NATURAL LANGUAGE IN A DESKTOP ENVIRONMENT

Proceedings of HCI89, 3rd International Conference on Human-Computer Interaction

Marilyn A. Walker

Hewlett Packard Laboratories
Filton Rd., Stoke Gifford
Bristol, England BS12 6QZ, U.K.
lyn%lwalker@hplb.hpl.hp.com

Abstract

Different modes of interaction are better suited for different tasks and both natural language and direct manipulation have strengths and weaknesses as interface technologies. Previous studies in the evaluation of interactive natural language interfaces have noted certain benefits of natural language access to information, but have neglected to describe the features of natural language that provides these benefits. We have identified a set of communicative features of natural language, such as quantification and discourse reference, that are difficult to support with direct manipulation. The analysis has implications for determining the tasks in a personal information environment that natural language is suited for.

1 Introduction

In the human-computer communication community, some have touted natural language interaction (NLI) as a critical component of future interfaces. They assume, in the main, that some users cannot or will not learn a formal method of interacting with a system; Natural Language interfaces are targeted on these users. Within the community of natural language researchers, the utility of natural language as an interface technology is rarely questioned. The evaluation of systems has focused on comparisons of the input-output characteristics of one natural language interface compared with another or on the limitations of current NLI systems versus the full range of human conversation (Wahlster, 1986; Cohen et al., 1982; Petrick, 1976; Woods, 1977).

In contrast, proponents of alternatives to natural language have doubts about both the feasibility and the desirability of NLI systems. Shneiderman states that since computers display information 1000 times faster than humans can enter commands, it is reasonable to just put more information on the screen and let people look for the piece that they need

((Shneiderman, 1986), p. 165). He gives us reports of user's 'joy' in using a direct manipulation interface to a system, of feelings of control and mastery. His favourite example of direct manipulation is the driving of an automobile and he cites the ridiculousness of an interface to the operations of driving that requires the user to say *Now left 30 degrees.* Similarly, the clumsiness of a word processor where every command has to be executed via an English specification is apparent. These examples are well-chosen and situation specific. However, the awkwardness of searching through a hierarchy of directories and files to find one with a particular known attribute is not commonly recognized. To the contrary, Shneiderman observes that the process of looking for an object in a container hierarchy may make users feel as though they have more control over the system, because the search is accomplished in small steps and the user can backup at any time.

Since there are no studies to date of the use of natural language in a typical personal information environment such as the Xerox Star or the Apple Macintosh, we lack functional comparisons. Most of the reports of benefits of one interaction style versus another are anecdotal. This has made the debate of natural language versus direct manipulation more religious than scientific.

Our hypothesis is that different modes of interaction are better suited for different tasks and that natural language and direct manipulation both have strengths and weaknesses as interface technologies. Whereas the case for direct manipulation has been well argued (Shneiderman, 1986; Hutchins et al., 1986), we would like to know which aspects of natural language provide the greatest benefit. The question is what communicative features provided by natural language are useful in an application, or alternatively, for what tasks in an application is natural language a suitable interface and for what tasks might we want to use a different interaction mode such as direct manipulation.

If one is willing to imagine a personal information environment with enough underlying knowledge about its own objects and operations to enable search on the items in a file or mail system (Malone et al., 1987), then since files and mail folders are personal databases, studies on NLI for database query may be relevant. We can classify these (as per (Whittaker and Stenton, 1989)) as studies on (a) LEARNABILITY (Kelly and Chapanis, 1980; Hendler, 1983; Ogden and Brooks, 1983), (b) COVERAGE (Diaper, 1986; Guindon, 1988; Whittaker and Stenton, 1989; Ogden and Sorknes, 1987; Kelley, 1983), and (c) COMPARISON (Jarke et al., 1985; Small and Weldon, 1983; Walker and Whittaker, 1989).

The most relevant of these studies are COMPARISON studies, which address the question of how NLI compares with another interface for a particular task[1]. In most of these, subjects try to complete some well-defined task that is supposedly typical of the domain. One result of these studies is that researchers have noted the efficiency of natural language (Jarke et al., 1985), but they do not say why it takes fewer input tokens, that is they do not provide an analysis of the **features** that NLI contributes to the solution of the given task in the domain. In addition, these studies generally report on the number of queries per task, how much time per query, how many errors, or time-to-solution for a task, but it is difficult to interpret such global measures. First, these measures conflate error analysis

[1]Many researchers have noted that there might be a trade-off between the learnability of an interface and the complexity of the operations that are supported well by that interface. The assumption that NLI is learnable, and indeed is nearly as learnable as a direct manipulation interface, is implicit in advocating the potential benefits of NLI in a personal information environment. Coverage studies are not relevant for our purposes since they do not allow comparisons across different interface modalities.

with the analysis of benefits, since if it takes longer to do a task using a particular interface modality, we cannot tell whether it was repeated error or lack of support for operations that contributed to the total time. Second, we are unable to assess how the task determines the underlying operations and which operations are in some sense independent of the task.

Therefore we propose a set of linguistically-based features which correspond to operations one might want to perform to accomplish a task. The existence of such a feature set give us one way to compare difference interfaces as well as a framework within which we can analyse and design experimental tasks. This feature set is independent of any particular application or domain, but we have ignored linguistic features that seem to be irrelevant to the tasks in a typical personal information environment where direct manipulation has come to be the preferred style of interface.

2 Communicative Features

One of the benefits of direct manipulation is that 'what you see is what you get', i.e. its very directness, but some researchers might say that 'what you see is **all** you get' (Hulteen, 1988). Language takes us out of the realm of the here and now, visually present. Some of the communicative features have to do with being able to select an item with language rather than directly. This ability comes in handy for finding items, rather than manipulating them once they are already found. Some of the features relate to the flexibility of the interaction with the system, and as well to the forms in which items are presented to the user. In addition, we claim that the interplay of the features contributes to the efficiency of natural language that has been noted in studies of NLI for database query. The list of features given below is not claimed to be an exhaustive one, but neverthless is a beginning of a taxonomy of such communicative features.

2.1 The features

The features are:

- DEFINITE DESCRIPTION: Finding something with a particular attribute is a type of **reference** problem. Language allows us to access an object, such as a file, via a description. We don't have to be able to see it. DEFINITE DESCRIPTION allows one to refer to objects by their attributes , e.g. *the file whose write date is 8/15/88*, instead of searching for it. We say the object is REALIZED (Grosz et al., 1983) by its description.

- DISCOURSE REFERENCE: The use of a pronoun such as *they, it, their, him* picks out something already in context. This allows one to refer to previously evoked objects. For instance, we have *Create a file named Jazz. Send IT to Steve.* It gives us a convenient shorthand for referring. By allowing questions to range over a set of objects gathered as the result of some previous command, e.g. *Get me the messages from Eric. Are any of THEM about today's meeting?*, discourse reference allows one to point to more than one object at a time, using language, and the object only needs to be realized, not visible.

- TEMPORAL SPECIFICATION: Temporal specifications allow one to specify that an action, corresponding to a series of mouse clicks, should take place at some time in the future, or repetitively. So, one might say, *Send CHI-draft to Jarrett ON MONDAY* or *Send a copy of the proposal to Susan EVERY WEEK.*

- QUANTIFICATION: Quantifiers are words like *all, every, each, any, some, most, how many, how much.* Quantifiers often let a user get information about the state of the desktop without pulling up items to the top of the desktop; they can probe this state as well as operate on it. For example, *HOW MANY folders named CHI-draft are there?*, or *Are there ANY files whose write date was yesterday?*. Quantifiers like *all* and *every* support operations on sets of items, e.g. *Send a copy of proposal-1 to EVERY person in the dialogue project*, instead of wading up and down a menu hierarchy or scrolling through a mail browser of messages and visually checking or performing some operation on each one.

- COORDINATION: This is exemplified by the use of words such as *and* and *or.* User can specify the same operation on more than one entity at a time, e.g. *Send the ACL-draft AND Phil's message to Steve*, or specify different operations on the same entity, e.g. *Latex file Jazz AND print it.* Coordination can be used to implicitly create sets of objects that do not share any obvious attributes other than that the user chose to group them.

- NEGATION: Negation is exemplified by words like *not, except* and *without.* An example might be *Are there any messages I haven't answered?*. It allows one to collect desktop items that do NOT have a particular attribute or to exclude particular items from a set.

- COMPARATIVES: Comparatives come in two types and allow users to make comparisons. These comparatives may range over attributes of desk-top objects, as in *Are there any messages OLDER than two months?*, or they may pertain to quantities of the objects themselves, e.g. *Are there MORE THAN two messages from Eric?.*

- SORTING EXPRESSIONS: These are often present in language by the use of *by* phrases, such as *by date* or *by topic*, For example, *Show me the messages in my in-tray by topic.* This allows one to display items in useful ways.

2.2 Features discussion

2.2.1 Here and Now

It is difficult to imagine any way to duplicate some of these features by using a combination of icons and mouse clicks. Features like DEFINITE DESCRIPTION let us operate on items that are not **here**, whereas TEMPORAL SPECIFICATION lets us specify actions that are not intended to be executed **now**. These are directly contrary to the three principles of direct manipulation, (a) continuous representation of the objects and actions of interest, (b) physical actions or labeled button presses instead of complex syntax, (b) rapid incremental reversible operations whose impact on the object of interest is immediately visible ((Shneiderman, 1986), p. 201).

Visual inspection is the only way to perform some of these operations in a direct manipulation environment. Visual inspection can replace features such as DEFINITE DESCRIPTION, NEGATION and COMPARATIVES, once we have found the relevant set of items and brought them to the surface of the desktop. We can visually check whether the item satisfies some combination of attributes that might be specified using one of these features. What visual inspection does not help us do is to find particular items according to their attributes.

2.2.2 Limited Support

A limited version of some features are provided by other interfaces. Some systems may provide limited notions of these communicative features. For instance, the lack of any type of DEFINITE DESCRIPTION was so bothersome to users of the Xerox Star system, that the reference icon, providing special support for one facet of this feature, was added to forestall one from having to look through a series of file drawers and folders to access a file when one knew exactly where it was stored(Rosenberg, 1988). A reference icon served as a pointer whose referent could be specified simply by typing its full pathname. Opening the reference icon then showed the contents of the referent. However, pathnames and names are restrictive ways of specifying a definite description since these ways of referring only access two of the possible attributes, ie. name and location, that might be associated with an object.

Another example is that a default for SORT EXPRESSIONS is usually provided by many kinds of file browsers or other modes of display, normally being chronological or alphabetical order. If we represent other kinds of information about the items in our desktops then we may want to view them one way one time and another way the next. Since any type of definite descriptor can be used in combination with a sorting operator such as *arrange*, a default presentation may not be adequate.

Finally, the facility to select and manipulate sets of items rather than single items, such as that provided by COORDINATION, some kinds of QUANTIFIERS and some DEFINITE DESCRIPTORS is usually available in particular applications like paint or draw programs, where one can group two items and operate on both at the same time.

2.2.3 Combination

Combining the features together provides the greatest increase in functionality. Often it is the presence of one that pushes the utility of another over some critical threshhold. There is a synergistic effect. For instance, without the facility to collect a group of objects via a description, DEFINITE DESCRIPTION, the ability to quantify over them, QUANTIFICATION, would not be as useful, e.g. *Send a copy every document in the CHI folder to Steve* needs both the use of *every* and the ability to specify documents described as a *document in the CHI folder*.

COORDINATION not only allows one to create adhoc sets of objects, but it also shows up in combination with the other features such as definite descriptions and sort expressions. The coordination of definite descriptions can give us multiple keys to access a document

by. User studies have shown this to be important (Malone, 1983). The coordination of sort expressions gives us an ever increasing ability to view a set of objects in a structure that is useful to us, e.g. *by topic and by date*.

2.2.4 Summary

In summary, direct manipulation gives us (a) visual inspection as the only way to perform some of these operations, (b) a limited version of others, (c) no way at all to do others. In addition, the combination of the communicative features in a flexible manner gives the greatest benefits. Some of the features are mainly about SELECTION, and obviously for single visible items, pointing is the ultimate in specificity. However, COORDINATION, some kinds of QUANTIFIERS and some DEFINITE DESCRIPTORS allow selection and manipulation of sets of items. In addition, if we have situations where there is too much information or it is not easily found, then the functionality of just getting exactly the right set of objects onto the desktop in the first place is extremely useful. However, even if the greatest payoffs of NLI are not in selection, we would like to test whether the operations provided by COMPARATIVES, DISCOURSE REFERENCE, COORDINATION, SORTING EXPRESSIONS, TEMPORAL SPECIFICATION and QUANTIFICATION, which are not just about selection, are useful. These might be features that we would selectively add to direct manipulation based environments for the greatest benefits.

3 Conclusion

We would like to validate the utility of these features, in particular to determine if the quantity of desktop items is what makes a qualitative difference, i.e. how much clutter on a computer desktop can one deal with?

If we use these features to design experimental tasks then we may avoid some errors in task design. For instance, sometimes the goal in task design is to avoid biasing a user's syntactic constructions, but we need to be careful about the other effects this might have. For example, a task that requires subjects to ask questions that will allow them to fill in the missing values on the form may eliminate NEGATION because we are only interested in positive values. It may also eliminate QUANTIFICATION, COMPARISON and COORDINATION because we are interested in attributes of just one object. In a comparison of NLI with the database query language SQL, Small and Weldon restricted NLI by specifying that the English queries must specify ALL the information given in a SQL query (Small and Weldon, 1983). For instance, subjects were not allowed to assume the STAFF table was the one to use just because it was the only one with an AGE column, so that *Find the doctors whose age is over 35*, would result in an error. This type of restriction automatically limits the use of DEFINITE DESCRIPTIONS, DISCOURSE REFERENCE, NEGATION, and COMPARATIVES.

We might expect certain features to only be elicited by certain tasks or in particular environments. For instance, a feature like SORTING EXPRESSIONS is not likely to be frequently used if the desired result of the experimental task is completely specified because subjects will have no motivation for looking at the data in different ways. It might also be that frequent use of SORT only shows up in field studies, since in that case we could expect

subjects to be really interested in the data in order to satisfy some higher level goal.

These features may also provide a framework for the analysis of the results of empirical studies. In a field study that we conducted, we analysed user's interactions with a database using some of the same features presented here (Walker and Whittaker, 1989). We found these features to be a useful way of categorising the data since we were able to show that the users who persisted with an error-prone system were using it in a different way, i.e. exploiting different features, than those who gave up on using the system.

These communicative features are worth cataloguing in order to guide future research in natural language, since it must be admitted that not all of the features are available in their full generality in any existing NLI interface. But if we are willing to combine language with other interfaces we may find a useful way to extract benefits from what is already available, and avoid potential pitfalls. These features give us a way of thinking about the functionality described, and other ways we might provide it. The future of interface technology is surely in mixed media; we need to have a way to talk about the features that iconic interfaces provide and we should develop those aspects of linguistic interfaces that can give us the greatest benefit.

4 Acknowledgements

Jarrett Rosenberg, Phil Stenton, Steve Whittaker, Eric Hulteen, Peter Williams, Andy Hunter and Nick Haddock have all provided thoughtful commentary on various versions of this paper. Some of the ideas presented here were developed while working with the Natural Language project at HPLabs Palo Alto, Ca.

Bibliography

Phillip R. Cohen, C. Raymond Perrault, and James F. Allen 1982. 1982. Beyond question answering. In Wendy Lehnert and Martin Ringle, editors, *Strategies for Natural Language Processing*, pages 245–274. Lawrence Erlbaum Ass. Inc, Hillsdale, N.J.

Dan Diaper. 1986. Identifying the knowledge requirements of an expert system's natural language processing interface. In M.D. Harrison and A.F. Monk, editors, *People and Computers, Designing for Usability*, pages 263–280. Cambridge University Press, Cambridge, U.K.

Barbara J. Grosz, Aravind K. Joshi, and Scott Weinstein. 1983. Providing a unified account of definite noun phrases in discourse. In *Proc. 21st Annual Meeting of the ACL, Association of Computational Linguistics*, pages 44–50.

Raymonde Guindon. 1988. How to interface to advisory systems? users request help with a very simple language. In *Proc. Annual Meeting of the Computer Human Interaction of the ACM*, pages 191–196.

James A. Hendler. 1983. The effects of limited grammar on interactive natural language. In *Proc. Annual Meeting of the Computer Human Interaction of the ACM*, pages 190–192.

Eric Hulteen. 1988. October. Personal Communication.

Edwin L. Hutchins, James D. Hollan, and Don A. Norman. 1986. Direct manipulation interfaces. In Don A. Norman and Stephen W. Draper, editors, *User Centered System Design: New Perspectives on Human-Computer Interaction*. Lawrence Erlbaum Associates, Hillsdale, New Jersey.

Matthias Jarke, Jon A. Turner, Edward A. Stohr, Yannis Vassiliou, Norman H. White, and Ken Michielsen. 1985. A field evaluation of natural language for data retrieval. *IEEE Transactions on Software Engineering*, SE-11, No.1:97–113.

J. F. Kelley. 1983. An empirical methodology for writing user-friendly natural language computer applications. In *Proc. Annual Meeting of the Computer Human Interaction of the ACM*, pages 193–196.

M.J. Kelly and A. Chapanis. 1980. Limited vocabulary natural language dialogue. *International Journal of Man-Machine Studies*, 6:71–86.

Thomas W. Malone, Kenneth R. Grant, Franklyn A. Turbak, Stephen A. Brobst, and Michael D. Cohen. 1987. Intelligent information-sharing systems. *Communications of the ACM*, 30(5):390–402.

Thomas W. Malone. 1983. How do people organize their desks? implications for the design of office information systems. *ACM Transactions on Office Information Systems*, 1(1):11–112.

William C. Ogden and Susan R. Brooks. 1983. Query languages for the casual user: Exploring the ground between formal and natural languages. In *Proc. Annual Meeting of the Computer Human Interaction of the ACM*, pages 161–65.

William C. Ogden and Ann Sorknes. 1987. What do users say to their natural language interfaces? In *Human-Computer Interaction, Interact 1987, Elsevier Science Publishers B.V. North Holland*, pages 561–66.

S.R. Petrick. 1976. On natural language based computer systems. *IBM Journal of Research and Development*, pages 314–325.

Jarrett Rosenberg. 1988. October. Personal Communication.

Ben Shneiderman. 1986. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison Wesley.

Duane W. Small and Linda J. Weldon. 1983. An experimental comparison of natural and structured query languages. *Human Factors*, 25(3):253–263.

Wolfgang Wahlster. 1986. The role of natural language in advanced knowledge-based systems. In H. Winter, editor, *Artificial Intelligence and Man-Machine Systems*, pages 62–83. Springer, Berlin.

Marilyn Walker and Steve Whittaker. 1989. When natural language is better than menus: A field study. Technical Report HPL-BRC-TR-89-020, HP Laboratories, Bristol, England.

Steve Whittaker and Phil Stenton. 1989. User studies and the design of natural language systems. In *Proc. 4th Conference of the European Chapter of the ACL, Association of Computational Linguistics*, pages 116–123.

William A. Woods. 1977. A personal view of natural language processing. *ACM-SIGART Newsletter*, 61:17–20, February.