COMMENTING ON ACTION: CONTINUOUS LINGUISTIC FEEDBACK GENERATION

Wim Claassen*, Edwin Bos*, Carla Huls*, Koenraad de Smedt**

**Nijmegen Institute for Cognition and Information*
PO Box 9104, 6500 HE Nijmegen, The Netherlands
Phone: +31 80 612618
E-Mail: bos@nici.kun.nl

***Leiden University*
Unit of Experimental Psychology
P.O.Box 9555, 2300 RB Leiden, The Netherlands
Phone: +31 71 273407
E-mail: desmedt@rulfsw.leidenuniv.nl

ABSTRACT

Action mode interfaces, in which the user achieves his goals by manipulating representations, suffer from some fundamental disadvantages. In this paper, we present a working prototype of a system for Continuous Linguistic Feedback Generation (CLFG), a facility that addresses some of the major disadvantages. CLFG generates natural language descriptions of the actions the user is performing. These descriptions are presented in both the visual and audio channels. The knowledge sources and algorithm that enable CLFG to provide concise and relevant information are described in detail.

KEYWORDS: natural language generation, multimodal interfaces, action mode.

1 INTRODUCTION

During the last decade, several categorizations for interfaces have been made (e.g., Hutchins, Hollan, & Norman, 1986; Frohlich, 1991). They all seem to converge toward two basic modes of interaction. Hutchins, Hollan, and Norman (1986) describe the two modes of interaction as metaphors: the metaphor of interface as *conversation* and the metaphor of interface as *model world*. We use a slightly different terminology based on the interaction mode instead; we distinguish between the *language mode* and the *action mode*. The distinctive criterion is whether the user can achieve his goal by using a linguistic description or by a non-verbal manipulation of representations (usually icons).

In the language mode, the user and the system engage in a dialogue of sorts. In practice, the user often formulates commands and queries in a formal or natural language. Natural language (NL) in particular is an essential component of recently developed multimodal interfaces, e.g., XTRA (Allgayer, Jansen-Winkeln, Reddig, and Reithinger, 1989), Cubricon (Neal and Shapiro, 1988), ALFresco (Stock, 1991). NL is a powerful and efficient medium to convey complex, abstract information. It is suited to efficiently communicate abstract or imaginary concepts, quantities and functional properties. Examples are *a good idea, the file I'm going to make tomorrow, all the Lisp files created by Koen, the message for Theo,* etc.

In the action mode, on the other hand, the system and the user communicate by effecting changes in a representation which is usually presented visually. In this mode, referring to unique objects is achieved in a most natural way by pointing. Information about shapes, textures, or hierarchical structures can be communicated almost ideally in a visual representation.

Recently, the introduction of so-called *multimodal* user interfaces has been proposed (e.g., Claassen, Bos, & Huls, 1990; Neal & Shapiro, 1988; Schmauks & Reithinger, 1988; Walker, 1989). The main reason to combine the two modes is that the disadvantages of both modes are considered largely complementary. The shortcomings of one mode of interaction can thus be countered by providing the other mode as well.

In this paper, we argue for the incorporation of a particular linguistic facility into action mode based interfaces. The proposed facility provides NL comments on the user's actions. Our basic assumption is that linguistic conveyance of information that is not present in the visual model world will yield fewer errors and improves the achievement of user goals. NL is a channel well suited to conveying such supplemental information.

Systems that are, to some extent, related to our work exist to date. Multimedia applications that combine graphics with language, albeit usually prerecorded speech or canned phrases, have become quite common nowadays. More interactive is the *SOCCER* system (André et al., 1988) which automatically generates descriptions of scenes that have been analyzed by a vision system. The *WIP* project (Wahlster et al., 1991) and the *Information Presentation* project (Arens, 1992) are aimed at multimodal descriptions in instructive manuals. Our work differs from these systems especially in its goal to enhance user interaction with systems dynamically.

The rest of this paper is structured as follows: in section 2, we will discuss the disadvantages of action mode based interfaces into more detail. In section 3, we will provide an overview of a working prototype of a multimodal interface. This multimodal interface is called EDWARD (Bos, Claassen, Huls, forthcoming) and is under development at NICI. In section 4, we will present the Continuous Linguistic Feedback Generator (CLFG), a facility of EDWARD that comments on user action. In section 5, we will elaborate on the improvements CLFG offers for action mode based interfaces. In the final section, some conclusions are drawn and opportunities for further research are sketched.

2 DISADVANTAGES OF ACTION MODE

Although more and more interactive systems adopt the action mode, this does not mean that the action mode is free of disadvantages. Bos (in press-a) lists several communicative features that give rise to disadvantages of the action mode. Here we will elaborate on some of them, in particular those that seem dissolvable by commenting on user action:

1 *Representation limitations.* The capacity of the graphical representational system is often insufficient. E.g., not all attributes of a file can be revealed by its icon. In contexts with a high information density, it is impossible to design representations in such a way that all types of information that could be relevant can immediately be observed by inspection.

2 *Specification beyond the 'here and now'.* A model world does not allow for the opportunity to abstract from the 'here and now' of it (Walker, 1989). It is important to note that items that are present in the model world need not be directly available: Often only a partial view on the world of action can be offered.

3 *Speech acts.* The underlying intention of an expression in the interface language, that is, the type of speech act (Searle, 1969), can hardly be revealed in the action mode. It is not immediately apparent whether a graphical representation or manipulation is intended as a comment, warning, explanation or question.

4 *Constraining reasons.* Action mode interfaces often constrain user actions that would result in erroneous states or illegal commands. In many desktop interfaces, e.g., icons cannot be dragged off the desktop. However, this raises a new problem. Users want to be informed *why* their action is constrained. This is not easy to achieve in the action mode.

5 *Unintended actions.* Since we have less control over what we do than over what we say, unintended actions occasionally occur. Typically, uncontrolled movements that are interpreted by the system (e.g., accidental release of a mouse button), result in undesired action execution.

6 *Inverse WYSIWYG.* In all action mode interfaces we know of, it is possible to perform actions that do not actually mean what the user might think. The user might erroneously apply an inverse WYSIWYG: What You Don't See You Haven't Got. Empirical data supporting this view can be found by Hensgens (Hensgens), who finds, e.g., some users trying to delete an arc in a graphical tree-editor by making the arc invisible.

3 A SYSTEM COMMENTING ON ACTION

Having described the disadvantages of the action mode, we now present the multimodal interface EDWARD (section 3.1), which comprises a component named CLFG (section 3.2) that comments on user action.

3.1 Overview of EDWARD

The multimodal interface EDWARD (Bos, Claassen, Huls, forthcoming) is composed of two subsystems, each providing unimodal interaction, coupled by a dialogue manager. The first module is the Dutch natural language dialogue system DoNaLD (Claassen & Huls, 1991), and the second is the graphical graph-editor Gr^2 (Bos, in press-b). EDWARD is implemented in Allegro Common Lisp and runs on a DECstation. The linguistic descriptions are either in written form (user input, system output) or spoken (currently system output only). EDWARD is a generic, domain independent interface. The current example domain which EDWARD is applied to is a hierarchichal file system consisting of various types of files, folders, mail messages, etc.

The user can manipulate the objects in the domain in four ways: by pointing at objects and mimicking actions (action mode), by selecting actions from menus, by entering a command in a command language (formal language mode) or by addressing the system in natural language (natural language mode). These modes can be used separately but they also can be combined in one multimodal expression, e.g. by typing the expression *put them there*↗, where ↗ denotes a simulated pointing gesture. The manipulation of the objects in either way results in an update of the model world as well as in an update of the internal knowledge base, which, in our example domain, implies an operation on the file system. Figure 1 presents a snapshot of an interaction.

Figure 1: A screen dump of EDWARD. The user has selected a file icon (black) and is dragging it to the copier icon (bottom left).

The frame in which EDWARD is displayed consists of three areas. The topmost area contains iconized trace windows. The area occupying most of the screen is the graphics display, which is a window called Gr^2. The example shown in Figure 1 represents a hierarchy of directories (depicted as bookcases in a tree structure), and files (e.g., reports, papers, emails, and books). The viewport shows only part of the model world, which, in principle, extends indefinitely. In the bottom corners of the viewport, a copier and a garbage container are displayed. The bear icon, at the bottom in the middle, represents the system.

Using a mouse, the user can manipulate the graphical representations of the domain objects by pointing, clicking and dragging. At the bottom of the Gr^2 window a mouse documentation bar is presented. The bottom area of the EDWARD window is occupied by natural language interaction windows. In the left one, labeled Dialogue (where *Invoer* means input, *Uitvoer* means output), the user can enter NL commands, questions, or assertions. At the lower right is the output window labeled CLFG for the system's comments on user action.

EDWARD's CLFG (Continuous Linguistic Feedback Generator) comments on the actions the user performs in the model world presented in the window labeled Gr^2. When the user selects an object, e.g. the file *donald* in Figure 1, CLFG presents additional information about this object, in this case the file type, authors and topic: *het spinrapport van Wim en Carla over dialoogsystemen* (the SPIN report by Wim and Carla about dialogue systems). This expression is both visually displayed in the CLFG window and auditorily presented by a speech synthesizer.

Suppose the user continues his interaction by dragging the selected file icon towards the copier icon, located on the lower left in the Gr^2 window. When the selection area of the copier icon is entered, CLFG presents information about this action (again both visually and auditorily), in this case saying *Je gaat het copieren* (You are about to copy it).

3.2EDWARD's internal architecture

Figure 2 presents an overview of EDWARD's internal architecture. The arrows represent the information flow between the main components. EDWARD's central component is the *dialogue manager*. It coordinates input and output expressions and it controls the linguistic and graphics processes. The *context model,* the *knowledge base,* and the *lexicon* are maintained by the dialogue manager. In addition, the dialogue manager makes sure that the display is always up to date. The *language interpreter* and the *language generator* consult the context model, the knowledge base, and the lexicon.

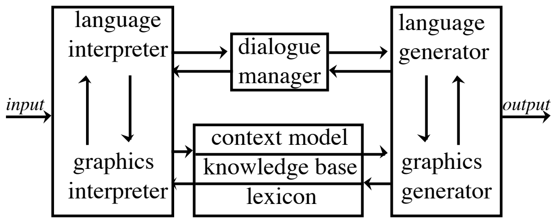

Figure 2. The internal architecture of the multimodal interface EDWARD.

The knowledge base stores the permanent generic and specific world knowledge of the system, whereas the context model temporarily memorizes which objects from the knowledge base have been referred to in the dialogue. In the lexicon, morho-phonological and syntactic features of words are represented, as are links between words and the knowledge base, representing lexical meaning.

The knowledge base is a semantic network implemented in CommonORBIT (De Smedt, 1987), a frame-based language somewhat similar to KL-ONE (Brachman & Schmolze, 1985). The nodes in the network represent classes and instances of entities and relations. E.g., the class <*person*> contains two subordinate classes <*man*> and <*woman*>*;* and the concept of sending an object to someone is represented by a generic relation called <*send*>. Individual objects in the domain are represented by instances, e.g., an individual who is a man might be represented as <*man#24*>. If he sends a message, a relation instance is created, e.g., <*send#89*>.

EDWARD's knowledge base is linked to the visually displayed model world. Firstly, EDWARD contains knowledge as to which classes of *entities* are displayed as icons. E.g., books are represented by book icons, email messages by enveloppes, project reports by document icons with a centered rectangle, but there are no icons for departments and abstract entities. Secondly, two classes of *relations* are known to be visualized, viz. *name-relations* (by means of attributed labels), and *contain-relations* (by means of arcs).

The knowledge base contains additional information about classes that is relevant for content selection in CLFG. Among this information is the *dynamic extent* of relations. Some concepts are more static (e.g., name-relations), some tend to be more sensitive to changes (e.g., icon positions), and others last only for a very short time (e.g., copy-relations). Another distinction concerns the *importance* of concepts as information conveyers within the domain. E.g., *name-relations* are more important than *icon positions:* Reference to a file is comprehended easier if its name is used (e.g. *donald*) than if its position in the model world is used (*e.g. the file just above the copier*). In the knowledge base, concepts are ranked according to this criterion.

EDWARD also knows facts about individual instances. It knows, e.g., that <*spin-report#12*> has a <*name-relation*> with the name *donald*, a <*contain-relation*> with <*directory#784*> (which has a <*name-relation*> with *claassen*), and a <*copy-relation*> with <*spin-report#63*> (which is named *gr2* and in which <*man#25*> is the *actor*, that is, the user who performed this copy action). And, of course, EDWARD also knows whether the object's icon is displayed in the current view-port, and if so, what its exact position is. EDWARD maintains a history of facts whose time span have elapsed (e.g., copy actions, old names). All this can be relevant for the formulation of expressions in CLFG, as will be explained below.

The second knowledge source EDWARD uses is the context model, which is a representation of current topics in the linguistic and non-linguistic context. The context model is a necessary requisite for interpreting and generating referential expressions with pronouns (e.g., *he, that one*). Whether a concept is a current topic in the context model is determined by its *salience*. The notion of salience has two important characteristics. In the first place, the salience of an instance at a given moment is determined by a rich diversity of factors, differing in importance (e.g., recency of mention, visibility on the screen). The second notable characteristic is that salience is a graded notion. We elaborated on the work on Alshawi (1987) who provided a general framework for modeling salience that does justice to both characteristics mentioned above. We refer to Claassen (1992) for a detailed description of EDWARD's context model. For descriptions of other features of EDWARD, e.g., the interpretation of deictic expressions or the anticipation of actions, we refer to Claassen, Bos & Huls (submitted) or Bos (submitted) respectively.

4 EDWARD'S CLFG

Now let us take a closer look at how CLFG operates in EDWARD. Two stages can be distinguished: composing the contents of the feedback (what to say), and constructing the linguistic form (how to say it).

4.1 What to say

There are two types of user action in the action mode: (1) selection of the object(s) to operate on, and (2) manipulation of the selected object(s). CLFG acts differently for each type of user action.

Selection. If the user points at an object in order to select it for action, CLFG describes this object. We believe that in order to be cooperative, CLFG must produce concise and relevant output. The algorithm we have devised to decide what to say is as follows.

First of all, if the object selected is an arc, CLFG will express the relation represented by that arc, e.g., *claassen bevat dialogue* (claassen contains dialogue). If the object selected is of a different kind, all information concerning the object is collected, i.e., all relations the object is involved in. Since this collection is usually very large, a selection must be made. Therefore, the collected relations are ordered in priority, using several heuristics. As was discussed above, the knowledge base contains several possible ranking criteria.

Initially, the *importance* information specified for the domain at hand in the knowledge base is used. This brings <*name-relation*> at the top. Next, relations that are known to provide information that is already visible on the screen are reset to the lowest priority. It seems useless for CLFG, e.g., to say *donald* upon selection of the icon with the same label. A further change in order is achieved by using the context model. It seems useless to say things that just have been said. If the user,e.g., has just sufficiently been told that something is *the report reviewed by Koen*, it does not make much sense to have CLFG say *You are about to copy the report reviewed by Koen*. Rather, using a pronoun might be more appropriate, e.g. *You are about to copy it.*

Finally, EDWARD's dialogue manager decides how many in this ordered list of relation instances will actually be presented to the user. The default number is 2 (one specifying the object's class, another a remarkable detail). However, if this number is too small to describe the object uniquely, more relations are used. If, e.g., Olga sent two emails, then using the send relation <*send#45*>, resulting in *Olga's email,* is clearly insufficient. CLFG searches for other relations making the referent unique, e.g., the topic relation <*topic#67*>, resulting in *Olga's email over Koen* (Olga's email about Koen). For a detailed description of EDWARD's algorithm to find distinguishing information, we refer to Claassen (1992).

Manipulation. Once an object has been selected for action, and is being manipulated, CLFG describes the action which is about to be performed, e.g. *Je gaat Olga's email weggooien* (You are about to throw away Olga's email). We assume that no further information about the selected object is required in this expression. However, in domains with only a few actions or with only very obvious actions, CLFG might provide additional information about the selected object, that is, present more relations from the ordered list.

CLFG describes actions by their meaningful effect in the interface language. Dragging a file icon to the garbage container icon, e.g., means deleting the file. In order to fully exploit CLFG's advantages, the feedback description of an action is best presented while the action is still ongoing. If CLFG waits until the action has been completed and the system has interpreted it and executed the corresponding command (e.g., deleting the file), the user usually needs to spend some effort to undo the effects in case of an error. By commenting on an ongoing action, errors need not be undone but can be prevented. Often, however, the meaning of an ongoing action is not obvious right from the beginning. E.g., one can drag a file icon in order to either copy it or delete it. CLFG comments on an action as soon as its meaning is clear, that is, as soon as the mouse cursor enters the selection area of either the copier icon or the garbage container icon.

Apart from semantic information, CLFG might also be enabled to provide information of lower levels of the interface language. It might provide syntactic information, e.g., *Next, move mouse cursor into selection area of copier icon;* or lexical information, e.g., *The right mouse button is pressed.* But these kinds of information are considered less helpful: most errors can be prevented by providing information about action semantics. Therefore, CLFG currently only presents semantic information.

4.2How to say it

Having decided what to say, CLFG must decide how to say it. The most important criterion in formulation is conciseness. EDWARD's NL generator meets this criterion by using the context model to construct an optimal referring expression. However, its application is not straightforward. The distinction between selection and manipulation, introduced in the previous section, must be made here as well, because both actions require different mechanisms to achieve conciseness. Upon *selection*, CLFG can be concise by using an elliptical sentence, referring to the object only instead of the entire action. There is no need to say *You selected Koen's file;* rather, *Koen's file* will suffice. But it is inappropriate to be even more concise and refer to the file with the pronoun *it,* as EDWARD's generator would do in a dialogue the user and the system are engaged in (e.g. in the lower left window of Figure 1). CLFG appears to set up its own context, and thus pronominalization is forbidden for first time reference. In subsequent phrases, however, pronominalization is allowed (e.g., *Koen's file and its copies*). Thus, when describing the second type of actions, viz. *manipulation*, CLFG can pronominalize references to selected objects (e.g., *You're about to delete them*).

For the lexico-syntactic issues that are dealt with by the Dutch generator, e.g., word order, lexical ambiguity, sentence and word stress computation for speech, as well as for the internal representation format for denoting preverbal messages, we refer to Claassen and Huls (1991).

With respect to the presentation channel, visual or audio, we decided to use both. Each has its own advantages. The main advantage of speech is that it uses a channel which is not occupied by the user during action. The user does not have to pay special attention to receive the information. The main advantage of the written CLFG output is that it remains more accessible for inspection, because it is less volatile. The Dutch speech synthesizer that is used applies the method of *allophone synthesis* (Boves et al., 1987).

It is important to note that, although the description of CLFG we have given is entirely sequential, CLFG generates its output incrementally, that is, piecemeal, in the sense that output can begin before the entire syntactic structure of the sentence or even the entire content is completely specified. The three subprocesses involved in generation, which we call conceptualizer, formulator, and articulator, run in parallel, and produce processable fragments as soon as possible (De Smedt, forthcoming; De Smedt & Kempen, 1987). The main advantage of this approach is that the user will be provided with information, albeit incomplete, much sooner, and will therefore be in a more favorable conditon to avoid errors.

5 IMPROVING THE ACTION MODE

EDWARD's CLFG dissolves some of the disadvantages the action mode suffers from, as listed in section 2.

(1) *Representation limitations.* CLFG provides supplemental information that could not easily be represented in the model world. Authorship of documents, e.g., is not presented visually, but is made available upon selection. CLFG thus prevents reference errors due to manipulation of the wrong object. CLFG also prevents the user from having to perform special actions in order to obtain another view on the model world (e.g., one in which authors are visualized, at the cost of other information, e.g., date of creation). We concede that CLFG does not entirely neutralize this disadvantage, because selection is still required in order to get additional information, and even then there is no guarantee that CLFG presents the information required.

(2) *Specification beyond the 'here and now'.* CLFG can provide information about objects outside the current viewport or outside the current temporal setting. E.g., it can easily refer to all text files irrespective of whether they are currently visible in the viewport. CLFG can also easily refer to states prior to the current one. E.g., *the directory that contained donald.*

(3) *Speech acts.* CLFG clarifies the effect of an action. E.g., by saying *You are about to copy ...,* it is clear that the interface expression the user is constructing is a directive and not a question or a comment.

(4) *Constraining reasons.* CLFG provides information on *why* a particular user action could not be completed when it is constrained by the system. E.g., if the user redirects an arc to the copier icon, CLFG responds that directories can only contain files, not copiers.

(5) *Unintended actions.* CLFG can prevent some errors and misconceptions that are due to unintended actions. If the user, e.g., clicks in the scroll area by mistake, CLFG responds *You scrolled to the left.* Thus the user won't be confused by the new view on the model world: his questions (what happened?, where am I?) are immediately answered by CLFG.

(6) *Inverse WYSIWYG.* Though we have not yet implemented this feature, it is definitely feasible to have CLFG comment on actions that result from inverse WYSIWYG. This could prevent errors. If the user, e.g., drags a node in such a way that the arc representing its *contain-relation* becomes too small to be visible, CLFG might warn that the *contain-relation* is still there, though not visible.

6 DISCUSSION

We have described a multimodal interface with a facility called Continuous Linguistic Feedback Generator that comments on user actions performed in the model world. By commenting on action, several types of errors typical for the action mode can be prevented. We have implemented a prototype demonstrating the feasibility and potential of this facility. Although all examples in this paper are drawn from the file system domain, the system is fully generic and domain independent.

At least three issues are worth further investigation. The first issue concerns multimodal descriptions. There are cases when the limitations of language sometimes become apparent. In particular, referring to specific positions is difficult. Although CLFG can use deictic expressions such as *boven de container* (above the garbage container), accurate absolute reference requires the use of a coordinate system, e.g., *the file at [104,37],* which is not always desirable. As we have already indicated, the use of multimodal deictic expressions is a suitable solution. E.g. the expression *here*↗ with an accurate simulated pointing gesture to the specific position seems better. But it must be noted that simulated pointing gestures are in the visual channel and thus require visual attention which may be occupied.

The second issue is user modeling. Currently, CLFG 'abuses' EDWARD's context model to decide whether or not the user already knows a certain fact. If something has just been said, and is therefore salient, the user is considered to know it, and therefore CLFG will prefer not to express it. However, a more detailed and better user model is required: some kinds of information are easier to remember than others, and some have explicitly been brought under the user's attention, while others, might never have been in focus. CLFG should have access to this kind of knowledge, in order to make better decisions in its content planning.

The third issue worth further research concerns the style of language production. EDWARD's generator operates in an incremental fashion. However, even though this provides certain advantages, it implies the risk that occasionally the formulator faces a dead end in mid-sentence, if the user decides not to complete an action already partially commented on by CLFG. Unfortunately, the current system is not capable of making *self-corrections* (De Smedt & Kempen, 1987), i.e., backtracking to some point and reformulating. Within our current domain, the action syntax is too simple to allow for significant changes, and there is no clear need for backtracking. In other, richer domains, self-repairs might indeed be necessary.

The next phase of the project, however, does not focus on any of these issues. We will in the near future evaluate the usefulness of CLFG in practice. We expect to find some clues for fine-tuning the algorithm for content planning. In addition, we will attempt to assess and quantify the system's error prevention potential.

ACKNOWLEDGEMENTS
This work elaborates on De Smedt, Desain, and Geurts (1987), who invented the concept of linguistic feedback in action mode interfaces and helped starting up our research project. This research was carried out within the framework of the research programme *Human-Computer Communication using natural language* (MMC). The MMC program is sponsored by SPIN Stimuleringsprojectteam Informaticaonderzoek, Digital Equipment B.V., BSO, and Sun Microsystems B.V. We wish to thank Gerard Kempen for comments on a preliminary version of this paper.

REFERENCES

1 Allgayer, J., Jansen-Winkeln, R., Reddig, C. & Reithinger, N. (1989). Bidirectional use of knowledge in the multi-modal NL access system XTRA. Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, Detroit, MI, pp. 1492-1497.

2 Alshawi, H. (1987). Memory and context for language interpretation. Cambridge (UK): Cambridge University Press.

3 André, E., Herzog, G., & Rist, T. (1988) On the simultaneous interpretation and natural language description of real world image sequences: the system SOCCER. Proceedings of the ECAI-88. London: Pitman, pp.449-454.

4 Arens, Y. (1992) Multimedia presentation planning as an extension of text planning. In: Dale, R., Hovy, E. Rösner, D., & Stock, O. (Eds.) Aspects of automated natural language generation. Proceedings of the 6th international workshop on natural language generation, Trento, Italy, April 5-7. Berlin: Springer, pp. 300-301@.

5 Bos, E. (in press). A thorough investigation of direct manipulation. Communication and Cognition - Artificial Intelligence, 9, 2-3.

6 Bos, E. (in press). A multimodal graph-editor. In L. Neal & G. Szwillus (Eds.). Syntax-directed editing. New York: Academic Press.

7 Bos, E., Claassen, W., & Huls, C. (forthcoming) EDWARD: A multimodal interface. SPIN/MMC Research Report. Nijmegen, The Netherlands: NICI. To be submitted to International Journal of Man-Machine-Studies.

8 Boves, L., J. Kerkhoff, & H. Loman (1987). "A new synthesis model for an allophone based text-to-speech system", In J. Laver & M.A. Jack (eds.) Proceedings of the European Conference on Speech Technology, Edinburgh. Vol. II, 385-388.

9 Brachman, R. & Schmolze, J. (1985). An overview of the KL-ONE knowledge representation system. Cognitive Science, 9, 171-216.

10 Claassen, W. (1992). Generating referring expressions in a multimodal environment. In: Dale, R., Hovy, E. Rösner, D., & Stock, O. (Eds.) Aspects of automated natural language generation. Proceedings of the 6th international workshop on natural language generation, Trento, Italy, April 5-7. Berlin: Springer, pp. 247-262.

11 Claassen, W., Bos, E. & Huls, C. (1990). The Pooh Way in human-computer interaction: towards multimodal interfaces. SPIN/MMC Research Report no. 5. Nijmegen, The Netherlands: NICI.

12 Claassen, W. & Huls, C. (1991). DoNaLD: A Dutch Natural Language Dialogue system. SPIN/MMC Research Report no. 11. Nijmegen, The Netherlands: NICI.

13 De Smedt, K. (1987) Object-oriented programming in Flavors and CommonORBIT. In: Hawley, R. (ed.) Artificial Intelligence programming environments (pp. 157-176). Chichester: Ellis Horwood.

14 De Smedt, K. (forthcoming) Parallelism in incremental sentence generation. In: Adriaens, G. & Hahn, U. (eds.) Parallel models of natural language computation. Norwood, NJ: Ablex.

15 De Smedt, K., Geurts, B. & Desain, P. (1987). *Waiting for the gift of sound and vision: On natural-language sentence production in multimodal interfaces*. Paper presented at the ESPRIT Workshop on Natural Language and Dialogue, Brussels, October 1 1987.

16 De Smedt, K. & Kempen, G. (1987) Incremental sentence production, self-correction and coordination. In: Kempen, G. (Ed.) Natural language generation: new results in artificial intelligence, psychology and linguistics (pp. 365-376).Dordrecht: Kluwer Academic Publishers.

17 Frohlich, D.M. (1991) The design space of interfaces. In: Kjelldahl, L. (Ed.) Multimedia: principles, systems and applications. Berlin: Springer-Verlag.

18 Hensgens, J. (1989) Een theoretisch en empirisch onderzoek naar de gebruikers-interface van een boomeditor. (A theoretic and empirical study of the user interface of a tree-editor.) Master thesis Cognitive Science, University of Nijmegen.

19 Hutchins, E.L., Hollan, J.D. & Norman, D.A. (1986) Direct manipulation interfaces. In: Norman, D.A. & Draper, S.W. (Eds.) User centered system design: New perspectives on human computer design. Hillsdale, NJ: Lawrence Erlbaum Associates.

20 Neal, J.G. & Shapiro, S.C. (1988). Intelligent multi-media interface technology. Proceedings of the workshop on Architectures for Intelligent Interfaces: Elements and Prototypes, Lockhead AI Center, Monterrey, CA, 1988 (pp. 69-91).

21 Searle, J.R. (1969) Speech acts. Cambridge: Cambridge University Press.

22 Schmauks, D., & Reithinger, N. (1988) Generating multimodal output - Conditions, advantages and problems. Proceedings of the COLING-88, Budapest 1988, pp. 584-588.

23 Stock, O. (1991). Natural language and exploration of an information space: the ALFresco interactive system. Proceedings of the Twelfth International Joint Conference on Artificial Intelligence, Sydney, Australia (pp. 972-978).

24 Wahlster, W., André, E., Bandyopadhyay, S. Graf, W., Rist, T. (1991) WIP: The coordinated generation of multimodal presentations from a common representation. DFKI (University of Saarbrücken) Technical Report.

25 Walker, M.A. (1989) Natural language in a desktop environment. In: Salvendy, G. & Smith, M.J. (Eds.) Designing and using human-computer interfaces and knowledge based systems. Proceedings of the HCI International '89, Boston, MA. Amsterdam: Elsevier Science Publishers B.V. (North-Holland).