

Linköping Electronic Articles in
Computer and Information Science
Vol. 1(1996): nr 1

Customizing Interaction for Natural Language Interfaces

Lars Ahrenberg
Nils Dahlbäck
Arne Jönsson
Åke Thurée

Department of Computer and Information Science
Linköping University
Linköping, Sweden

Linköping University Electronic Press
Linköping, Sweden

<http://www.ep.liu.se/ea/cis/1996/001/>

*Published on October 1, 1996 by
Linköping University Electronic Press
581 83 Linköping, Sweden*

**Linköping Electronic Articles in
Computer and Information Science**
ISSN 1401-9841
Series editor: Erik Sandewall

©1996 *Lars Ahrenberg, Nils Dahlbäck, Arne Jönsson, Åke Thuré*
Typeset by the authors using TeX
Formatted using étendu style

Recommended citation:

*<Authors>. <Title>. Linköping electronic articles
in computer and information science, Vol. 1(1996): nr 1.
<http://www.ep.liu.se/ea/cis/1996/001/>. October 1, 1996.*

This URL will also contain a link to the authors' home pages.

*The publishers will keep this article on-line on the Internet
(or its possible replacement network in the future)
for a period of 25 years from the date of publication,
barring exceptional circumstances as described separately.*

*The on-line availability of the article implies
a permanent permission for anyone to read the article on-line,
to print out single copies of it, and to use it unchanged
for any non-commercial research and educational purpose,
including making copies for classroom use.*

*This permission can not be revoked by subsequent
transfers of copyright. All other uses of the article are
conditional on the consent of the copyright owners.*

*The publication of the article on the date stated above
included also the production of a limited number of copies
on paper, which were archived in Swedish university libraries
like all other written works published in Sweden.
The publisher has taken technical and administrative measures
to assure that the on-line version of the article will be
permanently accessible using the URL stated above,
unchanged, and permanently equal to the archived printed copies
at least until the expiration of the publication period.*

*For additional information about the Linköping University
Electronic Press and its procedures for publication and for
assurance of document integrity, please refer to
its WWW home page: <http://www.ep.liu.se/>
or by conventional mail to the address stated above.*

Abstract

Habitability and robustness have been noted as important qualities of natural-language interfaces. In this paper we discuss how these requirements can be met, in particular as regards the system's ability to support a coherent and smooth dialogue. The discussion is based on current work on customizing a dialogue system for three different applications.

We adopt a sublanguage approach to the problem and propose a method for customization combining bottom-up use of empirical data with a global pragmatic analysis of a given application. Finally, we suggest three design principles that have emerged from our work called the sublanguage principle, the asymmetry principle and the quantity principle.

This is an extended version of a paper with the same title presented at the Workshop on Pragmatics in Dialogue, The XIV:th Scandinavian Conference of Linguistics and the VIII:th Conference of Nordic and General Linguistics, Göteborg, Sweden, August 16-21, 1993.

1 Introduction

Research on computational models of discourse can be motivated from two different standpoints. One approach is to develop general models and theories that apply to all kinds of agents and situations. The other is to develop accounts of specific discourse genres (Dahlbäck & Jönsson, 1992). It is not obvious that the two approaches should produce similar computational theories of discourse and we believe it is important to distinguish the two tasks from each other. Moreover, in the case of dialogues for natural-language interfaces (NLIs), which is our prime concern in this paper, there is not merely the question of modelling some external linguistic reality but also an important element of design, linguistic as well as otherwise.

The following requirements are widely recognized as being important for NLIs.

- habitability: the user should conveniently be able to express the commands and requests that the background system can deal with, without transgressing the linguistic capabilities of the interface (Watt, 1968);
- efficiency: the NLI should not slow down the interaction with the background system noticeably;
- robustness: the system should be able to react sensibly to all input (cf. Hayes & Reddy, 1983);
- transparency: the system's capabilities and limitations should be evident to the user from experience;

In this paper we discuss ways of making teletype natural-language interfaces satisfy these requirements in the context of applications which belong to the domain that Hayes and Reddy (1983) call simple service systems, i.e. systems that “require in essence only that the customer or client identify certain entities to the person providing the service; these entities are parameters of the service, and once they are identified the service can be provided” (*ibid.* p. 252). These systems exhibit the kind of dialogue that Van Loo and Bego (1993) term parameter dialogue.

The requirement that puts the highest demand on linguistic competence is the one concerning habitability. However, habitability does not necessarily imply that the system must understand any relevant request. Where extended linguistic coverage comes in conflict with either robustness, transparency or efficiency, it may be compromised (though cf. Ogden, 1988). This trade-off between requirements reconfirms the need for good design. The importance of habitability, however, suggests that a dialogue system must handle those phenomena that occur frequently in typed human-computer interaction correctly and efficiently, so that the user does not feel constrained or restricted when using the interface. The trade-off implies that the

interface should not waste effort on complex computations in order to handle irrelevant or rare phenomena. For instance, the system need not be able to handle features such as jokes or surprise, when these are not demanded by the purposes of the system.

On these grounds we have adopted a sublanguage approach (Grishman & Kittredge, 1986) to dialogue systems of this kind. All aspects of linguistic communication, including interaction patterns and the use of indexical language is assumed to depend on the application and domain. For this reason the interface system must be designed to facilitate customization to meet the needs of different applications. Moreover, we must find methods that allow us to determine what the needs of a given application are. Such a method, based on Wizard of Oz-simulations, will be outlined below.

The rest of this paper is organized as follows. The next section briefly describes our system and applications, and the ways in which the system can be customized. The following section presents ways of meeting the requirements listed above as they apply to dialogue behaviour. In the third and final section we discuss our solutions from a more general perspective and propose three design principles relating to the notion of sublanguage, the quantity of information, and asymmetries between users and systems.

2 The LINLIN model

The natural language interface LINLIN (Ahrenberg, Jönsson, & Dahlbäck, 1990; Jönsson, 1991, 1993a); is designed to facilitate customization to various applications. Dialogue in LINLIN is modelled using dialogue objects which represent speech acts and speech act sequences. The dialogue objects are structured in terms of parameters that represent their properties and relations. A dialogue manager, which is the major controlling module of the system, records instances of dialogue objects as nodes of a dialogue tree as the interaction proceeds. The dialogue tree constitutes the global context of the dialogue.

The dialogue objects are divided into three main classes on the basis of structural complexity. There is one class corresponding to the size of a dialogue and another corresponding to the size of a discourse segment (cf. Grosz & Sidner, 1986). An initiative-response (IR) structure is assumed (cf. adjacency-pairs Schegloff & Sacks, 1973) where an initiative opens a segment by introducing a new goal and the response closes the segment (Dahlbäck, 1991). The third class corresponds to the size of a single speech act, or dialogue move. Thus, a dialogue is structured in terms of discourse segments, and a discourse segment in terms of moves and embedded segments. Utterances are not analysed as dialogue objects, but as linguistic objects which function as vehicles of one or more moves.¹

¹The use of three levels for the hierarchical structuring of the dialogue is motivated from the analysis of the corpora. There is no claim that they are sufficient

2.1 Customization

Dialogue objects have been determined for three different applications on the basis of a corpus of 30 dialogues collected in Wizard of Oz-experiments (cf. Dahlbäck, Jönsson, & Ahrenberg, 1993; Fraser & Gilbert, 1991). 10 dialogues were used for each application. In one application, BILDATA, the system is a database providing information on properties of second-hand cars. The other two applications are both concerned with the travel domain. In one of them, TRAVEL1, users can only gather information on charter trips to the Greek Archipelago, while in the other, TRAVEL2, they can also order such a charter trip.

For the purpose of customization, two kinds of information can be obtained from a corpus:

- First, it can be used as a source of phenomena which the designer of the natural language interface was not aware of from the beginning, e.g. in the TRAVEL1 information system some users unexpectedly tried to make orders in spite of the fact that the system does not support it.
- Second, it can be used to rule out phenomena that do not occur in the corpus, especially those requiring sophisticated reasoning to be handled correctly.

Another matter is whether the corpus should serve as the only resource for determining what to include in the system, or whether the corpus data needs to be augmented somehow. The first, rather extreme stand is taken by Kelley (1983) who proposes a method, the User-Derived Interface (UDI), for acquiring the lexical and grammatical knowledge of a natural language interface in six steps. The first two steps are mainly concerned with determining and implementing essential features of the application. In the third step, known as the first Wizard of Oz-step, the subject interacts with what they believe is a natural language interface but which in fact is a human simulating such an interface. This provides data that are used to build a first version of the interface (step four). Kelley starts without grammar or lexicon. The rules and lexical entries are those used by the users during the simulation. In step five, Kelley improves his interface by conducting new Wizard of Oz simulations, this time with the interface running. However, when the user/subject enters a query that the system cannot handle, the wizard takes over and produces an appropriate response. The advantage is that the user's interaction is not interrupted and a more realistic dialogue is thus obtained. This interaction is logged and in step six the system is updated to be able to handle the situations where the wizard responded.

LINLIN is customized to a specific application using a process inspired by the method of User-Derived Interfaces. However, there is

for all types of dialogue, and even less so, to any type of discourse.

a drawback to Kelley's method as a very large corpus is needed for coverage of the possible actions taken by a potential user (cf. Ogden, 1988).

Our approach is thus less extreme. If a phenomenon is present in the corpus then it should be included. If it is not present, but occurs in other studies using similar background systems and scenarios and implementation is straightforward, the system should be customized to deal with it. Otherwise, if it is not present and it would increase the complexity of the system, it is not included. Knowledge from other sources is also employed (cf. Grishman, Hirshman, & Nhan, 1986). In the customization of LINLIN for the BILDATA and TRAVEL systems, knowledge on how the database is organised and also how users retrieve information from databases is needed.

3 Meeting requirements

In this section we discuss how the requirements listed in the introduction: habitability, efficiency, robustness and transparency, can be satisfied, in particular as they apply to the dialogue behaviour of the system.

3.1 Habitability

A dialogue object consists of parameters that specify different kinds of information. The parameters and their values can be modified for each new application, although some parameters are likely to be needed often. Customization of the Dialogue Manager to meet requirements on habitability involves two major tasks:

- Defining focal parameters of the dialogue objects and customizing heuristic principles for changing the values of these parameters.
- Constructing a dialogue grammar for controlling the dialogue, i.e. specify parameters that determine what actions to take in different situations.

Two focal parameters, used in all three applications, are Objects and Properties. They are focal in the sense that they can be in focus over a sequence of segments. Their basic function is to represent the information structure of a move. Objects identify, via description or enumeration, a set of primary referents, and Properties identify a complex predicate ascribed to this set (cf. Ahrenberg, 1987).

Two principles for maintaining the focus structure are utilized. A general heuristic principle is that everything not changed in an utterance is copied from one IR-node in the dialogue tree to the newly created IR-node. Another principle is that the value for Objects will be updated with the value from the module accessing the database, if provided.

Primary parameters for defining the dialogue grammar are *Type* and *Topic*. *Type* represents the illocutionary force of a move. Hayes and Reddy (1983, p 266) identify two sub-goals in simple service systems: 1) “specify a parameter to the system” and 2) “obtain the specification of a parameter”. Initiatives are categorized accordingly as being of two different types 1) update, U, where users provide information to the system and 2) question, Q, where users obtain information from the system. Responses are categorized as answer, A, for database answers from the system or answers to clarification requests. Other *Type* categories are Greeting, Farewell and Discourse Continuation (DC) (Dahlbäck, 1991). The latter type is used for system utterances that signal to the user that it is her turn.

Topic describes which knowledge source to consult. In our database applications three different topics are used: the background system for solving a task (T), the database model for queries about system properties, (S) and, finally, the dialogue tree for clarifications relating to the interpretation of moves (D).

A number of other parameters describing speaker, hearer, utterance content and so on, are also used. Although they provide additional information for the dialogue manager, the structure of the dialogue is largely captured through the parameters *Type* and *Topic*.

3.1.1 The focus structure

In the BILDATA application, task-related questions are about cars. Thus, the *Objects* parameter holds descriptions of, or explicit lists of cars while the *Properties* parameter, holds a set of car properties. In the TRAVEL systems, on the other hand, users switch their attention between objects of different kinds: hotels, resorts and trips. This requires a more complex behaviour of the *Objects* parameter and the use of domain knowledge for focus tracking (Jönsson, 1993b).

The general focusing principles need to be slightly modified to apply to the BILDATA and TRAVEL applications. For the BILDATA application the heuristic principles apply well to the *Objects* parameter. An intensionally specified object description provided in a user initiative will be replaced by the extensional specification provided by the module accessing the database. For the TRAVEL applications the principles for providing information to the *Objects* parameter are modified to allow hotels to be added if the resort remains the same.

For the BILDATA application the heuristic principles for the *Properties* parameter need to be modified. The modification is that if the user does not add new cars to *Objects*, then the attributes provided in the new user initiative are added to the old set of attributes. This is based on the observation that users often start with a rather large set of cars and compare them by gradually adding restrictions (cf. Kaplan, 1983), for instance using utterances like *Remove all small-sized cars*. We call such responses *cumulative* as they summarize all information obtained in a sequence of questions. For the TRAVEL

| Statistics on focusing heuristics | | | |
|--------------------------------------|---------|---------|---------|
| | BILDATA | TRAVEL1 | TRAVEL2 |
| Fully specified initiatives | 52% | 48% | 47% |
| Initiatives requiring local context | 40% | 42% | 45% |
| Initiatives requiring global context | 4% | 2% | 5% |

Table 1: User-initiatives classified according to context-dependence.

applications the copy principle holds without exception.

The results from the customizations showed that the heuristic principles worked well, minimizing the need to search the global context for referents of indexical constructions. See Table 1.

In the TRAVEL2 system there is one more object; the order form. A holiday trip is not fully defined by specifying a hotel at a resort. It also requires information concerning the actual trip: length of stay, departure date and so on. The order form is filled with user information during a system controlled phase of the dialogue.

3.1.2 The dialogue structure

The dialogue structure parameters Type and Topic also require customization. In the BILDATA system the users never update the database with new information, but in the TRAVEL2 system where ordering is allowed the users update the order form. Here another Type is needed, CONF, which is used to close an ordering session by summarizing the order and implicitly prompt for confirmation. For the ordering phase the Topic parameter O for order is added, which means that the utterance affects the order form.

The resulting grammars from the customizations of all systems are quite simple. The most common segment consists of a task-related initiative followed by an answer from the database, Q_T/A_T ², sometimes with an embedded clarification sequence, Q_D/A_D . In BILDATA 60% of the initiatives start segments of this type. For TRAVEL 83% of the initiatives in the non-ordering dialogues and 70% of the ordering dialogues are of this type. Other task related initiatives result in a response providing system information, Q_T/A_S , or a response stating that the initiative was too vague, Q_T/A_D . There are also a number of explicit calls for system information, Q_S/A_S . See Table 2 for a summary of the statistics on dialogue structure.

In the work by Kelley on lexical and grammatical acquisition, the customization process was saturated after a certain number of dialogues. The results presented here indicate that this is the case also

²For brevity, when presenting the dialogue grammar, the Topic of a move is indicated as a subscript to the Type. Labels of IR-segments have the form of a pair of move labels separated by a slash (/).

| Statistics on dialogue structure | | | |
|-----------------------------------|---------|---------|---------|
| | BILDATA | TRAVEL1 | TRAVEL2 |
| Number of rules | 15 | 12 | 14 |
| Q_T/A_T | 60% | 83% | 70% |
| Q_D/A_D | 12% | 2% | 2% |
| Q_S/A_S | 9% | 2% | 2% |
| Q_T/A_D | 7% | 2% | 3% |
| Q_T/A_S | 5% | 6% | 4% |
| Ordering rules | - | 1% | 17% |
| Others, e.g. greetings, farewells | 7% | 4% | 2% |

Table 2: Types of dialogue segments and their relative frequency in three different applications.

for the dialogue structure of our applications. From a rather limited number of dialogues, a context free grammar can be constructed which, with a few generalizations, will cover the interaction patterns occurring in the actual application (Jönsson, 1993a).

The IR-sequences found in the analysis of dialogue structure have a natural explanation if we consider the purpose of the system. Although the dialogue objects do not represent information on user's goals, it turns out that the user utterances can be classified into a few classes in goal-related terms. The segments can basically be divided into four classes, taking the user's initiative as the basis for the classification: (i) "proper" information requests that are satisfied by an answer with information from the database, (ii) successful queries about system properties, (iii) successful moves satisfying subordinate goals, such as greetings or discourse continuations; (iv) initiatives that transgress the system's knowledge and which require robust error handling.

Note that we can view the taxonomy as a preference order. There is a basic division between the simple, normal segment (Q_T/A_T with two moves) and the other segments, which in one way or another indicates that we have left the normal smooth interaction, and do something subordinate to the main purpose. From the point of view of efficiency the interaction is optimal when it consists of a sequence of Q_T/A_T -segments. If the user resorts to a Q_S , there is something about the system that she isn't aware of, but which he could be aware of. The third type is also of a subordinate, instrumental character, but really unnecessary for an experienced user. Uninterpretable moves are obviously the least preferred ones.

3.2 Optimal responses in normal mode

If the system determines that a certain user initiative is a complete and successful Q_T , the only remaining task is to derive an appropriate SQL-query and respond with the tuples that are returned from the database. One problem that the system can encounter in this process is ambiguity: a certain word or expression may be translated to the query language in more than one way. For instance, in the BILDATA database there is information on acceleration and top speed, so it is not clear what a user has in mind if he asks whether a certain car make is fast (Sw. *snabb*), or requests to see a list of fast cars. Similarly, the adjective *rymlig* (Eng. spacious) may pertain to the booth or the space inside the car. In this case, one option is to enter into a clarification sub-dialog. However, this takes time and it may be more swift to provide information on all aspects that are potentially relevant. The user may easily ignore information, if he is not interested.

A similar strategy is used if a question is about the same set of cars as a previous question (cf. 3.1.1). Then the information requested in the second question is added to that of the first. Some subjects actually mentioned this as a good feature of the system in their comments, as it facilitates comparisons and evaluations to have all relevant information in a single table.

In both of these cases the system provides more information to the user than she has actually asked for. Some other cases where this happens are described in the following section.

3.3 Communication of meta-knowledge

As regards the second type of user initiative, the system queries, there is first the problem of distinguishing them from the task-oriented queries. Descriptively, these queries have a different topic. The topic is recognized from the major constituents of the query, in particular main verbs and their complements.

Frequently the user starts a session by asking a question such as *Vilka bilar finns?* (What makes are there?) or *Finns det priser på både nya och begagnade bilar?* (Can you provide prices on both new and second-hand cars?) These questions are currently all answered by a pre-stored message giving a brief description of the contents of the database. This text usually gives more information than the user has asked for, and, as with the second example above, provides an answer only implicitly. However, it would easily be possible to make the responses more fine-grained given a sufficient empirical basis. Another option would be to display this text in a separate window on the screen so that the question need not arise as part of the dialogue.

The most common type of system query in the corpus concerns how information should be interpreted, e.g. *Förklara siffrorna för rostbenägenhet.* (Explain the numbers indicating susceptibility to

rust). These are recognized on the basis of the occurrence of a predicate that indicates that the interpretation of a certain value, or type of value, is at stake. The answers are then not obtained from the BIL-DATA database, but from a special file forming a part of the database model. Thus, there is one text informing on how rust is evaluated and what the various numbers in the rust column means, which is used as a response to all questions of type S relating to rust. It should be noted that the strategy of answering system queries by pre-stored standard answers has the effect that the analysis of a system question need not be as detailed as the analysis of a task-oriented question, thus contributing to efficiency.

3.4 Robustness

Although the model offers a general way to support robustness, i.e. that of engaging in a clarification sub-dialogue, this way is not always the best one. Many problems cannot be diagnosed correctly by the system, and other problems, such as mis-spellings, can be regarded as too small to warrant a clarification sub-dialog. Moreover, the model is restricted to information signalled verbally as part of a NL-dialog. But the system may communicate with the user by other means, e.g. in another window (pop-up window) or by non-verbal signals in the dialogue window. The purpose of this extra channel would be to give instructions and hints that enable the user to stay within the bounds of what the system allows with as little disturbance and time loss as possible. That is, it can be argued that it is preferable to solve problems by giving signals to the user that he need not respond to verbally, only interpret.

Below we discuss several types of transgressions that the user can make and discuss means to handle them. They relate to the lexicon, the sentence grammar, the dialogue grammar and the domain model, respectively.

It is common that the user enters a word that has no analysis in the lexicon. Considering the requirements on efficiency it is important that the transgression is detected as soon as possible. If lexical processing does not start until the user enters a carriage return, a lot of time may be wasted. The best one can achieve in this regard is to detect the mistake as soon as the user has pressed a character key that results in a substring with no continuation in the lexicon. As our current lexicon does not support error detection at this level, we go for the second best alternative, i.e. to detect errors when a word separator has been encountered. The error is presently signalled to the user by changing the font of the echoed input. There is no diagnosis performed by the system; this is a task given over to the user on the assumption that he knows better than the system what could be the trouble.³ We believe that a non-verbal signal is more adequate than

³Of course, this need not always be the case and we are currently looking into robust parsing techniques to support diagnosis and recovery (Ingels, 1993). How-

a verbal message. The important thing is to make the user aware of the lexical transgression while he is still engaged in writing.

Grammatical transgressions are handled similarly to lexical transgressions paying due regard to early detection. How soon you can detect a grammatical error depends on the parsing technique you use. To increase efficiency we are using a left-to-right, incremental chart-parser, which means that parsing starts as soon as the user enters some input (Wirén, 1992; Wirén & Rönnquist, 1993). The time interval between the user's pressing the carriage return key and the moment when parsing is finished is thus reduced noticeably. For an input to receive a grammatical analysis it is necessary that every word of the input receives an analysis as part of some phrase, possibly as a direct descendant of the category spanning the whole input. Thus, if a word is entered that cannot combine with active edges to its left, nor yield a phrase that can combine with those edges (top-down filtering is used), then the system knows that no complete parse can be obtained. Thus, many grammatical transgressions can be detected fairly quickly. Currently there is no diagnosis but the transgression is signalled to the user as soon as it is detected by inverting the dialogue window. This shows the user that the input cannot be interpreted, although it does not tell him what is at fault. We are contemplating giving information of the following kinds: information about expressions that could follow strings that are left unextended at the relevant vertex, and lists of example sentences that are somehow similar to what the user has written, e.g. that contain the same initial words.

Dialogue grammar transgressions occur when the user enters something which the system can only interpret as a move which is out of place. For example, the user may interpret a system message as a question and enter "Yes", while the system intended it as a discourse continuation and expected a question. The transgression is detected when the dialogue manager tries to connect the user move with the current dialogue tree. It is signalled to the user by an explicit error message, which entails a superficial analysis of the error and brings recovery with it. The system takes no notice of the error, however, and does not enter the segment in the dialogue tree. This means that the segment is treated as if it had never occurred for the purpose of focus tracking and expectations, i.e. in the same fashion as a lexical or grammatical transgression. The dialogue continues in the same state as before the transgressive move.

Another type of transgression is when the user requests information that is not in the database, although the user seems to believe so. Some examples in the BILDATA domain concern the colour of cars, or variation in the number of doors. This type of error is detected by using information in the database model, which also should represent

ever, under current circumstances, the user is better equipped to handle lexical transgressions.

common knowledge of the domain e.g. properties of cars that people like to inquire about. These properties can be added to the database model as they are discovered and be marked there as not corresponding to anything in the database. Thus, these transgressions are easily detected and diagnosed. The response is an A_S -move and is taken to be part of the dialogue, as the user is likely to refer to objects and properties mentioned in his request afterwards.

A question may refer to a set of properties, or to several objects, only one of which is not in the database. The general strategy is then to filter out those properties and objects that there is no information on, but supply an answer for those that are correct. In addition to the retrieved table, an error message is included informing about the transgression. This is in line with the preference ordering of IR-segments: we prefer database information to be exchanged as much as possible.

4 A set of principles for design

In this section we ask whether the solutions to the specific problems that we have discussed, conform to some general design principles. Indeed, we believe that they do. However, allowing users to participate in a coherent dialogue does not in itself guarantee that the requirements discussed above on habitability, robustness, transparency and efficiency are properly addressed. To do this we need to consider the overall design of the system.

4.1 The Sublanguage Principle

The language used in an interface dialogue is affected by several factors, e.g. that one participant is a human and the other a computer system, that the human partner is involved in a specific (work) task and that this task is related to a particular knowledge domain.

With reference to the well-known work on sublanguages (cf. Grishman & Kittredge, 1986) we refer to this guide-line as **The Sublanguage Principle**: Restrict and adapt the linguistic and general knowledge of the system to that which is needed to support the users' tasks.

Apart from the general conclusion that we have a more restricted development task to cope with than full NL-understanding and -interaction, we also see it as a virtue to restrict the linguistic and conceptual knowledge of the system to that which is needed for the task at hand, since the interpretations that the system makes of the user's utterances can be fewer (less ambiguity) and more specific (domain-specific meanings and categorizations can be used).

The Sublanguage Principle raises the question of what the necessary requirements are. All aspects of linguistic communication, including interaction patterns and the use of indexical language are assumed to depend on the application and domain. For this reason the interface system must be designed to facilitate customization

to meet the needs of different applications. Moreover, we must find methods that allow us to determine what the needs of a given application are. Such a method, is the Wizard of Oz-simulations. Our results so far indicate that for simple service systems the answers can be obtained with reasonable effort (Jönsson, 1993a).

We also take The Sublanguage Principle to imply that the semantics of the language used is jointly provided by the background system and the dialogue acts that are meaningful to perform in the context of the application. Different applications put different demands on the dialogue acts. For instance, the TRAVEL2 application needs to handle orders, which normally is not required of applications where ordering is not allowed. However, the simulated dialogues showed that some users in the TRAVEL1 application also tried to place an order, although the background system did not support that. Consequently, the interface needed to be equipped with mechanisms for (i) recognizing orders, and (ii) explaining the limits of this application to the subjects as a response. Thus, customization and empirical investigations are necessary to make The Sublanguage Principle support habitability in practice.

Similarly, the requirements on focus structure can vary from one application to another. In the CARS dialogues we have found that virtually all ellided plural NPs and plural third person pronouns and demonstratives such as *them* and *those* refer to a set of cars focused in the previous segment. This is a fact that we have exploited in the interpreter of the CARS system.

4.2 The Asymmetry Principle

An obvious, but nevertheless important observation on NLI and current NLP technology, is that the user and system have different abilities and knowledge. We note especially the following interesting asymmetries between the two participants of a NLI dialogue (c.f. Dahlbäck, 1987; Dahlbäck & Hägglund, 1987; Jönsson & Dahlbäck, 1988):

- The user reads and understands natural language at good speed; moreover she is able to select information on the basis of relevance. In contrast, understanding is a hard task for the computer, not to talk about the task of distinguishing important from irrelevant information;
- The user is often slow at writing on the terminal. In contrast, the interface can display a lot of information on the screen in virtually no time at all.
- The system's knowledge of the standard language and its common sense knowledge of the world is only partial, while the user's knowledge is much richer. This would be true as far as

language is concerned, even if we did not adhere to the sub-language principle. Moreover, it is difficult to make the system adapt to users with different linguistic and knowledge skills, while users generally have this capacity to adapt to the competence of others.

The guide-line that can be derived from these observations is simply that you should look for solutions that takes the strong and weak points of the two sides into consideration. More formally stated, we have **The Asymmetry Principle**: Design the interface in a way that exploits the relative strengths and circumvents the relative weaknesses of the two participants.

This principle must be used with some care, of course, and we must recognize that technical developments may make NLI's more powerful than they are today. For the present we generally want to minimize the inference tasks that the system has to cope with, such as diagnosing users' errors, inferring accurate representations of the user's goals and beliefs, instead relying on simpler means that clearly reveal the capabilities and limitations of the system, e.g. by providing examples of what the system can understand or by displaying help messages.

Another important consequence of this principle is that if there is a choice, prefer solutions that make the user learn from system contributions to solutions that require the system to learn from the user's contributions.

Answers are pre-stored and retrieved on the basis of the semantic representation of the query. The answer usually gives more information than the user has asked for (c.f. The Quantity Principle below). Another option would be to display this information in a separate window on the screen so that the question need not arise as part of the dialogue.

As previously mentioned, the most common type of query relating to system properties in the corpus concerns how displayed information should be interpreted, e.g.:

User *Explain the numbers indicating susceptibility to rust.*

System *The evaluation of rust susceptibility is based on information given by car owners as regards the corrosion of their cars. The ranking has been based on the following classes:*

- 1. Much worse than the average car,*
- 2. Worse than the average car,*
- 3. Average*
- 4. Better than the average car,*
- 5. Much better than the average car.*

The pre-stored messages presented from the system are often more complicated than one would expect that the parsing modules are able

to interpret. The design recommendations for natural language interfaces presented by (Ogden, 1988) suggest that the feedback provided by the system must be allowable as input expressions as there are studies showing that users often echo the form of the feedback (cf. Zoltan-Ford, 1984, 1991). However, the system performs a different communicative act when informing the user about properties of the system than when paraphrasing a user question, a kind of act that the user herself never performs. Consequently, users do not try to mimic the system's rather verbose messages explaining properties of the system when retrieving information from the background system. This observation is not only true for this kind of system messages, but seem to generalize to other kinds of pre-stored answers that stylistically deviates from the rest of the dialogue. We have for instance found that the users of the TRAVEL systems treat the long stretches of canned text with travel agency resort information differently from other system utterances, by rarely referring to contents of parts of the text using pronouns or other indexical expressions. They are, in a sense, encapsulated within the dialogue (Dahlbäck, 1991).

Another application of The Asymmetry Principle is the use of system initiatives to support efficiency. If the system does not only respond to users' inputs, but also occasionally takes the initiative, the interaction can be brought to an end quicker. Also, the interpretation of user utterances becomes simpler. Although users respond to a system initiative using a sentence fragment, for instance if the system asks *How long do you want to stay?* a typical user answer could be the fragment *Two weeks*. The interpretation of such a fragment is computationally easy to handle as the system knows what type of user input to expect. System initiatives also supports transparency as it tells the user what information is considered important in a particular application.

System responses and initiatives need not be restricted to the verbal channel. The system may communicate with the user by other means, e.g. in another window (pop-up window) or by non-verbal signals in the dialogue window. For example, in the CARS system, mis-spellings are signaled to the user by echoing input in a different font than normally. Generally speaking, the purpose of the extra channel is to give instructions and hints that enable the user to stay within the bounds of what the system allows with as little disturbance and time loss as possible. That is, it can be argued that it is preferable to solve problems by giving signals to the user that he need not respond to verbally, only interpret.

The asymmetry principle also implies that the way of expressing messages need not be the same for both participants. Answering a query with tables instead of with written text is simple to achieve for a computer, and is often preferred to a textual description of the contents of the table. The users of our CARS-system, where such an interaction was used, gave a very positive evaluation of this way of presenting the information, presumably since it is easier to read, and

makes the decision of the user, e.g. selecting the optimal solution based on the integration of the values on a number of parameters, simpler.

4.3 The Quantity Principle

A frequent interpretation problem in natural-language interfaces is ambiguity: a certain word or expression may be translated to the query language in more than one way. For instance, in the CARS database there is information on acceleration and top speed, so it is not clear what a user has in mind if she asks whether a certain car make is *fast*, or requests to see a list of fast cars. Similarly, the adjective *spacious* may pertain to the booth or the space inside the car. In this case, one option is to enter into a clarification sub-dialog. However, this takes time and it may be more swift to provide information on all aspects that are potentially relevant. The user may easily ignore information, if she is not interested.

A similar strategy is used if a question is about the same set of cars as a previous question. Then the information requested in the second question is added to that of the first. Although this obviously deviates from how a question is normally answered in a human conversation, some subjects actually mentioned this as a good feature of the system in their comments, as it facilitates comparisons and evaluations to have all relevant information in a single table, instead of having to integrate information from two displays, or having to memorize the contents of the previous answer.

These particular solutions are motivated by another design principle, which we term **The Quantity Principle**: The system may give more information to the user than has actually been requested provided it is potentially relevant and does not overload the user. Another case where the principle was applied was in the design of answers to queries about system properties, as illustrated in the previous section. The principle can be motivated from what we know about the average user: she has the ability to select (within limits, of course) what is relevant for her. The situation is similar to the one for information in tabular form, selection does not require excessive reading.

The Quantity Principle generally increases speed and supports robustness, while there may be a conflict with transparency, if the principle is not applied with care. But, as mentioned previously, users seem to be able to distinguish between different kinds of system utterances, and do not expect the system to accept as input everything it produces as output. We also note that it seemingly contradicts Grice's (Grice, 1975) second maxim of quantity: "Do not make your contribution more informative than is required". But then, Grice did not have human-computer dialogue in mind when he stated it, and, in fact, suggested that it could be subordinated to the maxim of relevance.

The intelligent help system GENIE (Wolz, 1993) is another exam-

ple where strategies are used that conform to The Quantity Principle. GENIE analyses user goals to produce not only accurate answers to a user query, but also to introduce new information related to the answer, for instance, to the query *How can I send a message to someone?* the system does not only answer with information on how to send e-mail to one person but also adds *Also, you can send mail to a group. To specify a group, separate the addresses with commas.* (*ibid.* p. 168)

5 Generalization to other kinds of interfaces?

All the examples in the paper have been taken from one kind of interface and two kinds of system, teletype NLI in conjunction with DB information retrieval or information retrieval and ordering. The obvious question is to what extent the suggested principles apply to other kinds of man-machine interaction.

Looking first at other kinds of systems using typed interaction, we believe that the principles apply here too, though their realization will take different forms. The interface for the SHERLOCK tutoring system (Lemaire & Moore, 1994) provides an illustration of this. They found in their analysis of human student-tutor interactions that tutors frequently refer to what they have said in a previous explanation. Another observation was that students frequently ask questions that refer to the tutor's previous explanations, mainly to request comparisons between the current and previous situation. Deciding not to mimic the human-human interaction, but instead exploit the unique attributes of the computer to make use of the dialogue history, the system cannot only refer to previous explanations but allow the user to visualize what has been said. We take this as an excellent application of The Asymmetry Principle in a domain different from ours.

Concerning spoken instead of written interaction, we believe that The Sublanguage Principle and The Asymmetry Principle are as valid for speech interfaces as they are for teletype interfaces. However, for speech output The Quantity Principle cannot be applied, for the simple reason that the user cannot receive and filter information with the same ease in the auditory channel as in the visual channel. However, when the screen is used to output text or graphics to the user, the principle applies with the same force.

Multi-modal interfaces provide means to utilize the Asymmetry and Quantity principles further, for instance using pop-up windows to communicate system information and hints, spoken output of error messages, text or graphics output combined with speech input etc. Examples of systems which use a variety of modalities for both interpretation and generation include AlFresco (Stock, 1991), XTRA (Wahlster, 1991), Voyager (Zue, 1994) and CUBRICON (Neal & Shapiro, 1991).

The main difference between these multi-modal systems and con-

ventional natural language interfaces is their ability to use a combination of input and output modalities such as speech, graphics, pointing and video output. Thus, more advanced interpretation and generation modules are required and principles for determining which media to utilize are needed (Arens, Hovy, & Vossers, 1993).

However, the dialogue and focus structures need not necessarily be more complicated. For instance, Voyager successfully utilizes the local context approach presented here (Seneff, 1992). Bilange (1991) presents a dialogue grammar for dialogue management for telephone interaction with an information retrieval application. Sitter and Stein (1992) present a model based on possible sequences of dialogue acts which are modeled in a transition network. In Stein and Maier (1993), Stein and Thiel (1993) that model is extended to handle multi-modal interaction as utilized in the MERIT system (Stein, Thiel, & Tißen, 1992). Traum and Hinkelman (1992) present a finite state machine for recognizing discourse units; units comprising more than a single utterance but which conveys one mutually understood intent. The conversation acts proposed by Traum and Hinkelman (1992) cover discourse obligations and are to be used in conjunction with beliefs and desires (cf. Traum & Allen, 1994) in a task-oriented spoken dialogue system. Another system is the Vehicle Navigation System (Novick & Sutton, 1994) where users can receive driving directions a step at a time by cellular telephone. This type of interaction requires a system capable of recognizing various acknowledgment acts and determining their applicability. A grammar based model for this is defined where exchanges are interpreted in terms of speech acts.

Thus, it seems that for simple service systems, the dialogue model presented here will be sufficient. However, for task-oriented dialogues, where the user's task directs the dialogue (Loo & Bego, 1993), a model of this and the user's goals need to be consulted in order to provide user-friendly interaction (cf. Burger & Marshall, 1993). This does not imply the necessity of a sophisticated model based on the user's intentions. A hierarchical structure of plans based on the various tasks possible to carry out in the domain might do just as well (cf. Wahlster, André, Finkler, Profitlich, & Th, 1993).

6 Conclusion

The proposed method of customization and the design principles should be regarded as tentative as no real users have actually put their hands on our system. Yet, taken together, the pragmatic observations on NLIs and the data from NLI dialogues that we have analysed suggest to us that many features, e.g. user models and general text inference components, that have been argued are important for good functionality of NLIs are not generally required for simple service applications. Instead our principles take us in another direction. Rather than modelling the user we prefer the system to present

a good model of itself (The Asymmetry Principle) and rather than inferring interpretations we rely on application-specific meanings (The Sublanguage Principle). To put it in a slogan-like phrase: if there is a choice, prefer global pragmatics at design time to local pragmatics at run-time.

Acknowledgements

This work results from a project on Dynamic Natural-Language Understanding supported by The Swedish Council of Research in the Humanities and Social Sciences (HSFR) and The Swedish National Board for Industrial and Technical Development (NUTEK) in their joint Research Program for Language Technology. We are indebted to the other members of NLPLAB for many valuable discussions on the topics of this paper.

References

- Ahrenberg, L. (1987). *Interrogative Structures of Swedish. Aspects of the Relation between grammar and speech acts*. Ph.D. thesis, Uppsala University.
- Ahrenberg, L., Jönsson, A., & Dahlbäck, N. (1990). Discourse representation and discourse management for natural language interfaces. In *Proceedings of the Second Nordic Conference on Text Comprehension in Man and Machine, Täby, Sweden*.
- Arens, Y., Hovy, E., & Vossers, M. (1993). On the knowledge underlying multimedia presentations. In Maybury, M. T. (Ed.), *Intelligent Multimedia Interfaces*, pp. 280–306. MITPress.
- Bilange, E. (1991). A task independent oral dialogue model. In *Proceedings of the Fifth Conference of the European Chapter of the Association for Computational Linguistics, Berlin*.
- Burger, J. D. & Marshall, R. J. . (1993). The application of natural language models to intelligent multimedia. In Maybury, M. T. (Ed.), *Intelligent Multimedia Interfaces*, pp. 174 – 196. MITPress.
- Dahlbäck, N. & Jönsson, A. (1992). An empirically based computationally tractable dialogue model. In *Proceedings of the Fourteenth Annual Meeting of The Cognitive Science Society, Bloomington, Indiana*.
- Dahlbäck, N. (1987). Kommunikation med datorer i naturligt språk - vad är det och vem behöver det?. Tech. rep. LITH-IDA-R-87-15, Department of Computer and Information Science, Linköping University.

- Dahlbäck, N. (1991). *Representations of Discourse, Cognitive and Computational Aspects*. Ph.D. thesis, Linköping University.
- Dahlbäck, N. & Hägglund, S. (1987). Människa och datorsystem i samverkan. Tech. rep. MDA-rapport 1987:16, Arbetsmiljöfonden and Styrelsen för teknisk utveckling.
- Dahlbäck, N., Jönsson, A., & Ahrenberg, L. (1993). Wizard of oz studies – why and how. *Knowledge-Based Systems*, 6(4), 258–266.
- Fraser, N. & Gilbert, N. S. (1991). Simulating speech systems. *Computer Speech and Language*, 5, 81–99.
- Grice, P. H. (1975). Logic and conversation. In Cole, P. & Morgan, J. L. (Eds.), *Syntax and Semantics (vol. 3) Speech Acts*. Academic Press.
- Grishman, R. & Kittredge, R. I. (1986). *Analysing language in restricted domains*. Lawrence Erlbaum.
- Grishman, R., Hirshman, L., & Nhan, N. T. (1986). Discovery procedures for sublanguage selectional patterns: initial experiments. *Computational Linguistics*, 12(3), 205–215.
- Grosz, B. J. & Sidner, C. L. (1986). Attention, intention and the structure of discourse. *Computational Linguistics*, 12(3), 175–204.
- Hayes, P. J. & Reddy, D. R. (1983). Steps toward graceful interaction in spoken and written man-machine communication. *International Journal of Man-Machine Studies*, 19, 231–284.
- Ingels, P. (1993). Robust parsing with charts and relaxation. In *NODALIDA '93: Proceedings from 9:e Nordiska Datalingvistikdagarna, Stockholm, June 1993*.
- Jönsson, A. (1991). A dialogue manager using initiative-response units and distributed control. In *Proceedings of the Fifth Conference of the European Chapter of the Association for Computational Linguistics, Berlin*.
- Jönsson, A. (1993a). *Dialogue Management for Natural Language Interfaces – An Empirical Approach*. Ph.D. thesis, Linköping University.
- Jönsson, A. (1993b). A method for development of dialogue managers for natural language interfaces. In *Proceedings of the Eleventh National Conference of Artificial Intelligence, Washington DC*, pp. 190–195.

- Jönsson, A. & Dahlbäck, N. (1988). Talking to a computer is not like talking to your best friend. In *Proceedings of the First Scandinavian Conference on Artificial Intelligence, Tromsø*.
- Kaplan, S. J. (1983). Cooperative responses from a portable natural language database query system. In *Computational Models of Discourse*, pp. 167–208. MIT Press.
- Kelley, J. F. (1983). An empirical methodology for writing user-friendly natural language computer applications. In *Proceedings of the CHI'83*, pp. 193–196.
- Lemaire, B. & Moore, J. (1994). An improved interface for tutorial dialogues: browsing a visual dialogue history. In *Proceedings of CHI'94, Boston*.
- Loo, W. V. & Bego, H. (1993). Agent tasks and dialogue management. In *Workshop on Pragmatics in Dialogue, The XIV:th Scandinavian Conference of Linguistics and the VIII:th Conference of Nordic and General Linguistics, Göteborg, Sweden*.
- Neal, J. G. & Shapiro, S. C. (1991). Intelligent multi-media interface technology. In Sullivan, J. W. & Tyler, S. W. (Eds.), *Intelligent User Interfaces*. ACM Press, Addison-Wesley.
- Novick, D. G. & Sutton, S. (1994). An empirical model of acknowledgement for spoken-language systems. In *Proceedings of the 32nd Conference of the Association for Computational Linguistics, New Mexico*.
- Ogden, W. C. (1988). Using natural language interfaces. In Helander, M. (Ed.), *Handbook of Human-Computer Interaction*. Elsevier Science Publishers B. V. (North Holland).
- Schegloff, E. A. & Sacks, H. (1973). Opening up closings. *Semiotica*, 7, 289–327.
- Seneff, S. (1992). A relaxation method for understanding spontaneous speech utterances. In *Paper presented at the Fifth DARPA Workshop on Speech and Natural Language*.
- Sitter, S. & Stein, A. (1992). Modeling the illocutionary aspects of information-seeking dialogues. *Information Processing & Management*, 28(2), 165–180.
- Stein, A. & Maier, E. (1993). Modeling and guiding cooperative multimodal dialogues. In *Proceedings of the AAAI Fall Symposium '93 on Human-Computer Collaboration: Reconciling Theory, Synthesizing Practice, Raleigh, NC, U.S.A.*
- Stein, A. & Thiel, U. (1993). A conversational model of multimodal interaction in information systems. In *Proceedings of the*

Eleventh National Conference of Artificial Intelligence, Washington DC, pp. 283 – 288.

- Stein, A., Thiel, U., & Tißen, A. (1992). Knowledge based control of visual dialogues in information systems. In *Proceedings of the 1st International Workshop on Advanced Visual Interfaces, Rome, Italy*.
- Stock, O. (1991). Natural language exploration of an information space: the alfresco interactive system. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence, Sydney, Australia*, pp. 972–978.
- Traum, D. R. & Allen, J. F. (1994). Discourse obligations in dialogue processing. In *Proceedings of the 32nd Conference of the Association for Computational Linguistics, New Mexico*, pp. 1–8.
- Traum, D. R. & Hinkelman, E. A. (1992). Conversation acts in task-oriented spoken dialogue. *Computational Intelligence*, 8(3), 575–599.
- Wahlster, W. (1991). User and discourse models for multimodal communication. In Sullivan, J. W. & Tyler, S. W. (Eds.), *Intelligent User Interfaces*. ACM Press, Addison-Wesley.
- Wahlster, W., André, E., Finkler, W., Profitlich, H.-J., & Rist, T. (1993). Plan-based integration of natural language and graphics generation. *Artificial Intelligence*, 63, 387 – 427.
- Watt, W. C. (1968). Habitability. *American Documentation*, July, 338–351.
- Wirén, M. (1992). *Studies in Incremental Natural Language Analysis*. Ph.D. thesis, Linköping University.
- Wirén, M. & Rönquist, R. (1993). Fully incremental chart-parsing. In *Third International Workshop on Parsing Technologies, Tilburg, The Netherlands and Durbuy, Belgium*.
- Wolz, U. (1993). Providing opportunistic enrichment in customized on-line assistance. In *Proceedings from the 1993 International Workshop on Intelligent User Interfaces, Orlando, Florida*.
- Zoltan-Ford, E. (1984). Reducing variability in natural -language interactions with computers. In *Proceedings of the Human Factors Society 28th Annual Meeting, Santa Monica, CA*, pp. 768–772.
- Zoltan-Ford, E. (1991). How to get people to say and type what computers can understand. *International Journal of Man-Machine Studies*, 34, 527–547.

Zue, V. W. (1994). Toward systems that understand spoken language.
IEEE Expert, 9, 51–59.