken to judge relevance would not be higher for visualization because we claimed that graphically displaying additional information would not take additional time to peruse by enabling set-at-a-time perusal. While this argument is certainly validated by the experimental results, we however see that visualization seems to decrease the time taken. We see only one explanation to this: Users consult visualization before they consult the titles, thereby not looking at the titles of those documents which are clearly non-relevant. Thus they save

Table 8: ANOVA for Interactive Recall "irecwu" (Unsure documents treated as relevant)

Source	DF	Adj SS	Adj MS	F	P
topic+ord	5	2.35410	0.47082	30.21	0.000
viz	1	0.42787	0.42787	27.46	0.000
topic+ord*viz	5	0.41879	0.08376	5.37	0.000

Table 9: Least squares means of iprecwu, iprecwu, irecwu, irecwu

viz	iprecwu	iprecwu	irecwu	irecwu
Without	0.6117	0.5753	0.4454	0.5484
With	0.6284	0.5947	0.5209	0.6108
Std error	0.007	0.006	0.009	0.008

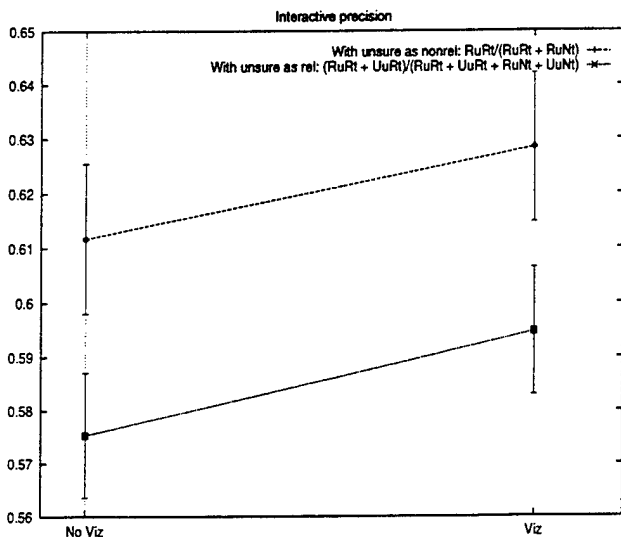


Figure 5: Effect of visualization on interactive precision (when Unsure documents are treated as relevant and non-relevant documents).

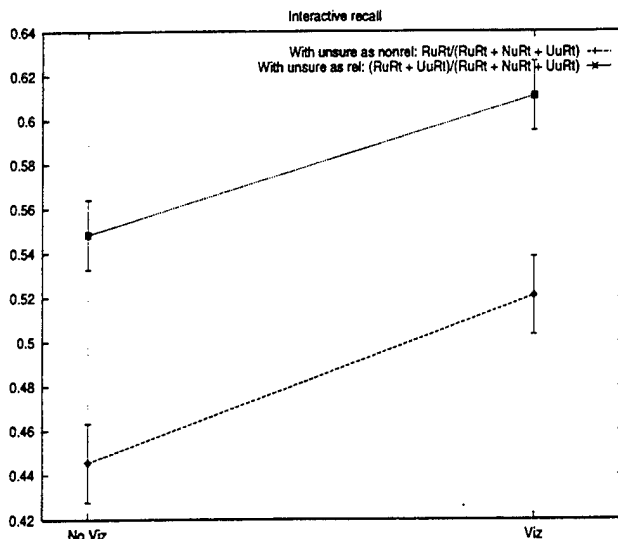


Figure 6: Effect of visualization on interactive recall (when Unsure documents are treated as relevant and non-relevant documents).

the time needed to read titles for those non-relevant documents. This is in agreement with the study by Saracevic [Sar69] which shows that minimal information is needed to say that a document is non-relevant. However, to say that a document is relevant, much more information is needed. This is also confirmed by the fact that the magnitude of time-decrease due to visualization is much higher in the low precision condition than in the high precision condition. On the whole we see confirmation of our argument about set-at-a-time perusal of documents in graphical displays.

The experiment also shows that users with the visualization tool did significantly better in accurate (both in terms of the aggregate "Accuracy" measure and in terms of the broken down measure of "Interactive Recall") identification of document relevance. The result about the influence of precision over relevance judgment Accuracy is in agreement with previous studies by Saracevic [Sar69], and Marcus et.al. [MKB78]. Their studies, like ours, also show that users are better able to judge non-relevance than relevance. However we do not see an interaction between precision and visualization on Accuracy. Thus visualization seems to help increase Accuracy to the same extent irrespective of the density of relevant documents. There is a marked difference in a user's ability to judge the relevance of relevant documents and non-relevant documents. Given this difference, we feel that precision (i.e., the density of relevant documents among the displayed documents) should be a variable that must be controlled in experiments that measure a user's ability to judge relevance. Further, care should be taken in making claims purely based on a compound measure such as "Accuracy" that combines both the ability to correctly identify relevant documents and the ability to correctly identify non-relevant documents.

We broke down the accuracy measure into two components: interactive precision and interactive recall to gain a better understanding of the relevance judgment process. While the effect of visualization tool was marginally significant for interactive precision, it was highly significant for interactive recall. Thus, we can safely say that the visualiza-

tion tool helps users in identifying more relevant documents out of the displayed documents. It also helps users in identifying them more quickly.

In an earlier paper [VB96], we discussed the difficulty of conducting interactive user experiments in IR. We mentioned the difficulty of huge inter-topic differences, inter-subject differences, the large number of subjects needed to account for these differences and how these factors severely affect the interactive track of TREC participants. There was also the problem of using appropriate measures to evaluate different user interface components and the lack of established metrics for these purposes. It is worthwhile noting how we approached these problems in the experiment described in this paper. At the outset, we had to be extremely specific in our claims about where the visualization tool would be of help. Having narrowed the scope of the experiment to these claims, we had to devise a scheme where inter-subject and inter-topic variability could be kept to a minimum. By restricting the task to relevance judgment of documents, we could safely construct a within-topic, within-subject experiment that would not threaten the extensibility of our inferences to the real world.

In addition, we do not know of any established performance metrics that measure the effectiveness of interactive display mechanisms in helping users identify all and only the relevant documents among the displayed documents. In the absence of established interactive metrics, we had to come up with our own measures of effectiveness of graphical displays (such as interactive precision and interactive recall). It remains to be seen if such choice of metrics are appropriate and if they are of real impact in terms of the quality of interaction of end-users. The lack of convincing answers to the above questions points to the acute need for more interactive experiments to study human interaction with ranked output IR systems and to study the effectiveness of emerging display mechanisms such as visualizations.

Acknowledgments

We deeply appreciate the help of Neff Walker who helped us in the design of the experiment. Many thanks to Nick Belkin who has been constantly supportive of the work and providing valuable feedback to revisions of this paper. Support from ARPA contract No. F33615-93-1-1338 to the first author is appreciated.

References

- [AB93] H.C. Arents and W.F.L. Bogaerts. Concept-based retrieval of hypermedia information - from term indexing to semantic hyperindexing. *Information Processing Management*, 29:387-396, 1993.
- [ACRS93] M. Aboud, C. Chrisment, R. Razouk, and F. Sedes. Querying a hypertext information retrieval system by the use of classification. *Information Processing Management*, 29:387-396, 1993.
- [BOB82] N.J. Belkin, R.N. Oddy, and H.M. Brooks. ASK for information retrieval: Parts I and II. *Journal of Documentation*, 38(2,3), 1982.
- [CCH92] J.P. Callan, W.B. Croft, and S.M. Harding. The INQUERY retrieval system. In *Third International Conference on Database and Expert Systems Applications*, September 1992.
- [CRM91] S. Card, G. Robertson, and J. Mackinlay. The information visualizer, an information workspace. In *Proceedings of CHI 91 Human Factors in Computer Systems.*, 1991.
- [Har96] D.K. Harman, editor. *The Fourth Text REtrieval Conference (TREC-4)*. NIST Special Publication, 1996.
- [Hea95] Marti A. Hearst. TileBars: Visualization of term distribution information in full text information access. In *Proceedings of CHI 95, Denver, Colorado.*, 1995.
- [HKW94] Matthias Hemmje, Clemens Kunkel, and Alexander Willet. LyberWorld - A visualization user interface supporting full text retrieval. In *Proceedings of the 17th Annual International Conference on Research and Development in Information Retrieval*, pages 249-259, 1994.
- [Jan91] Joseph W. Janes. Relevance judgements and the incremental presentation of document representations. *IPM*, 27(6):629-646, 1991.
- [Kor91] Robert Korfhage. To see, or not to see - is that the query? In *Proceedings of the 14th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 134-141, 1991.
- [MFH95] Sougata Mukherjea, James. Foley, and Scott Hudson. Visualizing complex hypermedia networks through multiple hierarchical views. In *ACM SIGCHI*, 1995.
- [MKB78] Richard S. Marcus, Peter Kugel, and Alan R. Benenfeld. Catalog information and text as indicators of relevance. *JASIS*, pages 15-30, Jan 1978.
- [RRS61] G.J. Rath, A. Resnick, and T.R. Savage. Comparisons of four types of lexical indicators of content. *American Documentation*, pages 126-130, April 1961.
- [Sar69] Tefko Saracevic. Comparative effects of titles, abstracts and full texts on relevance judgements. In *Proceedings of the ASIS*, pages 293-299, 1969.
- [Spo94] Anslem Spoerri. InfoCrystal: A visual tool for information retrieval and management. In *Human Factors in Computing Systems CHI 94 Conference Companion*, pages 11-12, 1994.
- [Tho73] C.W.N. Thompson. The functions of abstracts in the initial screening of technical documents by the user. *JASIS*, 24:270-276, 1973.
- [VB96] Aravindan Veerasamy and Nick Belkin. Evaluation of a tool for visualization of information retrieval results. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1996.

[Vee97] A. Veerasamy. *Visualization and User Interface Techniques for Interactive Information Retrieval Systems*. PhD thesis, Georgia Institute of Technology, Available at [ftp.cc.gatech.edu](ftp://ftp.cc.gatech.edu/pub/people/veerasam/thesis.ps.gz) in `/pub/people/veerasam/thesis.ps.gz`, March 1997.

PART III

**METADATA MANAGEMENT FOR
INTELLIGENT QUERY
PROCESSING**

PART III: METADATA MANAGEMENT FOR INTELLIGENT QUERY PROCESSING

In this part of the project we focused our attention on two problems -

- a. Improving the efficiency of query processing by avoiding unnecessary computation and using the semantics of data.
- b. Capturing semantic constraints at the instance level and maintaining them during the evolution of the database. These constraints constitute semantics that can be related to the various views of the database.

Whereas in parts 1 and 2 of the project we dwelled on the querying and integration of databases and the formulation of queries against text databases, in this part of the project we addressed an area known as **semantic query optimization**.

Most of the work to date on semantic query optimization has relied upon schema level semantic constraints. **Instance-based constraints** reflect a finer granularity of constraints that can be effectively utilized to provide additional information during query processing. Using instance based constraints, one may be able to avoid certain scans and searches which are known not to produce a meaningful answer to a given query. In [3.1], Pittges proposes *Metadata View Graph (MVG)* as a data structure to store these constraints and relate them to various query execution plans so that a constraint may be evaluated while computing certain intermediate results of a query. The paper also describes how these constraints are stored at compile time, maintained during run-time in response to updates to the databases, and used for query optimization. It thus makes a contribution at three levels: (i) creates a framework to define instance based constraints, (ii) provides a foundation that directs and integrates existing methods of constraint discovery, and (iii) proposes efficient techniques for a run-time retrieval of these constraints.

Metadata may be classified into semantic metadata which is in the form of rules (e.g. if `employee_type = manager` and `dept_number > 10` then `75000 < salary < 100000`) and structural metadata, which is in the form of indexes which relate query execution plans with view nodes and view caches. In [3.2], the inherent conflict of maintaining the semantic metadata before query execution and structural metadata during query execution, when the two overlap, has been tackled. The above conflict introduces inefficiencies in the processing of the update logs. [3.2] proposes strategies of overlapping update paths in the metadata view graphs.

This part of the research project resulted in the Ph.D. dissertation of Jeff Pittges [Pittges 1995] where algorithms have been presented for metadata view

graph construction, maintenance of this metadata and an efficient processing of updates. This work is unique in terms of its incorporation of instance level semantics of databases.

Additional References:

J. Pittges, " Metadata View Graphs: A Framework for Query Optimization and Metadata Management," Ph.D. Dissertation, Georgia Institute of Technology, November 1995.

PUBLICATIONS (PART3):

[3.1]. Jeff Pittges. " Maintaining Instance-Based Constraints for Semantic Query Optimization," In *Proceedings of the Sixth IFIP TC-2 Working Conference on Data Semantics (DS-6)* , Stone Mountain, Georgia, May 1995

[3.2]. " Maintaining Semantic and Structural Metadata in the Metadata View Graph," J. Pittges, L. Mark, and S. Navathe. In *Proceedings of the Seventh International Conference On Management of Data*, Pune, India, December 1995.

Maintaining Instance-Based Constraints for Semantic Query Optimization

Jeff Pittges

College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280
(404) 853-9381

`pittges@cc.gatech.edu`

<http://www.cc.gatech.edu/grads/p/Jeff.Pittges/pittges.html>

May 3, 1995

Abstract

Semantic Query Optimization has traditionally relied upon scheme-based integrity constraints that are valid for all instances of a database. *Instance-based constraints*, which are only valid for certain states of a database, contain more information than scheme-based constraints because they are specific to the current contents of the database. This makes instance-based constraints more useful to semantic query optimization. However, instance-based constraints are highly sensitive to any changes made to the database and must therefore be updated and validated before they can be applied.

A Metadata View Graph (MVG) is a metadatabase that stores instance-based constraints, along with statistical and structural metadata, for logical views of the database. Constraints at this level are even more useful to semantic query optimization because they are specifically tailored to the intermediate results of a query. This paper reviews existing methods for constraint discovery, describes how constraints are stored in the Metadata View Graph at compile-time, and describes how the MVG Framework retrieves and maintains instance-based constraints at run-time. The paper then analyzes how to apply updates to instance-based constraints in order to refresh them.

Keywords: Metadata View Graph, Instance-Based Constraints, Semantic Query Optimization, Constraint Discovery, Metadata Maintenance.

1 Introduction

Semantic query optimization [HZ80, Kin81, SO89, CGM90, SSD92, HK93] uses transformation rules to reformulate a query into a semantically equivalent query that is more efficient to execute. Traditionally, transformation rules have been derived exclusively from scheme-based integrity constraints that are valid for all instances of a database. Transformation rules based on this type of constraint are desirable because the rules remain valid when changes are made to the database since changes cannot violate the integrity constraints. Unfortunately, scheme-based constraints are typically so general that they are of little use. For example, an integrity constraint may require an employee's salary to be greater than zero.

Recently, a number of researchers [YS89, SSS92, SHKC93, HK94] have proposed methods for discovering *instance-based constraints* (also referred to as *dynamic constraints* in [YS89] and *derived constraints* in [SSS92]) which are only valid for particular instances of the database. Instance-based constraints contain more information than scheme-based constraints because they represent the actual contents of the database. For example, an instance-based constraint may assert that the salary of all employees is greater than or equal to \$22,000 and less than or equal to \$85,000. Although this constraint is more useful than the integrity constraint given above, instance-based constraints are sensitive to changes made to the database, so they must be maintained whenever the database is updated.

A Metadata View Graph is a metadatabase that maintains instance-based constraints for logical views of the database. Constraints at this level are even more useful to semantic query optimization because they are specifically tailored to the intermediate results of a query. For example, one view might represent graduate students and another view might represent faculty. Both views would maintain an instance-based constraint on salary, but the salary range for graduate students would be much less than the salary range for faculty members. Therefore, when given a query involving graduate students, the semantic query optimizer could use the salary constraint for graduate students to reformulate the query.

In addition, capturing instance-based constraints for views of the database allows the query context to influence the optimization process during semantic query optimization. The *profitability* [SO87] of a rule can be adjusted for each view. Therefore, although the same rule may appear in several views, the rule can be applied differently based on the query context represented by each view. The rules for a particular view can be ordered by associating a context-sensitive salience (priority) with each rule. In this way, the views partition the semantic constraints which allows the constraints to be precisely tailored to particular data sets. Partitioning reduces the number of rules which must be searched during semantic query optimization and improves rule selection by prioritizing rules according to the query context.

The rest of this paper is organized as follows. Section 2 describes the Metadata View Graph Framework. This section provides a structural description of the Metadata View Graph, reviews existing methods for discovering instance-based constraints, and describes how instance-based constraints are stored in the Metadata View Graph. The section also

describes how instance-based constraints are retrieved at run-time and used to select the best query execution plan. Section 3 presents the general problem of maintaining instance-based constraints, presents various representations that allow instance-based constraints to be maintained, and analyzes the problem of applying updates to instance-based constraints in order to refresh them. The last section summarizes the contributions of this paper and describes future tasks for this research.

2 The MVG Framework

The Metadata View Graph Framework [PMN95] supports the integration of various approaches to query optimization. As shown in Figure 1, the framework consists of an optimizer and a metadatabase. The MVG Framework has been developed with two objectives in mind: (1) improve query optimization, and (2) provide for a highly extensible query optimizer. Query optimization is improved by maintaining metadata, especially instance-based constraints which improve semantic query optimization, multiple query optimization, incremental query computation, and dynamic plans. Three types of knowledge are required by the query optimizer: (1) procedural knowledge specific to each type of query optimization, (2) control knowledge which integrates the various types of query optimization together, and (3) domain knowledge (metadata about the database). The MVG Framework maintains these three types of knowledge separately in order to facilitate a highly extensible architecture. The optimizer and metadatabase can be extended incrementally as new approaches to query optimization are developed.

This research focuses primarily on the metadatabase (the Metadata View Graph) which was inspired by considering six types of query optimization: syntactic, physical, semantic, dynamic, multiple, and caching and incremental query computation. Therefore, we treat the query optimizer as a black box that is capable of performing these six types of query optimization. Within that black box, the optimizers can be developed independently and loosely coupled, which requires less integration effort but reduces run-time efficiency, or the optimizers can be tightly coupled, which requires greater integration effort (each optimizer may have to be rewritten) but improves run-time efficiency. Although we envision a set of rule-based optimizers [Fre87, GD87, GM93], the actual implementation is irrelevant to our work on Metadata View Graphs.

2.1 Metadata View Graphs

A Metadata View Graph (MVG) is a collection of networks, as shown in Figure 2, for organizing and storing metadata (i.e., a metadatabase). The Metadata View Graph consists of four components: (1) a lexicon, (2) a semantic network, (3) a view network, and (4) a QEP Network of query execution plans.

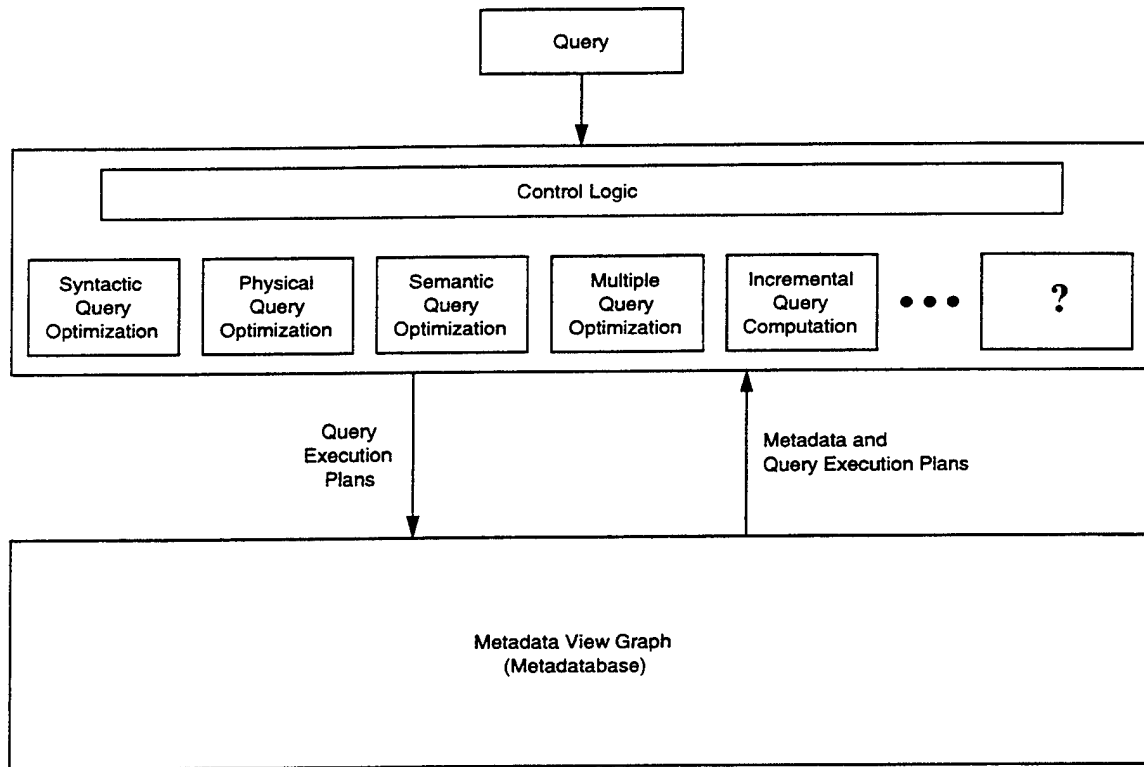


Figure 1: The MVG Framework.

The lexicon contains an entry for each term (word or phrase) recognized by the system (i.e., the system's vocabulary). A lexical entry provides information about the term, including a set of pointers to the semantic nodes that represent the term. In general, a lexicon will store any information about a term that is useful to the system.

Semantic networks represent domain knowledge about the concepts "understood" by the system. Each concept is represented as a node. Two nodes are linked together to represent their relationship to each other. In Figure 2, the semantic network is represented by nodes a_1 through a_8 and S_1 through S_4 . Nodes a_1 through a_8 represent the attributes that participate in one or more of the base relations (nodes R_1 through R_3). The attribute nodes are linked directly to their corresponding base relations.

The attribute names and types are specified in the data definition of the database. These nodes form the foundation of the semantic network. The network can be extended by defining nodes and links for application specific concepts and relationships. For example, two nodes representing STUDENT and ADVISOR could be connected by the links ADVISED-BY and ADVISOR-OF. The lexicon and semantic network are not relevant to the research presented in this paper.

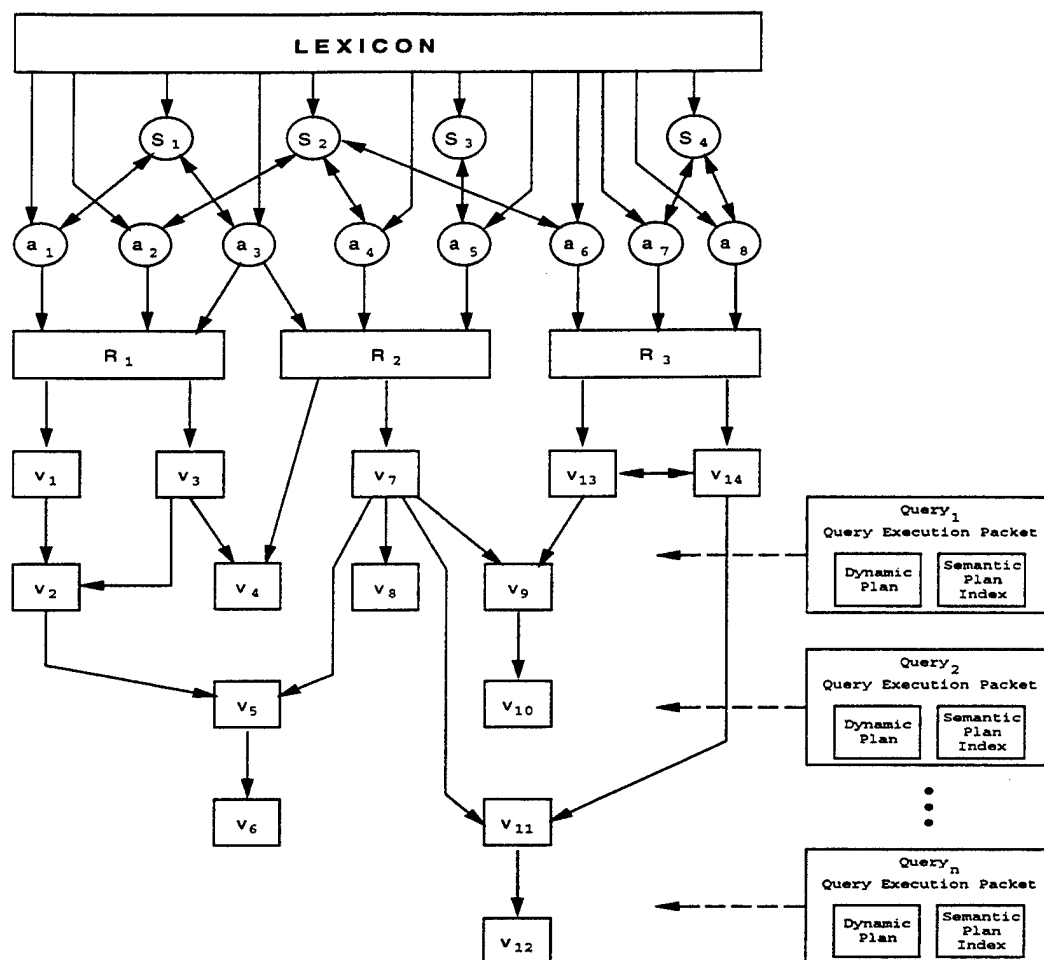


Figure 2: Conceptual Representation of a Metadata View Graph.

The View Network is an extension of Roussopoulos' Logical Access Path schema [Rou82]. The View Network stores semantic, statistical, and structural metadata that is useful to the query optimizer. The view nodes in the network, v_1 through v_{14} , represent *logical views* (intermediate results) and store metadata specific to the particular data set. A view is a projection of attributes which can be defined recursively as follows. All base relations are views. Additional views are the result of applying an operation (e.g., selection, projection, join) to a view or to a pair of views. The views represented by these nodes may or may not be materialized. The links represent logical operations and semantic relationships. The View Network, nodes R_1 through R_3 and v_1 through v_{14} , is essentially a collection of query graphs overlaid on top of each other where R_1 through R_3 represent base relations and v_1 through v_{14} represent the results of performing the operations specified by the links.

The View Network is a unified structure that applies to all of the application queries being served by the Metadata View Graph. The top level of the hierarchy consists of base relation nodes which anchor the Metadata View Graph and serve to connect the semantic network

to the View Network. The remaining nodes represent logical views. A view node is defined by the links connecting the node to the base relations. Figure 3 illustrates an example View Network along with a semantic network and lexicon.

The QEP Network stores two types of query execution plans, *dynamic plans* [GW89, CG94] and *semantic plans*. A semantic plan is a query execution plan that is semantically equivalent to the original query. Semantic plans are generated during semantic query optimization and depend on integrity constraints and instance-based semantic constraints.

A dynamic plan links several query execution plans together with choose-plan operators. An example of a dynamic plan is shown in Figure 4. Choose-plan operators allow a decision to be postponed until run-time when the run-time conditions are known. In order to select the best plan from the dynamic plan, the dynamic plan is traversed and the best path is chosen at each choose-plan operator. The statistics stored at the view nodes can guide the decision process. Therefore, the dynamic plan contains pointers to the view nodes with relevant metadata. If the statistics at a view node are out of date, the statistics must be updated before a decision can accurately be made.

The QEP Network maintains a separate *query execution packet* (i.e., a dynamic plan and semantic plan index) for each application query. Figure 4 illustrates how the query execution plans are linked to the logical access paths of the View Network so that the relevant view nodes can be retrieved for each plan. When a query is received at run-time, the query's execution packet is retrieved from the QEP Network and the metadata at the view nodes is used to select the best plan. When a semantic plan is selected as a candidate, the constraints it depends on must be updated and verified against the current state of the database before the plan can be executed. If one of the dependencies has been violated, then the plan is no longer guaranteed to be correct.

2.2 Using Metadata View Graphs

Metadata View Graphs are used at compile-time and run-time. Metadata is collected at compile-time and stored in the View Network. Metadata is retrieved at run-time and used to select the best query execution plan.

2.2.1 Compile-Time

The following high-level algorithm describes how the View Network and the QEP Network are constructed incrementally when a query is compiled.

1. When a query is received at compile-time, the query is optimized with conventional optimization techniques in order to generate a set of logical access paths.

2. The logical access paths are used to construct a separate View Network for the query being compiled. The existing MVG View Network is searched for (partial) matching view nodes.
3. The logical access paths are translated into query execution plans. The cost of each plan is estimated and the plans are filtered to remove any non-competitive plans.
4. Constraints and statistics are collected for the view nodes in the query's View Network. If a node already exists in the MVG View Network, metadata may not have to be collected for that node if the node's metadata is up to date.

In order to collect metadata, each logical access path will have to be executed. If the query being compiled contains variables, the query history will be used to substitute values for the variables. These are the variable bindings most likely to occur in future queries.

5. The constraints collected in step 4 are used by the optimizer to generate additional query execution plans. Semantic query optimization, multiple query optimization, and incremental query computation can apply the instance-based constraints that were collected.
6. If a new set of query execution plans are produced in step 5, the plans are evaluated using the statistics that were collected. The non-competitive plans are discarded and the query's View Network is modified to include any additional view nodes. Steps 4 and 5 are repeated until no new query execution plans are generated.
7. A query execution packet is created for the query. A dynamic plan is constructed for the non-semantic query execution plans, and an index is created for the semantic plans. The dynamic plan and the semantic plan index are stored in the query's execution packet.
8. The query's View Network is unified with the MVG's View Network (i.e., if a view node in the query's View Network does not already exist in the MVG's View Network, the node is added to the MVG's View Network at the correct location and the MVG View Network is reorganized).

Constraint Discovery

Our research does not address constraint discovery. This section describes existing methods for constraint discovery and discusses the advantages provided by the Metadata View Graph Framework.

Two primary problems hinder constraint discovery: (1) determining *where* to search, and (2) determining *what* to search for. Focusing on views of the database reduces the search space and produces more useful constraints. Searching smaller data sets, as opposed to searching the entire database, improves the performance of the discovery methods. Therefore, the view nodes of the Metadata View Graph determine where to search.

Two basic methods are used to determine what to search for: (1) query-driven methods, and (2) data-driven methods. Query-driven methods use a top-down process to search for constraints that would have been useful for previous queries. Data-driven methods use a bottom-up process to search (random) data sets for constraints. Metadata View Graphs provide a framework for integrating the top-down and bottom-up processes.

Query Driven Methods

[SSS92] presents a query-driven method for discovering constraints. Given a query, the semantic query optimizer identifies the *template* transformation rules that would have been useful to the optimizer, and the system discovers constraints that fit the rule templates. This strategy reduces the search space by only considering data sets that are relevant to queries that have been received. The disadvantage of this strategy is that a query will only benefit from the constraints that have been discovered if the query is similar to a previous query.

Reverse Engineering Method

After a query has been executed, [YS89, HK94] inspect the query result and attempt to discover relationships with other queries. For example, if the (intermediate) results of two queries are identical, then there must be some constraints relating the two queries. This is a type of query-driven approach that requires two similar queries before any constraints are discovered. In addition, this method requires that the results of previous queries be stored and matched against future queries.

Data Driven Methods

[SHKC93] has proposed a data-driven approach that uses grid files to inspect combinations of attribute values for a given data set. The zeros in the grid file indicate constraints. The advantage of this approach is that constraints can be found regardless of the query history. However, since it is impractical to search the entire database, there is no guarantee that the discovered constraints will apply to a query.

MVG Guidelines

The View Network is constructed for the application queries that have been compiled by the system, thus providing queries for the query-driven methods. In addition, the View Network identifies relevant data sets for the data-driven methods. Therefore, the View Network provides a foundation for integrating the top-down and bottom-up constraint discovery processes.

The semantic query optimization transformation types should be used to guide the discovery process. This guideline will focus the data-driven methods on constraints that are useful given the current structure of the database.

For example, one transition type attempts to introduce an index into the query condition. Therefore, the indexed attributes of each base relation should be explored since constraints involving these attributes could lead to rules that introduce indexes. Another transformation

attempts to eliminate operations such as a join between two views. In some cases, range constraints for the join attributes of each view can determine that the result of a join is empty in which case the operation can be eliminated.

The structure of the View Network, which consists of chains of nodes organized in a subsumption hierarchy, can also be used to guide the data-driven techniques. The constraints that exist at higher level nodes can be propagated to the nodes below, provided they apply to the nodes below, and then tightened to reflect the contents of the more restricted view. At the top level, the scheme-based integrity constraints that already exist for a database can be restricted to reflect the actual contents of the base relations.

Run-Time

When a query is received at run-time, the query's execution packet (i.e., the query's dynamic plan and its semantic plan index) is retrieved from the QEP Network along with any relevant view nodes from the View Network. The semantic plans are indexed so that the run-time bindings of the query can be used to select the semantic plans that match the conditions of the query. A semantic plan contains a set of pointers to the instance-based constraints it depends on (i.e., the constraints used to generate the plan). These dependencies must be verified for the current state of the database before a semantic plan can be executed. If any of the constraints that a plan depends on are no longer valid, the semantic plan is not guaranteed to be correct.

Each intermediate result in a query execution plan indexes a (possibly empty) set of view nodes with relevant metadata (i.e., metadata that is useful for predicting statistics about the intermediate result). When plans are being compared, the statistics at the view nodes are used to estimate the cost of each plan. However, before the plans can be evaluated, the metadata must be refreshed to reflect any updates made to the database.

The query optimizer selects the best non-semantic plan from the query's dynamic plan. If it is cost effective to update the instance-based constraints, then the constraints are updated and the semantic plans are evaluated. The best query execution plan is selected for execution. The Metadata View Graph adds the query to its query log along with a time-stamp and any other data which may have been collected during execution of the query.

Although this scheme moves most of the optimization effort to the compile phase, it does not preclude run-time optimization. For example, if several queries are received within a reasonable time frame, multiple query optimization can be performed.

2.3 Selecting a Query Execution Plan at Run-Time

Consider the two base relations and the *template query*, Q_1 , shown below. A template query contains one or more variables. Instantiations of a template query are received at run-time with all of the variables bound.

Relation	Attributes	Q_1 : Select	GPA
Students	snum, class, GPA, advisor	From	Employees, Students
Employees	enum, salary, dept, pos	Where	pos = student AND dept = var_1 AND enum = snum

Query Q_1 requests the grade point averages of the students in the var_1 department who are employed. Figure 3 illustrates part of a view network that supports this query. In this example, there are four departments (Psychology, Math, Computer Science, and Business). The View Network is not required to contain a view node for every department. Only the most frequently accessed views, based on the query history, will be represented in the view network. For example, only three class views ($v_9 - v_{11}$) are represented for the Student base relation.

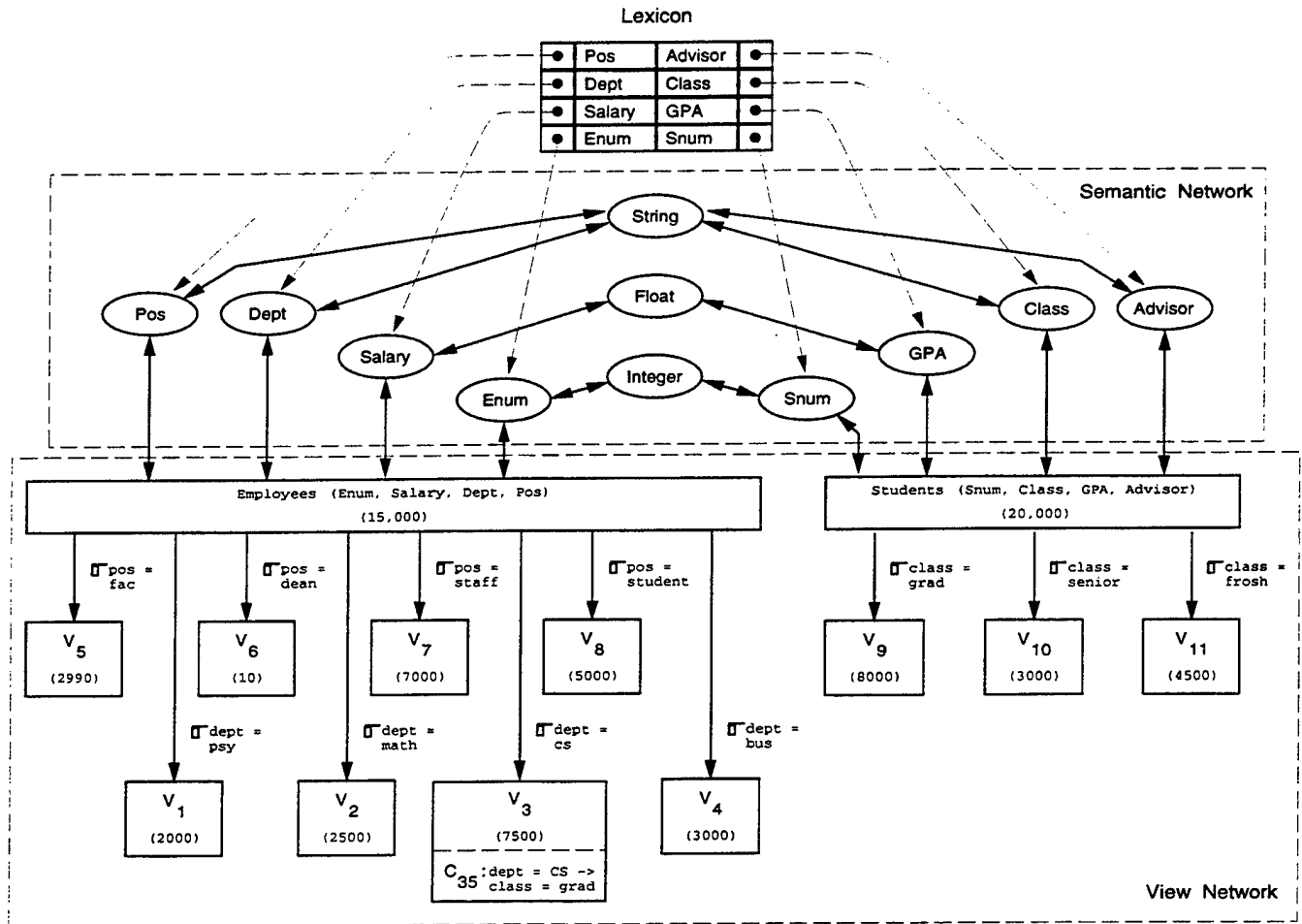


Figure 3: Example Lexicon, Semantic Network, and View Network.

Figure 4 shows the Query Execution Packet for query Q_1 . The packet contains the dynamic plan and the semantic plan index. There is one choice to be made in the dynamic plan. The

two selections can be performed in either order. The first two filter operations point to the view nodes that contain statistics to determine which path to take. Once var_1 is bound at run-time, the choice is obvious. If $DEPT = CS$, then $POS = STUDENT$ produces a smaller intermediate result (5000 tuples) than $DEPT = CS$ (7500). For the other three departments, however, selecting the department first produces the smaller intermediate result.

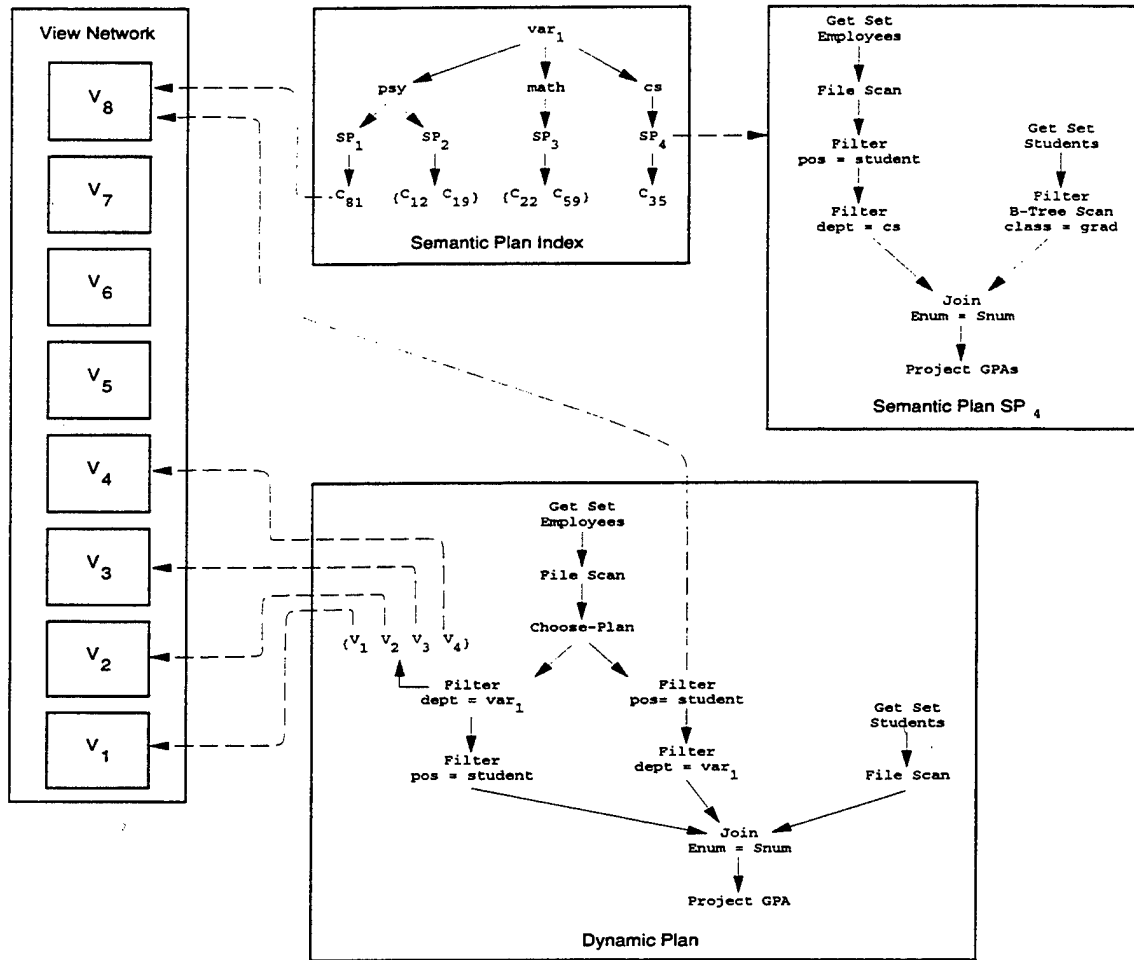


Figure 4: A global perspective of selecting a query execution plan at run-time.

Semantic plans are indexed by the bindings of the variables. The packet contains four semantic plans, SP_1 , SP_2 , SP_3 , and SP_4 , corresponding to the variable bindings $var_1 = PSY$, $var_1 = MATH$, and $var_1 = CS$ respectively. There are two semantic plans for the PSY binding. Each plan points to the set of constraints that the plan depends on.

Assume an instantiation of query Q_1 is received at run-time with var_1 bound to CS. The semantic plan index would return plan SP_4 which depends on constraint C_{35} which is stored at view node v_3 . The semantic plan SP_4 is shown in Figure 4. Constraint C_{35} , shown in Figure 3, states that if the department is CS then the class must be GRAD. In other words, only graduate students from the computer science department are employed.

Assuming there is an index on the CLASS attribute of the Student base relation, the semantic plan SP_4 can use that index to reduce the size of the join between the Student base relation and the view of employed computer science students. Furthermore, because the semantic plan was designed for a specific binding of var_1 , the choose-plan operator can be omitted since a drastic change in the database would be required before the number of students would be greater than the number of computer science employees.

Note that constraint C_{35} is not an integrity constraint. Undergraduate computer science students can be employed. Therefore, if the database were updated to include an employed undergraduate from the computer science department, the semantic plan SP_4 would no longer be valid. Consequently, the constraint C_{35} must be updated and verified against the current state of the database before the semantic plan can be selected for execution. If an update invalidates a constraint, then none of the semantic plans that depend on the invalid constraint can be executed because the plans are not guaranteed to be correct.

3 Maintaining Instance-Based Constraints

When a semantic plan is selected from the QEP Network, the constraints that the plan depends on must be refreshed and verified for the current state of the database. Constraints may be out of date if one or more updates have been received by the system since the last time the constraints were refreshed. However, only a subset of the updates will apply to a view node based on the definition of the node (e.g., $GPA \geq 3.5$). Therefore, the updates must be *filtered* to remove the *irrelevant updates* (i.e., those updates that do not apply to the view node being updated) [BLT86]. An example is provided below.

Consider the Student base relation and the view node shown in Figure 5. The view node shown in this example contains seven constraints, a tuple count, a distribution profile, and a view cache pointer (since the pointer is nil, there is no view cache for this node). The view is defined for students with a GPA of 3.0 or greater. The constraints and tuple count at the view node can be verified by selecting the tuples from the base relation (as shown) that satisfy the definition of the node (i.e., $GPA \geq 3.0$). An update contains a unique time-stamp, which indicates when the update was received, along with the tuple to be inserted or deleted. Updates to the database are maintained in a set of logs, one log per base relation. In order to refresh the metadata at this node, the updates in the base relation update log must be filtered to select the updates that satisfy the definition of the node. Each update with a GPA of 3.0 or greater can then be applied to each metadatum at the node.

Before the updates can be filtered and applied to the metadata, the update logs and the view nodes to be updated must be retrieved from disk. The cost of these disk accesses dominates the cost of the update process. Therefore, maintaining metadata can be divided into three subproblems: (1) managing the update logs, (2) managing the view nodes, and (3) refreshing the metadata at a view node. The rest of this paper analyzes the problem of refreshing constraints once the updates and the constraints are in main memory.

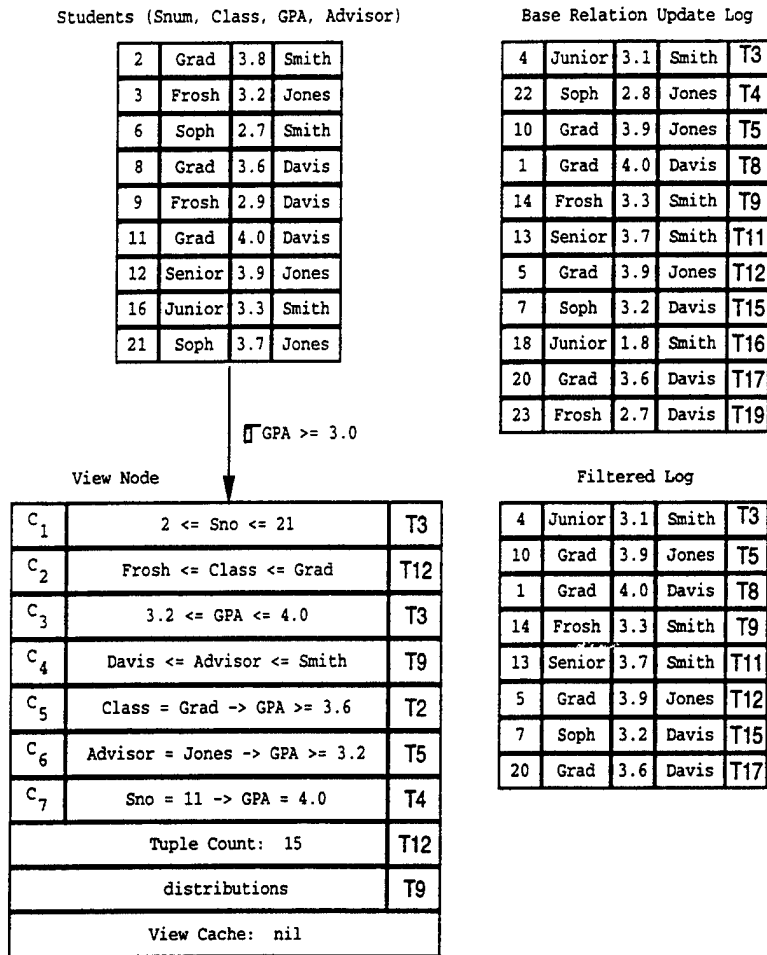


Figure 5: The Student base relation, base relation update log, view node, and filtered log.

3.1 Representing Constraints

Constraints are represented in First Order Predicate Logic. However, constraints must maintain additional information in order to be maintained efficiently. This section begins by considering the affects that insertions and deletions have on instance-based constraints, and then the section considers several representations that improve the maintainability of instance-based constraints.

3.1.1 Update Affects

The Metadata View Graph stores semantic query execution plans that depend on certain conditions. Consider a semantic plan, SP_1 , that requires that all graduate students have a GPA greater than 3.0. If all of the graduate students in the database have a GPA of 3.2 or

greater, then the instance-based constraint, $C_1: \text{CLASS} = \text{GRAD} \Rightarrow \text{GPA} \geq 3.2$, indicates that the semantic plan SP_1 is valid.

Assume there is only one graduate student with a 3.2 GPA and assume that the student is deleted. The deletion does not have to be applied to C_1 because no deletion could violate the required condition, $\text{CLASS} = \text{GRAD} \Rightarrow \text{GPA} > 3.0$. In general, unless a constraint implies existence, deletions do not have to be applied to valid constraints because deletions cannot invalidate the required conditions with respect to the semantic plans that are stored in the Metadata View Graph.

However, assume a graduate student with a 2.8 GPA is inserted into the database. The insertion must be applied to C_1 because the update violates the required condition for SP_1 thus invalidating the plan. Now assume that the graduate student with GPA 2.8 is deleted and assume that the remaining graduate students all have a GPA greater than 3.0. The required condition for SP_1 is now satisfied. Therefore, this deletion should be applied to C_1 in order to validate the plan.

When insertions are made to the database, it is easy to modify constraints because all of the necessary information is contained in the update. However, when deletions occur, the constraint must represent additional information in order to *recompute* the correct constraint.

3.1.2 Efficient Representations

Consider a student base relation with four attributes: student number (Snum), class (e.g., frosh, grad), GPA, and advisor. Consider a view node defined for $\text{GPA} \geq 3.9$ and assume that constraint C_1 is stored at the view node. Assume that student 4 is a graduate student advised by Smith with a 3.92 GPA. If student 4 is inserted at time T_2 , constraint C_1 can be modified as shown below. However, if student 4 is deleted at time T_3 , the constraint cannot be recomputed.

$T_1: C_1: \text{Class} = \text{Grad} \Rightarrow \text{Advisor} = \text{Jones}$

$T_2: \text{Insert: Snum} = 4, \text{Class} = \text{Grad}, \text{GPA} = 3.92, \text{Advisor} = \text{Smith}$

$T_2: C_1: \text{Class} = \text{Grad} \Rightarrow (\text{Advisor} = \text{Jones}) \text{ OR } (\text{Advisor} = \text{Smith})$

$T_3: \text{Delete: Snum} = 4, \text{Class} = \text{Grad}, \text{GPA} = 3.92, \text{Advisor} = \text{Smith}$

$T_3: C_1: \text{Class} = \text{Grad} \Rightarrow \text{Advisor} = \text{Jones}$

When student 4 is deleted, the constraint should be modified as shown at T_3 . However, with this representation, the system must materialize the view ($\text{GPA} \geq 3.9$) in order to discover that there are no graduate students in the view who are advised by Smith. We will consider two solutions to this problem.

1. Recompute the constraints at run-time during query execution by testing all of the tuples in the intermediate result. This can be done by saving the intermediate result, possibly to disk, and recomputing all of the constraints at a node, or by recomputing a few constraints on the fly while the query is being executed.
2. When the attribute of a term can be enumerated, as in the example above, keep a counter for each term.

The first solution requires additional processing during run-time to recompute the constraints. We can assume that the processing cost is negligible or that the processing is performed when the system is idle. However, the problem with this solution is that the constraints are recomputed during or after query execution. Therefore, the constraints cannot be used for the given query.

Consider semantic plan SP_1 which requires that all graduate students have a GPA greater than 3.0. Assume that SP_1 is valid at time T_1 . At time T_2 , a graduate student with a 2.8 GPA is inserted which invalidates semantic plan SP_1 . At T_3 , however, the graduate student with GPA 2.8 is deleted. At this point, semantic plan SP_1 is valid, but the plan cannot be used because constraint C_1 cannot be recomputed until the query is executed.

Therefore, the constraints will have to be recomputed every time a deletion affects the view node. Consequently, a constraint may thrash between being valid and invalid and the semantic plans that depend on the constraint will never be usable even though the required conditions are met.

The second solution is preferable, but this representation does not apply to attributes with non-enumerable values. Consider the following representation for constraint C_1 . This representation maintains the number of tuples that apply to each condition. At time T_1 , there are 10 graduate students in the view and all 10 are advised by Jones.

T_1 : C_1 : Class = Grad (10) \Rightarrow Advisor = Jones (10)

T_2 : Insert: Snum = 4, Class = Grad, GPA = 3.92, Advisor = Smith

T_2 : C_1 : Class = Grad (11) \Rightarrow (Advisor = Jones (10)) OR (Advisor = Smith (1))

T_3 : Delete: Snum = 5, Class = Grad, GPA = 3.94, Advisor = Jones

T_3 : C_1 : Class = Grad (10) \Rightarrow (Advisor = Jones (9)) OR (Advisor = Smith (1))

T_4 : Delete: Snum = 4, Class = Grad, GPA = 3.92, Advisor = Smith

T_4 : C_1 : Class = Grad (9) \Rightarrow Advisor = Jones (9)

Student 4 is inserted at T_2 and the constraint is modified. There are now 11 graduate students in the view, 10 are advised by Jones and 1 is advised by Smith. Student 5 is deleted at T_3

and the constraint is modified accordingly. Once again there are 10 graduate students in the view, but now there are 9 students advised by Jones and 1 advised by Smith. Finally, student 4 is deleted at T_4 . The constraint is modified to reflect that all 9 graduate students in the view are advised by Jones.

This representation works well for attributes with values that can be enumerated. Some constraints have attribute values that cannot be enumerated. For example, consider the range constraint on GPA shown below.

T_{10} : C_2 : GPA \geq 3.56

T_{11} : Insert: Snum = 4: GPA = 3.5

T_{11} : C_2 : GPA \geq 3.5

T_{12} : Delete: Snum = 4: GPA = 3.5

Initially, the lowest GPA for the given view is 3.56. When student 4 is inserted at time T_{11} , the lowest GPA becomes 3.5. However, when student 4 is deleted at T_{12} , there is no way to determine the exact value of the lowest GPA in the view without materializing the view. However, constraint C_2 still represents a lower bound on GPA. The constraint asserts that all of the GPAs are greater than or equal to 3.5, but the constraint does not represent the exact value of the lowest GPA.

The representation shown below maintains a *boundary value list* of the lowest GPAs. This representation can be used to determine the exact lower bound provided the boundary value list is not empty.

T_{10} : C_2 : GPA \geq (3.56, 3.57, 3.57, 3.58, 3.59)

T_{11} : Insert: Snum = 4: GPA = 3.5

T_{11} : C_2 : GPA \geq (3.5, 3.56, 3.57, 3.57, 3.58, 3.59)

T_{12} : Delete: Snum = 4: GPA = 3.5

T_{12} : C_2 : GPA \geq (3.56, 3.57, 3.57, 3.58, 3.59)

Initially the boundary value list contains the 5 lowest GPAs in the view. When student 4 is inserted at T_{11} , student 4's GPA is the lowest GPA in the view. Therefore, student 4's GPA is added to the front of the boundary value list. When student 4 is deleted at T_{12} , one instance of the value 3.5 (student 4's GPA) is removed from the boundary value list.

If the boundary value list becomes empty, because all of the values are deleted and no insertions replace them, then the view must be materialized in order to determine the exact lower bound. The last value in the boundary value list can be retained in order to provide a lower bound for the constraint. In this case, the retained value must be flagged as invalid so that it can be removed when the boundary value list is recomputed. If an insertion is received with a GPA less than or equal to the retained value, then the inserted GPA will replace the invalid GPA and the constraint will once again reflect the exact lowest GPA in the view.

For example, consider constraint C_2 shown above and assume that the students with GPA 3.56, 3.57, 3.57, 3.58, and 3.59 are deleted in that order. C_2 will become $C_2: \text{GPA} \geq (3.59)$, and the value 3.59 will be flagged as invalid. C_2 now represents an inexact lower bound. Now assume that a student with GPA 3.55 is inserted. The value 3.59 will be replaced by 3.55 and C_2 will represent the lowest GPA in the view, $C_2: \text{GPA} \geq (3.55)$.

Consider constraints C_3 and C_4 shown below. Each of these constraints involve attributes that require boundary value lists.

$C_3: (\text{GPA} \geq 3.54) \text{ AND } (\text{GPA} \leq 3.58) \Rightarrow \text{Advisor} = \text{Jones } (3)$

$C_4: (\text{SALARY} \geq 12\text{K}) \text{ AND } (\text{SALARY} \leq 15\text{K}) \Rightarrow (\text{GPA} \geq 3.2) \text{ AND } (\text{GPA} \leq 3.8)$

The counter (3) associated with C_3 's condition ($\text{Advisor} = \text{Jones}$) indicates that three students satisfy the constraint. Without boundary value lists for GPA, as shown below, the system cannot determine the middle GPA without materializing the view. Constraint C_4 would require four boundary value lists, one list for each term.

$C_3: (\text{GPA} \geq (3.54, 3.55)) \text{ AND } (\text{GPA} \leq (3.55, 3.58)) \Rightarrow \text{Advisor} = \text{Jones } (3)$

3.2 Refreshing Constraints

This section presents an algorithm for applying an update to an instance-based constraint in order to refresh the constraint. Consider constraint C_5 , shown below, which asserts that if a student's GPA is greater than 3.9, then the student is advised by Jones.

$C_5: \text{GPA} > 3.9 \Rightarrow \text{ADVISOR} = \text{JONES}$

The left hand side of a constraint is a list of terms. Each term represents a condition. The right hand side of a constraint is also a list a terms, but the terms on the right hand side represent assertions. Constraint C_5 has one condition on the left hand side ($\text{GPA} > 3.9$) and one assertion on the right hand side ($\text{ADVISOR} = \text{JONES}$).

An update applies to a constraint if it satisfies all of the conditions on the left hand side. As soon as one condition is not satisfied, the update can be disregarded with respect to the constraint being refreshed. If an update satisfies the conditions of the constraint, then the assertions on the right hand side must be considered. In this example, if the update's GPA

is greater than 3.9 and the advisor is Jones then the update does not affect the constraint. However, if the update satisfied the condition ($GPA \geq 3.9$) and the advisor is not Jones, then there are three options: (1) modify the assertion, (2) modify the condition, or (3) modify both the assertion and the condition by creating another constraint.

For example, assume a student is added to the database with a GPA of 3.92 and the student is advised by SMITH. The student's GPA satisfies the condition of the constraint, but the assertion no longer holds. The following two constraints can be created.

$GPA > 3.92 \Rightarrow ADVISOR = JONES$

$GPA > 3.9 \Rightarrow (ADVISOR = JONES) \text{ OR } (ADVISOR = SMITH)$

The following algorithm applies one update to an instance-based constraint.

- 1 If the update satisfies the condition then
- 2 If the assertion no longer holds then
- 3 Modify the constraint

As described above, an update may alter the representation of a constraint without affecting the validity of the constraint. For example, a deletion may add or remove a value from the boundary value list. In this case, the constraint would have to be modified, but the assertion would still hold.

Range Constraints

Specific algorithms can be developed to efficiently process some of the constraint types. For example, if there are range constraints to be updated, instead of processing each update individually for each range constraint, a single pass through the updates can collect the high and low values for each attribute with a range constraint. The high and low values can then be applied to each constraint. Furthermore, the pass that collects the high and low values can be performed during the filtering process. As the new updates are filtered, all of the range constraints at a node can be updated with only a few additional operations.

For example, consider a view node that represents graduate students and assume that the lowest GPA in the view is 3.1. Constraint C_6 represents the lower bound on GPA at the view, $C_6: GPA \geq 3.8$. Assume the following six insertions are received by the system: (grad, 3.3), (grad, 3.0), (frosh, 3.6), (soph, 3.4), (senior, 2.7), (grad, 3.8). In order to update the graduate student view, these updates must be filtered to remove the students that are not graduates. When the updates are filtered (i.e., tested for $CLASS = GRAD$), the system can maintain the lowest GPA for the graduate student updates. When these six updates are filtered, the three irrelevant updates will be removed and the lowest graduate GPA for the insertions will be recored as 3.0. The lowest GPA for the updates (3.0) will be compared with the GPA range constraint at the view node (3.1), and the range constraint will be modified if the insertions have a lower GPA than the current range constraint. In this example, the range constraint will be modified to reflect the 3.0 GPA that has been inserted.

4 Conclusion

Instance-based constraints are more useful to semantic query optimization because they contain more information than scheme-based constraints. This paper presented a framework for maintaining instance-based constraints. The Metadata View Graph Framework makes three contributions: (1) the framework maintains instance-based constraints for logical views of the database, (2) the framework provides a foundation that directs and integrates existing methods for constraint discovery, and (3) the framework allows instance-based constraints to be retrieved efficiently at run-time.

The problem of maintaining instance-based constraints in the Metadata View Graph can be decomposed into three sub-problems: (1) manage the update logs, (2) manage the view nodes, and (3) refresh the instance-based constraints. This paper analyzed the third sub-problem and considered various representations that improve maintenance efficiency.

Future research will develop efficient strategies for managing update logs and view nodes (i.e., the first two subproblems). Future research will also continue to analyze instance-based constraints in order to develop more efficient update strategies, such as the strategy presented for range constraints, and further classify the properties of instance-based constraints with respect to semantic query optimization and maintenance.

5 Acknowledgements

The author wishes to thank Leo Mark and Shamkant Navathe for their many comments and discussions regarding this research. The author has been generously supported by BNR Inc., the research and development subsidiary of Northern Telecom, and is especially grateful to Robert Bloedon and Deborah Stokes. The author also acknowledges the support of the Advanced Project Research Agency under contract number F33615-93-1-1338. The current work is part of the project entitled: "A Knowledge Based Approach to Integrating and Querying Distributed Heterogeneous Information Systems."

References

- [BLT86] J. Blakeley, P. Larson, and F. Tompa. Efficiently updating materialized views. In C. Zaniolo, editor, *Proceedings of the 1986 ACM SIGMOD International Conference on the Management of Data*, pages 61-71, Washington, D.C., May 1986.
- [CG94] Richard L. Cole and Goetz Graefe. Optimization of dynamic query evaluation plans. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 150-160, Minneapolis, Minnesota, May 1994.

- [CGM90] S. Chakravarthy, J. Grant, and J. Minker. Logic based approach to semantic query optimization. *ACM Transactions on Database Systems*, 15(2):162–207, June 1990.
- [Fre87] J.C. Freytag. A rule-based view of query optimization. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 173–180, San Francisco, May 1987.
- [GD87] G. Graefe and D. DeWitt. The exodus optimizer generator. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 160–171, San Francisco, May 1987.
- [GM93] G. Graefe and W.J. McKenna. The volcano optimizer generator: Extensibility and efficient search. In *Proceedings of the IEEE Conference on Data Engineering*, pages 209–228, Vienna, April 1993.
- [GW89] G. Graefe and K. Ward. Dynamic query evaluation plans. In *Proceedings of the 1989 ACM-SIGMOD International Conference on the Management of Data*, pages 358–366, Portland, Oregon, 1989.
- [HK93] C. Hsu and C.A. Knoblock. Reformulating query plans for multidatabase systems. In *Proceedings of the Second International Conference on Information and Knowledge Management*, Washington, DC, 1993.
- [HK94] Chun-Nan Hsu and Craig Knoblock. Rule induction for semantic query optimization. *Machine Learning*, pages 1–10, 1994.
- [HZ80] M. Hammer and S.B. Zdonik. Knowledge-based query processing. In *Proceedings of the Sixth International Conference on Very Large Data Bases*, pages 137–147, Montreal, October 1980.
- [Kin81] J. King. Quist: A system for semantic query optimization in relational databases. In *Proceedings of the Seventh International Conference on Very Large Data Bases*, pages 510–517, 1981.
- [PMN95] J. Pittges, L. Mark, and S. Navathe. Metadata view graphs: A framework for query optimization and metadata management. *ACM Transactions on Information Systems*, 1995. Unpublished – submitted, Nov. 1994.
- [Rou82] N. Roussopoulos. The logical access path scheme of a database. *IEEE Transactions on Software Engineering*, SE-8(6):563–573, November 1982.
- [SHKC93] S. Shekhar, B. Hamidzadeh, A. Kohli, and M. Coyle. Learning transformation rules for semantic query optimization: A data-driven approach. *IEEE Transactions on Knowledge and Data Engineering*, 5(6):950–964, 1993.
- [SO87] S.T. Shenoy and Z.M. Ozsoyoglu. A system for semantic query optimization. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 16(3):181–195, December 1987.

- [SO89] S.T. Shenoy and Z.M. Ozsoyoglu. Design and implementation of a semantic query optimizer. *IEEE Transactions on Knowledge and Data Engineering*, 1(3):344–361, 1989.
- [SSD92] Shashi Shekhar, Jaideep Srivastava, and Soumitra Dutta. A formal model of trade-off between optimization and execution costs in semantic query optimization. *Data and Knowledge Engineering*, 8:131–151, 1992.
- [SSS92] M. Siegel, E. Sciore, and S. Salveter. A method for automatic rule derivation to support semantic query optimization. *ACM Transactions on Database Systems*, 17(4):563–600, December 1992.
- [YS89] C. Yu and W. Sun. Automatic knowledge acquisition and maintenance for semantic query optimization. *IEEE Transactions on Knowledge and Data Engineering*, 1(3):362–375, September 1989.

Maintaining Semantic and Structural Metadata in the Metadata View Graph Framework

Jeff Pittges
Leo Mark
Shamkant B. Navathe

College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280
{pittges, leomark, sham}@cc.gatech.edu

November 13, 1997

Abstract

The Metadata View Graph is a metadatabase capable of maintaining semantic and structural metadata for views of a database. Semantic metadata provides dynamic rules which are used during query optimization and structural metadata provides indexes which are used during query execution. Since both types of metadata represent the current contents of the database, both types of metadata must be maintained when the contents of the database change.

Although both types of metadata use the same update logs, these logs are typically processed twice because the semantic metadata must be maintained *before* query execution while structural metadata is usually maintained *during* query execution. However, when a query execution plan requires both types of metadata, it is most efficient to process the update logs once and maintain both types of metadata at the same time. This creates a conflict when the update paths for the semantic and structural metadata overlap.

This paper presents several methods for efficiently maintaining semantic and structural metadata. The paper analyzes the maintenance cost for both types of metadata and proposes two strategies for processing overlapping update paths.

Keywords: Metadata Maintenance, View Maintenance, Constraint Maintenance, Dynamic Semantic Rules, View Cache, Query Optimization.

1 Introduction

Dynamic rules (semantic metadata) are used by semantic query optimization [HK94, HK93, S⁺93, S⁺92, CGM90, SO89, YS89] to reformulate a query into a semantically equivalent query that is more efficient to execute. Since dynamic rules represent the current contents of the database, dynamic rules must be maintained before the query is executed in order to guarantee that the reformulated query is correct. For example, consider a query that requests the grade point average of the students who work in the computer science department. Figure 1 illustrates two query execution plans for this query. The *non-semantic plan* is produced directly from the original query expression. The *semantic plan* is produced by applying transformation rule R_{35} which states that only graduate students work in the computer science department.

The non-semantic plan performs the join operation with the entire Student base relation whereas the semantic plan only joins the graduate students. More importantly, the semantic plan may greatly reduce the number of pages retrieved from the Student base relation if the database maintains an index on the CLASS attribute. Note, however, that the dynamic rule (R_{35}) is only valid for particular instances of the database. If an undergraduate student from the CS department is later employed by the school, then the rule is no longer valid and the semantic plan cannot be used to answer the query. Therefore, the dynamic rule must be maintained before the query is executed.

Incremental query computation [Rou91] maintains views in order to answer queries more efficiently. An index (structural metadata) called a *view cache* maintains a set of pointers into the base relation tuples that belong to a given view. The view cache is used to materialize the view efficiently during query execution. For example, consider a view representing a join between two base relations, R_i and R_j . For each tuple in the view, the view cache will contain a pair of tuple IDs $\langle TID_i, TID_j \rangle$ where TID_i and TID_j are pointers to the tuples in R_i and R_j that form the tuple in the view. View caches are maintained incrementally during query execution by propagating relevant updates while the view is being materialized. Updating the view cache is often more efficient than recomputing

The rest of the paper is organized as follows. Section 2 presents the Metadata View Graph Framework and describes how metadata is stored, retrieved, maintained, and used during query processing. The framework is presented to help the reader appreciate the problems addressed by the paper. Section 3 specifies the maintenance problem, analyzes the maintenance cost for both types of metadata, and proposes several strategies and algorithms for maintaining semantic and structural metadata. The last section contains our concluding remarks.

2 The Metadata View Graph Framework

As illustrated in Figure 2, the Metadata View Graph [Pit95] is a metadatabase capable of maintaining semantic, statistical, and structural metadata for views of a database. The Metadata View Graph consists of four components: (1) a lexicon, (2) a semantic network, (3) a view network, and (4) a QEP Network. The lexicon maintains the system's vocabulary and contains information for each word or phrase that is recognized by the system interface. The semantic network captures domain knowledge and can be used to disambiguate queries. The lexicon and semantic network are used for query interpretation and will not be discussed further.

The View Network is an extension of Roussopoulos' Logical Access Path schema [Rou82]. The view nodes in the network ($V_1 - V_{14}$ in Figure 2) represent views of the database and store metadata specific to the particular data set. The links represent logical operations and semantic relationships. The View Network is essentially a collection of query graphs overlaid on top of each other with each view node representing an intermediate result.

The QEP Network maintains a separate *query execution packet* for each application query. Query execution packets store two types of query execution plans, *non-semantic plans* and *semantic plans*. Non-semantic plans are produced by conventional query optimization and do not require maintenance. Semantic plans are produced by semantic query optimization. Consequently, semantic plans depend on the dynamic rules that were used to reformulate the original query. These rules must be

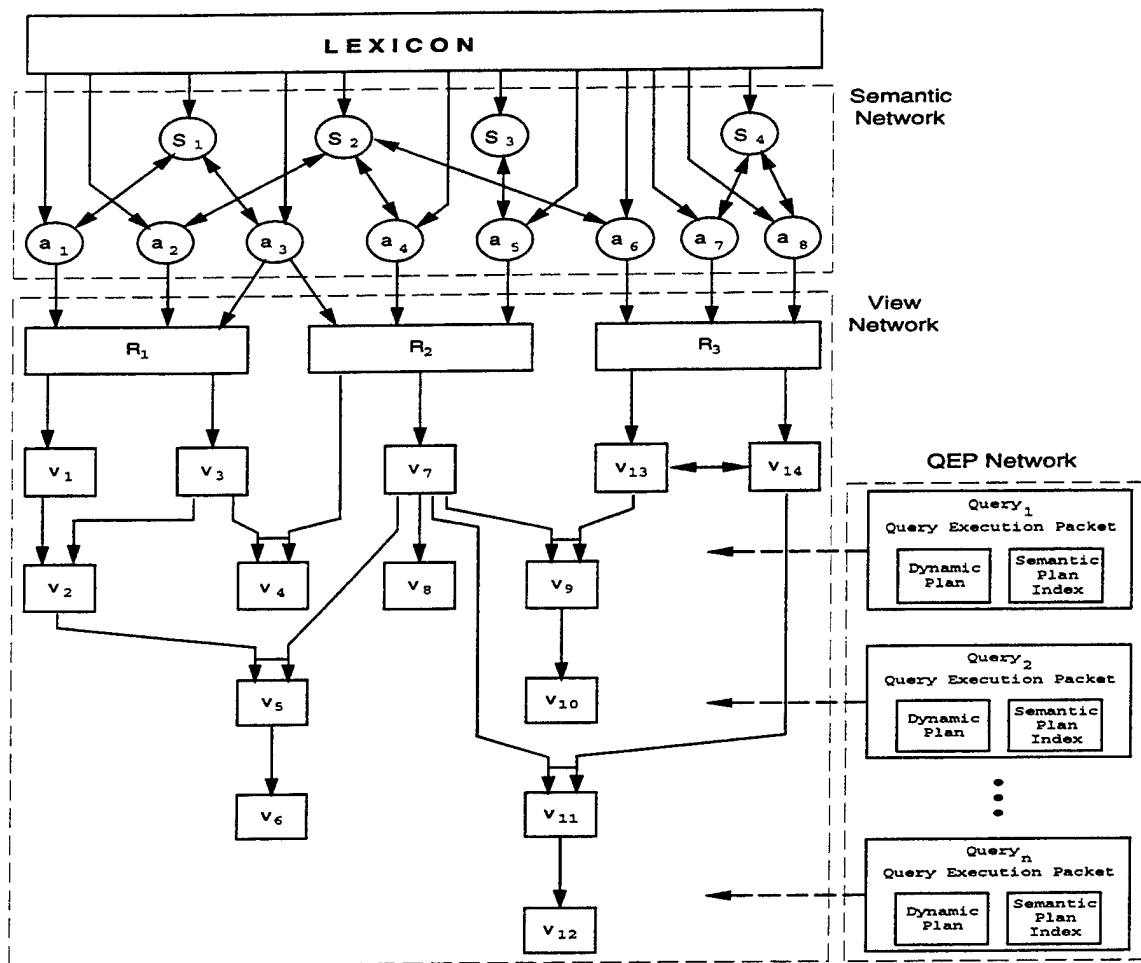


Figure 2: Conceptual representation of the Metadata View Graph.

updated and validated before a semantic plan can be executed. Therefore, semantic plans contain pointers to the rules that they depend on in order to efficiently retrieve and maintain those rules before the query is executed.

2.1 Query Optimization

When a query is processed at compile-time, several query execution plans are generated and metadata is collected for each query execution plan. We assume that existing methods have been used to discover the dynamic rules that are used to generate semantic plans. The most efficient query

execution plans are stored in the QEP Network. When a query is processed at run-time, the query's execution packet is retrieved from the QEP Network along with any relevant view nodes from the View Network. The semantic plans are indexed such that the run-time bindings of the query can be used to select the semantic plans that match the conditions of the query.

An Example

Figure 3 illustrates part of a View Network that supports our example query Q_1 . The figure also shows semantic plan SP_4 which depends on rule R_{35} stored at node V_3 . Rule R_{35} was used to reformulate Q_1 by introducing the selection condition $CLASS = GRAD$. When Q_1 is processed at run-time, semantic plan SP_4 will be retrieved from the QEP Network along with view node V_3 . If maintenance is cost effective, the metadata at node V_3 will be maintained, and if rule R_{35} is valid, then SP_4 will be executed. If R_{35} is not valid, the non-semantic plan for Q_1 will be executed.

Relation	Attributes	Q_1 : Select	GPA
Students	snum, class, GPA, advisor	From	Employees, Students
Employees	enum, salary, dept, pos	Where	pos = student AND dept = var_1 AND enum = snum

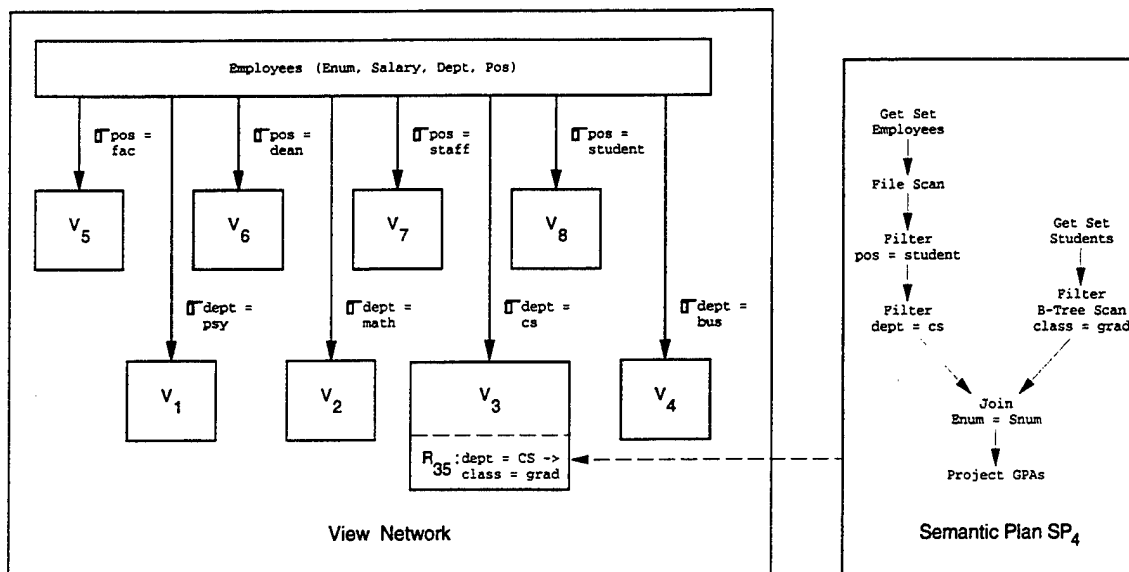


Figure 3: Semantic plan SP_4 depends on dynamic rule R_{35} which is stored in the View Network.

3 Maintaining Semantic and Structural Metadata

When a database evolves from one database state x to another database state y due to an update (i.e., a tuple is inserted, deleted, or modified), the metadata in the View Network may become invalid. In general, two actions must occur before the metadata can be used: (1) the metadata that is affected by the update must be identified, and (2) if the metadata is no longer valid, the metadata must be modified or invalidated.

Rather than consider each update as it is received, it is more efficient to process a batch of updates for a particular set of metadata when the metadata is needed. This is referred to as the *deferred update strategy* [Rou91]. This approach requires the system to maintain an update log for each base relation. When an update is made to a base relation, the system writes the update to the base relation's update log. When the metadata at a view node needs to be maintained, all of the updates for the base relations from which the view is derived are processed. However, only a subset of the updates will apply to a given view node based on the definition of the node (e.g., $GPA \geq 3.0$). Therefore, the updates must be *filtered* to remove the *irrelevant updates* [BLT86].

Figure 4 illustrates the contents of a base relation and a view node for two states of an example database. The figure also illustrates an update log for the base relation which contains a number of tuples to be inserted into the base relation. State x represents the database before the updates are received, and state y represents the database after the updates have been processed and the metadata at the view node has been maintained.

The view node shown in this example contains seven rules, a tuple count, a distribution profile, and a view cache pointer (since the pointer is nil, there is no view cache for this node). The view is defined for students with a GPA of 3.0 or greater. The rules and tuple count at the view node can be verified for states x and y by selecting the tuples from the base relation (as shown) that satisfy the definition of the node (i.e., $GPA \geq 3.0$). An update contains a unique time-stamp, which indicates when the update was received, along with the tuple to be inserted, deleted, or modified.

Students (Snum, Class, GPA, Advisor)

2	Grad	3.8	Smith
3	Frosh	3.2	Jones
6	Soph	2.7	Smith
8	Grad	3.6	Davis
9	Frosh	2.9	Davis
11	Grad	4.0	Davis
12	Senior	3.9	Jones
16	Junior	3.3	Smith
21	Soph	3.7	Jones

⌈ GPA >= 3.0

View Node

C ₁	2 <= Sno <= 21	T3
C ₂	Frosh <= Class <= Grad	T12
C ₃	3.2 <= GPA <= 4.0	T3
C ₄	Davis <= Advisor <= Smith	T9
C ₅	Class = Grad -> GPA >= 3.6	T2
C ₆	Advisor = Jones -> GPA >= 3.2	T5
C ₇	GPA = 4.0 -> Sno = 11	T4
Tuple Count: 15		T12
distributions		T9
View Cache: nil		

State X

Base Relation Update Log

4	Junior	3.1	Smith	T3
22	Soph	2.8	Jones	T4
10	Grad	3.9	Jones	T5
1	Grad	4.0	Davis	T8
14	Frosh	3.3	Smith	T9
13	Senior	3.7	Smith	T11
5	Grad	3.9	Jones	T12
7	Soph	3.2	Davis	T15
18	Junior	1.8	Smith	T16
20	Grad	3.6	Davis	T17
23	Frosh	2.7	Davis	T19

Filter
(GPA >= 3.0)

4	Junior	3.1	Smith	T3
10	Grad	3.9	Jones	T5
1	Grad	4.0	Davis	T8
14	Frosh	3.3	Smith	T9
13	Senior	3.7	Smith	T11
5	Grad	3.9	Jones	T12
7	Soph	3.2	Davis	T15
20	Grad	3.6	Davis	T17

Updates

Students (Snum, Class, GPA, Advisor)

1	Grad	4.0	Davis
2	Grad	3.8	Smith
3	Frosh	3.2	Jones
4	Junior	3.1	Smith
5	Grad	3.9	Jones
6	Soph	2.7	Smith
7	Soph	3.2	Davis
8	Grad	3.6	Davis
9	Frosh	2.9	Davis
10	Grad	3.9	Jones
11	Grad	4.0	Davis
12	Senior	3.9	Jones
13	Senior	3.7	Smith
14	Frosh	3.3	Smith
16	Junior	3.3	Smith
18	Junior	1.8	Smith
20	Grad	3.6	Davis
21	Soph	3.7	Jones
22	Soph	2.8	Jones
23	Frosh	2.7	Davis

⌈ GPA >= 3.0

View Node

C ₁	1 <= Sno <= 23	T19
C ₂	Frosh <= Class <= Grad	T19
C ₃	3.1 <= GPA <= 4.0	T19
C ₄	Davis <= Advisor <= Smith	T19
C ₅	Class = Grad -> GPA >= 3.6	T19
C ₆	Advisor = Jones -> GPA >= 3.2	T19
C ₇	GPA = 4.0 -> Sno = 1 or 11	T19
Tuple Count: 23		T19
distributions		T19
View Cache: nil		

State Y

Figure 4: The base relation update log contains a set of insertions that are filtered and applied to the view node as the database evolves from state *x* to state *y*.

By propagating the updates through the View Network, the system can identify which metadata is affected by an update. When an update applies to a node, the metadata can be tested to determine which metadata is no longer valid, and the system can modify or invalidate the incorrect metadata. Before the updates can be filtered and propagated through the View Network, the update logs and the view nodes to be maintained must be retrieved from disk. The cost of these disk accesses dominates the cost of the maintenance process.

Allocating and Processing Update Logs

Our objective is to minimize the average maintenance cost per query. This is achieved by creating additional update logs in the View Network. The additional logs form a chain in which each log only considers the updates in the log above. Once an update is filtered, it is never considered by an update log lower in the network. Although this strategy may create additional work for an individual query, the maintenance cost is amortized thus reducing the average cost per query.

Update logs are allocated throughout the View Network at compile-time based on the estimated selectivity of the view nodes. As described in [Pit95], when the selectivity at a node, with respect to the log above, is less than 50 percent (i.e., less than half of the updates apply to the node), an update log is created at the view node. These logs store relevant updates in order to reduce the number of log pages that must be read to maintain the rest of the network during future maintenance cycles.

The cost of maintaining the view nodes is dominated by the cost of reading and writing the update logs. Log maintenance is also a significant factor in the cost of maintaining view caches. Although the view nodes and view caches are separate data structures, they both depend on the same update logs. Therefore, maintenance costs can be minimized by efficiently utilizing the update logs (i.e., maintaining the view nodes and view caches together when an update log is retrieved).

Running Example

Before we present the cost formulas for maintaining view nodes and view caches, the following example provides some numbers to help interpret the formulas. This example considers the chain of nodes V_1, V_2, V_3 , and V_4 shown in Figure 5. We will continue to denote the nodes in the View Network as V_i unless it is necessary to distinguish between the view nodes (VN_i) and view caches (VC_i). As shown in Table 1, we assume that the base relation contains one million tuples. We estimate the tuple count for each of the four views by assuming that the selectivity factor at each level of the network is 50 percent (i.e., exactly half of the tuples at a given node apply to the nodes directly below).

	Tuples	Log Tuples	Log Pages	View Cache Pages (n_v)	View Cache Partitions (p_v)
Base Relation	1,000,000	10,000	1000		
Node V_1	500,000	5000	500		
Node V_2	250,000	2500	250	1000	50,000
Node V_3	125,000	1250	125		
Node V_4	62,500			250	6250

Table 1: Parameters for the running example.

In order to estimate the number of log pages that will be read and written, we assume that one percent of the base relation has changed. Therefore, the base relation update log contains 10,000 updates. Assuming 10 updates per page, 1000 pages will be read from the base relation update log. The number of updates for each view were estimated assuming a 50 percent selectivity factor between each view. In order to estimate the number of pages in the view caches (n_v), we assume that each pointer requires 5 bytes and that each TID in the view cache requires 3 pointers. Therefore, each entry in the view cache requires 15 bytes. Assuming a page size of 4K, each view cache page contains 250 entries. In order to estimate the number of partitions (p_v) in the view cache, we assume that VC_2 references half of the base relation pages (50,000) and we assume that each tuple in VC_4 is written on its own page (6250 partitions). Finally, we assume that each view node requires one page ($N_r = 1$).

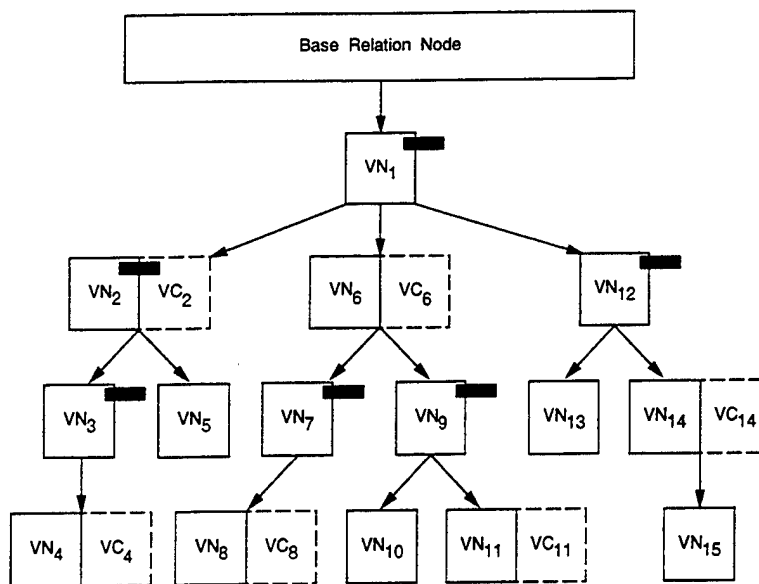


Figure 5: Example View Network with view nodes, view caches, and update logs.

Maintaining View Nodes

In order to maintain a view node, all of the logs along the update path must be retrieved and processed. Processing begins at the top of the network with the base relation update log. The updates in the base relation update log are filtered and the relevant updates are written to the next log. This process continues until the last update log in the update path has been processed. The updates in the last log that have not been applied to the metadata at the view node being maintained are applied to the metadata. The following formula estimates the cost of maintaining a view node.

$$L_r + L_w + 2N_r$$

L_r represents the number of log pages read, L_w represents the number of log pages written, and N_r represents the number of node pages that are read. In the worst case, each node page will be modified and written back to disk. The cost formula allows for this by doubling the number of node pages that are read. In our running example, the cost of maintaining VN_4 is $L_r + L_w + 2N_r = 1000 + 875 + 2(1) = 1877$.

From these numbers, it is clear to see that the cost of updating a view node is dominated by the cost of reading and writing the update logs (1875 pages to process the update logs versus 2 pages to read and write the view node). Therefore, whenever the update logs are processed, all of the view nodes along the update path should be maintained. The view nodes along the update path for V_4 (VN_1, VN_2 , and VN_3) can be maintained by reading (and possibly writing) 3 more pages.

Maintaining View Caches

View caches contain pointers to the base relation tuples that participate in a view. When an update is made that affects the view, the view cache must be maintained. (i.e., the insertions that apply to the view must be added to the view cache and the deletions must be removed). Maintaining a view cache is typically more efficient if the view is materialized, especially if the view joins two or more base relations.

With the deferred update strategy, view caches are updated when the view is materialized during query execution. Three items are retrieved when a view cache is maintained: (1) the view cache, (2) the update logs along the update path that support the view cache, and (3) the pages of the base relation that are indexed by the view cache. A view cache is *optimal* if it can be materialized without reading the same base relation page more than the minimum number of times required. This is achieved by partitioning the TIDs of the view cache into equivalence classes in which all of the TIDs access the same base relation page.

[Rou91] provides a detailed analysis of incremental update algorithms for view caches. The cost of these algorithms depends on four parameters: (n_v) the number of disk pages in the view cache V being maintained (L_r) the number of log pages read, (L_w) the number of log pages written, and (p_v) the number of partitions in V . The following formula estimates the cost of materializing a unary view cache (selection or projection).

$$L_r + L_w + 2n_v + p_v$$

Each view cache page may be modified and written back to disk. The cost formula allows for this by doubling the number of view cache pages that are read. Since the view cache is updated during query execution, the cost of materializing the view ($n_v + p_v$) is absorbed by the cost of executing the query. Therefore, the actual maintenance cost consists of reading and writing the update logs plus the cost of writing the modified pages of the view cache ($L_r + L_w + n_v$). In our running example, the cost of maintaining VC_4 is $L_r + L_w + 2n_v + p_v = 1000 + 875 + 2(250) + 6250 = 8625$.

These numbers indicate that the cost of updating a view cache is dominated by the cost of retrieving the tuples from the base relation (p_v). However, in this example we have considered the worst case (i.e., that each tuple is stored on its own page). If we assume that each partition contains two tuples, the cost of retrieving the tuples is cut in half ($p_v = 3125$), and the cost of reading and writing the update logs (1875) begins to approach the cost of materializing the view. As the size of the views decreases, the cost of reading and writing the update logs becomes a greater factor.

3.1 Maintaining View Nodes and View Caches

Having reviewed the methods for maintaining view nodes and view caches independently, the remainder of this paper considers how to efficiently maintain them together. Since the cost of maintaining view nodes and view caches greatly depends on the cost of reading and writing the update logs, it is desirable to maintain the metadata at the view nodes and view caches whenever an update log is retrieved. For example, in Figure 5, if a query requires the metadata at VN_{14} to be maintained, then the base relation update log, plus the update logs at VN_1 and VN_{12} , must be retrieved. When these update logs are processed, it would be most efficient to maintain $VN_1, VN_{12}, VN_{13}, VC_{14}$, and VN_{15} . However, since the query only requires the metadata at VN_{14} , the additional view nodes and view caches can be maintained if time permits,

The following analysis assumes that a query has been received at run-time and the query's execution package has been retrieved. Within this context, this section will consider three maintenance

scenarios. In all three scenarios, the query execution plan under consideration uses a view cache. In the first scenario, the query execution plan does not depend on any rules. Therefore, the view cache will be maintained during query execution and the view nodes along the update path may be maintained when the update logs are processed. In the second and third scenarios, the query execution plan does depend on rules which must be updated before the query execution plan can be executed. In the second scenario, the update paths for the view node and the view cache do not overlap. Therefore, each update path can be maintained separately. In the third scenario, the update paths do overlap which creates an interesting conflict.

Scenario 1: Non-Semantic Plans

In the first scenario, the query optimizer has selected a non-semantic plan that uses a view cache. Since the plan does not depend on any rules, none of the view nodes need to be maintained. Therefore, the view cache is maintained during query execution as described above. When the update logs are retrieved and processed to maintain the view cache, there is an opportunity to maintain the view nodes along the view cache update path. The cost of maintaining the view cache is: $L_r + L_w + 2n_v + p_v$.

Since the update logs have already been retrieved, the cost to maintain the view nodes is the cost of reading and writing the view node pages ($2N_r$ in the worst case). The total cost to update the semantic and structural metadata along the view cache update path is: $L_r + L_w + 2n_v + p_v + 2N_r$. Since the cost of maintaining the view nodes is dominated by the cost of reading and writing the update logs, the view nodes along the view cache update path should always be maintained whenever the view cache is maintained.

Figure 6 illustrates a View Network and two query execution plans. The non-semantic plan uses the view cache VC_2 shown in bold. Since this plan does not depend on any rules, VC_2 will be maintained during query execution. When the update logs are processed for VC_2 , the view nodes along the update path (VN_1 and VN_2) can be maintained. Since the logs have already been

```

Select   Snum
From     Students, Employees
Where    Advisor = Jones AND Class = Grad AND Snum = Enum

```

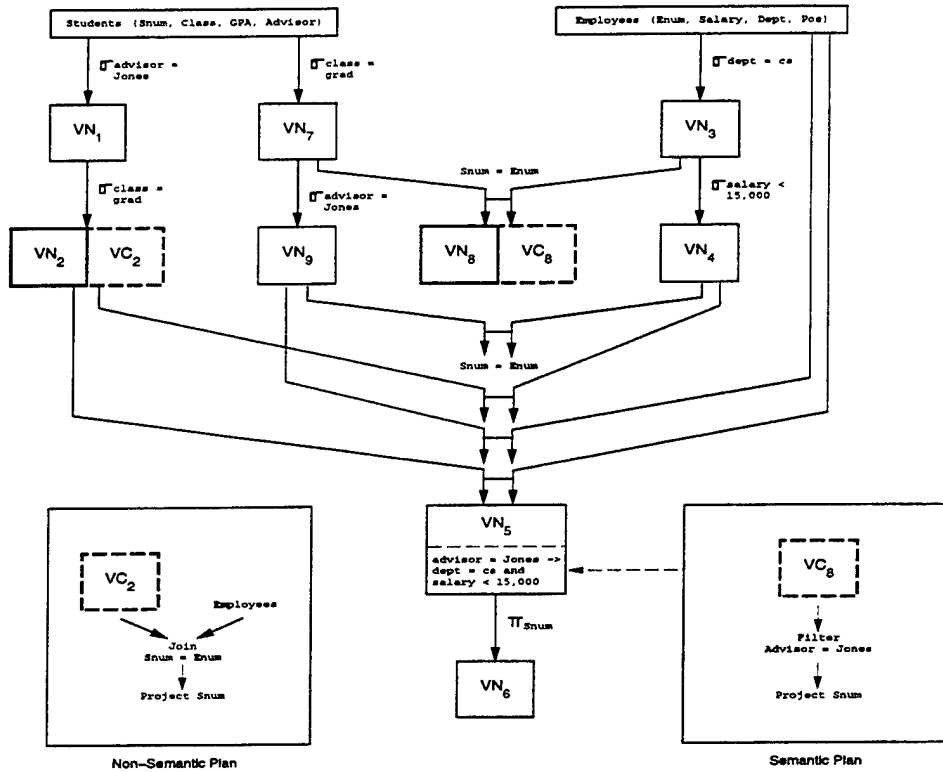


Figure 6: Example View Network and Query Execution Plans.

retrieved and processed, the cost of maintaining the metadata at VN_1 and VN_2 is the cost of reading and writing VN_1 and VN_2 .

There are two advantages to maintaining the view nodes when the view caches are maintained. First, although there is no benefit to the current query in this scenario, maintaining the view nodes will reduce the maintenance cost for future queries. Second, rules can be recomputed if the view is materialized when the view node is maintained. For example, consider a rule that maintains the minimum GPA for a view of students and assume that the student with the lowest GPA is deleted. If the view is materialized when the rule is maintained, the new minimum GPA can be recomputed.

```

INPUT: LOGS,           /* a list of update logs to be processed */
          NODES,         /* a list of node groups to be maintained */
          VIEW-CACHES;  /* a list of view caches to be maintained */

```

```

view-cache-update(LOGS, NODES, VIEW-CACHES)

```

```

  For each log in LOGS
    read the log
    filter the log
    If there are updates to be added to the log
      write the log
    maintain the log's node group
    If the log serves a view cache in VIEW-CACHES
      maintain the view cache

```

Algorithm 1: View Cache Update.

Scenario 2: Distinct Update Paths

The second scenario involves a semantic plan that depends on one or more rules stored at a view node. Therefore, the view node must be maintained before the semantic plan can be executed. In this scenario, the update paths for the view node and the view cache are distinct. Consequently, the update path for the view node can be processed before the query is executed and the update path for the view cache can be processed during query execution if the semantic plan is valid.

In Figure 6, the semantic plan uses the view cache VC_8 which contains an index for the graduate students who are employed by the computer science department. The semantic plan is only valid if the rule stored at VN_5 is valid. There are two possible update paths for VN_5 , one of which will be chosen at compile-time. If the update path for VN_5 goes through VN_1 and VN_2 , then the update path for VN_5 does not overlap with the update path for VC_8 . This scenario involves two steps: (1) maintain the view node, and (2) maintain the view cache provided the semantic plan is valid.

Step 1 (Maintain the View Node): In this example, the metadata at VN_5 will be maintained before the query is executed. View nodes VN_1 , VN_2 , and the view cache VC_2 can be maintained when the update logs are retrieved for VN_5 . The total cost to maintain the view nodes and view

```

INPUT: LOGS,           /* a list of update logs to be processed */
       NODES,         /* a list of node groups along the update path */
       VIEW-CACHES;  /* a list of view caches along the update path */

```

```

view-node-update(LOGS, NODES, VIEW-CACHES)

```

```

  For each log in LOGS
    read the log
    filter the log
    If there are updates to be added to the log
      write the log
    If time permits
      maintain the log's node group
    If the log serves a view cache in VIEW-CACHES AND time permits
      maintain the view cache

```

Algorithm 2: View Node Update.

cache is: $L_r + L_w + 2N_r + (2n_v + p_v)$, where $(2n_v + p_v)$ is the additional cost for the view caches.

Step 2 (Maintain the View Cache): The view cache VC_8 will be maintained during query execution. The view nodes VN_3 and VN_7 can be maintained when the update logs are retrieved to maintain VC_8 . The total cost to maintain the view nodes and view caches is: $L_r + L_w + (2N_r) + 2n_v + p_v$, where $(2N_r)$ is the additional cost to maintain the view nodes.

Scenario 3: Overlapping Update Paths

The third scenario is identical to the second scenario except that the update paths overlap (i.e., the update paths for the view node and view cache contain a common subpath). When the common subpath is processed, it is most efficient to retrieve the update logs once and process the view nodes and view caches together. However, since the plan being considered is a semantic plan, the view cache being maintained may not be used if the semantic plan is invalid.

In Figure 6, if the update path for VN_5 goes through VN_7 and VN_9 , then the update path for VN_5 overlaps with the update path for VC_8 . In this case, the update logs for VN_7 must be retrieved before the query is executed. Although it is most efficient to maintain VC_8 when the update log

```

INPUT: LOGS,           /* a list of update logs to be processed */
      NODES,           /* a list of node groups along the update path */
      VIEW-CACHES;    /* a list of view caches along the update path */

```

```

view-node-view-cache-update(LOGS, NODES, VIEW-CACHES)

```

```

  For each log in LOGS
    read the log
    filter the log
    If there are updates to be added to the log
      write the log
    maintain the log's node group
    If the log serves VC*
      maintain the view cache
    Else
      If the log serves a view cache in VIEW-CACHES AND time permits
        maintain the view cache

```

Algorithm 3: Combined view node and view cache update.

for VN_7 is processed, the view cache VC_8 should be maintained during query execution. If the rule at VN_5 is invalid, VC_8 will not be used to answer the query.

There are two solutions for this scenario, an optimistic approach and a pessimistic approach. The optimistic approach assumes that the semantic plan will be valid (i.e., the rule holds for the current state of the database). Therefore, VC_8 will be materialized and maintained when the update logs are processed for VN_7 . If necessary, the materialized view will then be stored on disk while the metadata at VN_5 is maintained. If the rule at VN_5 holds, then the semantic plan will be executed using the view cache. Otherwise, the non-semantic plan will be executed. The optimistic approach maintains the semantic and structural metadata before the query execution plan is selected. If the semantic plan is invalid, then the optimistic approach pays the price of maintaining a view cache that cannot be used for the given query. Although this will reduce the maintenance cost for future queries that use VC_8 , this can be an expensive price to pay for the current query.

The pessimistic approach assumes that the semantic plan will not be valid. Therefore, the metadata at VN_5 is maintained before a query execution plan is selected. If the semantic plan is valid, then the view cache at VC_8 will be maintained. This may require the update log for VN_7 to be retrieved a second time, but since the update log has already been processed, only the updates that are relevant to VN_7 will be retrieved. The cost of retrieving the update log is only significant if the size of the log approaches the size of the view cache.

4 Conclusion

Previous researchers have developed methods for discovering dynamic rules and for providing useful view caches. This paper has considered the problem of maintaining these two types of metadata together in the Metadata View Graph Framework which can be implemented as a database application or as an extension to the DBMS kernel. Because these two types of metadata are maintained at different times, a conflict arises when the update paths for the semantic and structural metadata overlap. Since both types of metadata depend on the same update logs, it is most efficient to maintain the metadata together when the update logs are retrieved. This paper presented various cost formulas and analyzed three scenarios. The first scenario showed that the view nodes along a view cache update path should always be maintained whenever the view cache is maintained. The second scenario showed that it is only cost effective to maintain a view cache along a view node update path when the size of the logs approaches the size of the view cache.

The third scenario presents a conflict because the update paths overlap. We have proposed two approaches to resolve this conflict. The optimistic approach is the most efficient, but the current query may incur a heavy penalty when the semantic plan is invalid. The pessimistic approach may be less efficient when the semantic plan is valid, but the additional cost is minimized by the previous maintenance cycle. The best approach can be selected at run-time based on cost estimates and other factors.

References

- [BLT86] J. Blakeley, P. Larson, and F. Tompa. Efficiently updating materialized views. In C. Zaniolo, editor, *Proc. of the 1986 ACM SIGMOD*, pages 61–71, Washington, D.C., May 1986.
- [CGM90] S. Chakravarthy, J. Grant, and J. Minker. Logic based approach to semantic query optimization. *ACM Trans. on Database Systems*, 15(2):162–207, June 1990.
- [HK93] C. Hsu and C.A. Knoblock. Reformulating query plans for multidatabase systems. In *Proc. of the Second Int. Conf. on Info. and Know. Management*, Washington, DC, 1993.
- [HK94] Chun-Nan Hsu and Craig Knoblock. Rule induction for semantic query optimization. *Machine Learning*, pages 1–10, 1994.
- [Pit95] J. Pittges. *Metadata View Graphs: A Framework for Query Optimization and Metadata Management*. PhD thesis, Georgia Institute of Technology, November 1995.
- [Rou82] N. Roussopoulos. The logical access path scheme of a database. *IEEE Trans. on Software Eng.*, SE-8(6):563–573, November 1982.
- [Rou91] N. Roussopoulos. An incremental access method for viewcache: Concept, algorithms, and cost analysis. *ACM Trans. on Database Systems*, 16(3):535–563, 1991.
- [S⁺92] M. Siegel et al. A method for automatic rule derivation to support semantic query optimization. *ACM Trans. on Database Systems*, 17(4):563–600, December 1992.
- [S⁺93] S. Shekhar et al. Learning transformation rules for semantic query optimization: A data-driven approach. *IEEE Trans. on Knowledge and Data Eng.*, 5(6):950–964, 1993.
- [SO89] S.T. Shenoy and Z.M. Ozsoyoglu. Design and implementation of a semantic query optimizer. *IEEE Trans. on Knowledge and Data Eng.*, 1(3):344–361, 1989.
- [YS89] C. Yu and W. Sun. Automatic knowledge acquisition and maintenance for semantic query optimization. *IEEE Trans. on Knowledge and Data Eng.*, 1(3):362–375, September 1989.

PART IV

**DEVELOPMENT OF AN
INTELLIGENT INTERACTIVE
TOOL FOR ENGINEERING
DESIGN**

PART IV: DEVELOPMENT OF AN INTELLIGENT INTERACTIVE TOOL FOR ENGINEERING DESIGN

In order to support our work on HIPED, we engaged in additional development and experimentation with the Interactive Kritik system. The Interactive Kritik system combines autonomous engineering design capabilities with an interactive explanatory interface.

Within the HIPED project, Interactive Kritik served as the primary motivating example of an intelligent design system which could require information from heterogeneous database sources. The papers in Part I focus heavily on how such requests are made and processed. The papers in this section provide more elaborate background on the Interactive Kritik system, per se.

In our work on Interactive Kritik, we have considered two primary applications of an interactive intelligent design system:

- 1) Expert designers might use such a system to make suggestions about a design. Such an application requires an explanatory interface because users are unlikely to trust results which are not clearly explained. Paper [4.1] describes Interactive Kritik from the perspective of this goal.
- 2) Design students might use such a system to learn about the design process. Such an application obviously requires an explanatory interface to enable students to see how a design was developed. Paper [4.2] describes Interactive Kritik from the perspective of this goal.

The relevance of HIPED to the former goal is extremely strong: true expert design often requires the use of large volumes of knowledge, e.g. libraries of components; such knowledge is generally only available in large sets of distributed, heterogeneous sources. The latter goal could potentially be accomplished without access to large volumes of information; students could be provided with very small, local libraries which contained enough information to solve the problems provided. This would, however, make this design instruction a very restricted, artificial process which largely negates the advantages of using an interactive tutoring system in the first place. In contrast, providing students with a realistic design environment in which a nearly unlimited variety of components, etc. were available could allow more productive learning. Again, HIPED provides a mechanism by which such information can be accessed.

Our work on Interactive Kritik has led to a number of additional important results. One of these results involves the value of encapsulation of reasoning episodes into explicit knowledge structures. We label these structures "meta-cases", i.e. stored instances not merely of past design results but rather of entire past design processes. Paper [4.3] examines meta-cases in

more detail. From the perspective of HIPED, meta-cases can be viewed as a sophisticated form of caching; an entire process, potentially involving one or more requests to external heterogeneous data sources, is stored away as a single, complex meta-case.

Another important issue in design computing is the role of function in design. Throughout our work, we have claimed that functional representations, i.e. those which explicitly specify the role that individual elements have in a complex system, provide a wide variety of capabilities. Paper [4.4] addresses a particular application of functional representations: support for explanation of reasoning. The argument is that by explicitly representing what each element of a knowledge system does, explanations can show not only what the system does but also why. For the purposes of HIPED, these functional descriptions play a potentially key role in integration of knowledge systems. An automated reasoner can only integrate multiple, heterogeneous knowledge systems to the extent that it understands what these knowledge systems do.

PUBLICATIONS (PART4):

- [4.1] A. K. Goel, A. Gomez, N. Grue, J. W. Murdock, M. Recker, and T. Govindaraj. Explanatory Interface in Interactive Design Environments. In *Proceedings of the Fourth International Conference on AI in Design*, Palo Alto, June 1996.
- [4.2] A. K. Goel, A. Gomez, N. Grue, J. W. Murdock, M. Recker, and T. Govindaraj. Towards Design Learning Environments - Explaining How Devices Work. In *Proceedings of the International Conference on Intelligent Tutoring Systems*, Montreal, Canada, June 1996.
- [4.3] A. K. Goel and J. W. Murdock. Meta-Cases: Explaining Case-Based Reasoning. In *Proceedings of the Third European Workshop on Case-Based Reasoning*, Lausanne, Switzerland, 1996.
- [4.4] A. K. Goel, A. Gomez, N. Grue, J. W. Murdock, and M. Recker, Functional Explanations in Design. In *Program of the 1997 International Joint Conference on Artificial Intelligence Workshop on Functional Reasoning*, 1997.

EXPLANATORY INTERFACE IN INTERACTIVE DESIGN ENVIRONMENTS

ASHOK GOEL, ANDRÉS GÓMEZ DE SILVA GARZA, NATHALIE GRUÉ,
J. WILLIAM MURDOCK AND MARGARET RECKER

*College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332, USA*

AND

T. GOVINDARAJ
*School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332, USA*

Source: Fourth International Conference on Artificial Intelligence in Design, AID '96, Stanford, California, June 24 - 27, 1996. John S. Gero and Fay Sudweeks, eds. Boston: Kluwer Academic Publishers, 1996.

Abstract.

Explanation is an important issue in building computer-based interactive design environments in which a human designer and a knowledge system may cooperatively solve a design problem. We consider the two related problems of explaining the system's reasoning and the design generated by the system. In particular, we analyze the content of explanations of design reasoning and design solutions in the domain of physical devices. We describe two complementary languages: task-method-knowledge models for explaining design reasoning, and structure-behavior-function models for explaining device designs. INTERACTIVE KRITIK is a computer program that uses these representations to visually illustrate the system's reasoning and the result of a design episode. The explanation of design reasoning in INTERACTIVE KRITIK is in the context of the evolving design solution, and, similarly, the explanation of the design solution is in the context of the design reasoning.

1. Background, Motivations and Goals

Effective communication of both the design process and the design product is critical in collaborative design. Communicating the process of design and the evidence that the product satisfies its requirements can help build confidence in the design. When members of a design team work on different parts of a design problem, this kind of communication about one part of the problem can help in constraining other parts of the problem. In addition, explanation of design reasoning and its result can enable reuse of parts of the reasoning/result in subsequent design projects. Within the course of a design project, the explanation can enable reflection, support the detection of flaws, and suggest remedies for fixing them.

This is no less true of collaboration between a human designer and a knowledge system in the context of computer-based interactive design environments. When a human designer and a knowledge system are cooperatively addressing a design problem, the system must be able to explain to the designer precisely what it is doing, how and why. In addition, the system must be able to justify why the design solution it has proposed is acceptable for the given problem. Without this the user will have little confidence in the design and may be unable to detect potential flaws in it. Building usable interactive design environments thus requires both a theory of design explanations and the creation of explanatory interfaces.

The issue then becomes how may a knowledge system explain both its reasoning and the design solutions it proposes. This issue has several related but distinct facets pertaining to the content, generation, and presentation of explanations. To illustrate, let us consider the problem of explaining the design of a gyroscope. The explanation may specify how the design works, how its structure delivers its functions, how its design satisfies its requirements. Within a knowledge system, knowledge of the gyroscope's behaviors may be represented as a causal network, or generated at run-time from a representation of its design structure. To the user, the system may present the explanation in text form, or as graphics, or in some other modality such as animation. Our research on design explanations centers on the content of explanations presented to the user, and the content and representation of design knowledge and reasoning needed for generating the explanations.

The content of explanation and justification of design solutions, such as that of a gyroscope, depends both on the design phase and the design domain. For example, the explanation of the result of preliminary design is different from that of the result of configuration design: the former pertains to the function and structure of the design while the latter refers to its geometry. Similarly, the content of a justification for the design of gyroscope is different from that of an office building or a software interface. This is because the relationships between the function and the structure of the gyroscope design are fundamentally different from the function-structure relations in the design of an office building or a software interface. Our work focuses on the preliminary (conceptual, qualitative)

design of physical devices such as electrical circuits, heat exchangers, and angular momentum controllers. The input to this task is a specification of the desired functions, and the output is a specification of a structure that can deliver the desired functions.

We are developing an interactive design and learning environment called INTERACTIVE KRITIK. When complete, INTERACTIVE KRITIK is intended to serve as an interactive constructive design environment. At present, when asked by a human user INTERACTIVE KRITIK can invoke a knowledge-based design system called KRITIK3 to address specific kinds of design problems. KRITIK3 evolves from KRITIK, which has been extensively described elsewhere (e.g., [Goel 1991, 1992; Goel and Chandrasekaran 1989, 1992].) INTERACTIVE KRITIK provides an explanatory interface to KRITIK3. In particular, it provides visual explanations and justifications of both KRITIK3's reasoning and the solutions it proposes. In addition, it enables the user to explore the system's design knowledge and also the design of the device generated by the system. A key feature of INTERACTIVE KRITIK is that explanation of the design reasoning is presented in the context of the evolving design solution, and, similarly, explanation of the design solution is presented in the context of the reasoning that led to it.

2. INTERACTIVE KRITIK

INTERACTIVE KRITIK's architecture consists of two agents: a design agent in the form of KRITIK3¹ and an interface agent². Figure 1 illustrates INTERACTIVE KRITIK's architecture. The solid lines in the figure represent data flow while dotted lines represent control flow.

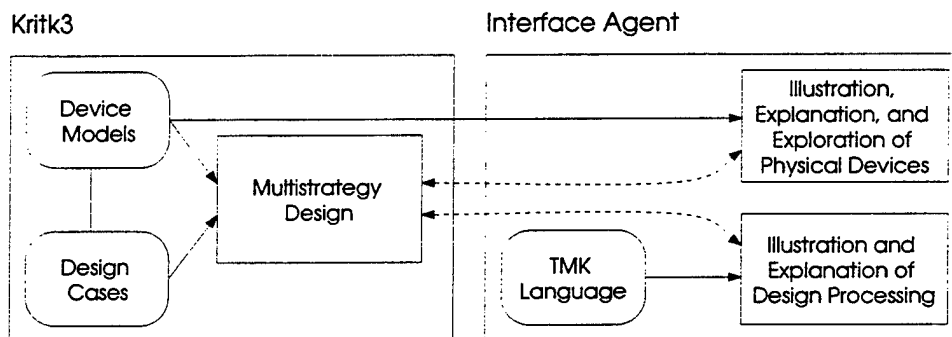


Figure 1. INTERACTIVE KRITIK's Architecture

¹KRITIK3 runs under Common Lisp using CLOS.

²The interface is built using the Garnet tool [Myers and Zanden 1992].

2.1. STRUCTURE-BEHAVIOR-FUNCTION MODELS IN INTERACTIVE KRITIK

We use structure-behavior-function models (SBF models) [Chandrasekaran et al 1993; Goel 1991, 1992] for explaining and justifying designs of physical devices. The SBF model of a device provides a functional and causal explanation of how the device works, how its structure delivers its functions. This explanation makes explicit the functional and causal roles played by each structural element in the device design. Since KRITIK3 addresses the function-to-structure design task, and because the SBF model of a design created by the system explains how the proposed structure delivers the desired functions, the SBF model provides a justification for the design.

The SBF model of a device explicitly represents (i) the function(s) of the device, (ii) the structure of the device, and (iii) the internal causal behaviors of the device. The internal causal behaviors specify how the functions of the structural elements of the device are composed into the device functions. As a simple (almost trivial) example, let us consider the SBF model of an electrical circuit that produces light of intensity 9 lumens.

Structure: The structure of a device in the SBF language is expressed in terms of its constituent components and substances and the interactions between them. Components and substances can interact both *structurally* and *behaviorally*. For example, electricity can flow from battery to bulb only if they are structurally connected, and only if supported by the function allow electricity of switch that connects the battery and the bulb.

Function: The function of a device in the SBF language is represented as a schema that specifies the input behavioral state of the device, the behavioral state it produces as output, and a pointer to the internal causal behavior of the design that achieves this transformation. Both the input state and the output state are represented as *substance schemas*. The input state specifies that the substance electricity has the property *voltage* and the corresponding parameter, *10 volts*. The output state specifies the property *intensity* and the corresponding parameter, *9 lumens*, of a different substance, light. Finally, the slot by-behavior points to the causal behavior that achieves the function of producing light. Figure 2 illustrates INTERACTIVE KRITIK's visual representation of the function of the electrical circuit.

Behavior: The SBF model of a device also specifies the internal causal behaviors that compose the functions of device substructures into the functions of the device as a whole. In the SBF language, the internal causal behaviors of a device are represented as sequences of *transitions* between *behavioral states*. The annotations on the state transitions express the *causal*, *structural*, and *functional contexts* in which the state transitions occur and the state variables get transformed. The causal context specifies *causal relations* between the variables in preceding and succeeding states. The structural context specifies different *structural relations* among the components, the substances, and the different spatial locations of the

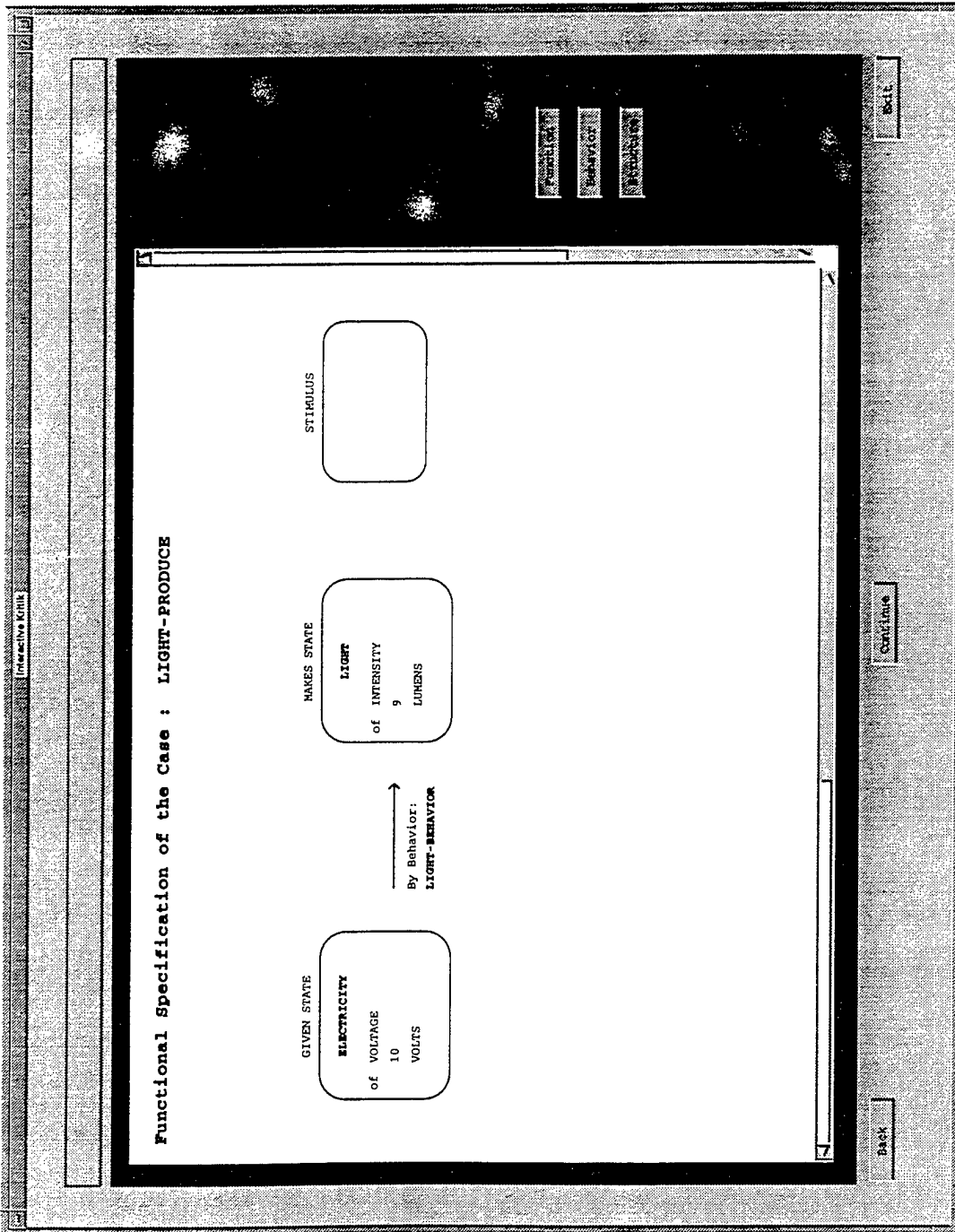


Figure 2. The Function of an Electrical Circuit

device. The functional context indicates which functions of which components in the device are responsible for the transition. The behaviors are organized along the flow of specific substances through the device.

Figure 3 illustrates INTERACTIVE KRITIK's visual representation of *Light-Behavior*, the causal behavior that explains how light is generated. The state transition in this behavior has three annotations Using Function, Under Condition Transition, and Parametric Equation as indicated in the side bar on the top right of the figure. In the screen shot depicted, the description of one of these annotations, Using Function, is displayed in the pop-up dialog box in the right center of the figure. This description explains that the transition occurs due to the function *Bulb-Function-Light* component *Bulb*. Although not shown in Figure 3, the description for *Under Condition Transition* specifies that the transition is contingent on the flow of electricity through the bulb as detailed in a separate behavior labeled *Electricity-Behavior*. Similarly, the description for *Parametric Equation* specifies the specific equation relating the state variables.

The use of SBF models for explanation of designs is consistent with Simon's [1981] notion of functional explanations of artifacts. He has argued that explanations of artifacts pertain to, and are referenced by, the purpose of the artifact. This leads us to hypothesize that *SBF models capture the content of explanation of a device design at the "right" level of abstraction for comprehension by human designers.*

2.2. TASK-METHOD-KNOWLEDGE MODELS IN INTERACTIVE KRITIK

We use task-method-knowledge models (TMK models) [Chandrasekaran 1989, 1990; Goel and Chandrasekaran 1992] for explaining and justifying reasoning about a design problem. The TMK model provides a functional and strategic explanation of design reasoning in terms of the task, the methods used to accomplish the task, the subtasks spawned by the methods, and the knowledge used by the methods. Since subtasks are spawned by the methods available to the reasoner, the TMK model also provides a justification of specific tasks addressed by the reasoner in terms of the methods that spawn the tasks. Similarly, since methods serve tasks and are afforded by the available knowledge, the TMK model provides a justification of the use of specific methods by the reasoner in terms of the tasks being addressed and the knowledge that affords the methods.

The TMK model of design reasoning has three main elements. The first element, the task, is characterized by the types of information it takes as input and gives as output. KRITIK3 addresses the functions-to-structure design task in the domain of physical devices. This task takes as input a specification of the functions of the desired design. It has the goal of giving as output the specification of a structure that delivers the desired functions. The second element in the TMK model is the method. A method is characterized by (i) the type of knowledge it

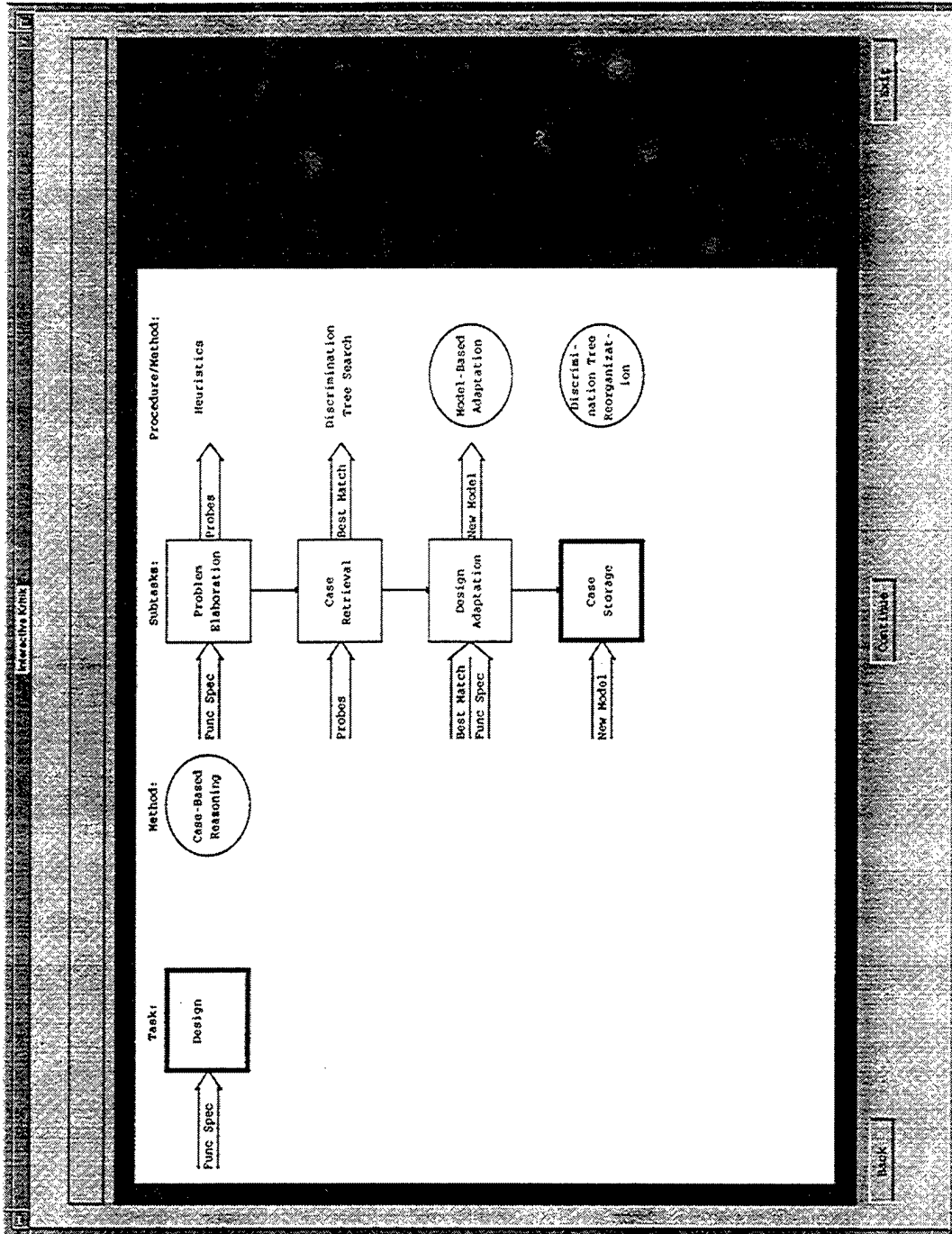


Figure 4. The Overall Design Task

uses, (ii) the subtasks (if any) it sets up, and (iii) the control it exercises over the processing of subtasks. KRITIK3 uses the method of case-based reasoning for addressing the function-to-structure design task. Figure 4 illustrates INTERACTIVE KRITIK's visual representation of this method. The figure shows that the method sets up the subtasks of problem elaboration, case retrieval, design adaptation, and case storage. It also shows the order in which these tasks are executed. In addition, it shows the input-output specification of these tasks. For example, the task of design adaptation takes as input the specification of the desired functions and the best matching case retrieved from the case memory. It gives as output an SBF model for a candidate design as indicated in Figure 4.

The third element in the TMK model is knowledge. A specific type of domain knowledge is characterized by its content, by its form of representation, and by its organization. Consider the example of diagnostic knowledge. In some domains, heuristic associations that directly map signs and symptoms into fault categories may be available. In a knowledge system, this associative knowledge might be represented in the form of production rules and organized as an unordered list. KRITIK3 contains two kinds of domain knowledge: past design cases and case-specific SBF models. We already have briefly described the representation and organization of the SBF models. Design cases are indexed by the functions delivered by the stored designs, and organized as leaf nodes of a discrimination tree.

The design of the KRITIK family of systems embodies a TMK model of of function-to-structure design of common physical devices [Goel and Chandrasekaran 1992]. We derived this model by analysis of the above task domain using the following methodology [Chandrasekaran 1989, 1990]:

Task Identification: First, the task is specified in terms of the generic types of information it takes as input and the generic types of information desired as its output.

Knowledge Identification: Next, the domain is analyzed in terms of the kinds of knowledge available in it.

Method Identification: Then, the different methods afforded by the different kinds of available knowledge are identified. This step also involves the identification of the subtasks that each method may set up.

Method Selection: Next, since more than one method may be feasible, the criteria for selecting a specific method is specified. These criteria may include factors such as properties of the desired solution and computational properties of the methods.

Recursive Task-Domain Analysis: Finally, the above steps are repeated for each of the subtasks that the selected method sets up.

This recursive decomposition of the given task continues up to an "elementary" level at which the domain affords knowledge that can "directly" map the input to the (sub)task into its desired output. At this level, no method is needed; instead, a procedure directly applies the relevant knowledge to solve the task. The recursive

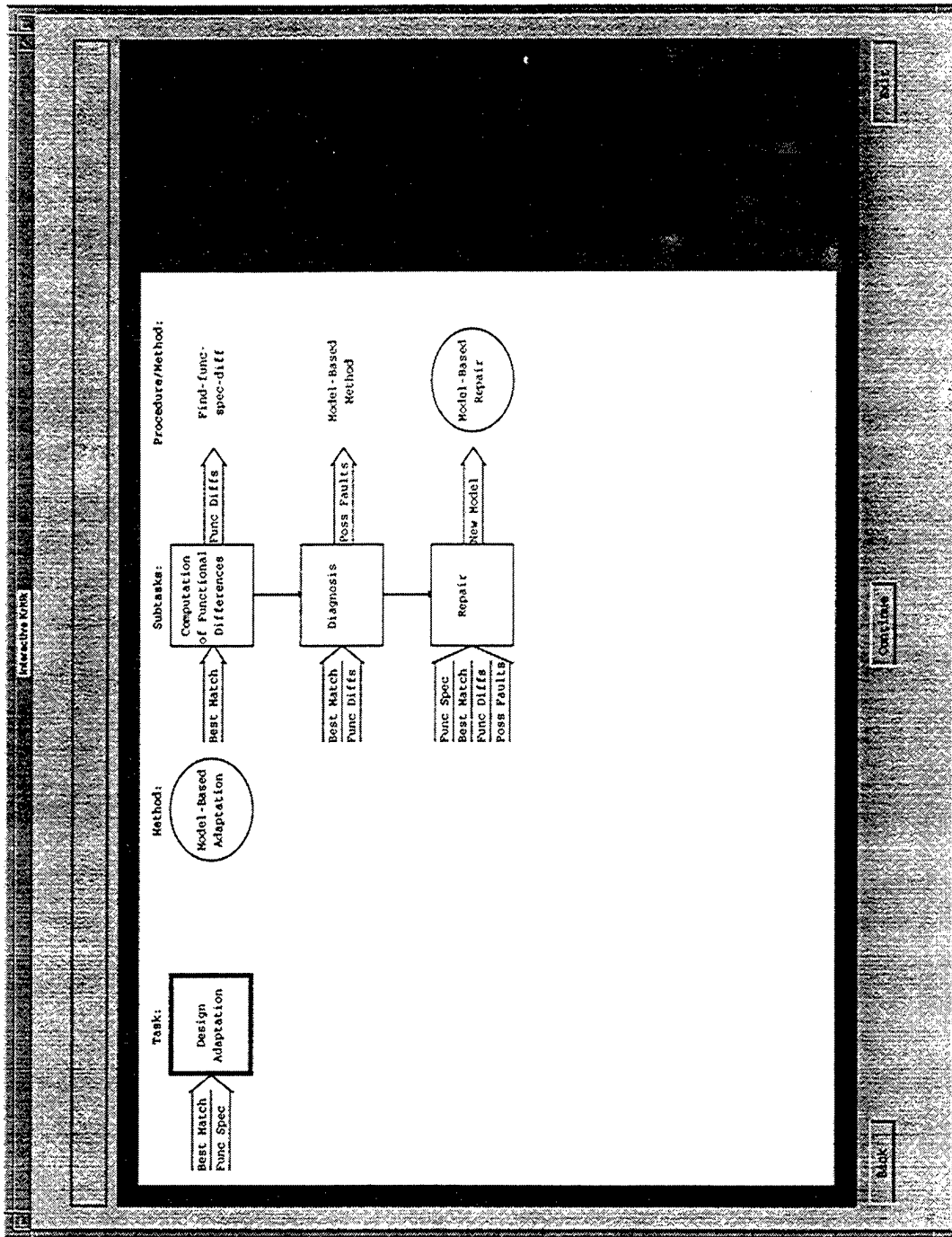


Figure 5. The Design Adaptation Task

task decomposition results in a task-method-subtask tree. For example, design adaptation is a subtask of the design task set up by the case-based method as illustrated in Figure 4. KRITIK3 uses a model-based method for addressing the task of adaptation as Figure 5 illustrates. The model-based method sets up its own subtasks of the design adaptation task. The first of these subtasks is the computation of differences between the desired function and the function delivered by the design retrieved from the case memory. KRITIK3 uses a simple pattern matching procedure for this task.

The TMK language for describing a knowledge system's reasoning is consistent with Marr's [1977] task-level and Newell's [1982] knowledge-level analyses of intelligent agents. Marr proposed that the reasoning of an intelligent agent can be analyzed at three levels. At the highest level is a specification of the tasks addressed and the mechanisms used by the agent. At the next level are the specific algorithms and data structures that the mechanism uses. At the lowest level is the architecture (or language) of implementation. Similarly, Newell proposed several levels of analysis of intelligent agents. The highest level in his scheme pertains to the agent's goals and the knowledge that enables the accomplishment of the goals. The next level concerns the symbolic structures that implement the mechanisms of the higher level. The next lower level specifies the physical devices that implement the symbolic structures, and so on. Marr suggested that the highest level in his scheme, the task-level, constituted the computational theory of the agent. Similarly, Newell suggested that the highest level in his scheme, the knowledge level, constituted the computational theory of the agent. This leads us to the hypothesis that *TMK models capture the content of explanation of design reasoning at the "right" level of abstraction for communication with human designers.*

3. The Explanatory Interface in INTERACTIVE KRITIK

The explanatory interface in INTERACTIVE KRITIK not only explains and justifies design reasoning and device designs, but also enables the user to explore the device designs and to reflect on the reasoning.

3.1. DESIGN EXPLANATION IN INTERACTIVE KRITIK

The interface agent in INTERACTIVE KRITIK has access to all the knowledge of KRITIK3 including its design cases and device models. It uses KRITIK3's SBF models of physical devices to graphically illustrate and explain the functioning of the devices to the users. It also graphically illustrates and explains the reasoning of the system in generating a new design. Within the context of a design episode, INTERACTIVE KRITIK provides graphical representations of both the designs retrieved from the case memory and the new designs created. Thus it provides representations of intermediate designs in addition to the final designs. The dif-

ferent design versions are presented as the design reasoning unfolds, i.e., in the context of the design subtask at hand.

The working of a device is illustrated to the user on several interrelated screens. One screen represents the device function; Figure 2 is an example of INTERACTIVE KRITIK's screen illustrating the function of an electrical circuit. The means by which the function of a device is achieved is explained by the internal causal behaviors in the SBF device model. Figure 3 shows an illustration by INTERACTIVE KRITIK of the main behavior, *Light-Behavior*, of the electrical circuit that produces light. A different screen shows the secondary behavior, *Electricity-Behavior*, of this device: the behavior of the electricity in this circuit.

KRITIK3's reasoning is illustrated on multiple screens identifying the tasks that the system performs while solving a problem and the methods it uses, as indicated in Figures 4 and 5. For each (sub)task, INTERACTIVE KRITIK illustrates the *reasoning state* both before and after the accomplishment of the (sub)task. The reasoning state specifies the task context and the method context. In addition, when appropriate, INTERACTIVE KRITIK illustrates the design knowledge available to KRITIK3. For example, in explaining the task of case retrieval, it graphically illustrates the case memory.

3.2. DESIGN EXPLORATION IN INTERACTIVE KRITIK

INTERACTIVE KRITIK enables the user to browse through different facets of a device design. This includes not only the final design proposed by KRITIK3 but also the intermediate designs it may have generated, for example, the design retrieved from the case memory. Exploration of a given design through browsing is enabled by the SBF model for the design.

As we explained in Section 2.1, different parts of an SBF model are closely interrelated. For example, the specification of a function in the SBF model acts as an index to the causal behaviors that accomplish the function. Also, the specifications of the state transitions in a causal behavior act as indices into the functional specifications of the structural components of the device. In addition, the description of a device component contains a specification of its functions, and points to the causal behaviors of the device in which the component plays a functional role. This indexing scheme enables the user to browse through the SBF model of the design.

The initial view of an SBF model is a representation of the device's functional specification, as in Figure 2. From here the user can push interface buttons to move among the functional, behavioral, and structural representations of the device. Additionally, the user can click on the name of the behavior by which the function is achieved (e.g., *Light-Behavior* in Figure 2) and "jump" directly to that behavior. Figure 3 illustrates the *Light-Behavior* screen. This screen presents *Light-Behavior* and labels all other behaviors (in this case, just the *Electricity-Behavior*) which the

user can select to jump to a different behavior. When a user clicks on a particular transition a menu pops up allowing the user access to a variety of options relating to that transition, as indicated in Figure 3. This allows direct access to structural and behavioral information relating to that transition. For example, if the transition selected is dependent on another behavior, the user can jump directly to that behavior. The structure screen provides similar capabilities for looking at the components of a device and the connections between them.

3.3. DESIGN REFLECTION IN INTERACTIVE KRITIK

The explicit SBF representation of a design enables the user to inspect each element and aspect of the device design. Similarly, the explicit TMK representation of the trace of design reasoning enables the user to inspect each task, method, knowledge source, and reasoning state. This enables the user to reflect on the design reasoning. For example, the user can examine the TMK reasoning trace and detect potential flaws in it.

As we mentioned in Section 2.1, the SBF model of a device design not only explains how the device works but also justifies the design by showing how its structure delivers the desired functions. And as we mentioned in Section 2.2., the TMK model not only explains the reasoning of KRITIK3 but also justifies the tasks it sets up and the methods it uses. In addition, the user can also ask INTERACTIVE KRITIK for a justification for specific reasoning choices. As an example, consider the situation in which KRITIK3 retrieves a design case from its case memory. The TMK trace shows the user the probe KRITIK3 had prepared to retrieve a case and the case the system actually retrieved from its case memory. The user can now ask why did KRITIK3 retrieve this particular design case. Since the reasoning trace explicitly specifies the probe prepared by KRITIK3, and how the system's retrieval method probed the case memory - the branches it followed, the matches it made, and their results - the trace provides a justification for why the particular case best matches the given problem.

3.4. CRITIQUE

There is still a great deal of work to be done on INTERACTIVE KRITIK's user interface. Some issues which would need to be addressed before the system could be used as a practical tool include the improved display of the structure of a device, the building of better graphical representations, and provision of additional interaction capabilities. More importantly, INTERACTIVE KRITIK needs to be formally evaluated in a real world setting. But this kind of evaluation also requires additional work on the user interface.

4. Discussion

This research builds on earlier work on three topics at the intersection of AI and Design: design methods and process models, design knowledge and device models, and interactive design environments.

Design Methods and Process Models: A major goal of AI research on design has been to develop computational methods and process models for design. This has led to the development of several computational methods for design; examples include heuristic search [Stallman and Sussman 1977], heuristic association [McDermott 1982], and plan instantiation and expansion [Brown and Chandrasekaran 1989, Mittal, Dym and Morjaria 1986]. Recent research on case-based design (e.g., [Goel and Chandrasekaran 1992, Maher, Balachandran and Zhang 1995, Navinchandra 1991]) has led to the development of multi-strategy process models for design. KRITIK3 is a multi-strategy process model of design in two senses. First, while the high-level design process in KRITIK3 is case-based, the reasoning about individual subtasks in the case-based process is model-based. For example, KRITIK3 uses SBF device models for adapting a past design and for evaluating a candidate design. Second, design adaptation in KRITIK3 involves multiple modification methods. While all modification methods make use of SBF device models, different methods are applicable to different kinds of adaptation tasks.

A closely related research direction concerns the language for specifying the computational methods and process models for design. McDermott [1982] describes R1's method for configuration design in the language of constraints of a design problem, components available in the design domain, heuristic associations pertaining to the constraints and the components, and selection and activation of the associations. But this language is much too specific to R1's method. This method-specificness of the language becomes a major problem for describing and explaining multi-strategy process models such as KRITIK3.

Task-level [Marr 1977] (or, equivalently, knowledge-level [Newell 1982]) accounts make a clearer separation between knowledge-based reasoning and its implementation in a knowledge system. In the mid-eighties, Chandrasekaran [1988] proposed the language of Generic Tasks for analyzing and modeling knowledge-based problem solving, and showed that this language enables more perspicuous explanations [Chandrasekaran, Tanner, and Josephson 1989]. In the late eighties, Chandrasekaran [1990] related Generic Tasks with task structures: [Chandrasekaran 1989] describes a high-level task structure for design; [Goel and Chandrasekaran 1992] describe a fine-grained task structure for case-based design. In their work on the elevator design project called VT, McDermott and his colleagues [McDermott 1988, Marcus et al 1988] described a similar task-oriented language for analyzing knowledge-based design.

Our TMK models represent a generalization of task structures based on Generic Tasks. Also, our hypothesis that TMK models provide the "right" level of

abstraction for explaining knowledge-based reasoning is based in part on earlier work on explanation in the Generic Task framework. But TMK models make the specific role played by a particular type of knowledge more explicit than earlier models. Consider, for example, the functional role of an SBF model of a past design in KRITIK3. Since the SBF model is associated with the past case, it affords a method for adapting the past design. The TMK model makes this affordance explicit. Thus, while task structures are useful for explaining the control of reasoning in terms of task-method interactions, TMK models are also useful for explaining knowledge-method interactions. In particular, they enable the explanation of the organization and indexing of different kinds of knowledge, the kinds of knowledge available for addressing a task, and the methods that become feasible because of the available knowledge.

Design Knowledge and Device Models: Explanation of physical devices has been a major topic of research not only in AI and in Design but also in Cognitive Engineering. AI research on device modeling and explanation can be traced as far back as Hayes [1979] work on "naive physics" in which he described a component-substance ontology. At about the same time, de Kleer developed the method of qualitative simulation for diagnosing electrical circuits [de Kleer 1984]. This work led to the no-function-in-structure principle [de Kleer and Brown 1984] which states that the behaviors of each structural component must be represented in a manner independent of their functional contexts.

In contrast, in the early eighties, Chandrasekaran and his colleagues developed the Functional Representation (FR) scheme [Sembugamorthy and Chandrasekaran 1986, Chandrasekaran et al 1993] in which the functions are not only represented explicitly, but also used to reference the causal behaviors responsible for their accomplishment. The causal behaviors in turn reference the functions of the device substructures. Since the function of a substructure refers to the causal behaviors that result in it, this gives rise to a hierarchical organization of the device model. Also in the mid-eighties, Bylander proposed a taxonomy of primitive behaviors [Bylander 1991] based in part on Hayes' component-substance ontology. He also described a method of composing the primitive behaviors into more complex behaviors. Our SBF models evolve from Chandrasekaran's Functional Representation scheme and Bylander's ontology of behaviors. In particular, they use FR's organizational scheme in which the device functions act as indices to the causal behaviors and the causal behaviors index the functions of device substructures. The specification of the functions, behaviors and structure in SBF models, however, is based on Bylander's well-defined behavioral ontology.

In Cognitive Engineering, Rasmussen [1985] proposed a hierarchical organization for presenting device knowledge to human users. His device models also specify the structure, the behaviors, and the functions at each level in the hierarchy. Our hypothesis that SBF models provide the "right" level of abstraction for explaining the working of a device to a human user is supported by

Rasmussen's empirical work. Govindaraj [1987] has used similar hierarchical organization schemes for enabling engineering students to explore the design of complex devices containing hundreds of components. Following his device models, the causal behaviors in our SBF models too are organized along the flow of specific substances in the device.

In Design research, [Gero et al 1991] and [Umeda et al 1990] have also described FBS models (for function-behavior-structure). While the details of the representation schemes differ, in both their FBS models and in our SBF models, behavior mediates between function and structure. Indeed, a major theme of our work on the KRITIK family of systems has been that while the design task takes a functional specification as input and gives a structural specification as output, much of the design reasoning is at the intermediate behavioral level.

Interactive Design Environments: A core issue in interactive design environments is how human designers and knowledge systems may share design responsibilities. AI research on interactive design environments covers a broad range of human/system responsibility sharing. At one extreme, the system acts as a knowledge source but leaves almost all reasoning to the human designer. Traditionally, knowledge bases for design have contained knowledge of design components and materials. But recent work on design knowledge bases has focused on providing human designers with access to libraries of design cases; examples include CADET [Sycara et al 1991], CADRE [Hua and Faltings 1992], CASECAD [Maher, Balachandran and Zhang 1995], FABEL [Voss et al 1994], Archie [Pearce et al 1992], AskJef [Barber et al 1992], and ArchieTutor [Goel et al 1993]. At the other extreme of this spectrum are autonomous knowledge systems that perform almost all design reasoning by themselves. Human interaction with these systems is limited to formulating design problems, supplying the problems to the system, and receiving the solutions generated by the system; examples include R1, AIR-CYL, and the original KRITIK system.

In between these two extremes lies a large range of potential sharing of responsibility between the system and the user. An important goal of design environments in the middle of this spectrum is to enable humans to construct new designs. Fischer et al's [1992] JANUS and Steinberg [1987] VEXED are two examples of constructive design environments. The goal of the INTERACTIVE KRITIK project is also the building of a constructive design environment. We will not describe here how, when completed, INTERACTIVE KRITIK may enable a human to construct new designs (but see [Grué 1994]). Instead, we focus the rest of this discussion on the issue of explanatory interface in the current version of INTERACTIVE KRITIK since this is already operational.

Mostow [1989] has argued that when a knowledge system in an interactive design environment proposes a solution to a design problem, then the system should also provide the human designer with an explanation of the reasoning that led to the solution. His BOGART system uses derivational analogy [Carbonell

et al 1989] for generating solutions to design problems. Following the theory of derivational analogy, BOGART provides the human designer with an explanation of its reasoning in the form of a derivational record. The derivational record contains a trace of the system's reasoning in the language of design goals, operators, and heuristics for goal decomposition and operator selection.

We share the premise that in any interactive design environment, the knowledge system must be able to explain its reasoning. However, we believe that the language of goals, operators and heuristics is too low level to be accessible and comprehensible to human designers, especially novice designers. Instead, we hypothesize that the TMK language is at the "right" level of abstraction. More importantly, we believe that in addition to explaining its reasoning, the knowledge system must also be able to justify the design solution it proposes. INTERACTIVE KRITIK uses SBF models for justifying its design solutions.

Further, we believe that it is critical that the explanation of design reasoning should be grounded in the context of the evolving design solution, and, similarly, the explanation of the evolving design should be grounded in the context of the design reasoning that led to it. The advantages of situating design explanations in this way are two fold. First, situating the explanation of design reasoning in the context of the evolving design solution makes the explanation more meaningful. This is because the explanatory terms can now get their meaning from the specific parts of the design to which they refer. Second, situating the explanation of the design solution in the context of the design reasoning makes the explanation more complete because of the availability of previous versions in the evolution of the design solution.

4.1. CONCLUSIONS

Interactive design environments typically contain knowledge systems as major components. A human designer may use the interactive environment for design construction and experimentation. The knowledge systems may help automate specific and selected portions of this process, leading to human-system cooperative design. This raises the issues of usability and learnability of the knowledge systems. Human designers are unlikely to work with these systems if they cannot easily use them and also easily learn how to use them. Designers are more likely to use these systems if they can form a mental model of how the system works, how it reasons about problems, and if they can develop some confidence in the solutions generated by the system.

So the issue becomes how might a knowledge system enable the user to form a mental model of its reasoning, how might it explain its reasoning and justify its answers. Our work on INTERACTIVE KRITIK is based on three related ideas:

1. Explanations of a knowledge system need to capture the functional and strategic content of reasoning in addition to its knowledge content. Task-method-knowledge

models enable this kind of task-level and knowledge-level explanation, which facilitates effective communication between the system and the user.

2. Explanations of physical systems need to capture the functionality and causality of the systems in addition to their structure. Structure-behavior-function models enable this kind of explanation at a level of abstraction that facilitates effective communication between the system and the user.

3. Explanation of design reasoning needs to be grounded in the context of the evolving design solution, and, similarly, the explanation of the evolving design needs to be grounded in the context of the design reasoning that led to it.

INTERACTIVE KRITIK demonstrates the computational feasibility of these ideas.

ACKNOWLEDGMENTS

Much of this research was done during 1993-94 when all the authors were with Georgia Institute of Technology in Atlanta, Georgia, USA. Andrés Gómez is now with the Key Centre of Design Computing, University of Sydney, Sydney, Australia; Nathalie Grué is now with the Institute for Learning Sciences, Northwestern University, Evanston, Illinois, USA; and Margaret Recker is now with Victoria University, Wellington, New Zealand. This work has benefited from contributions by Sambasiva Bhatta, Michael Donahoo, Vinay Pandey, and Eleni Stroulia. It has been funded in part by a grant from the Advanced Research Projects Agency and partly by internal seed grants from Georgia Tech's Educational Technology Institute, College of Computing, Cognitive Science Program, and Graphics, Visualization and Usability Center.

References

- J. Barber, M. Jacobson, L. Penberthy, R. Simpson, S. Bhatta, A. Goel, M. Pearce, M. Shankar, and E. Stroulia. Integrating Artificial Intelligence and Multimedia Technologies for Interface Design Advising. *NCR Journal of Research and Development*, 6(1):75-85, October 1992.
- D. Brown and B. Chandrasekaran. *Design Problem Solving: Knowledge Structures and Control Strategies*. Pitman, London, UK, 1989.
- T. Bylander. A Theory of Consolidation for Reasoning about Devices. *Man-Machine Studies*. 35:467-489, 1991.
- J. Carbonell, C. Knoblock and S. Minton. PRODIGY: An Integrated Architecture for Planning and Learning. *Architectures for Intelligence*, Van Lehn (editor), Lawrence Erlbaum, 1989.
- B. Chandrasekaran. Generic Tasks as Building Blocks for Knowledge-Based Systems: The Diagnosis and Routine Design Examples. *Knowledge Engineering Review*, 3(3):183-219, 1988.
- B. Chandrasekaran. Task Structures, Knowledge Acquisition and Machine Learning. *Machine Learning*, 4:341-347.
- B. Chandrasekaran. Design Problem Solving: A Task Analysis. *AI Magazine*, pp. 59-71, Winter 1990.
- B. Chandrasekaran, M. Tanner, and J. Josephson. Explaining control strategies in problem solving. *IEEE Expert*. 4(1):9-24, 1989.
- B. Chandrasekaran, A. Goel, and Y. Iwasaki. Functional Representation as Design Rationale. *IEEE Computer*, 48-56, January 1993.
- J. de Kleer. How Circuits Work. *Artificial Intelligence*, 24:205-280, 1984.
- J. de Kleer and J. Brown. A Qualitative Physics Based on Confluences. *Artificial Intelligence*, 24:7-83, 1984.
- G. Fischer, J. Grudin, A. Lemke, R. McCall, J. Ostwald, B. Reeves and F. Shipman. Supporting Indirect Collaborative Design with Integrated Knowledge-Based Design Environment. *Human-*

- Computer Interactions*, 7(3):281-314, 1992.
- J. Gero, H. Lee and K. Tham. Behavior: A Link Between Function and Structure in Design. *Proc. IFIP WG 5.2 Working Conference on Intelligent CAD*, Columbus, Ohio, pp. 201-230, September 1991.
- A. Goel. A Model-based Approach to Case Adaptation. *Proc. Thirteenth Annual Conference of the Cognitive Science Society*, Lawrence Erlbaum Associates, pp. 143-148, August 1991.
- A. Goel. Representation of Design Functions in Experience-Based Design. *Intelligent Computer Aided Design*, D. Brown, M. Waldron and H. Yoshikawa (editors), North-Holland, pp. 283-308, 1992.
- A. Goel and B. Chandrasekaran. Functional Representation of Designs and Redesign Problem Solving. *Proc. Eleventh International Joint Conference on Artificial Intelligence*, Morgan Kaufmann Publishers, pp. 1388-1394, 1989.
- A. Goel and B. Chandrasekaran. Case-Based Design: A Task Analysis. In *Artificial Intelligence Approaches to Engineering Design, Volume II: Innovative Design*, Tong and D. Sriram (editors), Academic Press, pp. 165-184, 1992.
- A. Goel, M. Pearce, A. Malkawi and K. Liu. A Cross-Domain Experiment in Case-Based Design Support: ARCHITUTOR. *Proc. AAAI Workshop on Case-Based Reasoning*, pp. 111-117, 1993.
- T. Govindaraj. Qualitative Approximation Methodology for Modeling and Simulation of Large Dynamic Systems: Applications to a Marine Power Plant. *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-17 No. 6, pp. 937-955, 1987.
- N. Grué. Illustration, Explanation and Navigation of Physical Devices and Design Processes. M.S. Thesis, College of Computing, Georgia Institute of Technology, June 1994.
- P. Hayes. Naive Physics Manifesto. *Expert Systems in the Microelectronics Age*, Edinburgh University Press, Edinburgh, UK, pp. 242-270, 1979.
- K. Hua and B. Faltings. Exploring Case-Based Building Design - CADRE. *AI(EDAM)*, 7(2):135-143, 1993.
- J. McDermott. R1: A Rule-Based Configurer of Computer Systems. *Artificial Intelligence*, 19:39-88, 1982.
- J. McDermott. Preliminary Steps Towards a Taxonomy of Problem Solving Methods. *Automating Knowledge Acquisition for Expert Systems*, S. Marcus (editor), Kluwer, Boston, MA, 1988.
- M.L. Maher, M.B. Balachandran, and D. Zhang. *Case-Based Reasoning in Design*, Erlbaum, Hillsdale, NJ, 1995.
- S. Marcus, J. Stout, and J. McDermott. VT: An Expert Elevator Designer that Uses Knowledge-Based Backtracking. *AI Magazine*, 9(1):95-112, 1988.
- D. Marr. Artificial Intelligence — A Personal View. *Artificial Intelligence*, 9(1), 1977.
- S. Mittal, C. Dym and M. Morjaria. PRIDE: An Expert System for the Design of Paper Handling Systems. *Computer*, 19(7):102-114, 1986.
- J. Mostow. Design by Derivational Analogy: Issues in the Automated Replay of Design Plans. *Artificial Intelligence*, 1989.
- B. Myers and B. Zanden. Environment for rapidly creating interactive design tools. *Visual Computer*, 8:94-116, 1992.
- D. Navinchandra. *Exploration and Innovation in Design: Towards a Computational Model*, Springer-Verlag, New York, 1991.
- A. Newell. The Knowledge Level. *Artificial Intelligence*, 18(1):87-127, 1982.
- M. Pearce, A. Goel, J. Kolodner, C. Zimring, L. Sentosa and R. Billington. Case-Based Design Support: A Case Study in Architectural Design. *IEEE Expert*, 7(5):14-20, 1992.
- J. Rasmussen. The Role of Hierarchical Knowledge Representation in Decision Making and System Management. *IEEE Trans. Systems, Man and Cybernetics*, 15:234-243, 1985.
- V. Sembugamoorthy and B. Chandrasekaran. Functional representation of devices and Compilation of Diagnostic Problem Solving Systems. *Experience, Memory and Reasoning*, J. Kolodner and C. Riesbeck (editors), Erlbaum, Hillsdale, NJ, pp. 47-73, 1986.
- H. Simon. *The Sciences of the Artificial (2nd ed.)*, MIT Press, 1981.
- R. Stallman and G. Sussman. Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis. *Artificial Intelligence*, 9:135-196, 1977.
- L. Steinberg. Design as Refinement Plus Constraint Propagation: the VEXED Experience. *Proc.*

- Sixth National Conference on Artificial Intelligence*, pp. 830-835, 1987.
- K. Sycara, D. Navinchandra, R. Guttal, J. Koning, and S. Narsimhan. CADET: A Case-Based Synthesis Tool for Engineering Design. *Expert Systems*, 4(2):157-188, 1991
- Y. Umeda, H. Takeda, T. Tomiyama, and H. Yoshikawa. Function, Behavior and Structure. In *Proc. Fifth International Conference on Applications of AI in Engineering*, 1:177-193, 1990.
- A. Voss, C-H Coulon, W. Grather, B. Linowski, J. Schaaf, B. Barstsch-Sporl, K. Borner, E. Tammer, H. Durscke, and M. Knauff. Retrieval of Similar Layouts - About a Very Hybrid Approach in FABEL. *Proc. Third International Conference on AI in Design*, Lausanne, pp 625-640, August 1994.

Towards Design Learning Environments - I: Exploring How Devices Work

Ashok K. Goel¹, Andrés Gómez de Silva Garza¹, Nathalie Grué¹, J. William Murdock¹, Margaret M. Recker¹, and T. Govindaraj²

¹ Artificial Intelligence Group
College of Computing
Georgia Institute of Technology
801 Atlantic Drive, Atlanta, Georgia 30332-0280
² Center for Human-Machine Systems Research
School of Industrial and Systems Engineering
Georgia Institute of Technology

Source: Third International Conference on Intelligent Tutoring Systems, ITS '96, Montreal, Canada, June 12 - 14, 1996. Published as Intelligent Tutoring Systems, Lecture Notes in Computer Science 1086. Claude Frasson, Gilles Gauthier, Alan Lesgold, eds., New York: Springer, 1996.

Abstract. Knowledge-based support for learning about physical devices is a classical problem in research on intelligent tutoring systems (ITS). The large amount of knowledge engineering needed, however, presents a major difficulty in constructing ITS's for learning how devices work. Many knowledge-based design systems, on the other hand, already contain libraries of device designs and models. This provides an opportunity for reusing the legacy device libraries for supporting the learning of how devices work. We report on an experiment on the computational feasibility of this reuse of device libraries. In particular, we describe how the structure-behavior-function (SBF) device models in an autonomous knowledge-based design system called KRITIK enable device explanation and exploration in an interactive design and learning environment called INTERACTIVE KRITIK.

1 Motivations and Goals

Design, construction, evaluation, and use of intelligent tutoring systems (ITS) raises a variety of complex issues. Examples include cognitive issues pertaining to how humans solve problems, comprehend, and learn; user interface issues relating to interaction and communication between humans and computers; and knowledge system issues pertaining to the content, representation, organization, and access of knowledge in the computer. Within the context of knowledge system issues, a common difficulty is the enormous amount of knowledge engineering required to construct an ITS for a particular class of users in a specific class of task domains. One potential solution to this problem is to design reusable

ITS's. In this paper, we explore another potential solution, namely, the reuse of knowledge systems already built for one set of goals to address related ITS tasks.

In particular, we are interested in the question of whether device libraries in autonomous knowledge-based design systems can be reused for supporting interactive learning of the way devices work. We have developed a family of autonomous knowledge-based device design systems called KRITIK (Goel and Chandrasekaran 1989, 1992; Goel 1991, 1992). KRITIK addresses the extremely common functions-to-structure design task in the domain of simple physical devices. Its high-level process for this design task is case-based: it designs new devices by adapting the designs of old devices. Its method for adapting old designs is model-based: it uses case-specific device models for deciding on the design modifications needed for the current problem. Thus KRITIK contains (i) a library of design cases and device models, (ii) a case-based process model of design, and (iii) a family of model-based methods for design adaptation. In this paper, we examine how an interactive version of KRITIK can enable the learning of how devices work. An accompanying paper will address the related issue of learning about design processes and methods.

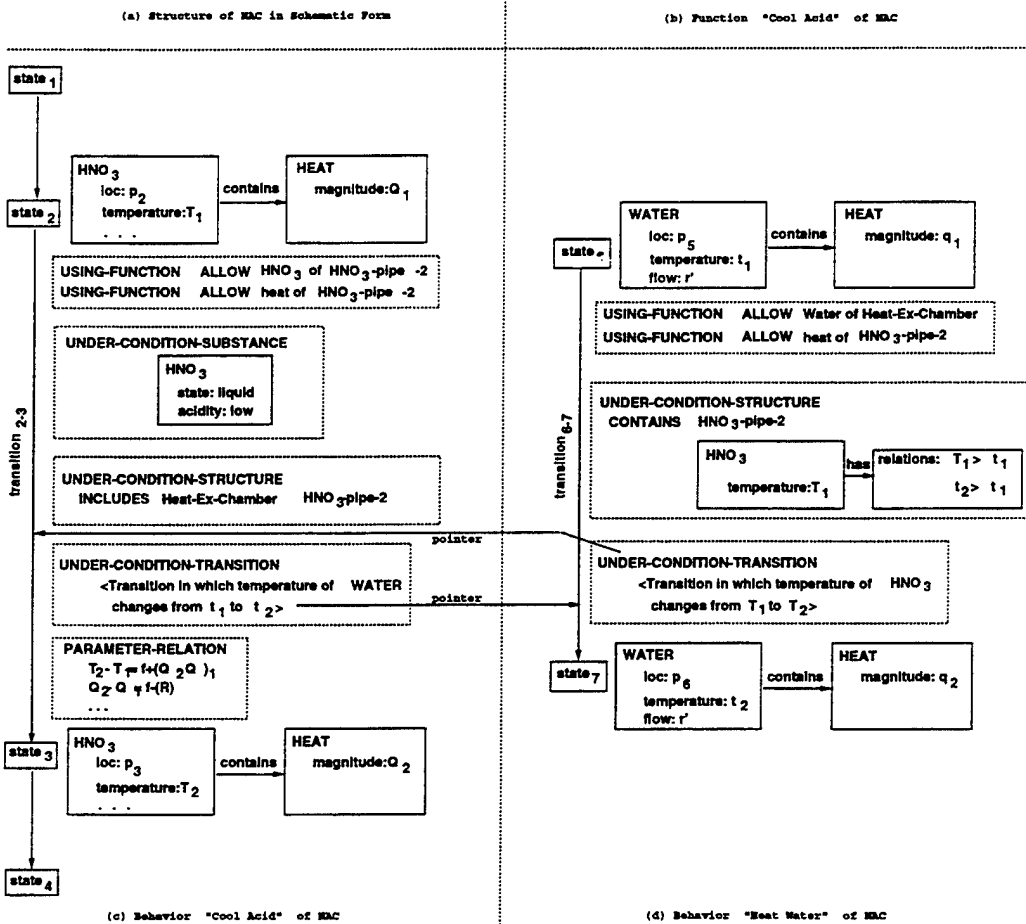
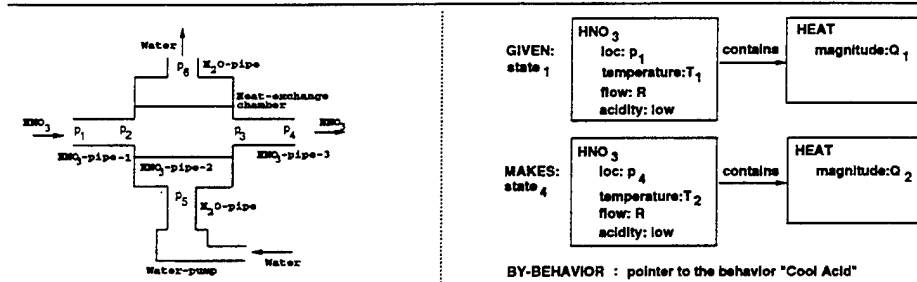
We are developing an interactive design and learning environment called INTERACTIVE KRITIK. The new environment provides a user with access to the device models in KRITIK. It also provides explanations of how the devices work and enables the user to explore the device models.

2 KRITIK

KRITIK³ contains a library of devices and associated structure-behavior-function (SBF) models. The structure-behavior-function (SBF) model of a device, such as gyroscope, explicitly represents (i) the function(s) of the device, (ii) the structure of the device, and (iii) the internal causal behaviors of the device. The internal causal behaviors specify how the functions of the structural components of the device are composed into the device functions. An SBF device model is organized hierarchically so that the device functions reference the causal behaviors responsible for their accomplishment and the causal behaviors index the functions of the device substructures. As a simple example, let us consider the SBF model of a device that cools nitric acid.

Structure: The structure of a device in the SBF language is expressed in terms of its constituent components and substances and the interactions between them. Figure 1(a) shows a diagrammatic view of the structure of a nitric acid cooler. Within the device, substances can interact both *structurally* and *behaviorally*. For example, water can flow from pump to chamber only if they are structurally *connected*, and due to the function *allow water* of the pipe that connects them.

³ The current version of KRITIK runs under Common Lisp using CLOS.



Note: All locations are with reference to components in this design.
All labels for states and transitions are local to this design.

Fig. 1. SBF Model of a Nitric Acid Cooler

Function: The function of a device in the SBF language is represented as a schema that specifies the input behavioral state of the device, the behavioral state it produces as output, and a pointer to the internal causal behavior of the design that achieves this transformation. Figure 1(b) illustrates the function of the nitric acid cooler. Both the input state and the output state are represented as *substance schemas*. The input state specifies that the substance HNO_3 at location p_1 in the topography of the device (Figure 1(a)) includes the property **temperature** and the corresponding parameter T_1 . Similarly the output state specifies that this property now has the value T_2 . Finally, the slot *by-behavior* points to the causal behavior that achieves the function of cooling acid.

The devices and their SBF models are indexed by the *functions* delivered by the devices. Thus the existing nitric acid cooler is indexed by the function illustrated in Figure 1(b). The functions, in turn, act as indices into the internal causal behaviors of the SBF model through their *by-behavior* slot.

Behavior: The SBF model of a device also specifies the internal causal behaviors that compose the functions of device substructures into the functions of the device as a whole. In the SBF language, the internal causal behaviors of a device are represented as sequences of *transitions* between *behavioral states*. The annotations on the state transitions express the *causal*, *structural*, and *functional contexts* in which the state transitions occur and the state variables get transformed. The causal context specifies *causal relations* between the variables in preceding and succeeding states. The structural context specifies different *structural relations* among the components, the substances, and the different spatial locations of the device. The functional context indicates which functions of which components in the device are responsible for the transition. Figure 1(c) shows the causal behavior that explains how heat is decreased in the nitric acid. The first two states describe the state of the acid prior to entering the chamber while the last two describe its state after the chamber. The annotation *under-condition-transition* on *transition₂₋₃* between *state₂* and *state₃* indicates that the transition occurs due to the action of the water behavior. Similarly, the annotation *under-condition-structure* specifies that the involved components need to be connected in order for the transition to occur.

3 INTERACTIVE KRITIK

INTERACTIVE KRITIK's architecture consists of two agents: a design reasoning agent in the form of KRITIK and an user interface agent⁴. The architecture is illustrated in Figure 2; in this figure solid lines represent data flow while dotted lines represent control flow.

The interface agent in INTERACTIVE KRITIK has access to all the knowledge of KRITIK. It uses KRITIK's SBF device models to graphically illustrate and explain the functioning of the devices to users. Additionally, as we will describe in

⁴ The interface is built using the Garnet tool (Myers and Zanden 1992).

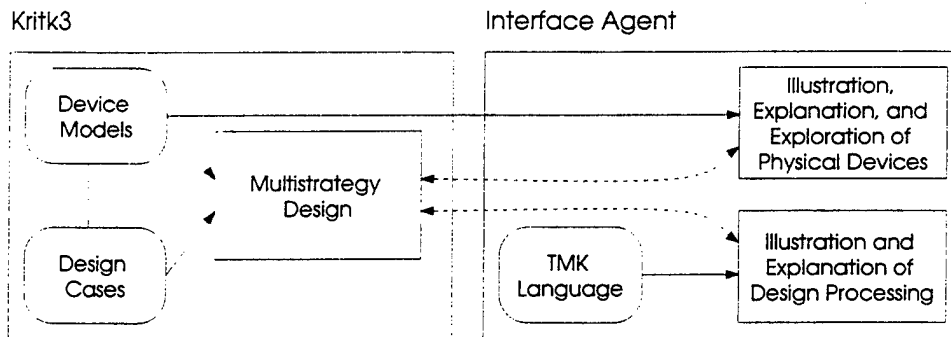


Fig. 2. INTERACTIVE KRITIK's Architecture

an accompanying paper, the interface agent uses task-method-knowledge (TMK) models to describe KRITIK's reasoning.

3.1 Device Explanation in INTERACTIVE KRITIK

INTERACTIVE KRITIK uses SBF device models to explain how a device works to a user. The SBF model provides a functional and causal explanation of how the device works in terms of its function, its structure, and its causal behaviors that specify how the functions of the structural elements get composed into the functions of the device. INTERACTIVE KRITIK illustrates the SBF model of a device to the user on several interrelated screens that illustrate the device structure, functions, and behaviors. For example, Figure 3 shows the illustration of part of the behavior of the nitric acid cooler that explains how water is heated; a different screen shows the primary behavior of this device, the cooling of the acid.

3.2 Device Exploration in INTERACTIVE KRITIK

INTERACTIVE KRITIK also enables the user to browse through different aspects of a device design. This exploration of a given device too is enabled by the SBF model. As we explained in Section 2, the SBF language provides a vocabulary for cross-indexing different parts of an SBF model. For example, the *by-behavior* slot in the specification of a function in the SBF model acts as an index to the causal behaviors that accomplish the function (see Figure 1b). Also, the *using-function* slot in the specifications of the state transitions in a causal behavior acts as an index into the functional specifications of the structural components of the device (see Figure 1c). In addition, the *under-condition-transition* slot in the specifications of the state transitions in a causal behavior acts as an index into specific transitions in other causal behaviors of the devices (see Figure 1d). The description of a device component contains a specification of its functions, and points to the causal behaviors in which the component plays a functional role.

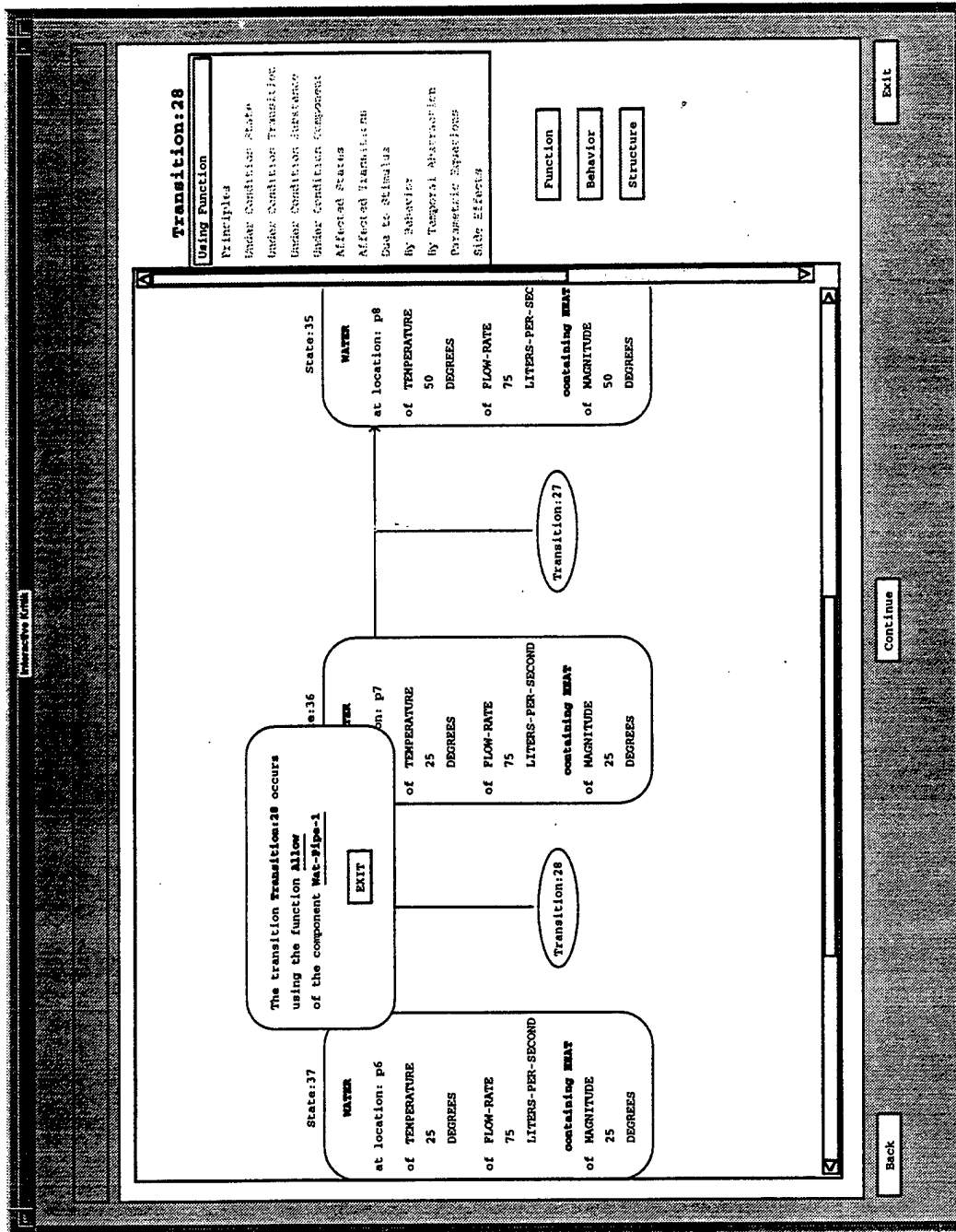


Fig. 3. A Behavioral Transition within a Nitric Acid Cooler

This organizational scheme enables the user to browse through the SBF model of the design. The initial view of an SBF model in INTERACTIVE KRITIK is a representation of the device's functional specification. From here the user can use push interface buttons to move among the functional, behavioral, and structural representations of the device. Additionally, the user can click on the name of the behavior in the *by-behavior* slot in the functional specification, and "jump" directly to that behavior. Figure 3 illustrates a behavior screen. When a user clicks on a particular transition a menu pops up that provides additional information about the transition (as illustrated in Figure 3), and allows direct access to structural and behavioral information relating to that transition. For example, if the transition is dependent on another behavior, the user can jump directly to that behavior by clicking on the name in the *under-condition-transition* slot. The structure screen provides similar capabilities for inspecting the components of a device and the connections between them.

4 Related Work

Explanation of physical devices is a classical issue in intelligent tutoring systems. SOPHIE, designed to teach troubleshooting of electrical circuits, was perhaps the first intelligent tutoring system to encounter this problem (Brown, Burton and de Kleer 1982). Early work on SOPHIE motivated much artificial intelligence and cognitive science research on "qualitative physics" and "naive physics." For example, de Kleer (1984) developed the method of *qualitative simulation* for diagnosing and predicting the behavior of electrical circuits, while Forbus (1984) developed a *qualitative process theory* to describe the behavior of physical processes as opposed to physical devices.

KRITIK's theory of SBF device models evolves from the Functional Representation (FR) scheme (Sembugamoorthy and Chandrasekaran 1986, Chandrasekaran et al. 1993). In FR, the functions are not only represented explicitly, but also used as indices to causal behaviors responsible for their accomplishment. SBF device models build on the FR scheme in three dimensions. First, SBF models are based on a well-defined component-substance ontology in which the structure of a device is viewed as constituted of components, substances and relations between them. This ontology enables explicit representation of behavioral states. Second, SBF models use Bylander's (1991) taxonomy of primitive behaviors to classify the device functions. This taxonomy enables more explicit representation of state transitions. Third, SBF models use Govindaraj's (1987) organization of causal behaviors along the flow of specific substances in the device.

The use of SBF models for device illustration, explanation and exploration is similar to Rasmussen's (1985) earlier work in cognitive engineering. Rasmussen proposed a hierarchical organization for presenting device knowledge to human users. His hierarchically-organized device models specify the structure, the behaviors, and the functions at each level in the hierarchy. TURBINIA-VYASA (Vasandani and Govindaraj 1994), uses a similar organizational scheme in a

computer-based instructional system that trains operators to troubleshoot and diagnose faults in marine power plants. But while TURBINIA-VYASA was engineered specifically as an ITS, INTERACTIVE KRITIK reuses KRITIK's knowledge for the ITS task.

ASKHOWITWORKS (Kedar et al. 1993) is a recent prototype of an interactive manual for physical devices. It indexes device information by the kinds of questions and answers that occur in typical dialogs, and enables navigation of the indexed material through question asking. While ASKHOWITWORKS takes an issue-centered view of device explanations, INTERACTIVE KRITIK takes an artifact-centered view. The latter is a natural result of reusing device libraries in knowledge-based design systems for supporting the learning of device models.

5 Conclusions

Knowledge-based support for learning about physical devices is a classical problem in research on intelligent tutoring systems (ITS). The large amount of knowledge engineering needed, however, presents a major difficulty in constructing ITS's for learning how devices work. Many knowledge-based design systems, on the other hand, already contain libraries of device designs. This provides an opportunity for reusing the design libraries for supporting the learning of how devices work. Our work on INTERACTIVE KRITIK represents an experiment in this reuse of libraries of device designs and associated structure-behavior-function (SBF) models.

There is still a great deal of work to be done on device explanation and exploration within INTERACTIVE KRITIK. Some issues which would need to be addressed before the system could be used in a real world setting include the display of the structure of a device, the building of a better user interface, and provision of additional interaction capabilities. However, our preliminary work on INTERACTIVE KRITIK does indicate the computational feasibility of using SBF models for explaining what a device does and how it does it, and for enabling the user to explore the device model.

Acknowledgments

Much of this research was done during 1993-94 when all the authors were with Georgia Institute of Technology in Atlanta, Georgia, USA. Andrés Gómez is now with the Key Centre of Design Computing, University of Sydney, Sydney, Australia; Nathalie Grué is now with the Institute for Learning Sciences, Northwestern University, Evanston, Illinois, USA; and Margaret Recker is now with Victoria University, Wellington, New Zealand. This work has been funded in part by a grant from the Advanced Research Projects Agency (research contract #F33615-93-1-1338) and partly by internal seed grants from Georgia Tech's Educational Technology Institute, College of Computing, Cognitive Science Program, and Graphics, Visualization and Usability Center.

References

- Brown, J.S., Burton, R., and de Kleer, J.: Pedagogical Natural Language and Knowledge Engineering Techniques in SOPHIE I, II, III. Intelligent Tutoring Systems, S. Derek and J. S. Brown, (Ed), Academic Press, New York (1982)
- Bylander, T.: A Theory of Consolidation for Reasoning about Devices. *Man-Machine Studies* **35** (1991) 467-489
- Chandrasekaran, B., Goel, A., and Iwasaki, I.: Functional Representation as a Basis for Design Rationale. *IEEE Computer* **26**(1) (January 1993) 48-56
- de Kleer., J.: How Circuits Work. *Artificial Intelligence* **24** (1984) 205-280
- Forbus, F.: Qualitative Process Theory. *Artificial Intelligence* **24** (1984) 85-168
- Goel, A.: A Model-based Approach to Case Adaptation. Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society, Lawrence Erlbaum Associates (1991) 143-148
- Goel, A.: Representation of Design Functions in Experience-Based Design. *Intelligent Computer Aided Design*, D. Brown, M. Waldron and H. Yoshikawa (editors), North-Holland (1992) 283-308
- Goel, A., Chandrasekaran, B.: Functional Representation of Designs and Redesign Problem Solving. Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, Morgan Kaufmann Publishers (1989) 1388-1394.
- Goel, A., Chandrasekaran, B.: Case-Based Design: A Task Analysis. *Artificial Intelligence Approaches to Engineering Design*, Volume II: Innovative Design, Tong and D. Sriram (editors), Academic Press (1992) 165-184
- Govindaraj, T.: Qualitative Approximation Methodology for Modeling and Simulation of Large Dynamic Systems: Applications to a Marine Power Plant. *IEEE Transactions on Systems, Man and Cybernetics*, **SMC-17 No. 6** (1987) 937-955.
- Grué, N.: Illustration, Explanation and Navigation of Physical Devices and Design Processes. M.S. Thesis, College of Computing, Georgia Institute of Technology (June 1994)
- Kedar, E., Baudin, C., Birnbaum, L., Osgood, R., and Bareiss, R.: *ASKHOWITWORKS: An Interactive Intelligent Manual for Devices*. INTERCHI'93 (1993)
- Myers B., Zanden, B.: Environment for rapidly creating interactive design tools. *Visual Computer* **8** (1992) 94-116
- Rasmussen, J.: The Role of Hierarchical Knowledge Representation in Decision Making and System Management. *IEEE Trans. Systems, Man and Cybernetics* **15** (1985) 234-243
- Sembugamoorthy, V., Chandrasekaran., B.: Functional representation of devices and Compilation of Diagnostic Problem Solving Systems. *Experience, Memory and Reasoning*, J. Kolodner and C. Riesbeck (editors), Elbaum, Hillsdale, New Jersey (1986) 47-73
- Vasandani, V., Govindaraj, T.: Knowledge structures for a computer-based training aid for troubleshooting a complex system. *The Use of Computer Models for Explanation, Analysis and Experiential Learning*, D. Towne (editor) NATO ASI Series F, Programme AET, Springer-Verlag (1994)

This article was processed using the L^AT_EX macro package with LLNCS style

Meta-Cases: Explaining Case-Based Reasoning

Ashok K. Goel and J. William Murdock

Artificial Intelligence Group
College of Computing
Georgia Institute of Technology

Source: Third European Workshop on Case-Based Reasoning, EWCBR '96, Lausanne, Switzerland, November 14 - 16, 1996. Published as *Advances in Case-Based Reasoning, Lecture Notes in Computer Science 1168*. Ian Smith, Boi Faltings, eds., New York: Springer, 1996.

Abstract. AI research on case-based reasoning has led to the development of many laboratory case-based systems. As we move towards introducing these systems into work environments, explaining the processes of case-based reasoning is becoming an increasingly important issue. In this paper we describe the notion of a meta-case for illustrating, explaining and justifying case-based reasoning. A meta-case contains a trace of the processing in a problem-solving episode, and provides an explanation of the problem-solving decisions and a (partial) justification for the solution. The language for representing the problem-solving trace depends on the model of problem solving. We describe a task-method-knowledge (TMK) model of problem-solving and describe the representation of meta-cases in the TMK language. We illustrate this explanatory scheme with examples from INTERACTIVE KRITIK, a computer-based design and learning environment presently under development.

1 Background, Motivations and Goals

One goal of AI research on case-based reasoning is to develop theories for designing useful and usable interactive case-based environments. In an interactive case-based environment, a human may acquire knowledge by navigating and browsing a case library, address a problem in cooperation with a case-based system, or learn about problem solving by observing the problem solving in the case-based system. The goal of designing case-based interactive systems that are both useful and usable raises the issue of *explaining* the reasoning of the case-based system. This issue is especially important in moving laboratory case-based systems into real work environments.

Explanation of reasoning is a recurrent theme in AI research. Consider, for example, the history of AI research on knowledge systems. Starting with MYCIN (Shortliffe 1976), which probably was the first useful and usable knowledge system, explanation became an increasingly important issue. In the context of MYCIN, for example, AI researchers first built an explanatory interface called GUIDON for tutoring medical students (Clancey 1987). Explanations in GUIDON initially were expressed in the language of goals, production rules, and

rule activation and selection. But the need for generating useful and usable explanations soon led to the theory of heuristic classification (Clancey 1985) that provided a task-level account of MYCIN's reasoning. This task-level model in turn led to the development of a new system called NEOMYCIN, and to an new explanatory interface called GUIDON-WATCH. In parallel, other AI researchers developed general task-oriented theories of knowledge-based problem-solving, for example, Chandrasekaran's theories of Generic Tasks (Chandrasekaran 1988) and Task Structures (Chandrasekaran 1989). Chandrasekaran, Tanner and Josephson (1989) in particular argued that explanations in interactive knowledge-based systems need to capture the functional and strategic content of problem solving at the task level.

Before we go further with this discussion, it may be useful to make some key distinctions. First, by "explanation," we mean a system's capability of generating *self-explanations*, not its ability to generate abductive explanations of external data. The generation of self-explanation is a meta-task that involves introspective meta-reasoning. Second, self-explanation includes both justification of generated solutions and justification of knowledge used in generating the solutions in addition to explanation of problem solving. In this paper, we focus on explanation of problem solving. Third, explanation in an interactive system involves the issues of content of explanations and modality of interaction. This paper focuses on explanatory content.

We are exploring the issue of explaining case-based reasoning in the context of an interactive design and learning environment called INTERACTIVE KRITIK. INTERACTIVE KRITIK directly evolves from a family of autonomous systems called KRITIK (Goel 1991, 1992; Goel and Chandrasekaran 1989, 1992). KRITIK and its successor systems combined case-based and model-based reasoning for functional design of physical devices: the high-level computational process is case-based, and structure-behavior-function device models provide (i) a model-based vocabulary for indexing, retrieving and storing design cases, and (ii) a set of model-based strategies for adapting a design case and evaluating the modified design. KRITIK3 provides both the case base and the case-based reasoner for INTERACTIVE KRITIK.

Our earlier work on case-based reasoning has naturally led us to the notion of *meta-cases* for the meta-task of explanation. A meta-case contains a trace of the processing in a problem-solving episode. We use the term 'meta-case' to distinguish it from an 'object-case' that may specify, say, a specific design. Our idea of a meta-case is related to Chandrasekaran's (1989) notion of Task Structures. A task structure of a problem solver specifies a recursive task-method-subtask decomposition that sets up a virtual architecture for the problem solver. The architecture is virtual because more than one method may be applicable to any (sub)task in the task structure. When a specific problem is presented to this virtual architecture, specific methods get selected, specific subtasks get spawned, and specific branches in the virtual architecture get instantiated. A meta-case corresponds to a specific instantiation of this virtual architecture for a particular problem. It follows that a meta-case is represented in the language of tasks and

methods.

The goal of this paper is describe the notion of meta-cases, and to illustrate the use of meta-cases for explanation in interactive systems through examples from INTERACTIVE KRITIK. The rest of this paper is organized as follows: in the next section, we describe a task-method-knowledge theory of problem solving, and, in section 3, we present an illustrative example from INTERACTIVE KRITIK. In section 4, we describe INTERACTIVE KRITIK and present an illustrative example from INTERACTIVE KRITIK. In section 5, we compare our work to related research and conclude the paper.

2 Task-Method-Knowledge Specification of Meta-Cases

AI research on knowledge systems has led to several task-oriented theories of problem solving, knowledge acquisition and explanation, for example, (Chandrasekaran 1988, 1989; McDermott 1988, Steels 1990; Wielenga, Schreiber and Breuker 1992). Although the different theories vary in many details, they all specify the content and organization of problem solving in terms of domain-independent classes of goals (called tasks) and task-specific patterns of inference (called methods). Chandrasekaran's (1989) theory of Task Structures provides the starting point for our work on modeling and explaining case-based reasoning. The main difference between our work and his theory is that our theory makes the content, form, organization of knowledge and its functional role in problem solving more explicit. For this reason, we call it the Task-Method-Knowledge (TMK) theory.

A task-method-knowledge (TMK) model of a specific problem solver has three main elements. The first element, the task, can be characterized by the types of information it takes as input and gives as output. For example, a common design task takes as input a specification of the functions desired of an artifact, and has the goal of giving as output a specification of the structure for the artifact that can deliver the desired functions. The second element in the TMK model is the method. A method can be characterized by (i) the type of knowledge it uses, (ii) the subtasks (if any) it sets up, and (iii) the control it exercises over the processing of subtasks. For example, the method of case-based reasoning uses knowledge of past cases, sets up the subtasks of retrieval, adaptation, evaluation and storage, orders these subtasks as listed here, and controls their processing so that the last three subtasks are processed only if the retrieval task fails to access an exactly-matching case that directly provides a solution to the given problem. In general, a number of methods may be applicable to a given task. The third element in the TMK model is knowledge. A specific type of knowledge can be characterized by its content, by its form of representation, and by its organization. To illustrate, consider the example of diagnostic knowledge. In some domains, models that specify how a device works may be available. In an interactive system, this knowledge system this knowledge may be represented in the form of directed acyclic graphs (DAGs) and the DAGs may be organized in a hierarchy, for example, an abstraction hierarchy.

Note that the task-method decomposition in a TMK model is recursive: since a method used for addressing a task spawns subtasks, the same task-method decomposition gets repeated for each of the subtasks. This recursive decomposition bottoms out when, for a given subtask, knowledge is available that directly solves the subtask, i.e., the knowledge directly corresponds to input-output specification of the task. We will use the term *procedure* to refer to this kind of method: a procedure does not spawn any subtasks. Also, we will use the term *strategy* to refer to subtrees in the task-method decomposition: a strategy is a specific task-method decomposition. Informally, a task in the TMK specification correspond to “goals” and the leaf-level subtasks correspond to the “operators” in means-ends analysis. Stroulia and Goel (1994a, 1994b) provide a semi-formal notation for representing tasks, methods, procedures, strategies, and knowledge.

A meta-case contains the trace of processing in a problem-solving episode. The TMK theory of problem solving provides a language for meta-cases in terms of tasks, methods and knowledge.

3 An Illustrative Example

To illustrate TMK models, we will briefly describe here the TMK model for KRITIK3, which provides the foundation for INTERACTIVE KRITIK. The primary task addressed by KRITIK3 is the extremely common functions-to-structure design task in the domain of physical devices. The functions-to-structure design task takes as input the functional specification of the desired design. For example, the functions-to-structure design of a flashlight may take as an input the specification of its function of creating light when a force is applied on a switch. This task has the goal of giving as output the specification of a structure that satisfies the given functional specification, i.e., a structure that results in the given functions.

KRITIK3's primary method for accomplishing this task is case-based reasoning. Its case-based method sets up four subtasks of the design task: *problem elaboration*, *case retrieval*, *design adaptation*, and *case storage* as illustrated in Figure 1. Note this figure shows only some of the high-level tasks and methods in KRITIK3's TMK model; it does not show the detailed decomposition of each task-method branch, nor does it show the kinds of knowledge that are used by the different methods. The rectangles in the figure represent tasks while the ovals represent methods; points beneath some of the rectangles/ovals in the figure indicate further decomposition of the tasks/methods.

The task of problem elaboration takes as input the specification of the desired function of the new design. It has the goal of generating a probe to be used by design-retrieval for deciding on a new case to use. KRITIK3 uses domain-specific heuristics to generate probes based on the surface features of the problem specification. The task of case retrieval takes as input the probes generated by the problem elaboration component. It has the goal of accessing a design case and the associated SBF model whose functional specification is similar to the specification of desired design. KRITIK3's case memory is organized in a

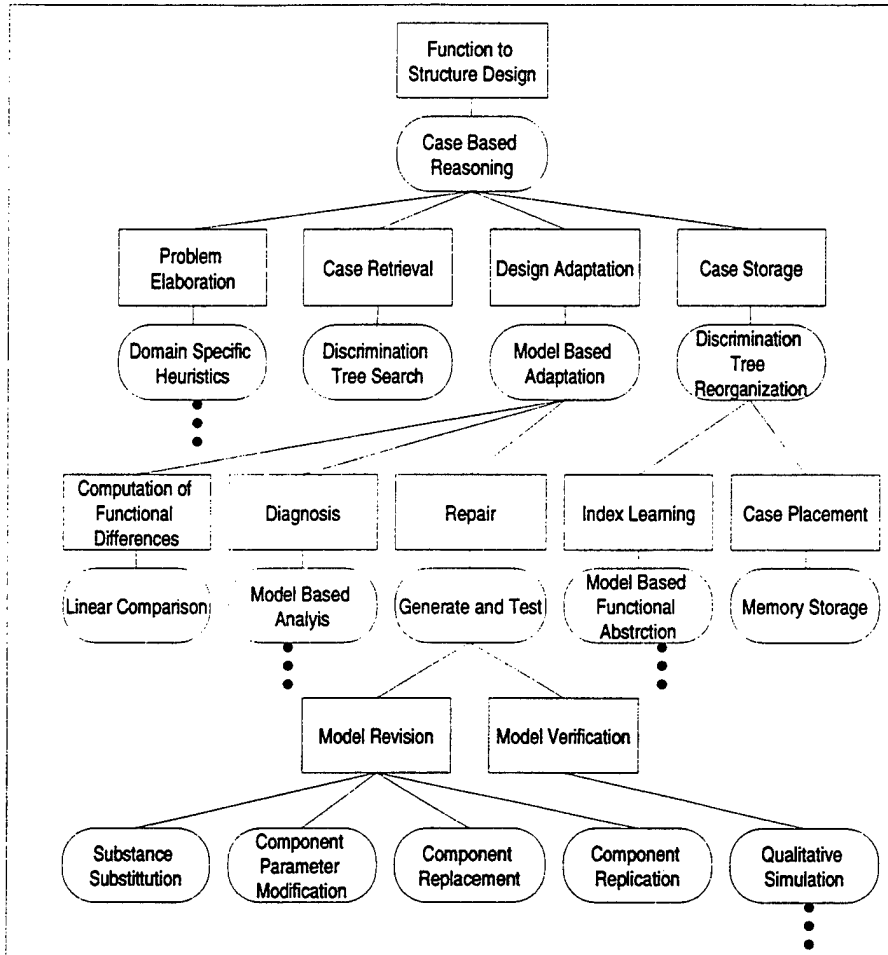


Fig. 1. The Tasks and Methods of KRITIK3

discrimination tree, with features in the functional specifications of the design cases acting as the discriminants. Its retrieval method searches through this discrimination tree to find the case that most closely matches the probe.

The task of design adaptation takes as input (i) the specification of the constraints on the desired design, and (ii) the specifications of the constraints on and the structure of the candidate design. It has the goal of giving as output a modified design structure that satisfies the specified constraints. KRITIK3 uses a model-based method of design adaptation which divides the design task into three subtasks: *computation of functional differences*, *diagnosis*, and *repair*. The idea here is that the candidate design can be viewed as a failed attempt to accomplish the desired specifications. The old design is first checked to see how its functionality differs from the desired functionality. The model of the design is

then analyzed in detail to determine one or more possible causes for the observed difference. Lastly, KRITIK3 makes modifications to the device with the intent of inducing the desired functionality.

The method of repair used by KRITIK3 is *generate and test*. This method sets up two subtasks of the repair task: *model revision* and *model verification*. The task of model revision takes as input (i) the specification of the constraints on the desired design, and (ii) the model of the candidate design. It has the goal of giving as output a modified model that is expected to satisfy the constraints on the desired design. KRITIK3 knows of several model revision methods such as the substitution of one component for another or the replication of a component. KRITIK3 dynamically chooses a method for model revision at run time based on the results of the diagnosis task. Depending on the modification goals set up by the diagnosis task, the system may also use more than one model-revision method.

The task of model verification takes as input (i) the specification of the constraints on the desired design, and (ii) the specification of the structure of the modified design. It has the goal of giving as output an evaluation of whether the modified structure satisfies the specified constraints. KRITIK3 qualitatively simulates the revised SBF model to verify whether it delivers the functions desired of it.

The task of case storage takes as input (i) a specification of the case memory, and (ii) a specification of a new case. It has the goal of giving as output a specification of the new case memory with the new case appropriately indexed and organized in it. Recall that KRITIK3's case memory is organized in a discrimination tree. The system uses a model-based method for the task of storing a new case in the tree. This method sets up the subtasks of *indexing learning* and *case placement*. The SBF model of the new design case enables the learning of the appropriate index to the new case. This directly enables the task of case placement.

4 INTERACTIVE KRITIK

INTERACTIVE KRITIK is an interactive design environment that illustrates both KRITIK3's case-based reasoning and the device designs generated by the system (Goel et al 1995; Grue1994). When completed, INTERACTIVE KRITIK is intended to serve as a constructive design and learning environment. At present, when asked by a human user INTERACTIVE KRITIK can invoke KRITIK3 to address specific kinds of design problems. In addition, INTERACTIVE KRITIK can provide explanations and justifications of KRITIK3's reasoning and results, enable the human user to explore the system's design knowledge, and also enable the user to access a library of meta-cases for examining specific reasoning traces. In this section, we describe only how INTERACTIVE KRITIK explains the case-based reasoning in KRITIK3, not the device designs the system generates.¹

¹ INTERACTIVE KRITIK's explanation of devices is described in (Goel et al 1996).

4.1 Explanation of Case-Based Reasoning in INTERACTIVE KRITIK

INTERACTIVE KRITIK's architecture consists of two agents: a case-based design agent in the form of KRITIK3 and an user interface agent². The architecture of INTERACTIVE KRITIK is illustrated in Figure 2; in this figure solid lines represent data flow while dotted lines represent control flow.

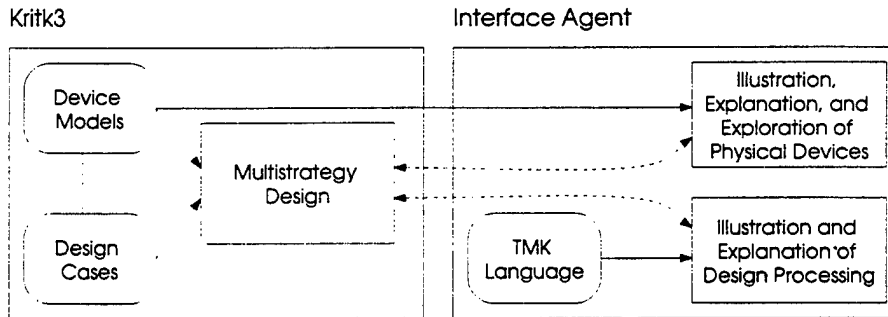


Fig. 2. INTERACTIVE KRITIK's Architecture

The interface agent in INTERACTIVE KRITIK has access to all the knowledge of KRITIK3 including its design cases and device models. It uses KRITIK3's structure-behavior-function (SBF) models of physical devices to graphically illustrate and explain the functioning of the devices to the users. It uses the TMK model of KRITIK3's case-based reasoning to graphically illustrate and explain how the system generates a new design. The trace of this reasoning is available for inspection in the form of a meta-case.

The application of multi-strategy case-based reasoning in INTERACTIVE KRITIK is illustrated to the user on several interrelated screens. Figure 3 shows the first task screen in INTERACTIVE KRITIK. It informs the user that the current task is the Design task. It also shows that KRITIK3 is planning to use the Case-Based Reasoning method, and displays the subtasks that are set up by this method: Problem Elaboration, Case Retrieval, Design Adaptation, and Case Storage.

INTERACTIVE KRITIK provides a set of screens for presenting the user with information about the input and output of the subtasks and uses highlighting features to inform the user of the reasoning state: which tasks have already been performed, what is the current task and what subtasks are left. For example, Figure 4 shows the representation of the subtasks set up by the *Model-Based Adaptation* method used for the *design adaptation* task. It illustrates a deeper level of KRITIK3's task-method decomposition.

² The interface is built using the Garnet tool (Myers and Zanden 1992).

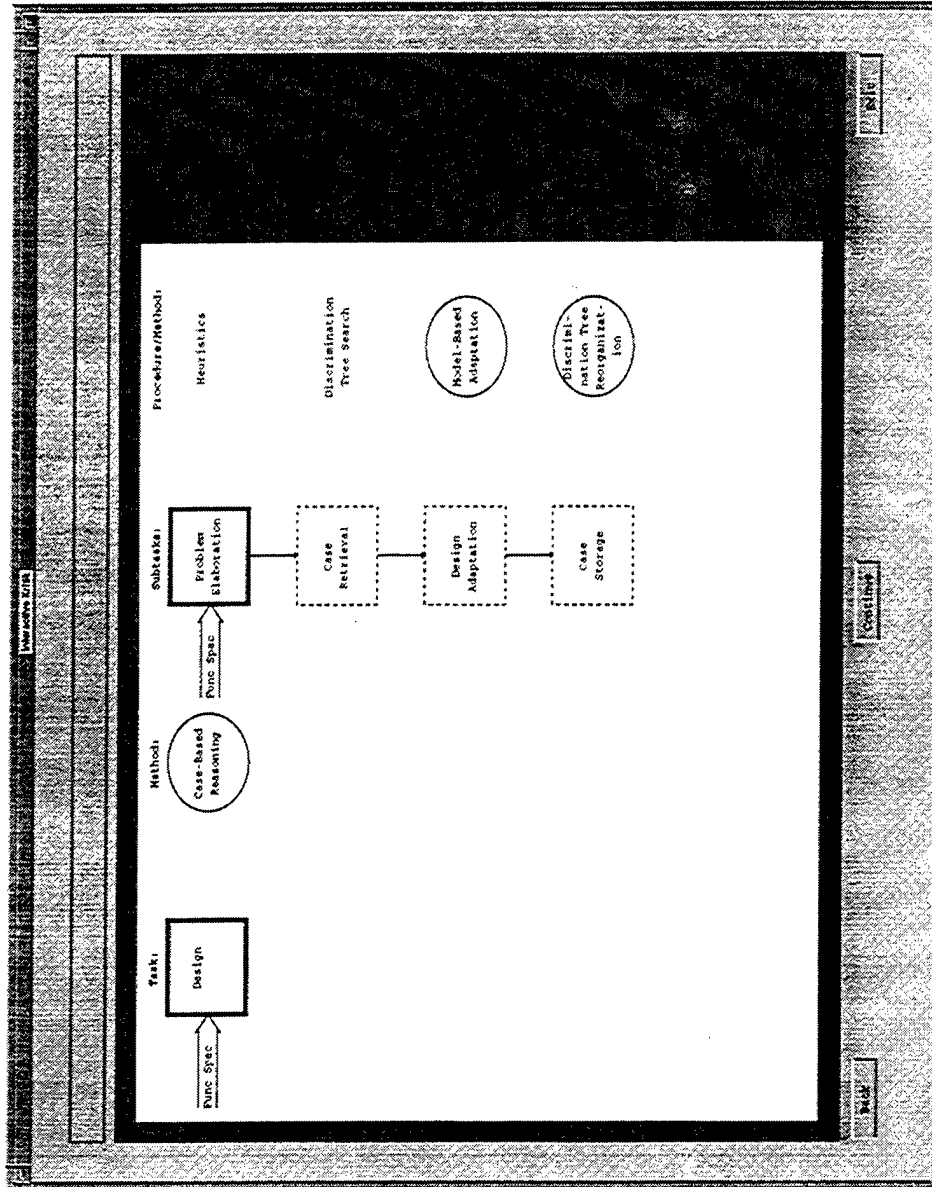


Fig. 3. The Overall Design Task

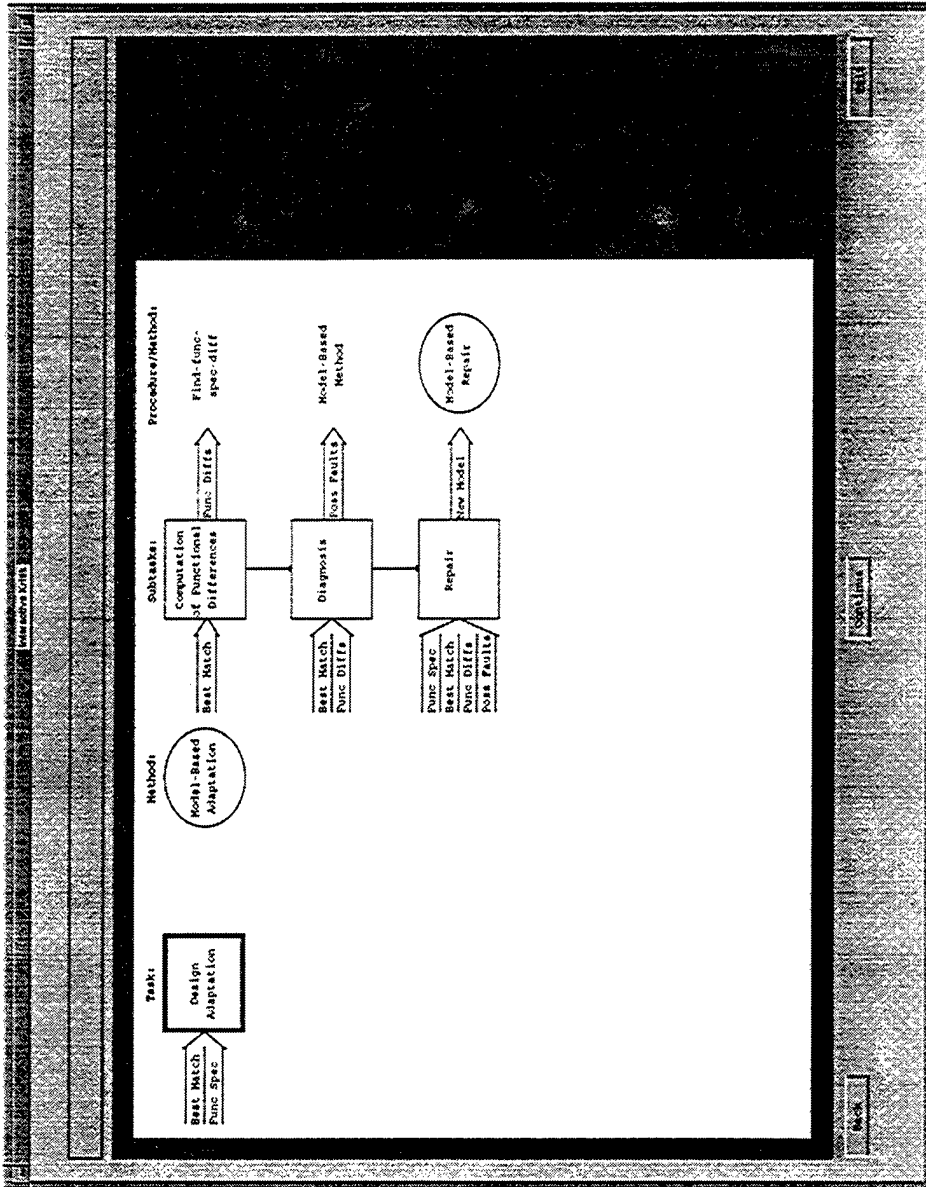


Fig. 4. The Design Adaptation Task

4.2 Reflection on Case-Based Reasoning in INTERACTIVE KRITIK

INTERACTIVE KRITIK makes available its reasoning traces in the form of meta-cases. The TMK representation of the trace of reasoning enables the user to inspect each task, method, knowledge source, and reasoning state. This enables the user to reflect on the design reasoning. For example, the user can examine the TMK reasoning trace and detect potential flaws in it.

The user can also ask INTERACTIVE KRITIK for a justification for some kinds of reasoning choices. As an example, consider the situation in which INTERACTIVE KRITIK is given a problem, INTERACTIVE KRITIK invokes KRITIK3 to solve the problem, and, during the course of reasoning, KRITIK3 retrieves a design case from its case memory. The meta-case for this design episode shows the user the probe KRITIK3 had prepared to retrieve a case and the case the system actually retrieved from its case memory. The user can now ask why did KRITIK3 retrieve this particular design case. Since the reasoning trace explicitly specifies the probe prepared by KRITIK3, and how the system's retrieval method probed the case memory - the branches it followed, the matches it made, and their results. In this way, the trace provides a justification for why the particular case best matches the given problem.

5 Discussion

In this section, we compare our work with related research on explanation in interactive case-based environments. In addition, we critique INTERACTIVE KRITIK and point to further work needed on it.

5.1 Related Research

We already have pointed out the relationship between our work on explanation of case-based reasoning and task-oriented theories of problem solving and explanation. In particular, our use of the TMK model for explaining case-based reasoning is an extension of Chandrasekaran, Tanner and Josephson's (1989) use of Task Structures for explanation of control strategies. The literature on the use of task models of problem solving for explanation and reflection is vast (e.g. (Arcos and Plaza 1994)), and we will not cover it here in its full generality. Instead, we focus on the relationship of our work with other interactive case-based problem-solving and design environments.

AI research has led to the development of several paradigms of case-based reasoning and numerous laboratory case-based systems. Kolodner (1993) provides a recent summary of the main paradigms and a compilation of the major systems. Maher, Balachandran and Zhang (1995) provide a recent summary of major case-based design systems, such as their own CADSYN and CASECAD systems, CADET (Sycara et al 1991), CADRE (Hua and Faltings), and FABEL (Voss et al 1994). None of these interactive design environments provide any kind of explanatory interface. This is also true of our own earlier work on interactive case-based design aiding systems such as Archie (Pearce et al 1992),

AskJef (Barber et al 1992) and ArchieTutor (Goel et al 1993). These systems provided human designers with access to design case libraries in different domains. AskJef, for example, used multi-media (text, graphics, animation and sound) for enabling the navigation and browsing of a library of annotated design cases in the domain of software interface design. While the case annotations provided an explanation of the designs, they did not provide an adequate explanation of case-based reasoning itself.

The JANUS system of Fischer et al (1992) and BOGART system of Mostow (1989) are two notable exceptions to this. Like INTERACTIVE KRITIK, both JANUS and BOGART provide explanation in the form of reasoning traces. Unlike INTERACTIVE KRITIK, the reasoning traces in both are part of the object cases themselves. Fischer et al have advocated that interactive design environments should provide access not just to a catalog of past designs but also to the reasoning that led to the specific designs in the catalog. Their JANUS system adopts the issue-based view of group problem solving (Rittel 1972), and provides a user with a trace of the issues that arose in a past design problem-solving episode, the arguments made for and against various design choices, and the justifications for the design decisions. Fischer et al argue that the issue-based trace of past design problem-solving episodes enables the user to make arguments for and against a specific design choice in the context of new problems, and, thus, empowers the user to create more effective designs.

Mostow adopts a similar stance towards the knowledge content of design cases in interactive design environments. Based on Carbonell's (1983, 1986) framework of derivational analogy, Mostow's BOGART system provides a user with traces of past design problem-solving episodes in the language of goals, operators, and heuristics for goal decomposition and operator selection. He argues that this derivational record of the problem solving in a past design case enables the user to more effectively transfer knowledge from the past case to the new problem.

While INTERACTIVE KRITIK shares this explanatory stance with JANUS and BOGART, we believe that the usefulness and usability of the reasoning traces used in these earlier systems are limited. The difficulty with the JANUS scheme is that it uses an informal language for representing the trace: what is (and what is not) a valid design issue, a valid argument for a design choice, a valid justification for a design decision? This informal specification may be the best that can be accomplished in recording the design rationale, i.e., the trace of decision making in a group. But in the case of explaining problem solving in an interactive system, it is possible to automate the process of explanation generation. And the difficulty with the BOGART scheme is that it represents the trace at too low a level of abstraction, e.g., operators, operator selection, and operator selection heuristics. This makes for a poor explanatory interface. Our argument mirrors Clancey's argument against the explanatory interface of his own Guidon system, which too explained problem solving in the language of goals, rules, rule activation, and rule activation heuristics. Thus JANUS's language for representing traces of design problem solving is too informal to be automated, and BOGART's language is too low level to be useful or usable in an interactive

setting. The TMK language for specifying meta-cases in INTERACTIVE KRITIK, we believe, addresses both shortcomings.

5.2 Critique

There is still a great deal of work to be done on INTERACTIVE KRITIK's user interface. As we mentioned in the introduction, so far we have focused on the content and generation of explanations, not on the display and presentation of explanations. Some issues which would need to be addressed before INTERACTIVE KRITIK can be used as a practical tool include the improved display of explanations, the building of better graphical representations, and provision of additional interaction capabilities. We recognize that INTERACTIVE KRITIK needs to be formally evaluated in a real world setting. But this kind of evaluation too requires additional work on the user interface.

5.3 Conclusions

Explanation is an important issue in the design of interactive case-based environments. In fact, if past experience in use of knowledge systems in real work environments is any guide, then explanation of problem solving is a critical issue in moving case-based systems out of the laboratory. Past experience with knowledge systems also indicates that explanations need to capture the functional, strategic and knowledge content of reasoning at the *task level*.

Meta-cases that contain reasoning traces of problem solving provide one way for explaining case-based problem solving. But to be useful and usable, meta-cases need to specify the trace at the task level. The Task-Method-Knowledge is a general task-level model of problem solving that sets up a virtual architecture for the problem solver. Meta-cases correspond to a specific instantiation of this virtual architecture for a particular problem. This insures that the meta-cases specify the task-level content and organization of reasoning. INTERACTIVE KRITIK demonstrates the computational feasibility of using meta-cases for explaining case-based reasoning.

Acknowledgments

Sambasiva Bhatta, Andrés Gómez, Murali Shankar, and Eleni Stroulia contributed to the programming of KRITIK3, while Michael Donahoo, Andrés Gómez, Gregory Grace, and Nathalie Grué contributed to the programming of INTERACTIVE KRITIK. This work has benefited from many discussions with T. Govindaraj and Margaret Recker. It has been funded in part by a grant from the Advanced Research Projects Agency (research contract #F33615-93-1-1338) and partly by internal seed grants from Georgia Tech's Educational Technology Institute, College of Computing, Cognitive Science Program, and Graphics, Visualization and Usability Center.

References

- Arcos, L. and Plaza, E. A Reflective Architecture for Integrated Memory-Based Learning and Reasoning. In *Lecture Notes in Artificial Intelligence - 837*, pp. 289-300, Berlin: Springer-Verlag, 1994.
- Barber, J., Jacobson, M., Penberthy, L., Simpson, R., Bhatta, S., Goel, A., Pearce, M., Shankar, M., and Stroulia, E. Integrating Artificial Intelligence and Multimedia Technologies for Interface Design Advising. *NCR Journal of Research and Development*, 6(1):75-85, October 1992.
- Carbonell, J. Learning by Analogy: Formulating and Generalizing Plans from Past Experience. *Machine Learning: An Artificial Intelligence Approach*, R. Michalski, J. Carbonell, and T. Mitchell (editors). Palo Alto, CA: Tioga, 1983.
- Carbonell, J. Derivational Analogy: A Theory of Reconstructive Problem Solving and Expertise Acquisition. *Machine Learning: An Artificial Intelligence Approach, Volume II*, R. Michalski, J. Carbonell, and T. Mitchell (editors). San Mateo, CA: Morgan Kaufman, 1986.
- Chandrasekaran, B. Generic Tasks as Building Blocks for Knowledge-Based Systems: The Diagnosis and Routine Design Examples. *Knowledge Engineering Review*, 3(3):183-219, 1988.
- Chandrasekaran, B. Task Structures, Knowledge Acquisition and Machine Learning. *Machine Learning*, 4:341-347.
- Chandrasekaran, B. Design Problem Solving: A Task Analysis. *AI Magazine*, 59-71. Winter 1990.
- Chandrasekaran, B., Tanner, M., and Josephson, J. Explaining Control Strategies in Problem Solving. *IEEE Expert*, 4(1):9-24, 1989.
- Clancey, W. Heuristic Classification. *Artificial Intelligence*, 27(3): 289-350, 1985.
- Clancey, W. *Knowledge-Based Tutoring: The Guidon Program*. Cambridge, MA: MIT Press, 1987.
- Fischer, G., Grudin, J., Lemke, A., McCall, R., Ostwald, J., Reeves, B. and Shipman, F. Supporting Indirect Collaborative Design with Integrated Knowledge-Based Design Environment. *Human-Computer Interactions*, 7(3):281-314, 1992.
- Goel, A. A Model-based Approach to Case Adaptation. *Proc. Thirteenth Annual Conference of the Cognitive Science Society*, Lawrence Erlbaum Associates, pp. 143-148, August 1991.
- Goel, A. Representation of Design Functions in Experience-Based Design. *Intelligent Computer Aided Design*, D. Brown, M. Waldron, and H. Yoshikawa (editors), North-Holland, pp. 283-308, 1992.
- Goel, A. and Chandrasekaran, B. Functional Representation of Designs and Redesign Problem Solving. *Proc. Eleventh International Joint Conference on Artificial Intelligence*, Morgan Kaufmann Publishers, pp. 1388-1394, 1989.
- Goel, A. and Chandrasekaran, B. Case-Based Design: A Task Analysis. In *Artificial Intelligence Approaches to Engineering Design, Volume II: Innovative Design*, Tong and D. Sriram (editors), Academic Press, pp. 165-184, 1992.
- Goel, A., Pearce, M., Malkawi, A. and Liu, K. A Cross-Domain Experiment in Case-Based Design Support: ARCHIE TUTOR. *Proc. AAAI Workshop on Case-Based Reasoning*, pp. 111-117, 1993.
- Goel, A., Gomez, A., Grue, N., Murdock, J. W., Recker, M., and Govindaraj, T. Design Explanations in Interactive Design Environments. In *Proc. Fourth International Conference on AI in Design*, Palo Alto, June 1996.

- Gru, N.é. Illustration, Explanation and Navigation of Physical Devices and Design Processes. M.Thesis, S., College of Computing, Georgia Institute of Technology, June 1994.
- Hua, K. and Faltings, B. Exploring Case-Based Building Design - CADRE. *AI(EDAM)*, 7(2):135-143, 1993.
- Kolodner, J. *Case-Based Reasoning*, Sam Mateo, CA: Morgan Kaufman, 1993.
- McDermott, J. Preliminary Steps Towards a Taxonomy of Problem Solving Methods. *Automating Knowledge Acquisition for Expert Systems*, S. Marcus (editor), Kluwer, Boston, MA, 1988.
- Maher, M. L., Balachandran, M. B., and Zhang, D. *Case-Based Reasoning in Design*, Erlbaum, Hillsdale, NJ, 1995.
- Mostow, J. Design by Derivational Analogy: Issues in the Automated Replay of Design Plans. *Artificial Intelligence*. 1989.
- Myers, B. and Zanden, B. Environment for rapidly creating interactive design tools. *Visual Computer*, 8:94-116, 1992.
- Pearce, M., Goel, A., Kolodner, J., Zimring, C., Sentosa, L. and Billington, R. Case-Based Design Support: A Case Study in Architectural Design. *IEEE Expert*. 7(5):14-20, 1992.
- Rittel, H. On the Planning Crisis: System Analysis of the First and Second Generations. *Bedriftsokonomien*, 8:390-396, 1972.
- Shortliffe, E. Computer-Based Medical Consultation: MYCIN, New York: American Elsevier, 1976.
- Steels, L. Components of Expertise. *AI Magazine*, 11(2):29-49, 1988.
- Stroulia, E. and Goel, A. A Model-Based Approach to Reflective Learning. In *Proc. 1994 European Conference on Machine Learning*, Catania, Italy, April 1994, pp. 287-306; available as *Lecture Notes in Artificial Intelligence 784 - Machine Learning*, F. Bergadano and L. De Raedt (editors), Berlin: Springer-Verlag, 1994.
- Stroulia, E. and Goel, A. Reflective Self-Adaptive Problem Solvers. In *Proc. 1994 European Conference on Knowledge Acquisition*, Germany, September 1994; available as *Lecture Notes in Artificial Intelligence - A Future for Knowledge Acquisition*, L. Steels, G. Schreiber, and W. Van de Velde (editors), Berlin: Springer-Verlag, 1994.
- Voss, A., Coulon, C-H, Grather, W., Linowski, B., Schaaf, J., Barstsch, Spori, B., Borner, K., Tammer, E., Durscke, H., and Knauff, M. Retrieval of Similar Layouts - About a Very Hybrid Approach in FABEL. *Proc. Third International Conference on AI in Design*, Lausanne, pp 625-640, August 1994.
- Wielinga, B., Schreiber, G. and Breuker, J. KADS: A Modelling Approach to Knowledge Acquisition. *Knowledge Engineering*, 4:5-53, 1992.

Functional Explanations in Design

Ashok K. Goel¹, Andrés Gómez de Silva Garza,
Nathalie Grué, J. William Murdock, and Margaret M. Recker
Intelligence and Design Group, College of Computing, Georgia Institute of Technology

Source: IJCAI-97 Workshop on Modeling and Reasoning about Function

Abstract

A key step in explaining how something works is explaining what that thing was intended to do. This is equally true of physical devices and of abstract devices such as knowledge systems. In this paper, we consider the problem of providing functionally oriented explanations of a knowledge-based design system. In particular, we analyze the content of explanations of reasoning in the context of the design of physical devices. We describe a language for expressing explanations: task-method-knowledge models. Additionally, we describe the INTERACTIVE KRITIK system, a computer program that makes use of these representations to visually illustrate the system's reasoning.

1 Introduction

One crucial aspect of the success or failure of an intelligent information system is the extent to which it enables users to understand what it is doing. Fortunately, many AI systems have an advantage which facilitates the development of effective explanatory interfaces: knowledge and reasoning is specifically designed from a functional perspective, i.e., individual elements of information and processing each contain a specific, rigorously defined purpose with respect to the overall computation. Hence it may be possible to use this understanding of purpose to describe how these elements are combined.

The particular task for which we have examined functional explanations of reasoning is design. Design is a common, everyday information processing activity. Two of the most obvious goals that a user might possess in understanding a knowledge-based design system include comprehension of physical devices and comprehension of design processes. We believe that device comprehension is an important goal in and of itself, but that understanding design process *must* be tightly coupled with some comprehension of the devices being designed. In this paper we address the issue of explaining design processes as well as the relationship between this task and that of conveying an understanding of the devices themselves.

The issue of how a knowledge system might explain its reasoning has two related but distinct facets pertaining to the content and modality of presentation of explanations to the human user, and the content and the representation of design knowledge and reasoning in the knowledge system. Our research on process explanations centers on the content of explanations presented to the user, and the content and representation of design knowledge and reasoning needed for generating the explanations.

The presentation of a design, such as that of a gyroscope, depends both on the design phase and the design domain. The content of an explanation for the design of gyroscope is different from that of an office building or a software interface. This is because the relationships between the function and the structure of the gyroscope design are fundamentally different from the function-structure relations in the design of an office building or a software interface. Our work focuses on the preliminary (conceptual, qualitative) design of simple, common physical devices such as electrical circuits, heat exchangers and angular momentum controllers. The input to this task is a specification of the desired functions, and the output is a specification of a structure that can deliver the desired functions.

¹Contact: Ashok Goel, 110 College of Computing Building, Georgia Institute of Technology, 801 Atlantic Drive, Atlanta, Georgia 30332-0280, goel@cc.gatech.edu

We use task-method-knowledge models (TMK models) [Goel and Chandrasekaran 1992; Stroulia and Goel 1994, 1995] for describing reasoning about a design problem. The TMK model of a design reasoner provides a functional and strategic explanation of reasoning in terms of the task, the methods used to accomplish the task, the subtasks spawned by the methods, and the knowledge used by the methods. We are developing an interactive design and learning environment called INTERACTIVE KRITIK. A major component of INTERACTIVE KRITIK is a knowledge-based design system called KRITIK3. INTERACTIVE KRITIK provides visual explanations and justifications of both KRITIK3's reasoning and the solutions it proposes. A key feature of INTERACTIVE KRITIK is that explanation of the design reasoning in a design episode is situated in the context of the evolving design solution, and, similarly, explanation of the design solution is situated in the context of the reasoning that led to it. In this paper we examine the explanations of reasoning processes. It is our hypothesis that TMK models capture the content of explanation of a design episode at the "right" level of abstraction.

2 Task-Method-Knowledge Models

A task-method-knowledge (TMK) model of knowledge-based design has three main elements. The first element, the task, can be characterized by the types of information it takes as input and gives as output. For example, a common design task takes as input a specification of the functions desired of an artifact, and has the goal of giving as output a specification of the structure for the artifact that can deliver the desired functions. The second element in the TMK model is the method. A method can be characterized by (i) the type of knowledge it uses, (ii) the subtasks (if any) it sets up, and (iii) the control it exercises over the processing of subtasks. For example, the method of case-based reasoning uses knowledge of past cases, sets up the subtasks of retrieval, adaptation, evaluation and storage, orders these subtasks as listed here, and controls their processing so that the last three subtasks are processed only if the retrieval task fails to access an exactly-matching case that directly provides a solution to the given problem. In general, a number of methods may be applicable to a given task.

The third element in the TMK model is knowledge. A specific type of knowledge can be characterized by its content, by its form of representation, and by its organization. To illustrate, consider the example of diagnostic knowledge. In some domains, heuristic associations that directly map signs and symptoms into fault categories are available. In a knowledge system, this knowledge might be represented in the form of production rules and organized as an unordered list.

A TMK model is derived by analysis of the task and the domain. In describing the derivation of a TMK model, it is convenient to adopt the viewpoint of a system designer. Let us suppose that a system designer has to design a system for solving a given class of tasks in a given class of domains, for example, the functions-to-structure design task in the domain of physical devices. The designer may perform task and domain analysis as follows.

- **Task Identification:** First, the designer may specify the task in terms of the generic types of information it takes as input and the generic types of information desired as its output.
- **Knowledge Identification:** Next, the designer may analyze the domain in terms of the kinds of knowledge available in it.
- **Method Identification:** Then, the designer may identify different methods afforded by the different kinds of available knowledge. This step also involves the identification of the subtasks that each method may set up.
- **Method Selection:** Next, since more than one method may be feasible, the system designer may specify the criteria for selecting a specific method. These criteria may include the following.
 - *Properties of the Solution:* Different methods produce different types of solutions. Some methods may produce optimal solutions, while others may produce satisfying ones. Some methods may

produce precise answers, while the answers of others may be only qualitative. What, then, are the requirements on the properties of a solution to the given task?

– *Properties of the Process*: Different methods may require different computational resources. Some methods may be computationally so complex that they are pragmatically infeasible. Others may vary in the processing time and memory space they take. What, then, are the constraints on the availability of computational resources?

- **Recursive Task-Domain Analysis**: Finally, the above steps may be repeated for each of the subtasks that the selected method sets up. This recursive decomposition of the given task continues up to an “elementary” level of task/method decomposition. At the elementary level, the domain affords knowledge that can directly map the input to the (sub)task into its desired output. At this level, no method is needed; instead, a procedure directly applies the relevant knowledge to solve the task.

This task and domain analysis produces a TMK model for reasoning about the given task. The criteria for selecting a particular method from a store of methods applicable to a particular task in the TMK model implies that no one method may be “the correct method” for solving all instances of the task. This is because the choice of the method is constrained by the types of knowledge available in the domain of the task instance. For example, if knowledge of previous design cases is available in a given domain, then this knowledge affords the case-based method for addressing a given design task in that domain. If, however, knowledge of such design solutions is not available then the case-based method becomes infeasible. If the knowledge types used by more than one method are available, then the choice among the methods is based on the properties of the methods and the desired solution.

Thus, the TMK model specifies a virtual architecture for the reasoner. Given a specific instance of the task, the reasoner may dynamically and flexibly select and pursue different task-method branches. The trace of the reasoning on the task instance would specify the specific task-method subtree chosen by the reasoner. Thus the reasoning trace also gets expressed in the same TMK language. A detailed example of a TMK model is presented in Figure 1 and described in detail in Section 3.1.

3 INTERACTIVE KRITIK

INTERACTIVE KRITIK is a computer-based design environment. A major component of INTERACTIVE KRITIK is KRITIK3, an autonomous knowledge-based design system. When completed, INTERACTIVE KRITIK is intended to serve as an interactive constructive design environment. At present, when asked by a human user, INTERACTIVE KRITIK can invoke KRITIK3 to address specific kinds of design problems. In addition, INTERACTIVE KRITIK can provide explanations and justifications of KRITIK3’s design reasoning and results, and enable the human user to explore the system’s design knowledge.

3.1 KRITIK3

KRITIK3² evolves from KRITIK, an early multi-strategy case-based design system. Since KRITIK is described in detail elsewhere (see, for example, [Goel and Chandrasekaran 1989, 1992]), in this paper we only sketch the outlines of KRITIK3. We do this by describing KRITIK3’s TMK model.

KRITIK3 is a multi-strategy process model of design in two senses. First, while the high-level design process in KRITIK3 is case-based, the reasoning about individual subtasks in the case-based process is model-based; KRITIK3 uses device models described in the Structure-Behavior-Function (SBF) language for adapting a past design and for evaluating a candidate design. Second, design adaptation in KRITIK3 involves multiple modification methods. While all modification methods make use of SBF device models, different methods are applicable to different kinds of adaptation tasks.

²KRITIK3 runs under Common Lisp using CLOS.

The primary task addressed by KRITIK3 is the extremely common functions-to-structure design task in the domain of simple physical devices. The functions-to-structure design task takes as input the functional specification of the desired design. For example, the functions-to-structure design of a flashlight may take as an input the specification of its function of creating light when a force is applied on a switch. This task has the goal of giving as output the specification of a structure that satisfies the given functional specification, i.e., a structure that results in the given functions.

KRITIK3's primary method for accomplishing this task is case-based reasoning. Its case-based method sets up four subtasks of the design task: *problem elaboration*, *case retrieval*, *design adaptation*, and *case storage* as illustrated in Figure 1. This figure shows only some of the high-level tasks and methods in KRITIK3's TMK model; it does not show the detailed decomposition of each task-method branch, nor does it show the kinds of knowledge that are used by the different methods. The rectangles in the figure represent tasks while the ovals represent methods; points beneath some of the rectangles/ovals in the figure indicate further decomposition of the tasks/methods.

The task of problem elaboration takes as input the specification of the desired function of the new design. It has the goal of generating a probe to be used by design-retrieval for deciding on a new case to use. KRITIK3 uses domain-specific heuristics to generate probes based on the surface features of the problem specification. The task of case retrieval takes as input the probes generated by the problem elaboration component. It has the goal of accessing a design case and the associated SBF model whose functional specification is similar to the specification of desired design. KRITIK3's case memory is organized in a discrimination tree, with features in the functional specifications of the design cases acting as the discriminants. Its retrieval method searches through this discrimination tree to find the case that most closely matches the probe.

The task of design adaptation takes as input (i) the specification of the constraints on the desired design, and (ii) the specifications of the constraints on and the structure of the candidate design. It has the goal of giving as output a modified design structure that satisfies the specified constraints. KRITIK3 uses a model-based method of design adaptation which divides the design task into three subtasks: *computation of functional differences*, *diagnosis*, and *repair*. The idea here is that the candidate design can be viewed as a failed attempt to accomplish the desired specifications. The old design is first checked to see how its functionality differs from the desired functionality. The model of the design is then analyzed in detail to determine one or more possible causes for the observed difference. Lastly, KRITIK3 makes modifications to the device with the intent of inducing the desired functionality.

The method of repair used by KRITIK3 is *generate and test*. This method sets up two subtasks of the repair task: *model revision* and *model verification*. The task of model revision takes as input (i) the specification of the constraints on the desired design, and (ii) the model of the candidate design. It has the goal of giving as output a modified model that is expected to satisfy the constraints on the desired design. KRITIK3 knows of several model revision methods such as component replication or component replacement. KRITIK3 dynamically chooses a method for model revision at run time based on the results of the diagnosis task. Depending on the modification goals set up by the diagnosis task, the system may also use more than one model-revision method.

The task of model verification takes as input (i) the specification of the constraints on the desired design, and (ii) the specification of the structure of the modified design. It has the goal of giving as output an evaluation of whether the modified structure satisfies the specified constraints. KRITIK3 qualitatively simulates the revised SBF model to verify whether it delivers the functions desired of it.

The task of case storage takes as input (i) a specification of the case memory, and (ii) a specification of a new case. It has the goal of giving as output a specification of the new case memory with the new case appropriately indexed and organized in it. Recall that KRITIK3's case memory is organized in a discrimination tree. The system uses a model-based method for the task of storing a new case in the tree. This method sets up the subtasks of *indexing learning* and *case placement*. The SBF model of the new design case enables the learning of the appropriate index to the new case. This directly enables the task of case placement.

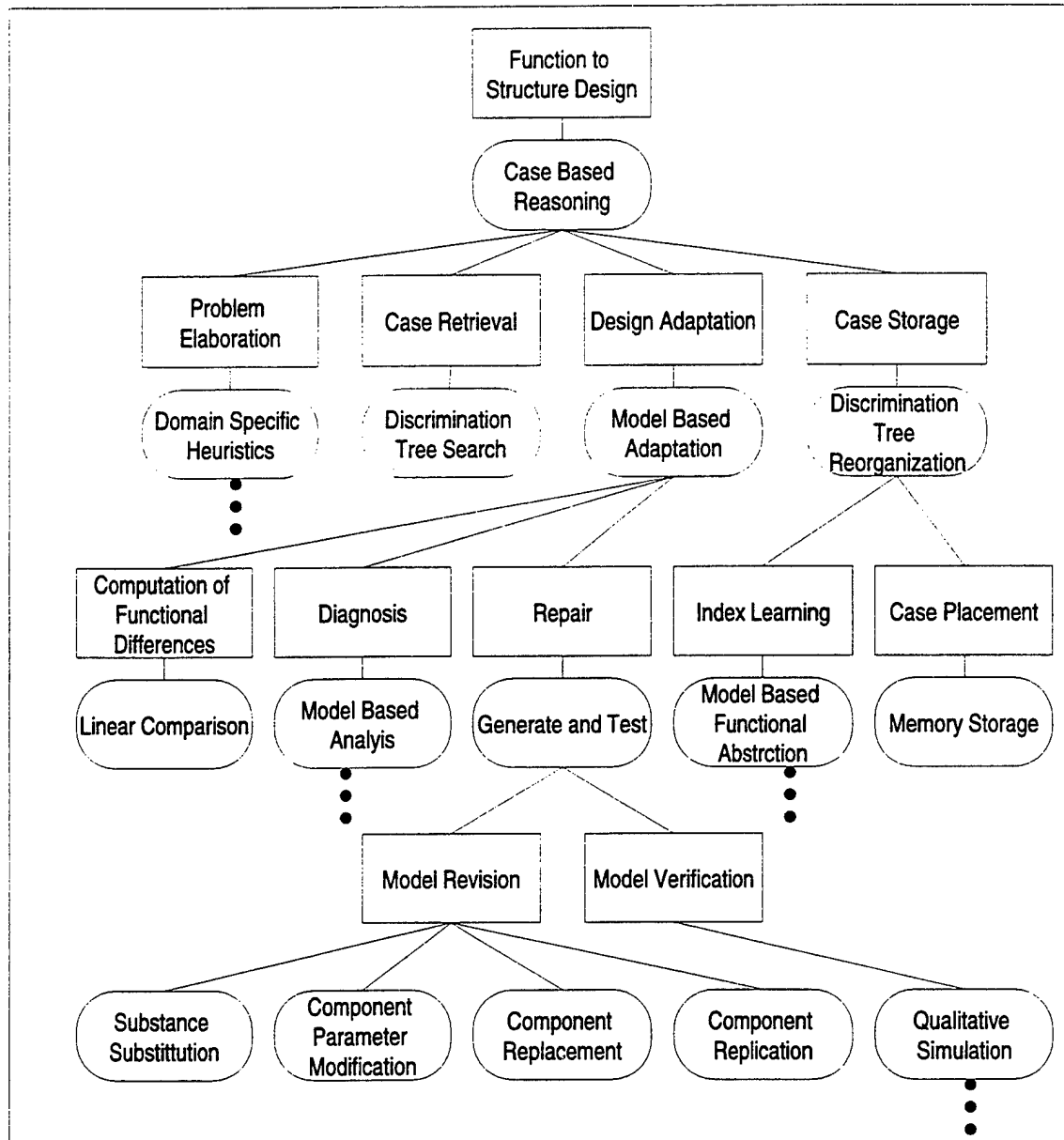


Figure 1: The Tasks and Methods of KRITIK3

3.2 Design Explanation in INTERACTIVE KRITIK

INTERACTIVE KRITIK's architecture consists of two agents: a design reasoning agent in the form of KRITIK3 and an user interface agent³. The architecture of INTERACTIVE KRITIK is illustrated in Figure 2; in this figure solid lines represent data flow while dotted lines represent control flow.

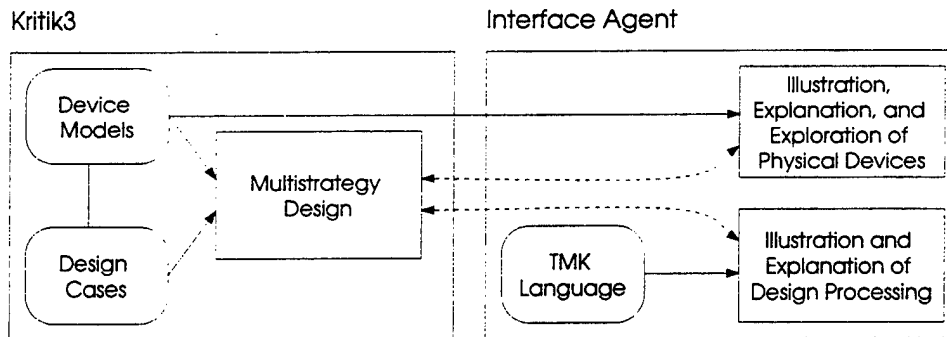


Figure 2: INTERACTIVE KRITIK's Architecture

The interface agent in INTERACTIVE KRITIK has access to all the knowledge of KRITIK3 including its design cases and SBF models. It also has a TMK model of KRITIK3's reasoning. It uses KRITIK3's SBF models of physical devices to graphically illustrate and explain the functioning of the devices to the users. It also uses the TMK model of KRITIK3's reasoning to graphically illustrate and explain how the system generates new designs.

Within the context of a design episode, INTERACTIVE KRITIK provides graphical representations of both the designs retrieved from the case memory and the new designs created. Thus it provides representations of intermediate designs in addition to the final designs. The different design versions are presented as the design reasoning unfolds, i.e., in the context of the design subtask at hand. The working of a device is illustrated to the user on several interrelated screens which are not described in detail here since the focus of this paper is on the explanations of reasoning; for examples of how INTERACTIVE KRITIK explains design products, see [Goel et al 1996].

The reasoning of KRITIK3 is specified by its TMK model. This reasoning is illustrated by INTERACTIVE KRITIK on screens identifying the tasks that KRITIK3 performs while solving a problem and the methods it uses. For each (sub)task, INTERACTIVE KRITIK illustrates the *reasoning state* both before and after the accomplishment of the (sub)task. By reasoning state, we mean the task context, the method context, and the available design information. Figure 3 shows the first task screen in INTERACTIVE KRITIK. It informs the user that the current task is the Design task. It also shows that KRITIK3 is planning to use the Case-Based Reasoning method, and displays the subtasks that are set up by this method: Problem Elaboration, Case Retrieval, Design Adaptation, and Case Storage.

INTERACTIVE KRITIK provides a set of screens for presenting the user with information about the input and output of the subtasks and uses highlighting features to inform the user of the reasoning state: which tasks have already been performed, what is the current task and what subtasks are left. For example, Figure 4 shows the representation of the subtasks set up by the *Model-Based Adaptation* method used for the *design adaptation* task. It illustrates a deeper level of KRITIK3's task-method decomposition.

4 Discussion

Explanation of problem solving has received considerable attention in knowledge-systems research. One issue in explaining knowledge-based design is the language for representing the design process. For example,

³The interface is built using the Garnet tool [Myers and Zanden 1992].

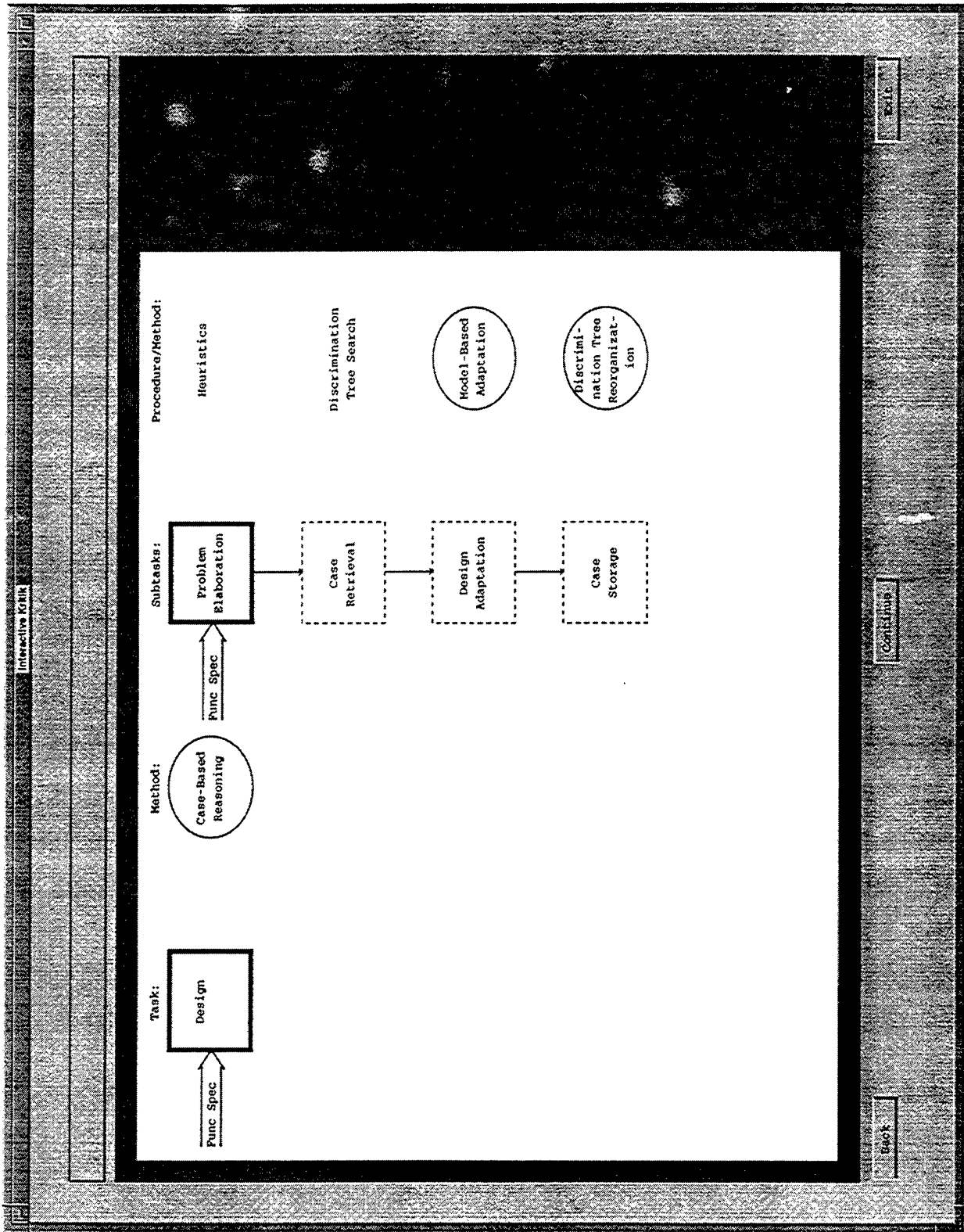


Figure 3: The Overall Design Task

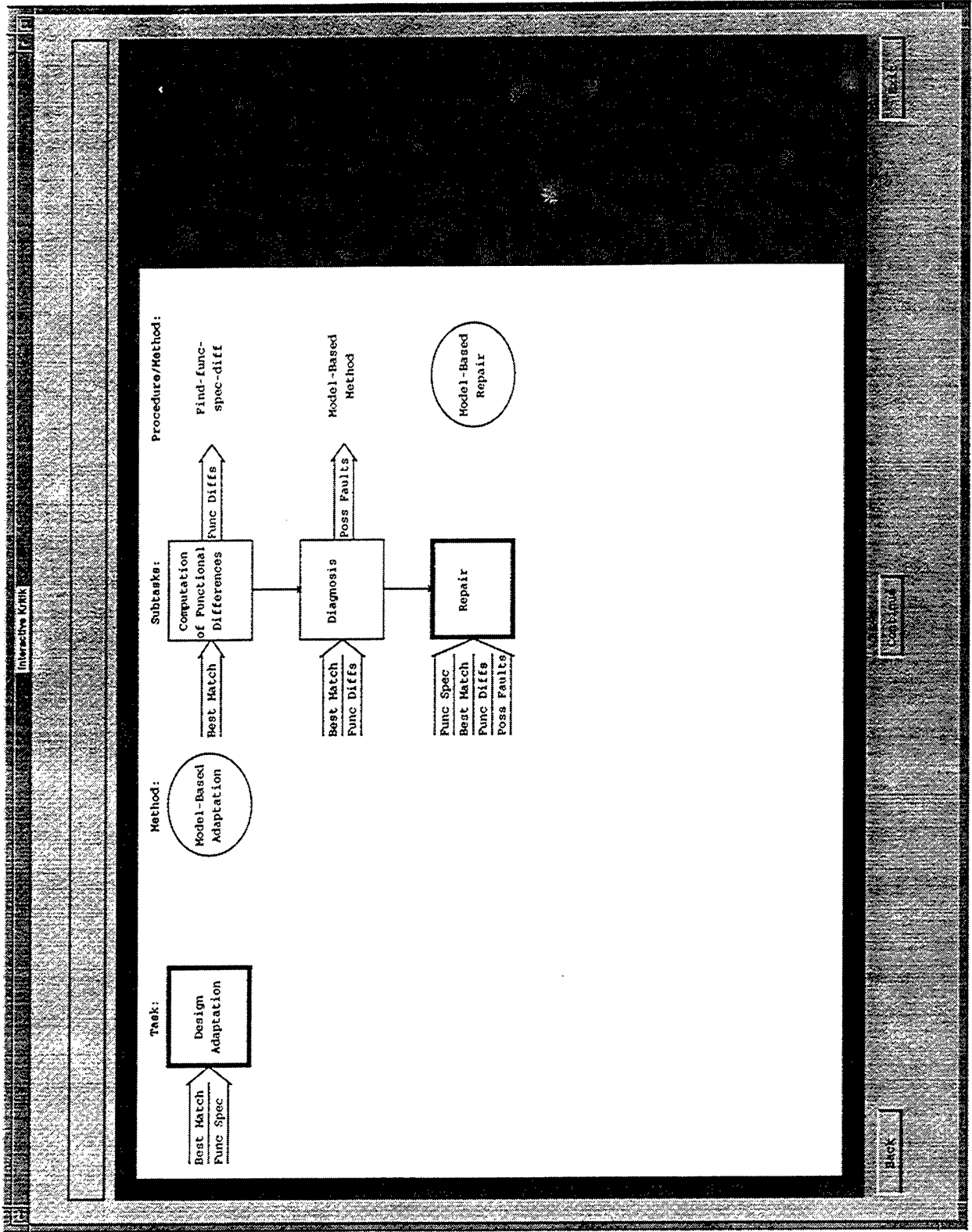


Figure 4: The Design Adaptation Task

McDermott [1982] describes R1's method for configuration design in the language of constraints of a design problem, components available in the design domain, heuristic associations pertaining to the constraints and the components, and selection and activation of the associations. But this language is much too specific to R1's method. This method-specificness of the language becomes a major problem for describing and explaining multi-strategy process models such as KRITIK3.

Task-level [Marr 1977] (or, equivalently, knowledge-level [Newell 1982]) accounts make a clearer separation between knowledge-based reasoning and its implementation in a knowledge system. In the mid-eighties, Chandrasekaran [1988] proposed the language of Generic Tasks for analyzing and modeling knowledge-based problem solving, and showed that this language enables more perspicuous explanations [Chandrasekaran, Tanner, and Josephson 1989]. In the late eighties, Chandrasekaran [1990] related Generic Tasks with task structures: [Chandrasekaran 1989] describes a high-level task structure for design; [Goel and Chandrasekaran 1992] describe a fine-grained task structure for case-based design. In their work on the elevator design project called VT, McDermott and his colleagues [McDermott 1988, Marcus et al 1988] described a similar task-oriented language for analyzing knowledge-based design.

Our TMK models represent a generalization of task structures based on Generic Tasks. TMK models make the specific role played by a particular type of knowledge more explicit than earlier models. Consider, for example, the functional role of an SBF model of a past design in KRITIK3. Since the SBF model is associated with the past case, it affords a method for adapting the past design. The TMK model makes this affordance explicit. Thus, while task structures are useful for explaining the control of reasoning in terms of task-method interactions, TMK models are also useful for explaining knowledge-method interactions. In particular, they enable the explanation of the organization and indexing of different kinds of knowledge, the kinds of knowledge available for addressing a task, and the methods that become feasible because of the available knowledge.

4.1 Critique

There is still a great deal of work to be done on the usability of interface for INTERACTIVE KRITIK; our current research has focused heavily on the *content* of the explanations and specific *implementation* issues at the level of what should the buttons on the screen be and where should they be located have been largely ignored. While these issues are not directly relevant to the *theory* which we present here, they would need to be addressed before the system could be used as a practical *tool*. In particular, open issues include the improved display of the structure of a device, the building of better graphical representations, and provision of additional interaction capabilities. More importantly, INTERACTIVE KRITIK needs to be formally evaluated in a real world setting. But this kind of evaluation also requires additional work on the user interface.

4.2 Conclusions

It is generally desirable for intelligent systems to be able to provide explanations of what they are doing. In these systems users are better able to understand the results of a task and are more likely to have greater confidence that these results are correct and meaningful. So the issue becomes how might a knowledge system enable the user to form a mental model of its reasoning, how might it explain its reasoning and justify its answers. Our work on INTERACTIVE KRITIK depends heavily on these two related ideas:

- Explanations of a knowledge system need to capture functional and strategic content of reasoning in addition to its knowledge content. Task-method-knowledge models enable this kind of explanation at a level of abstraction that facilitates effective communication between the system and the user.
- Explanation of design reasoning needs to be situated in the context of the evolving design solution, and, similarly, the explanation of the evolving design needs to be situated in the context of the design reasoning that led to it.

INTERACTIVE KRITIK demonstrates the computational feasibility of these ideas.

Acknowledgments

Much of this research was done during 1993-94 when all the authors were with Georgia Institute of Technology in Atlanta, Georgia, USA. Andrés Gómez is now with the Key Center for Design Quality, University of Sydney, Sydney, Australia; Nathalie Grué is now with the Institute for Learning Sciences, Northwestern University, Evanston, Illinois, USA; and Margaret Recker is now with Victoria University, Wellington, New Zealand. This work has benefited from discussions with T. Govindaraj. It has been funded in part by a grant from the Defense Advanced Research Projects Agency (research contract #F33615-93-1-1338).

References

- [Chandrasekaran 1988] B. Chandrasekaran. Generic Tasks as Building Blocks for Knowledge-Based Systems: The Diagnosis and Routine Design Examples. *Knowledge Engineering Review*, 3(3):183-219, 1988.
- [Chandrasekaran 1989] B. Chandrasekaran. Task Structures, Knowledge Acquisition and Machine Learning. *Machine Learning*, 4:341-347.
- [Chandrasekaran 1990] B. Chandrasekaran. Design Problem Solving: A Task Analysis. *AI Magazine*, pp. 59-71, Winter 1990.
- [Chandrasekaran, Tanner and Josephson 1989] B. Chandrasekaran, M. Tanner, and J. Josephson. Explaining control strategies in problem solving. *IEEE Expert*. 4(1):9-24, 1989.
- [Goel and Chandrasekaran 1989] A. Goel and B. Chandrasekaran. Functional Representation of Designs and Redesign Problem Solving. *Proc. Eleventh International Joint Conference on Artificial Intelligence*, Morgan Kaufmann Publishers, pp. 1388-1394, 1989.
- [Goel and Chandrasekaran 1992] A. Goel and B. Chandrasekaran. Case-Based Design: A Task Analysis. In *Artificial Intelligence Approaches to Engineering Design, Volume II: Innovative Design*, Tong and D. Sriram (editors), Academic Press, pp. 165-184, 1992.
- [Goel et al 1996] . Goel, A. Gómez de Silva Garza, N. Grué, J. W. Murdock, M. Recker, and T. Govindaraj. Explanatory Interface in Interactive Design Environments. Fourth International Conference on Artificial Intelligence in Design, AID '96, Stanford, California, June 24 - 27, 1996. John S. Gero and Fay Sudweeks, eds. Boston: Kluwer Academic Publishers, 1996.
- [McDermott 1982] J. McDermott. R1: A Rule-Based Configurer of Computer Systems. *Artificial Intelligence*, 19:39-88, 1982.
- [McDermott 1988] J. McDermott. Preliminary Steps Towards a Taxonomy of Problem Solving Methods. *Automating Knowledge Acquisition for Expert Systems*, S. Marcus (editor), Kluwer, Boston, MA, 1988.
- [Marr 1977] D. Marr. Artificial Intelligence — A Personal View. *Artificial Intelligence*, 9(1), 1977.
- [Myers and Zanden 1992] B. Myers and B. Zanden. Environment for rapidly creating interactive design tools. *Visual Computer*, 8:94-116, 1992.
- [Newell 1982] A. Newell. The Knowledge Level. *Artificial Intelligence*, 18(1):87-127, 1982.
- [Stroulia and Goel 1994b] E. Stroulia and A. Goel. Reflective Self-Adaptive Problem Solvers. In *Proc. 1994 European Conference on Knowledge Acquisition*, Germany, September 1994; available as *Lecture Notes in Artificial Intelligence - A Future for Knowledge Acquisition*, L. Steels, G. Schreiber, and W. Van de Velde (editors), Berlin: Springer-Verlag, 1994.
- [Stroulia and Goel 1995] . E. Stroulia and A. Goel. Functional Representation and Reasoning in Reflective Systems. *Journal of Applied Intelligence*, Special Issue on Functional Reasoning, 9(1): 101-124, 1995.

SECTION 3: CONCLUSIONS

This project has dealt with many important problem areas related to intelligent query processing in the context of heterogeneous databases.

In Part I of this project we addressed the issues of diversity of schemas of databases and showed that to support intelligent front ends like the KRITIK3 system for Engineering Design, it is essential to take a very flexible rule based approach where new rules of correspondence and mapping of information can be continually extended. We were able to demonstrate that for a given set of generic requests from a front end tool, we may have to perform matches on relation names, attribute names or actual values to find the relevant information. This is a new approach to dealing with the schema integration problem that has not been explored very much. The approach needs to be extended to a variety of data models including object oriented models. Our approach is limited presently to queries in SQL and rules that relate individual tables. This needs to be extended to other languages and establishment of correspondences or rules among sets of tables. Another possible extension is to tie the semantic constraints or rules from Part III of our work into Part I, thereby increasing the potential for optimization of queries on integrated databases. However, the whole area of how to extend the metadata view graph framework for multiple databases is an open research problem.

In the knowledge based system integration part of our work, we addressed the issues related to tying intelligent front ends to non-intelligent back ends and the dual problem of data integration and process (or method) integration. We attempted to extend reasoning in the context of engineering design by incorporating a large amount of external data from databases. This facility is typically absent in the tools like design assistants at the present time. A large payoff exists by extending the tools in A.I. like KRITIK3 with a collection of data sources. In Part II, where we dealt with textual information, the explanation of the ranking of documents was provided by means of visualization in the form of a histogram of words vs. top ranked documents. The explanation of how a user request is reasoned about helps the user in interpreting the answer obtained. In Part IV of the project a specification of the device meta model in the KRITIK3 system is utilized to explain the answer. Our work on method specific data to knowledge transformation illustrated how to convert data extracted from a (possibly legacy) database into a form appropriate for the processing method used within a knowledge based system. The extension of this work to generic cases of extracting data and incorporating into intelligent processes remains a challenging problem. The work takes a different flavor based on the knowledge representation schemes used and the types of reasoning employed.

Our work on part II of the project was done in the context of textual data. We evaluated user interface and visualization techniques for more efficient retrieval from document databases. The technique can be extended to any

databases at large. The experimental framework can be improved by setting up different controls, forming homogeneous groups of users, accounting for the effect of difficulty of topics, subject-topic interaction, etc. This work has great potential by combining it with the work from DARPA's TIPSTER program which has focused more on the retrieval technology and hardly any on user interface and visualization techniques. Our field studies and experiments with the interface and visualization ideas have clearly established that user productivity in terms of locating relevant information and evaluating relevance of information can both be improved by these techniques.

The work holds a lot of promise in terms of leading up to proper interfaces for accessing the information on the web. Another fruitful area of research is to extend the thesaurus concept graphs like the CYC system from MCC. With additional knowledge support, user activities in searching and browsing can be made more focused with tremendous productivity gains for knowledge workers. Accessing heterogeneous text in terms of HTML, SGML etc. is another possible extension of our research.

Part III of our work makes a contribution in the use of instance level knowledge about data in optimizing query processing. Work on semantic query optimization to date has considered only schema level information. We considered instance level constraints and their use in minimizing work during processing of queries and computations against views of data. The work can be extended to incorporate additional semantic constraints as well as inter-database constraints. More work can be done on classification of rules, linking of MVGs with dynamic execution plans and on implementing an integrated query processing system based on these ideas.

Overall, the current project has explored several issues central to the theme of the I3 (Intelligent Integration of Information) program at DARPA. Specifically, techniques have been developed and tested to help in solving problems like engineering device design, searching for information from large corpuses of text etc. Research advances were made in the areas of a flexible integration of database schemas using rules, query formulation for text databases using visualization, semantic optimization of queries using instance based constraints, and an integration of knowledge based and data based query processing.

SECTION 4: RECOMMENDATION

Our publication [1.1] has spelled out a number of open problems that still deserve further deeper investigation in the context of this project and the DARPA I3 program on the whole:

- Identification of which sources are relevant and which are not, based on a knowledge of the metadata. The relevance determination problem is addressed in [2.1,2.2] through visualization. Relevance of metadata and constraints to existing views is addressed in [3.1, 3.2]. Explanation of why a certain answer is given to a query is partly considered in [1.3] and in detail in [4.1] and [4.2].
- We used the thesaurus idea to help users in query formulation [2.1,2.2]. For matching user requests with available data sources, synonyms, thesauri and ontologies can be employed. If DARPA's work on Ontologies can be tied into our front end interface work, the resulting interface will be very powerful. Another possibility is to maintain user profiles which are kept up to date and based on these profiles, only parts of an ontology are accessed.
- The entire area of query formulation has been given scant attention in databases, and particularly in heterogeneous databases. We addressed the problem by using positive and negative feedback in the context of document databases in [2.1,2.2] and evaluated it to show that visualization techniques that provide relevance of retrieved documents to query words indeed help users in the reformulation task. Further work is necessary in determining how users should be guided in the query reformulation task.
- More research is necessary in automated knowledge acquisition from available knowledge sources (as pointed out in [1.1]). This knowledge comprises identification, content, description, and interrelationships among the data sources. How to acquire it and represent it is a worthwhile problem, particularly in the context of the world wide web which places a vast number of data sources at an average user's disposal.
- An incremental approach to adding and deleting sources from federations needs to be considered. We proposed a flexible approach to schema integration in [1.2, 1.4]. Adding new schemas translates into adding new rules in this approach.
- Dealing with external knowledge sources (such as catalogs, newsgroups, web sites) during query processing is a largely unexplored problem. It requires natural language understanding and concept derivation and knowledge acquisition that transcends traditional data processing. Our work in parts I and II is relevant to this problem.

- We addressed query optimization only at the level of semantic query optimization using metadata in [3.1, 3.2]. Using further knowledge about physical data sources and their organization, optimal strategies for query processing and optimization can be developed. There is a need to tie the work on multiple query optimization, parallel and distributed query optimization into heterogeneous databases and their query processing.

The research conducted here in four distinct parts has opened up many avenues for further long term research with a high potential impact.