From: AAAI-88 Proceedings. Copyright ©1988, AAAI (www.aaai.org). All rights reserved.

Specification By Reformulation: A Paradigm for Building Integrated User Support Environments[1]

John Yen, Robert Neches, and Michael DeBellis

USC / Information Sciences Institute
4676 Admiralty Way,
Marina del Rey, CA 90292

Abstract

Specification by reformulation is a general interface paradigm. It is an abstraction of *retrieval by reformulation*, a paradigm used in previous systems for assisting users in formulating database queries. *Specification by reformulation* serves as a general foundation upon which domain specific applications can be built. To illustrate its usage, we describe three services built within it: a database retrieval aid, a notecards facility, and an electronic-mail interface to an on-line procurement system. Building systems in this way illustrates the concept of an integrated user support environment - a set of cooperating tools for end users that can be extended by application builders.

1. Introduction

An integrated user support environment is a set of modular software tools that can interact and smoothly pass partial results back and forth [Neches 88]. It is similar to the concept of a tool-oriented programming environment such as Unix. However, whereas a programming environment consists of tools that are used primarily by computer programmers, a user support environment consists of higher-level tools that can cooperate and aid application users in their day-to-day activities. Just as good programming environments are marked by close integration between their tools (for example the ability to invoke an editor from a debugger in order to edit the source code of the function being examined in the debugger) the tools of a user support environment should be constructed in an open fashion so that they will naturally be able to work together.

In this paper, we will demonstrate a particular approach to constructing such environments. Our starting point is an interface paradigm called retrieval by reformulation that has been used for knowledge-based database retrieval systems. We will show how this paradigm can be abstracted to a general paradigm which we call *specification by reformulation*. By this we mean a paradigm of human/computer interaction in which the user develops by successive approximations a specification of the objects a system is to manipulate and/or the behavior it is to evince. In this style of interaction, the system provides an environment which facilitates the refinement of the specification, largely by generating feedback for the user about the specification in its current form and by providing guidance about means for modifying that specification.

The implementation of the specification by reformulation paradigm consists of a knowledge representation structure that represents the current specification, and general functions that provide feedback and guidance on refinement of the specification. Once an environment for operating within the specification by reformulation paradigm has been implemented, it is relatively easy to build applications on top of it. Although such applications will seem to have little in common on the surface, at a higher level of abstraction they will all utilize the same paradigm for accomplishing their various tasks. The one thing that they will have in common is that they will all depend on the structure of a database or knowledge base to successfully complete their tasks. Constructing such systems benefit users via a consistent interface and cooperating tools and system builders via modularity and reusability of code.

We will discuss the BACKBORD system [Yen 88], an implementation of the specification by reformulation paradigm. BACKBORD illustrates how the paradigm can be applied to database query formulation, knowledge base browsing, the creation and attachment of notes to a knowledge base, and the creation of mail messages. These services are tools which must operate in an integrated fashion within an intelligent workstation for procurement of standard electronic parts [Neches 88].

2. Specification by Reformulation

Specification by reformulation is an abstraction of retrieval by reformulation, a paradigm for assisting users in formulating database queries [Tou 82, Williams 84]. Retrieval by reformulation assists users who know what they are trying to retrieve but cannot construct a query to retrieve it, either because they do not understand the

[1]The research described in this paper was supported by DARPA under contract No. MDA903-86-C-0178, and by the Air Force Logistics Command under contract No. F33600-87-C-7047. Views and conclusions contained in this paper are those of the authors, and should not be interpreted as representing the official opinion or policy of the sponsoring agencies.

query language or because they lack knowledge about the structure of the database. The major techniques of the paradigm came from a psychological theory of human remembering [Williams 81]. Stelzner and Williams [Stelzner 86] developed the term *specification by reformulation* to refer to a generalization of the retrieval by reformulation paradigm used to develop knowledge base interfaces. By abstracting the retrieval by reformulation paradigm, we achieve a general paradigm for interacting with large amounts of stored data that is based on a psychological theory of human information retrieval.

2.1. Previous Work

Previous systems such as RABBIT [Tou 82, Williams 84], ARGON [Patel-Schneider 84] and *Intelliscope*™ (a recent commercial product of Intellicorp) have used retrieval by reformulation to aid database users. Such systems consist of a query (i.e., the description), a matching list, and an example. The query serves as the current context that the user has been able to establish and a description of what is being searched for in the database. The matching list contains all database records that match the description used for the last retrieval. The example shows the detailed content of one of the matching records.

The major idea is to provide interactive guidance on possible ways to reformulate the query. Using this guidance, users can modify the original query to better reflect their intent. For example, the user can select values from the example to further constrain the query. The refined query is then used for another retrieval. Reformulation and retrieval iterate until the user is satisfied with the retrieval results.

2.2. A General Interface Paradigm

Specification by reformulation provides a high level tool for applications utilizing databases or knowledge bases. The paradigm consists of alternating between (1) creating/refining a specification for achieving a user's goals and (2) obtaining feedback on the effect of the current specification and guidance about how it can be modified. When satisfied with this process, the user can then execute actions that utilize the resultant specification.

In a retrieval-by-reformulation aid, refinement means modifying the query, feedback is obtained by retrieving against the query, guidance is obtained through menus indicating ways that contents of information fields in the display can be used to modify the query, and actions consist of tools for graphically displaying the retrieved data. By generalizing the notions of queries, retrievals, and examples to those of specifications, feedback, and guidance, we can address applications well beyond database browsing. We next will describe an architecture for doing so, followed by some illustrative applications.

3. Representing Specification by Reformulation

This section elaborates the paradigm and describes its implementation in BACKBORD. Since BACKBORD operates on NIKL knowledge bases, a few terms in the NIKL knowledge representation language must be introduced. A NIKL knowledge base consists of *concepts* and *roles*, which correspond to frames and slots in frame-based systems. The *value restrictions* in NIKL are used both to provide values of a slot and to provide restrictions on possible values. In this paper all references to knowledge base objects will appear in **Bold** face. A more detailed description of the NIKL language is provided by [Moser 83]. The most significant aspect of NIKL related to our concerns is the presence of an automatic *classifier* [Schmolze and Lipkis 83], which utilizes the semantics of concept and role definitions to reason about where new concept descriptions fit with respect to pre-existing concepts in a subsumption (*isa*) hierarchy.

3.1. Refinement: Search in a Description Space

Specification by reformulation involves a search through a space representing an abstraction hierarchy of descriptions. At any given point in time, the current specification can be thought of as a node in that space. Based on the node's location, and the feedback obtained from it, the system helps users select operations to produce descriptions that are closer to expressing their intent. The specification consists of various roles. Each role itself has properties (information on possible values, restrictions on cardinality, etc.). Based on these properties the specification can always be classified to determine its current position in the knowledge base. Based on its classification, certain things can be said about the specification in its current state (Feedback). Based on the system's model of valid specifications, it can make inferences about how the specification needs to be changed in order to classify into a valid place (Guidance).

An application suitable for the paradigm must be formulated so that successful completion can be viewed in terms of one or more concepts to be found or created in the knowledge base. Once formulated in this manner, execution of the application takes the form of searching through the space of possible descriptions until the correct one is found. Thus, an application developer's task is to provide BACKBORD with: (1) a taxonomy of the concepts in the application domain (e.g., a knowledge base model of the information covered in a database); (2) mappings from class concepts in the taxonomy to procedures for obtaining feedback (e.g., functions for converting a concept into a database query); (3) associations between class concepts and actions applicable to members of that class (e.g., offering auto-dialing on data entries containing phone numbers).

With this information, the system can provide a general-purpose user interface for viewing specification

descriptions, obtaining feedback and guidance, modifying the descriptions, and invoking actions upon resultant descriptions. The interface is customizable within certain parameters, because an application developer specifies how feedback is generated from a description and what actions can be applied to it. However, the interface provides a framework in which application developers are freed from specifying much that would otherwise be required, and in which users see a high degree of consistency across individual tools.

3.2. Guiding the User Through the Search Space

Because of the general nature of the specification by reformulation paradigm, aids can be provided to help the user navigate through the space of possible descriptions which can be used with little or no modification by specific applications.

In BACKBORD the specification is represented by a structure with superconcepts, roles, and value restrictions for the roles. This is a NIKL concept[2] to be classified into the knowledge base. Figure 3-1 shows a BACKBORD screen browsing a database of standard electronic parts. The specification is displayed in a window called the description window. BACKBORD also has a scrollable window which contains all the objects that classify below the current specification (the window labeled "Matching Instances" in Figure 3-1). The user can select any one of these to be displayed in detail in the "Example" window, which appears just above the "Matching Instances" window.

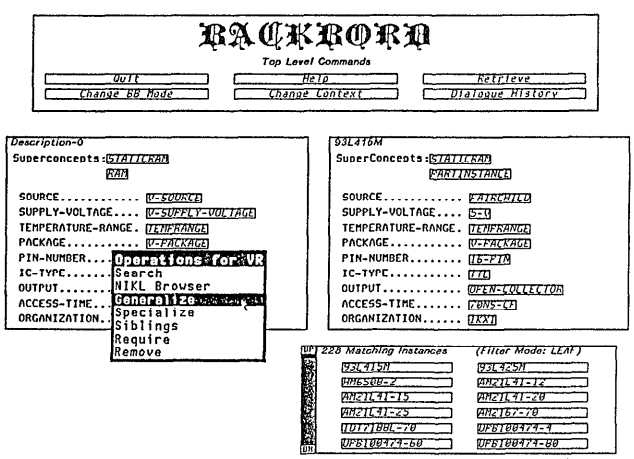

Figure 3-1: A BACKBORD screen for database browsing

[2]The specification is actually a special structure that allows disjunctive value restrictions and other facilities that are not available in the current version of NIKL.

The specification by reformulation paradigm as implemented in BACKBORD consists of operations on the specification and matching objects.

3.2.1. Operations on the Specification

BACKBORD allows the user to modify the superconcepts and value restrictions of the specification. These modifications take advantage of the fact that all superconcepts and value restrictions are concepts in the knowledge base. All such concepts will henceforth be referred to as *specification concepts*. The operations provided for modifying specification concepts are:

Generalize, Specialize, and *Siblings:* These options allow the user to replace a reference to a specification concept by a reference to one or more of its immediate parents, children, or siblings in the concept taxonomy.

NIKL Browser: This invokes the ISI Grapher [Robins 87] to give the user a graphical display of the knowledge base relative to the chosen specification concept. The user may then choose any concept from the graph to replace the specification concept.

Search: The search option creates a recursive call to BACKBORD with the specification concept as the starting specification.

3.2.2. Operations on the Matching Objects

The matching objects are used as prompts for things that should (or should not) be in the specification. One can modify or add a role or a superconcept to the specification by transferring the value from a matching object. One can also perform any of the operations described for specification concepts on a concept that serves as a value restriction or superconcept for a matching object.

4. Example Applications of Specification by Reformulation

Once we have implemented the specification by reformulation paradigm, building applications on top of it consists of developing customizations or extensions to the representation of the specification and/or the functions that implement feedback and guidance. The following are examples of applications in the BACKBORD system developed using this methodology.

4.1. Database Retrieval

When using BACKBORD for database retrieval, the specification represents a query to the database. The matching objects represent database instances that would be retrieved using the current specification. The specification is refined by selecting roles and role values from the matching instances, until it retrieves the instances that are desired.

This is very similar to retrieval by reformulation systems such as ARGON and RABBIT. The main difference between the database capabilities of BACKBORD and retrieval by reformulation systems is that the retrieval by reformulation systems worked in a mostly bottom up fashion. Although such systems created an internal knowledge base representation for the data being retrieved, the user was never able to explicitly view the structure of that representation. Thus, the feedback in these systems all came from the prompts provided by matching instances. In addition to this type of feedback, BACKBORD makes possible a top down manner of specification. The user is able to view the hierarchical representation of the database. In this way the user can modify the specification based on feedback from matching examples (bottom up) and by specializing the type of object that is being searched for (top down).

4.2. The Mail Interface

ISI's FAST project [Neches 88] provides price quotes and handles purchase requests for electronic parts via computer mail. The BACKBORD mail interface to FAST (see Figure 4-1) helps the user construct messages for part quotations and orders.

In the mail interface, the specification represents a message being constructed by the user and the matching instances are examples of previously completed messages that can be used to help construct the current message. The following describes the steps that were necessary in order to build the mail interface using the specification by reformulation paradigm:

4.2.1. A Model of Legal Specifications

New concepts describing the hierarchy of message types must be entered into the knowledge base. The concept (in this case **Message**) that should serve as the starting point when entering the mail interface must also be specified.

4.2.2. Feedback Procedures

The most important method of feedback is the retrieval of objects that classify under the specification. From these objects, the user can find a message or messages similar to the one being created. The default commands for manipulating examples allow the user to utilize information in the examples to modify the new message, e.g., by copying a field.

A type of feedback that is specific to the mail interface is the ability to compare the specification to valid message classes and advise the user on how to change the specification in order to classify it into a message type that is valid to send.

4.2.3. Associations Between Actions and Concepts

The actions associated with the mail interface concepts are sending and incorporating mail messages. This was the primary effort involved for this application. It

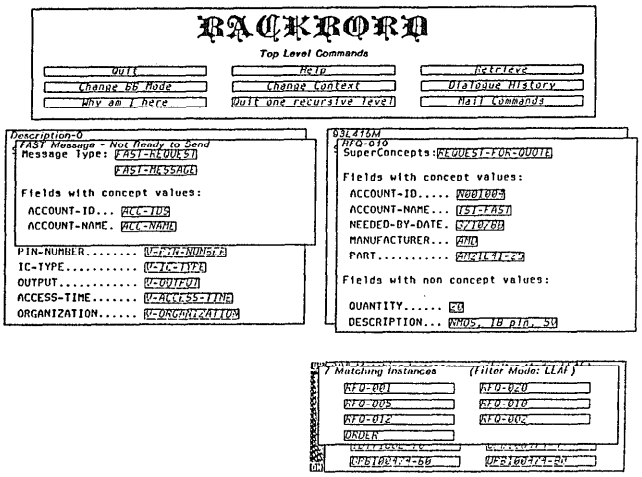

Figure 4-1: A BACKBORD screen for the mail interface

consisted of interfacing the mail concepts to the Unix mail facility. This included converting concepts to messages and messages to concepts and executing Unix commands (such as Send and Inc) from the Lisp Workstation. All of this is transparent to the user.

The following describes some of the advantages of building the mail interface using the specification by reformulation paradigm:

4.2.4. Integration

The mail interface illustrates the integrated user support environment concept. It is entered as a result of the user locating a **Part**, using BACKBORD's database browsing capabilities, and deciding to send a message to the FAST broker regarding the **Part**. Thus, the roles of the message specification that refer to the part (the **Part** and the **Manufacturer**) will by default be filled in with values from the **Part** found by the database query.

4.2.5. Consistent Interface

As can be seen from figure 4-1 the mail interface is very similar to the browsing interface (figure 3-1). The operations for constructing a mail message are also similar to those for constructing a query. All of the operations described in section 3.2.1 can be used to construct a message.

4.3. The Notecards Interface - TINT

The Intelligent Note Taker (TINT) enables users to create notes, attach notes to knowledge base objects, and retrieve notes relevant to an object. Notes are classified into a note taxonomy in the knowledge base. A detailed discussion of TINT can be found in [Harp 88]. For the note creation task, the specification represents a new

note to be created. The procedure for creating a note is almost identical to that for creating a mail message. Just as in the mail interface, the user must decide the type of note to create (using a taxonomy of note types) and correctly instantiate the fields of the note using objects from the knowledge base.

5. Conclusion

This paper has described specification by reformulation, an interface paradigm that extends the retrieval by reformulation paradigm into activities beyond query-based retrieval. Examples of the use of the paradigm were shown through BACKBORD. BACKBORD is an example of an integrated user support environment - a modular set of tools that naturally interact with each other. Such an environment has the following advantages:

Integration and Consistency: The user is provided with one environment with the same interface conventions and with integrated capabilities.

Reusability and Ease of Maintenance: Modules that accomplish very different tasks can all be based on the specification by reformulation paradigm. By having a general module that captures the user interface paradigm, the code size and maintenance problems for these application modules is significantly reduced.

BACKBORD is a domain-independent shell for information systems. In essence, BACKBORD is analogous to an expert system shell in that it separates domain-independent components from domain-specific feedback and actions, just as expert system shells separate a domain-independent inference engine from domain-specific rule bases. The mechanisms that implement these facilities make BACKBORD the beginning of an integrated user support environment. By building the appropriate knowledge base and adding domain-specific feedback and actions, system builders can extend BACKBORD to their own applications.

Acknowledgments

We would like to thank Brian Harp, John Granacki, and Paul Rosenbloom for their comments on earlier drafts of the paper.

References

[Harp 88] Harp, B. & Neches, R., "A Knowledge-based Notecard Environment," in *Proceedings of Workshop on Architectures for Intelligent Interfaces: Elements and Prototypes*, Monterey, California, March 1988.

[Moser 83] M.G. Moser, "An Overview of NIKL, the New Implementation of KL-ONE," in *Research in Natural Language Understanding*, Bolt, Beranek, and Newman, Inc., Cambridge, MA, 1983. BBN Technical Report 5421.

[Neches 88] Robert Neches, *FAST Workstation Project Overview*, USC/Information Sciences Institute, Technical Report ISI/RS-88-203, December 1988.

[Patel-Schneider 84] P. F. Patel-Schneider, R. J. Brachman, and H. J. Levesque, "ARGON: knowledge representation meets information retrieval," in *Proceedings of the First Conference on Artificial Intelligence Applications*, Denver, Colorado, December 1984.

[Robins 87] Gabriel Robins, "The ISI Grapher: a Protable Tool for Displaying Graphs Pictorially," in *Symboliikka '87*, Helsinki, Finland, August 1987. reprints available through USC/ISI technical report ISI/RS-87-196

[Schmolze and Lipkis 83] James Schmolze and Thomas Lipkis, "Classification in the KL-ONE Knowledge Representation System," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, IJCAI, 1983.

[Stelzner 86] Marilyn Stelzner, Michael D. Williams, *Specification by Reformulation, An Approach to Knowledge Based Interface Design*, Intellicorp, Mountain View, CA, 1986.

[Tou 82] Tou, F.F., M.D. Williams, R, Fikes, A. Henderson, and T. Malone, "RABBIT: An Intelligent Database Assistant," in *Proceedings AAAI-82*, pp. 314-318, 1982.

[Williams 81] M. D. Williams and J. D. Hollan, "The process of retrieval from very long term memory," *Cognitive Science* 5, 1981, 87-119.

[Williams 84] M. D. Williams, "What makes RABBIT run?," *Int. J. Man-Machine Studies* 21 , 1984, 333-352.

[Yen 88] John Yen, Robert Neches, Michael Debellis, "Backbord: Beyond Retrieval by Reformulation," in *Proceedings of the Workshop on Architectures for Intelligent Interfaces, Monterey, California*, 1988.