IMPULSE: A Display Oriented Editor for STROBE

Eric Schoen *Reid G. Smith*
Schlumberger-Doll Research
Old Quarry Road
Ridgefield, Connecticut 06877

ABSTRACT

In this paper, we discuss a display-oriented editor to aid in the construction of knowledge-based systems. We also report on our experiences concerning the utility of the editor.

1. Introduction

There is by now wide experience with construction of knowledge-based systems. Several authors have emphasized the importance of powerful tools for creation, modification, and maintenance of knowledge bases and related code (e.g., [Buchanan, 1982]). IMPULSE, a display oriented knowledge base editor, is one such tool. It provides a convenient user interface to the STROBE [Smith, 1983] structured object programming system running in Interlisp-D on the Xerox 1100 series scientific workstations.

In designing and implementing IMPULSE, we were concerned with meeting the following goals:

- Taking full advantage of the underlying knowledge representation language.
- Providing flexible tools for visualizing the potentially complex and varied hierarchical structurings of a knowledge base.
- Making full use of Interlisp-D's graphical facilities (windows, menus, bitmaps, data inspectors, etc.) to present information and commands in a form which both experts and novices can use comfortably, and to minimize keystrokes and other repetitive or unnecessary user operations.
- Preserving context while editing several parts of a knowledge base or knowledge bases simultaneously.
- Preventing the user from causing unintentional damage to a knowledge base.
- Furnishing tools for managing and editing multiple knowledge bases.

2. A Brief Overview of STROBE

STROBE is designed to be a data structuring and control *tool* built on top of Interlisp. Unlike LOOPS [Bobrow, 1982], which provides a syntax to suppress Lisp, STROBE supplies only a kernel of Lisp functions to manage inheritance, message-passing, and knowledge base construction for the Lisp programmer. The data-structuring components of STROBE are knowledge bases (KB's), objects, slots and facets.[1] Its inheritance mechanism, implemented at the facet level, supports multiple hierarchies. In addition to message-passing, STROBE allows procedure activation in conjunction with several types of data access and alteration (similar in concept to the Interlisp *Advise* facility).

3. The IMPULSE User Interface

The naked STROBE interface is intended for program-level interaction with knowledge bases. We felt that IMPULSE needed to provide a rich set of support functions to simplify the task of the knowledge base builder. While several character-oriented knowledge base editors are available (e.g., UNITS [Smith, 1980]), a serious limitation of these editors is their inability to display multiple editing contexts concurrently. A display oriented editor allows the builder/maintainer to edit simultaneously in as many windows as can be displayed. IMPULSE is implemented as four distinct levels: a top-level knowledge base manager, a knowledge base editor, an object/slot editor, and a facet editor. Each level consists of an editor window and one or more associated command menus. Any number of editor windows may be open simultaneously.

The **knowledge base manager** deals with KB's as indivisible entities. The manager presents a menu of loaded knowledge bases, which allows the user to select a specific knowledge base to be edited, stored, or copied. Additional commands in the associated menu allow new knowledge bases to be loaded, or created. The *Settings* command permits the user to modify the values of several STROBE global flags.

The **knowledge base editor** allows the user to modify the structure of an individual knowledge base. The information window provides an overview of the KB. One associated menu provides a set of KB-oriented commands: creating and editing objects, deleting the knowledge base, etc. The other menu is a list of objects in the KB. The user selects an entry from the object menu, and then selects a command to perform an operation on the selected object.

1. A knowledge base is made up of a number of interrelated objects. The objects encode *packets* of knowledge. The characteristics of an object and its links to other objects are encoded as a number of slots. The slots themselves have structure--facets--that can be used for annotatation.

Object and slot edit commands are issued from command menus associated with the **object/slot editor**. Its window contains a structured printout of an object's contents. The slot name captions in the bold font are *active regions* sensitive to picks with the mouse. Picking a slot caption inverts it and sets that slot as *current* for slot editing commands. The IMPULSE **facet editor** provides access to the contents of each facet in a slot. Facets are the least structured components in STROBE, and thus the data structures most like basic Interlisp datatypes. For this reason, the IMPULSE facet editor serves mainly as a link to the Interlisp-D data inspector [Burton, 1982].

One additional major component has proved very useful. Using the Interlisp-D Grapher package [van Lehn, 1982], tree and graph hierarchies can be drawn to aid in visualizing KB structure. The *Ancestry* and *Progeny* commands in the object/slot editor draw generalization and specialization trees. The *Display Slot Succession* command traces a tree of objects containing the selected slot name and pointed to by those slots. The nodes of these trees are object names; selecting a node starts an object/slot editor on the corresponding object. The *KB Struct. Graphs* command allows a user to supply arbitrary tree generating functions (e.g., to display object relationships in *parts* hierarchies); the system associates these functions with the knowledge base for menu selection in later sessions.

Figure 1 shows IMPULSE running on the Xerox 1100. The knowledge base manager (the small window at the upper right entitled "STROBE") indicates two loaded knowledge bases. The *Dipmeter Advisor* KB is selected, and its corresponding KB editor appears on the left side of the screen. The graph at the lower left side of the screen is a tree of the *Tectonic-Feature* object's progeny. Bold face nodes are class objects; normal face nodes are individuals.

Within *Dipmeter Advisor*, the *Fault* object has been selected for editing; its edit window appears at the lower right. The *Picture* slot of the *Fault* object is currently selected in the object editor window, and is expanded in the facet editor on the left. Note that the object/slot editor view of *Fault* is the bitmap, while the facet editor view is its Interlisp *print name*.

The *Late-Fault* object is displayed in an edit window above the *Fault* object. Slot names followed by uparrows (pointers) are slots whose values are obtained by STROBE via inheritance.

IMPULSE provides two means to aid the user in managing and conserving screen space. Any IMPULSE window can be shrunk to its *icon* (such as the two small windows above the KB manager in Figure 1). Also, IMPULSE will not create a menu of object names which extends below the bottom of the window with which it is associated; instead, large object menus are limited in size and made scrollable.

4. User Assistance in IMPULSE

Work of this type would ordinarily involve a tremendous amount of (error-prone) typing. For the IMPULSE user, the need for typing has been vastly reduced by an extensive set of menu commands. For those operations which absolutely require typein, we have supplied "smart" typein routines with command completion, partial name recognition, and spelling correction. For instance, when creating a new object, the user needs to supply a list of parent objects, whose names need be known only partially.[2]

2. The typein routines rely on the Interlisp TTYIN package [van Melle, 1982].

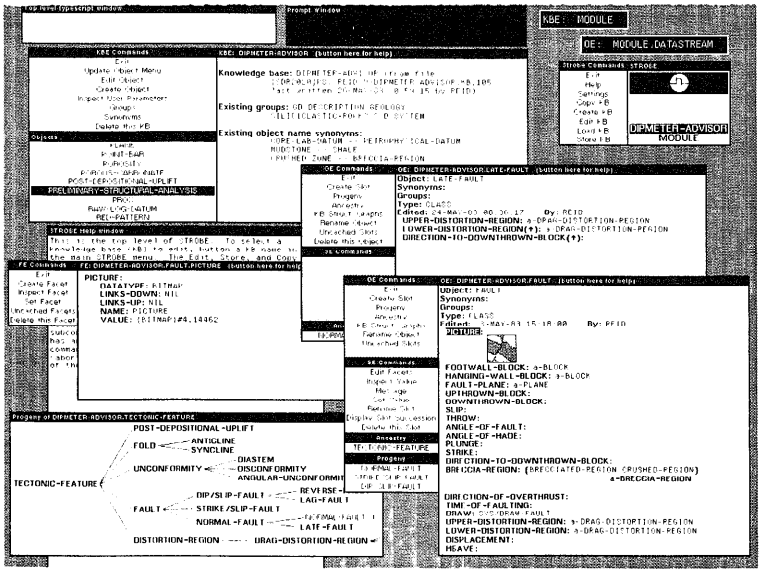

Figure 1. Stratigraphic Analysis Interaction

We felt it necessary to keep the user interface simple, and therefore reduced the size of command menus as much as possible. As a result, a number of menu entries have subcommands: when picked with the left mouse button, they cause the indicated command to be executed; when picked with the middle button, they bring up a submenu. For instance, the *Create Object* command normally prompts the user for the information necessary to create a STROBE object. Its subcommand menu gives the choice of creating an object of prespecified type—*class*, *individual*, or *description*—thus avoiding that question in the object creation dialogue. Other menu entries have more powerful subcommands: the *Ancestry* command has subcommands to allow interactive editing of an object's *generalizations* slot.

Another important feature is flexibility in selecting an object for editing. In IMPULSE, there are three such mechanisms:

- From the KB editor menu;
- From nodes in Grapher trees,
- From the direct progeny/ancestry menus associated with object/slot editors.

We have provided an on-line help mechanism for all editor levels. Each editor window title bar is sensitive to mouse button selections. Whenever the user buttons in this region, a pertinent help message is displayed in the IMPULSE help window, a scrollable window whose size and screen position is under user control.

Finally, IMPULSE tries to keep the user from causing inadvertant damage to a knowledge base. Requests to delete facets, slots, objects, or knowledge bases *must* be confirmed. When performing STROBE functions whose behavior is dependent upon the settings of argument flags (e.g. *RENAMEOBJECT* and *RENAMESLOT*), IMPULSE queries the user for the desired behavior.

5. Conclusion

We believe we have met the goals we set for IMPULSE. The editor has proved very simple to use. Although the user community is on the order of a dozen computer scientists, no user manual has been needed; instead, comprehensive on-line documentation guides the novice user. Interaction with the editor is rapid: keyboard input is required only where unavoidable, and is always backed up with spelling correction and partial name recognition.

IMPULSE has now been used to build several large knowledge bases (containing up to 700 objects). In each case, its abilities to display the structure of the knowledge base in a variety of ways, and to provide flexible access to knowledge base components have greatly reduced the magnitude of the task.

In addition to their utility to the builder/maintainer of knowledge bases, editors like IMPULSE can assist in the transfer of expertise from a domain expert to a program. This currently involves a computer scientist intermediary (*knowledge engineer*). One of the most useful roles played by the intermediary is to help provide a logical organization for the knowledge of the domain expert. This assistance is typically provided via many interactions. For each interaction, the intermediary gathers some understanding of a portion of the expert's knowledge, encodes it in a program, discusses the encoding and the results of its application with the expert, and refines the encoded knowledge. Discussion and refinement is facilitated when the knowledge is encoded in domain-specific terms, and when it is presented in forms familiar to the domain expert. Our early experience with IMPULSE is that its ability to simultaneously display different views of a knowledge base and its characteristic immediate feedback have enhanced interactions with our domain experts.

Acknowledgements

We greatly appreciate the contributions of Dave Barstow, Stephen Smoliar, Gilles Lafue, Stan Vestal, Tony Passera, and Scott Marks. Their valuable suggestions made in the course of testing early versions of IMPULSE guided much of our development effort. This paper has benefited from comments by Barstow and Brad Cox.

REFERENCES

D. G. Bobrow and M. J. Stefik, *A Virtual Machine for Experiments in Knowledge Representation.* Unpublished Memorandum, Xerox Palo Alto Research Center, April 1982.

B. G. Buchanan, New Research On Expert Systems. In J. E. Hayes, D. Michie, and Y-H Pao (Eds.), *Machine Intelligence 10.* New York: Wiley & Sons, 1982, pp. 269-299.

R. R. Burton, *The Data Inspector.* Interlisp-D Documentation Series, Xerox Palo Alto Research Center, March, 1982.

R. G. Smith, STROBE: Support for Structured Object Knowledge Representation. *Proceedings of the Seventh International Joint Conference on Artificial Intelligence,* August, 1983.

R. G. Smith and P. E. Friedland, *Unit Package User's Guide.* DREA Technical Memorandum 80/L, December 1980. Also published as HPP-80-28, Heuristic Programming Project, Stanford University.)

W. Teitelman, *Interlisp Reference Manual.* Xerox Palo Alto Research Center, October 1978.

K. van Lehn, *Grapher Documentation.* Interlisp-D Documentation Series, Xerox Palo Alto Research Center, September 1982.

W. van Melle, *TTYIN—A Display Typein Editor.* Interlisp-D Documentation Series, Xerox Palo Alto Research Center, June, 1982.