An Iterative Design Methodology for User-Friendly Natural Language Office Information Applications

J. F. KELLEY
IBM Thomas J. Watson Research Center

A six-step, iterative, empirical human factors design methodology was used to develop CAL, a natural language computer application to help computer-naive business professionals manage their personal calendars. Input language is processed by a simple, nonparsing algorithm with limited storage requirements and a quick response time. CAL allows unconstrained English inputs from users with no training (except for a five minute introduction to the keyboard and display) and no manual (except for a two-page overview of the system). In a controlled test of performance, CAL correctly responded to between 86 percent and 97 percent of the storage and retrieval requests it received, according to various criteria. This level of performance could never have been achieved with such a simple processing model were it not for the empirical approach used in the development of the program and its dictionaries. The tools of the engineering psychologist are clearly invaluable in the development of user-friendly software, if that software is to accommodate the unruly language of computer-naive, first-time users. The key is to elicit the cooperation of such users as partners in an iterative, empirical development process.

Categories and Subject Descriptors: D.m [**Software**]: *software psychology*; H.1.2 [**Models and Principles**]: User/Machine Systems—*human factors*; I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems—*natural language interfaces*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*language parsing and understanding*; I.6.3 [**Simulation and Modeling**]: Applications; K.6.3 [**Management of Computing and Information Systems**]: Software Management—*software development*

General Terms: Experimentation, Human Factors

Additional Key Words and Phrases: Natural language, limited context, naive user, discretionary user, iterative design, simulation, user-friendly, ease-of-use, empirical grammar, task analysis, engineering psychology.

1. INTRODUCTION

Computerized office applications employ one of three modes of input: menus, command languages, or natural language. According to one view, the suitability of a particular mode of input depends on the "semantic" and "syntactic knowl-

This work was mostly supported by a research agreement with the IBM Corporation Systems Products Division, under the supervision of Prof. A. Chapanis, Dept. of Psychology, The Johns Hopkins University. Portions of this material appeared in the *Proceedings of the CHI '83 Conference on Human Factors in Computing Systems*, Dec. 1983.
Author's address: IBM, Thomas J. Watson Research Center, Room ST-K40, P.O. Box 218, Yorktown Heights, NY 10598.

edge" of the user [11]. Menus might best serve a person who is not familiar with the computer's command structure (low syntactic knowledge) and who is uncertain how to proceed in solving his particular problem (low semantic knowledge); the menu tells the person which responses to make at any given time, leading him through the problem-solving process. Command languages are good for people who know what steps they want to undertake to solve a problem and are familiar with the computer's syntax for accomplishing each step. This view holds that a natural language interface might be appropriate for people who have a high level of semantic knowledge in a problem domain, but aren't familiar with any special computer syntax for achieving their goals.

While experts are pessimistic about the general implementation of a program that can "understand" and respond meaningfully to unrestricted natural utterances on a range of topics from naive users, such a program would certainly entice "a large number of people who are potential computer users [but] are unwilling to learn and then use a formal machine language." ([9], p. 314).

The principal purpose of the research reported here was to design and test a systematic, empirical methodology for developing context-dependent natural language computer applications. This paper describes that methodology and its successful use in the development of a natural language computer application: *CAL*, *C*alendar *A*ccess *L*anguage. The limited context or domain in which the application operates is the management of a personal calendar, or office appointment database by computer-naive business professionals.

Designers of natural language systems have, in the past, begun with a comprehensive model of the language expected in a particular domain and have built up recognition systems from there (see, for example [10, 13, 14, 15]). Contrary to that "armchair" method of program development, this research involves an iterative empirical development approach (which has recently been aptly discussed in [3]).

2. METHODOLOGY

The development process for CAL comprised six steps [5]. Central to the methodology is an experimental simulation which I call the *OZ* paradigm, in which experimental participants are given the impression that they are interacting with a program that understands English as well as another human would. In fact, at least in the earlier stages of development, the program is limping along, only partly implemented. The experimenter surreptitiously intercepts communications between participant and program, supplying answers and new inputs as needed. The six steps of program development are follows:

(1) *Task analysis.* Twenty-three business professionals were interviewed extensively to discover how they keep their appointment calendars. That information provided a starting point for the functional specification of a computerized calendar [4]. For the majority of the persons interviewed, calendars are indispensable to the conduct of their business and, in some cases, their personal lives. The data from this step show an unexpectedly large amount of diversity in the kinds of calendars people use and in the ways they use them. Substantially more than half of the respondents have more than one calendar, with two persons using as many as six calendars at once. Portability and access from diverse

locations are important for many. Concerns about privacy vary widely: some keep their calendars closely guarded, others allow free access to them. Relevant time spans covered by calendars are enormous. Some few people are concerned only with the current day and the day following, others may plan appointments a year or more in advance. A substantial number of appointments are changed after they have been made and, once again, the range is large, from about 2 percent for some persons to about 80 percent for others. Archiving, query patterns, and the insertion of correlated information into calendars also vary greatly among users.

(2) *Deep structure development.* In this second step of program development, the database manipulating functions were written in APL.

(3) *First run of OZ (simulation).* Here, *no* language processing components were in place. The experimenter simulated the system *in toto.* This simulation is similar to the ones used in [2] and [12].

(4) *First-approximation language processor.* The corpus of inputs obtained in step three was used to develop a first approximation of the language processing subroutines (described in [6]).

(5) *Second run of OZ (intervention).* This was the iterative design phase of program development. Fifteen participants used the program, and the experimenter intervened as necessary to keep the dialog flowing. As this step progressed, and as the dictionaries and functions were augmented, the experimenter was phased out of the communications loop.

(6) *Cross-validation.* The final program was tested with six additional participants to see how well it performed. In this step the program ran without any assistance from the experimenter. Various measures of program speed, "understanding," and efficiency were combined with the results of postsession interviews to evaluate CAL's success.

2.1 Apparatus

Participants and experimenter communicated via IBM 3277 displays and keyboards to an APLSV program residing in an IBM 370/168 host system. During the simulation and intervention steps of the development (first and second runs of OZ), all communications between the participant and the program were channeled via shared variables and appropriate software through the experimenter's workspace. Both participant and experimenter, working in separate rooms, communicated with the host system via a Bell System 4800 baud modem and an IBM 3271 controller. A two-way push-to-talk intercom was provided for voice communication between the participant and the experimenter. The keyboard used by the participants in the iterative design phase of OZ and in the cross-validation was modified by abbreviating the available editing functions to include only backspace, forward cursor motion, and deletion of one character at a time. Aside from the RESET and ENTER keys, all other function keys were masked and disabled.

In addition to his own communications terminal, the experimenter had a tandem terminal which was slaved to the participant's own and which echoed his or her keystrokes and displays. The slave terminal was useful in two ways. Its primary use was to prepare the experimenter by giving him an advanced view

of the message being composed by the participant. On a few occasions, the slave terminal allowed the experimenter to rescue the participant from certain difficult situations surreptitiously (e.g., correcting the cursor position and pressing the RESET key when the participant attempted to type in a nontyping area and had consequently "locked" the keyboard).

2.2 Problem Solving Task

Participants were asked to tell the computer about whatever routine and nonroutine appointments they had in the next two weeks or so. They were asked to try and get at least ten appointments into the computer. Pilot work showed that this minimal goal was sufficient to provide some focus for the participant in "trying out" the system. If the participants did not do so on their own, the experimenter called them on the intercom and dictated changes to, or searches of, their appointment calendars (e.g., "let's pretend that Bob Jones just called and wants to reschedule his 3 o'clock appointment with you to sometime on Thursday; when would be good?"). Thus the experimenter assured that each participant would attempt database retrieval, manipulation, and storage using his or her own data. Aside from the minimal goal provided for the participants, and the minimum prompting required to solicit changes to the data when necessary, an effort was made to avoid giving too much task structure to the participants for fear of priming them with language that was not their own. A two-page overview of the system was provided showing examples of how CAL could be used to store, retrieve, and change appointments. To control for the possibility that material in the overview might affect the language generated by the participants, questions about the form and frequency of manual use were included in the postsession interviews.

2.3 Participants

Consonant with the purpose of this research, an attempt was made to sample participants from various walks of life. Examples of professions included in the participant pool were Jesuit priest, symphony conductor, auto repair manager, real estate saleswoman, clothing store owner, clinical psychologist, architect, dental assistant, flight instructor, homemaker, bank manager, attorney, and an appointments secretary to a US senator. The participants represented a range of computer experience, with most having little or no experience whatsoever. In addition to these laboratory/interview participants, two computer industry professionals at the sponsoring organization were given access to the developmental versions of CAL. They used the program at their leisure, in an unmonitored mode (no experimenter, no interventions). Some of their dictionary entries and suggestions for improving the program were implemented during the iterative design phase of development.

2.4 Procedure

Each participant took part in a single experimental session. After a background questionnaire was filled out, a short (5 minute) interactive keyboard tutorial was run by the experimenter and the participant together. This introduced the participant to the concept of communicating with a computer program (e.g.,

"what is an ENTER key?") and to the abbreviated editing functions available on the modified 3277 keyboard. After the introductory tutorial, the participants were advised that the experimenter would be in the next room "keeping more or less of an eye on the printout of the session," and that they could call him on the intercom if they had any questions. They were not told of any potential participation on the part of the experimenter in the communication loop, nor were they advised of the existence of the slaved tandem terminal echoing their every keystroke in the experimenter's room.

During the iterative design (intervention) phase of OZ, the experimenter intervened in the session when the fledgling system made mistakes. After each of the intervention iterations (sessions), the dictionaries and programming of CAL were augmented and enhanced to accommodate the grammatical structures and functional requirements of the inputs from that session. In addition, a batch-type program was written allowing CAL to be subjected to a large corpus of difficult inputs from previous sessions to make sure that changes made in the programming or dictionaries would not interfere with the previous capabilities of the system.

After each of the 15 participants took part in an intervention session during the iterative design phase of CAL's development, the experimenter made the decision that the development had reached the point of diminishing returns (a judgment that was borne out in examination of the approach to asymptote dictionary growth, discussed in the Results section below). Very few interventions were required at this point, and the experimenter removed himself entirely from the communications loop, switching over to step 6 of the development (i.e., the cross-validation). Six participants were exposed to the same procedure as before, but CAL was no longer enhanced or augmented after each session. It was on the basis of data collected during this step that the performance of the program was assessed.

3. RESULTS

3.1 Participants' Performance

In the cross-validation phase of CAL's development project, six experimental participants spent an average of 65 minutes interacting with the program, entering a total of 2,429 words to perform 155 actions: describe a total of 87 appointments, ask for 32 displays of portions of the database, and initiate 36 changes in old appointments, among other things. Some actions required more than one message or input. The participants typed in 59, 50, 41, 80, 43, and 54 messages, respectively. Within those 327 messages, a total of 457 time phrases were recognized. A total of 161 context-establishing phrases (e.g., "I have an appointment ...," "Change my appointment with ...," "When do I," "Never mind") were recognized as well. Appendix A gives examples of user inputs that were correctly processed by CAL. Appendix B shows some inputs that the program failed to correctly process either during the iterative or cross-validation phases. A straightforward extension of the development methodology would have made CAL able to handle most of the examples. Some of them (such as the one-step change request) would have required a more extensive effort.

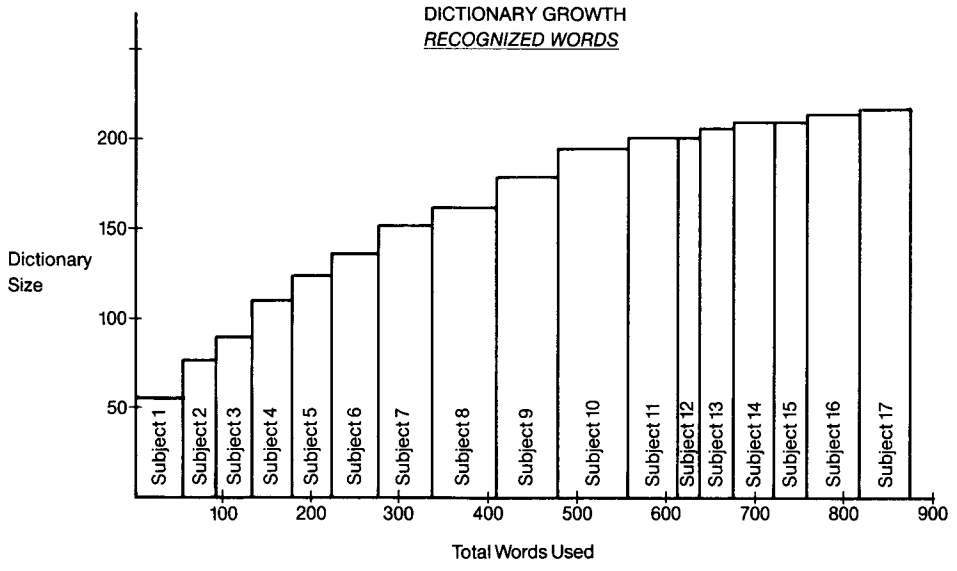

Fig. 1. Word dictionary size as a function of time (total number of word types entered into the system). The vertical divisions in this chart represent participant boundaries.

The participants took an average of 58 seconds to compose and enter a message; CAL responded within a few seconds.

3.2 Program Growth

Including all of the dictionaries and the programs for data collection and communications intervention, CAL occupied about 100K bytes of computer storage.

Only ten one-hour sessions or iterations brought dictionary growth to the point of diminishing returns during the iterative development phase, as evidenced by Figure 1. The figure shows the growth in number of unique, recognized words (types) in the master word dictionary as a function of time (as measured by the total number of word types entered into the system) as the development phase progressed through its iterations. The vertical divisions in the chart of Figure 1 represent participant (session, iteration) boundaries. Thus we can see that the first participant used 55 unique words (types) and added that many to the word dictionary; the second participant used 40 unique words, contributing 22 new entries to the dictionary. A chart of the growth of recognized word synonym categories would show a similar quick approach to asymptote.

Figure 2 shows a similar growth pattern for the number of recognized time phrases used by participants as the development proceeded. Another perspective on dictionary growth comes from the analysis of overlaps among the sets of words used in each session. Each participant contributed, on the average, 1.91 unique words to the total pool (mode = 1). This represents a measure of the acceleration of dictionary growth at asymptote; it means that most people used only one or two words that no one else used during the development and cross-validation phases. Kelley [6] contains a comprehensive description of CAL's word and phrase dictionaries.

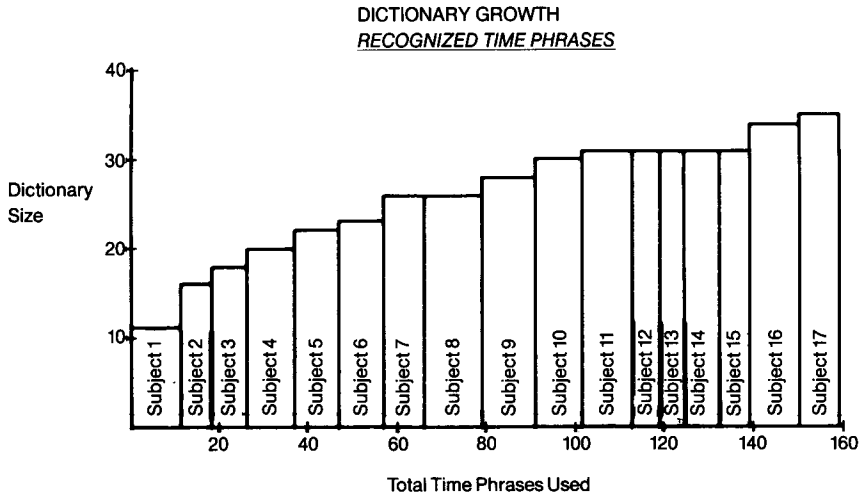

Fig. 2. Phrase dictionary size as a function of time (total number of word types entered into the system). The vertical divisions in this chart represent participant boundaries.

3.3 Program Performance

Errors. The participants' task was to enter some appointments into the computer and then query and update that database. The six participants in the cross-validation group stored 87 appointments in all. During that process, there were three occasions on which the program's failure to correctly process an unambiguous input resulted in the storage of an incorrect appointment. These all occurred when one participant typed in times without colons (e.g., "630" instead of "6:30"). I consider this a program error rather than a user error because I feel that this input would be unambiguous to a human being. As such, it should have been clear to CAL. A liberal estimate of CAL's "understanding"[1] based on this metric would be 97 percent.

More conservative estimates of the program's performance would take into account less serious breakdowns in understanding (i.e., those which do *not* result in incorrectly stored data). There were two times that the program actually stored an incorrectly processed appointment, but the user noticed the error (CAL displays its interpretation when it stores an appointment, even if it has no clarification questions) and subsequently corrected it. The two errors would bring the estimate of CAL's failure rate up from 3 percent to 5.7 percent. On two occasions the program failed to correctly process the time description in an unambiguous input, recognized that it was confused, and abandoned the attempt to store the appointment. In the first case, the user left out a space between two words: the month and the date. In the second case, CAL caught its own error when a participant entered a multiday appointment in an unrecognized format. It happened that none of the development participants had ever typed in a phrase

[1] I use the term here in the restricted sense used by Winograd [14] and others. Computers don't really "understand" anything in the same way people do.

in the form *month date to date* (e.g., July 15 to 29) before, and the time phrase dictionary had no corresponding entry. Including these as program errors brings the performance level down from 94 percent to 92 percent. There were five occasions when CAL made errors of varying magnitude, but, accommodating the possibility that there *might* be something wrong in its interpretation, engaged the user in a successful clarification dialog. This represents a cumulative error rate total of 14 percent. It is not clear whether these represent errors on the part of CAL, since the program acted just as one would expect a competent human appointments secretary to act when faced with a communications failure.

Efficiency. An analysis of the printouts of the cross-validation sessions showed that, on the average, 1.68 inputs were required to successfully store an appointment and 2.96 inputs were required to locate and change one (simple retrieval requests were typically one-step operations). A simplistic view of these numbers is as follows: about half of the times that an appointment is entered, CAL engages the user in further dialog in order to confirm its understanding of the input or, in a few cases, to warn of a potential scheduling conflict. Also, a change of an appointment usually requires three inputs, except for the few times (seven, to be precise) that users figured out how to skip a step taking only two inputs, and the fewer times (four) that the change became complicated requiring four or five steps.

3.4 Postsession Interviews

After each session, the 15 developmental participants in the intervention phase of OZ and the six cross-validation participants were interviewed on 16 topics. Not every participant had a response for every question. Due to the open-ended and in-depth nature of these interviews, the respondent was free to skip around or go off on a tangent. The interviewer's job was to try to bring the discussion back to the issues or categories at hand. In cases where statistics are provided—summarizing the participants' feelings about a topic—they are based on a subjective analysis of the transcripts of the audio tapes of interview sessions. Where the cross-validation participants' opinions, as a group, seem to differ from the attitudes expressed by the developmental participants (and where that difference is deemed relevant), the discrepancy is described.

The interview results are presented here with several goals in mind. First, comments on reaction to "bugs," response time, and experimenter interference could act as pointers for others contemplating use of the simulation techniques reported here. Second, individual comments on CAL's style and mode of operation might prove useful in the design of other office applications. Third, some of the comments point to potential strengths and weaknesses of natural language as a mode of input.

Effectiveness of simulation. One of the interesting things to come out of the less formal postsession interviews in the simulation phase of OZ (step 3 in the methodology), and borne out in the later steps, is the fact that participants quite readily accept the low-level deception inherent in the OZ paradigm. In spite of an occasional spelling error or other human fault in the "computer output" simulated by the experimenter, no subject ever seriously questioned the propo-

sition that there was a computer acting alone on the other end of the line. This relates to Weizenbaum's observation [13] that human parties to a communication interaction attribute all sorts of world knowledge and understanding to their partners. It almost seems to require a positive effort to convince participants that there is less to the computer program than meets the eye.

Utility. Three respondents felt that their old paper and pencil way of doing things was better than a computerized approach could ever be ("I think my little calendar that I have at home is much easier."). Seven saw advantages to CAL, but would want changes made (such as portable terminals and/or daily printouts) before using it for themselves. Nine participants were unequivocal in their praise of CAL's potential.

Keyboard and display. Thirteen participants volunteered their dismay with the locations of the ENTER and RESET keys on the IBM 3277 keyboards—right where their fingers expected to find the SHIFT keys. The other six respondents in this category did not. Aside from this, and a problem a few people had in hitting the carriage return (next line key) instead of the ENTER key, most participants found the pared-down keyboard/display system "real easy." One participant suggested that CAL allow semicolons in place of colons in times because shifting to upper case is "just one extra step that I like to take out." (This suggestion was accommodated in a later revision of the program.)

Output language style. The consensus was that CAL's output language was "very polite" and "friendly." "The friendliness level was appropriate, not overly unctuous." One computer-unsophisticated participant thought that CAL "was kind."

Input flexibility. "I was surprised when I said 'show calendar' and it did it. I thought I might have to say 'write calendar' or 'print calendar', but 'show' worked. 'Print' was the established term [on the one other computer I have used]." "It's more flexible than I thought it would be . . . I found it . . . refreshing that it is on a more human level [than other computers I have used]." The prevailing view was that CAL was flexible in terms of the variations it would accept. A few people (none in the cross-validation group) found the program "fairly demanding . . . if you didn't have it just right, it said 'I don't know what you're talking about'."

Perceived comprehension. In this category (which interacts somewhat with the previous one), three respondents (including one cross-validation participant) felt that CAL's comprehension of the English language left something to be desired ("I think that the [presence or absence of a] colon should not make that much difference . . . '630 pm' can't be anything but a time." (see page 32). Three cross-validation participants and one other were impressed with the program's performance ("It seemed to understand English very well"), and five people (including one cross-validation participant) were equivocal, seeing both good points and bad.

Perceived accuracy. Four members of the cross-validation group and four others said they were confident enough in CAL's accuracy to entrust their

appointments to the computer. One cross-validation participant and four others suggested that they would prefer a little more experience with the program first. One developmental participant was worried about finding "quite a few inaccuracies."

Use of "manual"/examples. While there was no manual provided per se, the two-page overview of the system contained some examples of language. Thirteen people (including the entire cross-validation group) mentioned that they felt good about experimenting with the system and trying out different ways of entering things (for other relevant work on exploration in learning, see [1] and [7]). One developmental participant specifically stated that he did *not* feel like experimenting with the system and would rather read about it ahead of time. Half of the participants in both groups indicated that they relied on the examples in the two-page printout. The other half "didn't look at it once," or turned to the examples on the rare occasion when they couldn't figure out how to phrase some input (usually one of the less frequently used inputs, such as the change or query requests). Though few were specifically asked, many of the participants seemed to find it reassuring to have a short description of the program, even if they didn't use it. Four cross-validation participants and three others mentioned modeling parts of their inputs on CAL's output language. One cross-validation participant specifically said that he did not do that.

Estimated training time. Most participants acknowledged that they were learning about the system as the one-hour session progressed. Three people said that they were comfortable enough with the system by the end of the session to use it (one felt that way after "10 or 15 minutes"). Seven thought that anywhere from 30 minutes to a couple of hours or half a day more would do the trick. Three people seemed to think it would take a week or two of regular use before they were comfortable with the system. No one thought that remembering how to use CAL after an absence of a week or so would pose any problems.

Reaction to "bugs". Seven participants commented that the "bugs" or unexplainable problems that sometimes crop up with computers didn't affect their attitudes much. Three people who encountered system or application software problems during their session volunteered that "my first impression was I must have typed in something wrong. I didn't blame the computer." Two people said that they were made "frustrated" or "nervous" by problems with the computer.

Response time. Half of the cross-validation participants and ten others found CAL's response time "too slow," several feeling that the response "really should be instantaneous." (The response times during the iterative development phase were longer than in the cross-validation, owing to the time lag involved in interventions.) The other half of the cross-validation group and four members of the developmental group felt that "it didn't bother me at all." One cross-validation member said, "it didn't really leave me waiting. It said 'please wait,' it's a nice machine."

Program assumptions. Human parties to communication are able to make inferences and assumptions, thus filling in missing pieces of information. CAL is endowed with some minor examples of the same ability. In order to lighten the

load on the user, CAL can proceed with an incomplete specification of a time interval, inferring the missing pieces on the basis of knowledge in its own world model (e.g., one such item of knowledge is that an appointment from "11 till 2 tomorrow" is probably from 11 *am* till 2 *pm*). While this feature of CAL might save user keystrokes, there is one potential negative impact in the exercise of this ability: CAL may guess wrong in some instances. The program *could* have been designed to double check each assumption it makes with the user, but that would add more work than it would save. How do users of the program feel about this? Five respondents are of a negative opinion: "I don't like a machine telling me what I think, which is what it's doing." Twelve felt that "it doesn't bother me at all" and "I definitely prefer the shorthand" of being able to specify times incompletely. One even went so far as to say: "I kind of like its spunk."

Dialog initiative. "I did begin to feel that I, after a while, could control it instead of it controlling me." "Often you have the computer print out 'is the time right?' or 'is this day correct?' and that's a good idea." Many people noticed the places where CAL would assume more control over the dialog (when it needed specific, mandatory, pieces of information, for instance), but that was not felt to be intimidating. In fact, some people appreciated the increased structure at confusing moments: "Actually, it's easier because it makes you think in patterns and whenever you think in patterns, you think faster; and the faster you can think, the more work you can get done."

Experimenter interference. The participants were told at the beginning of the session that the experimenter would be keeping an eye on the printout, but none of the participants maintained that awareness, or, if they were conscious of the indirect presence of the experimenter watching their progress, it did not affect them: "Once I got into CAL, I didn't think, really, about too much else. I was having too much fun."

Ease-of-Use. "It was very comfortable to use." "Once they know what they're doing with it, I think it would be very easy to use."

Grammatical quirks. Due to the simplicity of CAL's grammatical model, it sometimes makes minor errors in extracting the descriptions of an appointment once the time references are removed (i.e., the occasional stray comma or word finds its way into the description). In addition, there is sometimes a confusion of plurality when lists are printed ("Here are appointments number 1 . . ."). Three members of the development group and one member of the cross-validation group mentioned that they would probably go back and correct minor problems in the appearance of their stored appointments. Three people (one of whom was from the cross-validation group) did not notice CAL's "pidgin English" and 14 people did not care one way or the other. As the computer-naive psychiatrist put it: "I guess I don't expect a whole lot of sophistication from a computer."

General favorable comments. "I don't feel very comfortable with video games or anything like that. It makes me feel very tense, but this—I didn't feel tense at all." "I love it. At least it listens, it's better than most people!" "I really enjoyed it, thoroughly enjoyed it." "In fact," one computer-naive psychiatrist commented, "I was sitting here thinking . . . for the first time I thought it might be sort of fun to have a home computer."

4. DISCUSSION

CAL is a relatively small program that operates with a very unsophisticated language model. Yet, it did well in a controlled test of many aspects of its performance. This success is directly attributable to the empirical nature of the design process that gave birth to the program. How did the key phases of the design process contribute to this success?

The *task analysis* was indispensable for everything that followed. Its purpose was to determine what functions the computer application must have (i.e., what exactly the program is supposed to do). A single designer cannot adequately anticipate the needs of a population of users without consulting representatives of that group. For example, from the perspective of software design, it was easy to think of a program that, for the sake of advanced function, might require an ending time, however tentative, for each appointment entered by the user. However, it turned out that the participants in the task analysis found the prospect of such a demand totally unacceptable. Some people simply refused to think about the possible ending times for future appointments, at least until the appointment times drew near. Thus, I had to rethink my notions of the absolute nature of time specifications right from the outset.

A key role of the *simulation phase* was to provide a basis upon which to build the initial grammar. In contrast to previous natural language programs (e.g., LUNAR [15], SHRDLU [14]), CAL was not built on a model of a prespecified grammar. Rather, CAL uses what I have chosen to describe as an *empirically derived grammar.*

An unanswered question at the beginning of this project was whether this simplistic method of building a language model would be a never ending process of bringing people into the lab and augmenting the model to accommodate their new ways of saying things. It turns out that such is not necessarily the case, as evidenced by the limited number of iterations (10) that were required to reach asymptote.

It was surprising that a point of diminishing returns was reached after so few iterations (i.e., that each participant used only one to two words that no one else used). This result was not obvious in light of previous findings by Michaelis, Chapanis, Weeks, and Kelly [8] in their study of human-to-human problem-solving communications in three other contexts. That study showed a very small common vocabulary used by participants in each problem-solving domain. Among the many potential explanations for this difference (e.g., substantive differences between their problem domain and the calendar-keeping domain) there are two compelling possibilities: first, the participants in the Michaelis, et al. study were probably operating in a less familiar problem-solving area and, having lower levels of "semantic knowledge," tended to be more erratic in their language behaviors. Another possibility is that when people are (or think they are) communicating with a machine, as they were in the CAL study, they might tend to "normalize" their language (i.e., use fewer uncommon words).

It appears, at least in the calendar-keeping context, that people engaged in person–computer problem-solving tend to say things in predictable, systematic ways (witness the fact that the empirical approach was sufficient to prepare the computer for future inputs). However, a brief look at the unruly examples in

Appendix A shows that those predictable, systematic ways don't necessarily coincide with what is traditionally thought of as grammatically proper. The structure in these inputs is accessible from the empirical approach described here, but wouldn't necessarily be amenable to analysis by more traditional grammatical processing routines.

A natural consequence of the use of an empirical grammar is the inability to generalize the obtained language model beyond the context in which it was developed.[2] However, though CAL's grammar cannot be generalized, the systematic approach used to generate it can. The six steps of program development used here can just as easily, and presumably with comparably little investment in participant-hours, be applied to other office applications where natural language is appropriate.

During the *iterative design phase*, breakdowns in communications were not blamed on "user error," but were thought of as failures of CAL (or, more appropriately, of CAL's designer) to anticipate all the necessary variations in input structure. If people find it natural to express times with semi-colons rather than colons (and thus avoid the SHIFT function of the keyboard), and if that usage doesn't generate any unresolvable ambiguity (it doesn't), why force them to use colons? It doesn't cost anything to add that flexibility to the program.

While there were some informal long-term users of CAL, a more formal longitudinal study would be necessary to shed more light on how this natural language interface holds up with dedicated users over time. This would also provide an opportunity to give more thorough consideration to information retrieval in a realistic setting (e.g., the ways in which people check their calendars at the beginning of each day).

5. CONCLUSION

This study has shown several things about calendars, natural language, and software design. The task analysis [4] showed that calendars are indispensable in the office environment and that they are good candidates for computerization. While no controlled comparison was made with other modes of input, the interview results do indicate that natural language does hold some promise as an input mode, at least for the semantically knowledgeable, computer-naive business professionals represented in this study. Finally, the objective program performance results show that the object of CAL's design was met. Computer-naive users were indeed able to sit down at a terminal and have meaningful interactions with a computer, in their own natural language, from the very outset. Most of the participants in this experiment, including those who had expressed much trepidation over the prospect of dealing with computers, went out of their way to tell me how enthusiastic they were about the program and their accomplishments with it.

APPENDIX A. Inputs Correctly Processed by CAL

Comments (surrounded by angle brackets ⟨ ⟩) follow some of these examples. An asterisk (⟨*⟩) denotes inputs that were correctly understood by CAL but that

[2] Actually, that component of CAL's language model which deals with the interpretation of natural language date and time inputs would, in some degree, be generalizable to other contexts that involve dates and times (i.e., inventory control, payroll).

caused the program to double-check its understanding (e.g., "Please check the following time period... are the times of day correct?"). The use of lower-case "l" for "1" is noted by upper-case "L."

Storing Appointments with Beginning Time Only

July 13 9:45 Bob Jones ⟨*⟩
Weds, July 14,1982 Meet Susan Johnson at 2 p.m.
7/11/82 BSC Concert at 7 pm
get Missy from school at 3:30 pm on 4/29/82
Pick up Missy at 3:30, afternoon of April 29, L982
Tues: July 13, 1982 10:A.M.Call John Wilson at 555-3545
1:00 PM on Monday 6/14/82: production meeting
6/2/82,11:30 am ,lunch
tomorrow, call john at 7:30am
"Pick up Card" Vern's Birthday, Jul 11,1982 11:00 A.M.
July 13 5:00 PM Bill Jones "Prft Shrng"
Meeting with boss next Thurs. 9am
jon for dinner ,5:30 pm, tommorow ⟨note anticipated misspelling of "tomorrow".⟩
Appt, also July 15 See Dr. Wilson at 6:45 pm for dentist appt. (be sure to take papers)
3:30 p.m. Check with Valley Lighting to see if they need an order ⟨This input caused CAL, quiet properly, to ask for a date. The participant responded with the following:⟩
This appointment is for Tues.July 13, L982
Mon, Sep L3 Classes and lessons begin at Peabody;Orch. Rep. Session at 4:00 p.m., North Hall EP

Storing Appointments with Beginning and Ending Times

voice lesson 5:00 pm until 6:00 pm on 6/22/82, tuesday
6/15 "bsc" 7;30 to 10 pm
7/17/82 9:30 am till 12;30 pm Meet with Andy and Pete to see Walton load ⟨Note the use of a semi-colon in the second time.⟩
Mon 7/19/82 Call Bob Stevens 9:00 am till 9:30 am
"Chamber Singers Rehearsal" 7/19/82 from 7;30–10pm ⟨*⟩
7/15/82 13:00 THRU 15:00 GENERAL STAFF MEETING; 202-99 EEA
Wed meet in room 2–3 from 4 til 5
Friday, July 16, all day: dad's birthday

Storing Routine Appointments and Reminders

"Advanced Concepts Seminar" 3–4:15pm every thursday 9/2/82 through 5/26/83
"Lunch with the chief resident" every tuesday from 12–1pm beginning 7/13/82 and ending 6/28/83 ⟨*⟩
every Thursday from July 1 thru September 30, 1982; BME lab seminar—1:00 pm to 2:00 pm.
"bsc at trinity" 6:30 pm–10 pm every tue tomorrow through jun 12, 1984
Please enter every Friday from 8:00 p.m. to 9:30 p.m. for the remainder of the year that Angel has Girl Scouts.
7/17/82am "Vacation until sometime 7/24/82 ⟨*⟩
remind me send parents anniversary card
remind me 8/11/82 send birthday card to mama
remind me to "go to bank" on Friday
remind me to "buy birthday card for Pauline" next week
4/29/83 reminder to call Walt Anglo's potential tenant

Query and Change Requests

show cal
show cal Tuesday
show me July Appointments

show me my calendar from 7/11/82 until 8/31/82
show me all trinity dates
show meeting with boss
when is prft shrng
change 10
change 2 and 3
change Thomas
cancel 7 8 9 10 11 12
help
what day is it?

APPENDIX B. Inputs That Were Not Recognized by CAL

Comments (surrounded by angle brackets ⟨ ⟩) follow some of these examples, which represent failures of CAL in both the cross-validation phase and the development phase.

Tues July 13 9:45 AM Bob Jones 13 10:AM Bob Jones
change handball to no known ending time
show me the appointments so far for this year
July 15 ii:30 "handball Smith"
change Prft Shrng to Sept 12 10:AM
supervision with Dr. Smith every monday from 10 AM–11AM beginning 8/30/82 and ending after 6/83 ⟨No entry existed in the time phrase dictionary for the syntactic structure "6/83" because none of the development participants used that form (and the developer did not think of it). This is one of the very few times this occurred in the cross-validation.⟩
july 16 730–10 pm chamber rehearsal Trinity
change3 to read friday july 16 730 pm–10–pm chamber rehearsal trinity
Tues Jul13, 7:00 am–11:00 am—have computer signed out. ⟨CAL is usually pretty good at separating numbers from words when spaces are inadvertently left out. However, the program is also pretty good at accepting "L" for "1" (many people have learned to type on machines which *have* no "1" at all and have learned that they *must* use lower case "L") and those two provisions interact in this peculiar example.⟩
4/30/82 at 4:30 pm take Dick Jones back thru 3830-22 Latvia Rd ⟨To CAL, "22" looks a lot like "until 22:00".⟩
remind me the last day of every month through 4/30/2007 "mail mortgage check"
"Conference on the Ward: Tuesday 7.13.82 4:00 P.M.

REFERENCES

1. CARROLL, J.M., AND MACK, R.L. Learning to use a word processor: by doing, by thinking, and by knowing. In *Human Factors in Computer Systems.* J.C. Thomas and M. Schneider, Eds., Ablex, Norwood, N.J., 1984.
2. GOULD, J.D., CONTI, J., AND HOVANYECZ, T. Composing letters with a simulated listening typewriter. *Commun. ACM 26,* 4 (1983), 295–308.
3. GOULD, J.D. AND LEWIS, C. Designing for usability: key principles and what designers think. Res. Rep., IBM Thomas J. Watson Research Center, 1983.
4. KELLEY, J.F., AND CHAPANIS, A. How professional persons keep their calendars: implications for computerization. *J. Occupational Psychol. 55* (1982), 241–256.
5. KELLEY, J.F. Natural language and computers: six empirical steps for writing an easy-to-use computer application. Unpublished PhD dissertation, The Johns Hopkins Univ., 1983 (Can be obtained from University Microfilms International, 300 North Zeeb Road, Ann Arbor, MI 48106.)
6. KELLEY, J.F. CAL—Calendar Access Language: an APL program for processing natural language. IBM Res. Rep., in preparation, 1983.
7. MACK, R.L., LEWIS, C.H., AND CARROLL, J.M. Learning to use word processors: problems and prospects. *ACM Trans. Office Inf. Syst. 1,* 3 (1983), 254–271.
8. MICHAELIS, P.R., CHAPANIS, A., WEEKS, G.D., AND KELLY, M.J. Word usage in interactive

dialog with restricted and unrestricted vocabularies. *IEEE Trans. Professional Commun. PC-20,* 4 (1977), 214–221.
9. PETRICK, S.R. On natural language based computer systems. *IBM J. Res. Dev. 20* (1976), 326–334.
10. RAPHAEL, B. SIR: a computer program for semantic information retrieval. In *Semantic Information Processing.* M. Minsky, Ed., MIT Press, Cambridge, 1968.
11. SHNEIDERMAN, B. *Software Psychology: Human Factors Aspects of Computers and People.* Winthrop, Cambridge, 1980.
12. THOMAS, J.C. A method for studying natural language dialog. Res. Rep. RC 5882, IBM Thomas J. Watson Research Center, 1976.
13. WEIZENBAUM, J. Contextual understanding by computers. *Commun. ACM. 10* (1967), 474–480.
14. WINOGRAD, T. *Understanding Natural Language.* Academic Press, New York, 1976.
15. WOODS, W.A. *The LUNAR Sciences Natural Language Information Processing System: Final Report.* Bolt, Beranek & Newman, Inc., Cambridge, 1972.

Received September 1983; revised November 1983; accepted December 1983