

Multi-Dimensional User Models for Multi-media I/O in the Maintenance Consultant*

David N. Chin, Mitsuyuki Inaba, Harish Pareek,
Keiichi Nemoto, Michael Wasson, and Isao Miyamoto

Software Engineering Research Laboratory
Department of Information and Computer Sciences
University of Hawaii
2565 The Mall
Honolulu, HI 96822
email: Chin@Hawaii.Edu

Abstract

The MC (Maintenance Consultant) system helps users to maintain large software systems by providing access to a set of graphical models representing different aspects of the target software system. Users can combine natural language requests with pointing and MC responds either graphically, textually (tables and natural language), or in combination. Users can also ask about how to use various maintenance tools or invoke the tools. These multi-media capabilities depend on a number of user models that track static user information such as the user's job type, programming expertise, and security access level; the user's knowledge of different tools and products based on observing the amount and frequency of usage; the user's current task based on a process model for software maintenance; and the shared conversational and visual context.

Introduction

The MC (Maintenance Consultant) system provides multi-media help and access to information for users of the SMA (Software Maintenance Assistant) system (Huang, Sugihara, & Miyamoto 1991, 1992; Huang et al. 1992). Users of SMA include software engineers and their managers, who are engaged in maintaining software. SMA provides a set of automatic and semi-automatic reverse engineering tools to build up a database of information about all aspects of the maintained software system. MC is a multi-media interface (including graphical I/O and natural language I/O) that allows users access to information about the maintained software system, allows invocation of SMA tools, and allows users to ask about how to use SMA.

Typically, a software engineer starts SMA in order to implement a change request. After login to SMA, the user follows a set process to implement the requested change. Steps might include inspection of the code and/or derived models of the target software to estimate the scope of the change and its cost, reporting to the manager and obtaining approval for implementation

of the change, actual code modification, propagation of changes to other modules, and testing. To make this process easier, SMA provides a set of fifteen graphical models that describe different aspects of the target software such as the data structures, functions, control flow, user interface, database schema, etc. These models are represented in the MERA language (Takeda, Chin, & Miyamoto 1992) and displayed/edited using the meraTalk graphical editor. Many of the models can be automatically derived directly from COBOL and JCL source code. Other models require user assistance to create. By looking at these graphical models rather than directly at poorly structured source code, the software engineer gets a structured view of the target software at multiple levels of abstraction that is easier to understand and modify. The models are linked to one another and to the actual source code by automatically-derived linkage models.

At any time during the maintenance process, the user can type English in the MC window to request display of MERA models, invoke SMA tools, or ask about how to use SMA. Figure 1 shows the user's view of the meraTalk window after asking MC, "Can I look at the overview diagram for Flight-Reservation program?" Next, the user requests additional information by clicking the left mouse button on the file icon labeled TRANSACTION-FILE (meraTalk highlights the icon) and requests from MC, "I want to know the details of this file." MC understands the combination of "this file" and the user's pointing gesture (the mouse click) and sends a request to meraTalk to display the appropriate model.

In order to provide this type of multi-media interaction, MC models many aspects of its users, including the user's job, the user's expertise in using SMA/MC, the user's expertise in software engineering, the user's current tasks and goals, and the shared conversational and visual context. This paper describes how these user models are represented, acquired & maintained, and used to improve MC's multi-media user interaction.

User Knowledge

MC contains dynamic information about the user's knowledge of numerous SMA products (e.g. files) and tools. The user's knowledge of a particular product/tool

*This is a reprint of a paper in the *Proceedings of the Fourth International Conference on User Modeling*, Hyannis, MA, August, 1994, pp. 139-144.

The authors would like to thank Fujitsu and its subsidiary companies, the State of Hawaii, and SRA for their financial support.

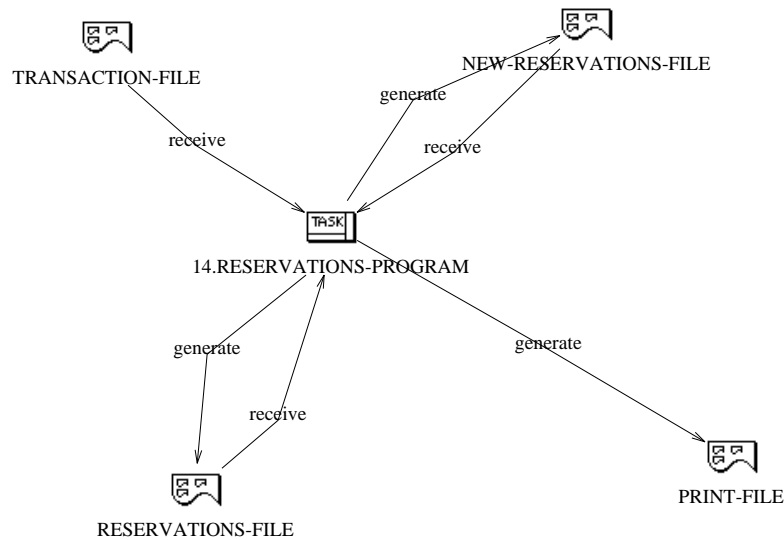


Figure 1: Top level Conceptual Process Model of Flight-Reservation program.

is estimated by observing how many times and how frequently the user has used the product/tool. The reasoning is: if the user uses a product/tool only a few times or very infrequently, then it is possible that the user invoked the tool/product by mistake and really does not understand the product or know how to use the tool. On the other hand, it is unlikely that the user would mistakenly use the same product/tool multiple times. Hence, if the user uses a product/tool many times, then MC assumes that the user is familiar with it.

representation

For every type of product handled by SMA and for every tool available for use in the SMA environment, MC stores information about the number of times that a user has seen/used the product/tool, the time of last use, and the usage frequency. Tools are classified into SMA tools, information presentation tools (MC and UNIX man), graphical-editing tools (the meraTalk editor) and text-editing tools (vi and emacs). SMA tools are subdivided into reverse engineering tools, the SMA object base, and tools for understanding code (the change impact analysis tool, the knowledge-based verification and validation tool, the code abstraction tool, and the performance evaluation simulator).

Products are divided into source files, CDF files (the representation of graphical information in SMA), and documents. Documents consist of man pages, KBVV reports (the knowledge-based verification and validation tool), and other text files. Source files include JCL and COBOL files. CDF files include regular files and linkage files. Regular CDF files are subdivided according to the fifteen different models currently used in SMA to represent different aspects of software systems. For example, DM (Data Model) describes data structures and their interrelationships, FM (Functional Model) describes the hierarchical functional require-

ments of the software in terms of input and output, FSM (Functional Structure Model) describes the relationship between functions and data objects, CPM (Conceptual Process Model) describes the high-level, real-world view of data flow and user interaction, PM (Performance Model) describes the performance parameters of subcomponents, UIM (User Interface Model) describes the layout of screens, user actions, and dynamic transitions in the user interface, IOPM (Internal Operational Profile Model) is an extended petri-net model describing control-flow, EOPM (External Operational Profile Model) describes the interaction of the software with outside entities such as users, machines, files, devices, and networks, and DBM (Database Scheme Model) describes the conceptual database scheme as an extended entity-relationship diagram. The linkage files relate objects in the regular files to one another and to the underlying source code.

acquisition & maintenance

MC's dynamic user model is updated whenever the user invokes a tool, imports a product into a tool (for processing or viewing), or exports a product from a tool. MC updates the time of last access, increments the count of how many times the user has seen the tool/product and computes the frequency of usage (which has the initial value of 0) based on the time since last access. Frequency of usage is computed as an exponential average using the formula:

$$frequency_{n+1} = \alpha(1/time) + (1 - \alpha)frequency_n$$

This allows MC to keep track of the usage frequency without needing to store the entire history of user interactions. Values of α close to 1 emphasize the most recent frequency information, while values of α close to 0 emphasize the historical record. Currently a value

of 1/2 is used in MC, however this definitely needs to be fine tuned through experimentation.

The likelihood that a user knows a product or tool is based on the following formula, where the units are number of times per week:

$$likelihood = 1 - \frac{1}{1.15(frequency + count/lifetime)}$$

Lifetime is the number of weeks since the SMA supervisor created the user's account. For example, a user who has seen a tool 5 times in the last week and has been active for a year would be considered to have a likelihood of .51 that the user knows the tool.

usage

The likelihood that the user knows about a particular type of tool or product is used to volunteer explanations of how to use tools and to decide the display mode (graphics, text, or both) for requested information. When telling the user to use a particular tool, MC will volunteer an explanation about how to use the tool if it is likely that the user is not familiar with the tool.

When the user asks for information, MC will decide whether to display the information graphically or textually (or as a mixture) depending on a combination of how likely the user understands that particular graphical format, how inherently understandable is the format (e.g., the DM model is very intuitive, but the IOPM model is not), and the user's primary display preferences (see the next Section). If the user's primary display preference is both, then the information is presented both graphically and repeated textually (possibly using a table). Otherwise, if the user is unlikely to understand the graphical format and the graphical format is inherently difficult or the user prefers text over graphics, then the information is given as text or as a table. Otherwise, if the user is unlikely to understand the graphical format but the graphical format is inherently simple or the user prefers graphics over text, then the information is presented graphically with additional textual explanation. Otherwise, if the user is likely to understand the graphical format and the user does not prefer text over graphics, then the information is presented graphically. Of course, if the user prefers another format, then the user is free to explicitly request MC to redisplay the information in the other format.

Initial Stereotypes

MC contains predefined static stereotypes of users based on job type and the user's level of expertise in software engineering (i.e., programming).

representation

The static user model includes information about the user's login name, personal name, job type (software engineer or manager), software engineering skill level (none, low, medium, or high), the software system that this user is responsible for maintaining (in case more

than one maintained software system is stored in SMA), the user's primary display preference (text, graphics, or both), and the user's security access level. This information is stored as a single relation per user within SMA's relational database and is retrieved as part of the SMA login procedure.

acquisition & maintenance

MC starts out with some initial information about the user that is typically provided by the SMA supervisor upon creating the user's account. The supervisor fills in information about the new user using an electronic form. This part of the user model is called the static user model because most of the information will rarely change over a period of a few years. Only the SMA supervisor or MC's inference routines are allowed to change the information in the static user model.

During a single session, MC keeps track of a user's requests for redisplay of presented information in other modalities. For example, if MC displays information to the user graphically, but the user later asks explicitly for a textual description, then MC increments the graphics-2-text counter. The text-2-graphics counter tracks the opposite requests. If the absolute difference between the two counters divided by the elapsed time of the current session is larger than some threshold (currently set at 5/hour), then MC will modify the user's primary display preference appropriately. Also, if the sum of the counters is larger than some threshold (currently set at 10/hour), then MC will modify the user's primary display preference to be mixed mode.

usage

The user's primary display preference is a factor in deciding how to display information to the user (the algorithm is described in the previous Section). The user's job type provides information about how the user will use SMA. In particular, a different process model (described in the following Section) can be selected for the user based on the user's job type. The user's software engineering skill level is used to provide an initial estimate of the user's knowledge of SMA tools and products when MC has not observed the user interacting with the particular tool/product enough. Following the double-stereotype approach (Chin 1989; Jameson 1992), SMA tools and products are classified according to inherent difficulty. If MC does not have enough observations of the user using the tool/product (i.e., count ≥ 4), then MC considers the user's software engineering skill level in guessing whether the user knows the tool/product. The higher the user's skill level, the more likely the user knows the tool/product.

The user's skill level and security level is used by MC to understand certain indirect speech acts (Searle 1975). Following the lead of (Allen 1979; Allen & Perreault 1980), the indirect speech act is analyzed based on the user's plans and goals. If the user is asking whether MC or the user can do X, then the user may be asking whether MC or the user has the ability or permission to

do X (the direct speech act) or the user may be asking MC to do X (the indirect speech act). The user's security level and software engineering skill level allow MC to estimate how likely it is that the user does not know whether he/she has permission to do X. If the user knows that he/she has permission, then the indirect speech act interpretation is chosen and vice-versa.

Task Model

MC contains a process model of the steps that a user will take in maintaining software. MC tracks the user's progress through the process model steps. Currently, MC only has a process model for programmers and does not have a process model for managers. However it would be simple to add such a process model if the necessary management tools (e.g. PERT chart editors, spreadsheets for cost analysis, personnel databases, etc.) were added to SMA.

representation

The process model is based on a Petri-net formalism and specifies the steps that users are supposed to follow in maintaining software. Each step specifies particular tools and the input/output products of the step. The Petri-net representation allows specification of parallel paths and external inputs such as supervisor approval. The process model may be changed or specialized by the SMA supervisor to reflect the software maintenance process requirements of the specific software company.

acquisition & maintenance

The process model is tracked by noting user invocations of tools, importing products and exporting products. User actions that do not match possible state transitions in the process model are ignored.

usage

The process model allows managers to keep track of the progress of team members. It is also used by MC to guide inexperienced users through the maintenance process, answering questions such as, "What do I do next?" Changes in process model tasks are used to signal changes in the dialog context (described in the next Section). Finally, the process model will eventually be used to provide additional context for the interpretation of user requests (not yet implemented).

Context

Modeling the context allows MC to understand natural language combined with pointing.

representation

The context model is split into two history lists. One list represents the linguistic context (LC) and the other list represents the visual context (VC). Both lists consist of entries representing semantic objects (not only physical objects, but also actions, events, etc.) or collections of objects. The LC list is built up using standard

linguistic techniques (c.f. Allen 1987 for an introduction). The VC list consists of all visible objects within the meraTalk window. Both history lists are ordered by salience. In the LC list, salience is equivalent to linguistic focus (Grosz 1977); in the VC list, the set of salient objects is the set of selected objects in the meraTalk editor.

The contents of the VC list are always included within the LC list. This is because the user can always refer to any visible objects. However, the most salient object in the VC list may not correspond to the most salient object in the LC list. For example, consider the following dialog:

User: I want to use the KBVV model checker on this file (*user highlights a file icon*). How do I run it?

In this example, the pronoun *it* refers to the KBVV model checker, which is the linguistic focus. The visual focus is established by the user highlighting the particular file icon.

acquisition & maintenance

Whenever MC's parser, PAU (Chin 1992), understands a phrasal unit (noun-phrase, verb-phrase, prepositional-phrase or sentence), the phrase and its meaning is passed to the Context component. Indefinite noun-phrases, verb-phrases, prepositional-phrases, and sentences are added to the LC list.

Unknown words are passed from PAU to the Context as potential names of objects (data structures, subroutines, files, etc.) stored in the SMA object base. If a search of the SMA object base finds the unknown word as a name of an object, then that object is added to PAU's temporary lexicon as the meaning of a Proper-name. This allows PAU to quickly understand the name next time it is used.

Whenever the user interacts with the meraTalk editor, this is passed to the Context component. Highlighting an object moves the object to the front of the VC list. Highlighting multiple objects creates a compound object (representing the set of highlighted objects) that is placed at the front of the VC list. When the user scrolls the meraTalk window, creates a meraTalk window, or deactivates a window, this causes objects to become visible or not visible. Newly visible objects are added to the VC and LC lists and objects that are no longer visible are removed from both lists. Unfortunately, obscuring part or all of a meraTalk window by other windows is not currently handled by MC (it should remove the obscured objects from the VC list). Also, multiple copies of meraTalk and multiple windows under the same meraTalk editor are not handled by MC.

When the user switches tasks in the process model, the LC list is cleared except for universal objects (such as the user and MC) and those objects that are also members of the VC list. This change corresponds to a switch in task context (Grosz 1977; Grosz & Sidner 1986).

The referents of the personal pronouns, *I* and *you* and their various other forms are initialized upon user login to correspond to the user and MC respectively.

usage

The context model is used to understand anaphoric references. The referent of deictic pronouns and noun-phrases (*this* and *that*) are given by the head of the VC list. The referent of prepositional-phrases involving spatial prepositions (e.g., *on*, *above*, *next to*, *to the right of*) are found in the VC by invoking the handler associated with the meaning of the preposition. For example, “the file to the right of the task,” is understood as a Right-of relation between a file and the highlighted task. The handler for the Right-of relation searches the VC list for the closest file object that is in the triangular region formed by 45 degree diagonal lines from the task toward the bottom right and top right corners and the right edge of the meraTalk window.

The referents of non-deictic definite pronouns and noun-phrases are found by searching for matching objects in the LC. Except for the personal pronouns referring to MC and the user, which are handled as special cases, matching is based purely on semantics. Characteristics such as gender and number that usually associated with words are encoded in the meaning as attribute (similar to a slot in a frame) values.

Comparison with Previous Research

The most similar system to MC is the UC (UNIX Consultant) system (Wilensky et al. 1988). Both MC and UC model the user’s knowledge and provide help in using the underlying system (SMA for MC and UNIX for UC). However, MC has multi-media capabilities, combining text and graphics for output and text and pointing for input. Also, MC provides direct access to information stored in SMA’s object base, whereas UC did not have access to UNIX. Both system’s user models employ double-stereotypes (one range for the user, another range for difficulty of the information). UC’s user modeling component, KNOWME (Chin 1989), estimated the user’s level of expertise by making inferences based on the user’s queries. MC directly observes the user’s interaction with SMA and estimates the user’s knowledge of specific tools and products based on amount and frequency of usage.

Other systems have combined pointing with natural language input. The XTRA system (Kobsa et al. 1976; Wahlster 1991) combined a range of pointing gestures with natural language. It was able to understand a variety of pointing gestures ranging from a point to a region and disambiguate the referent of the pointing action. In MC, immediate direct feedback (meraTalk highlights the selected objects) eliminates the need for pointing disambiguation. Also MC has a more sophisticated context model with separate history lists and separate foci for visual and linguistic contexts. MC also has a mechanism for switching linguistic contexts when the user switches tasks.

The VITRA project’s CITYTOUR and SOCCER systems (André et al. 1986; Retz-Schmidt 1988) handled interpretation and generation of spatial prepositions both for static descriptions and descriptions involving trajectories. MC also handles static descriptions, but does not handle trajectories. Also, MC does not worry about points of view since the point of view is always that of the user. However, MC has a more sophisticated context tracking mechanism.

The Integrated Interfaces project (Arens, Miller, & Sondheimer 1991), the COMET system (Feiner & McKeown 1990), the WIP system (Wahlster et al. 1993), the SAGE system (Roth, Mattis, & Mesnard 1991), and the CUBRICON system (Neil & Shapiro 1991) all combined natural language with graphical output. The Integrated Interfaces project lacked user and context models. COMET, SAGE, and WIP all implemented more sophisticated multi-media output, but were not interactive systems. Both CUBRICON and MC use two history lists, but CUBRICON does not have separate linguistic and visual foci. Also MC understands spatial prepositions, which are not handled in CUBRICON.

Conclusions

MC is not meant to serve as a complete interface for SMA. Rather, it provides an alternative to menu invocation/selection of tools/products or command-line invocation of tools with products (using the underlying UNIX shell). The user will choose MC only when it is easier to use natural language, perhaps in combination with pointing, to perform the task. Besides asking MC about how to use other SMA tools, users much prefer MC over searching a menu of products to find what they want. This is because there are typically a large number of alternative products and it is easier to find the particular product using natural language to specify constraints such as the relationship of the desired product to currently displayed products.

Users can independently invoke SMA tools other than MC without realizing that those tools may be constantly communicating with MC through the SMA ITC, Inter-Tool Communications, system (implemented with UNIX sockets). In most cases, the tools inform MC only upon invocation and whenever products are imported or exported from the tool. However, the meraTalk editor communicates every user action to MC and will accept MC commands such as loading new models or highlighting particular entities (icons) or relations (links).

An essential technological point that has greatly simplified implementation of MC and integration of MC with SMA is the shared usage of the MERA (Takeda, Chin, & Miyamoto 1992) language for representing both software models and MC knowledge throughout all SMA tools and products. MERA is implemented in CLOS (Common LISP Object System) for MC, in C data structures (meraTalk and other SMA tools), in text files (SMA products), and as database relations in the SMA object base.

Acknowledgements

MC would not be possible without the considerable efforts of the many other members of SERL (the Software Engineering Research Laboratory) who built the SMA system. In particular, Rajasekhar Kesarapalli, Taro Adachi, and Koji Takeda were instrumental in implementing the MC to meraTalk interface; Huang Ye, Hai Huang, and Tatuo Ishii developed the SMA Inter-Tool Communications interface; Takeshi Nishimura developed the process model for software maintenance; and Kamala Balaraman, Xiabo Wang, Huang Ye, Yoganathan Sivapathasundaram, and Yixin Zhou have contributed to earlier versions of MC.

References

- Allen, J. F. 1979. A Plan-Based Approach to Speech Act Recognition. Ph.D. diss., Dept. of Computer Science, Univ. of Toronto. Also available as Technical Report No. 131, Univ. of Toronto.
- Allen, J. 1987. *Natural Language Understanding*. Menlo Park, CA: Benjamin/Cummings.
- Allen, J. F., and Perrault, C. R. 1980. Analyzing Intention in Utterances. *Artificial Intelligence* 15: 143–178.
- André, E.; Bosch, G.; Herzog, G.; and Rist, T. 1986. Coping with the Intrinsic and Deictic Uses of Spatial Prepositions. In Ph. Jorrand and V. Sgureg (eds.), *Artificial Intelligence II, Proceedings of AIMSA-86*: 375–382.
- Arens, Y.; Miller, L.; and Sondheimer, N. 1991. Presentation Design Using an Integrated Knowledge Base. In J. W. Sullivan and S. W. Tyler (eds.), *Intelligent User Interfaces*. Reading, MA: Addison-Wesley, 241–257.
- Chin, D. N. 1989. KNOME: Modeling What the User Knows in UC. In A. Kobsa and W. Wahlster (eds.), *User Models in Dialog Systems*. Berlin: Springer-Verlag, 74–107.
- Chin, D. N. 1992. PAU: Parsing and Understanding with Uniform Syntactic, Semantic, and Idiomatic Representations. *Computational Intelligence* 8(3): 456–476.
- Feiner, S. K. and McKeown, K. R. 1990. Coordinating Text and Graphics in Explanation Generation. In Proceedings of the Eighth National Conference on Artificial Intelligence, 442–449.
- Grosz, B. 1977. The Representation and Use of Focus in Dialogue Understanding. Ph.D. diss., Univ. of California, Berkeley.
- Grosz, B., and Sidner, C. 1986. Attentions, Intentions, and the Structure of Discourse. *Computational Linguistics* 12: 175–204.
- Huang, H.; Sugihara, K.; and Miyamoto, I. 1991. Reconstructing Data Flow Diagram from Cobol Source Code. In Proceedings of International Conference for Young Computer Scientists, 198–201. Beijing, China.
- Huang, H.; Sugihara, K.; and Miyamoto, I. 1992. A rule-based tool for reverse engineering from source code to graphical models. In Proceedings of the Fourth International Conference on Software Engineering and Knowledge Engineering, 178–185. Capri, Italy.
- Huang, H.; Sugihara, K.; Takeda, K.; Yamamoto, K.; and Miyamoto, I. 1992. Reverse engineering tools in Software Maintenance Assistant. In Proceedings of the Fifth International Conference on Software Engineering & Its Applications (Toulouse '92), 401–410. Toulouse, France.
- Jameson, A. 1992. Generalizing the Double-Stereotype Approach: A Psychological Perspective. In Proceedings of the Third International Workshop on User Modeling: UM92, 69–83. Dagstuhl, Germany. Also available as E. André, R. Cohen, W. Graf, B. Kass, C. Paris, W. Wahlster (eds.), DFKI publication D-92-17.
- Kobsa, A.; Allgayer, J.; Reddig, C.; Reithinger, N.; Schmauks, D.; Harbusch, K.; and Wahlster, W. 1986. Combining Deictic Gestures and Natural Language for Referent Identification. In Proceedings 11th International Conference on Computational Linguistics, 356–361.
- Neil, J. G., and Shapiro, S. C. 1991. Intelligent Multi-Media Interface Technology. In Sullivan, J. W. and Tyler, S. W. (eds.), *Intelligent User Interfaces*. Reading, MA: Addison-Wesley, 11–43.
- Retz-Schmidt, G. 1988. Various Views on Spatial Prepositions. *AI Magazine* 9(2): 95–105.
- Roth, S. F.; Mattis, J.; and Mesnard, X. 1991. Graphics and Natural Language as Components of Automatic Explanation. In Sullivan, J. W. and Tyler, S. W. (eds.), *Intelligent User Interfaces*. Reading, MA: Addison-Wesley.
- Searle, J. R. 1975. Indirect Speech Acts. In P. Cole and J. L. Morgan (eds.), *Syntax and Semantics Vol. III: Speech Acts*. New York, NY: Academic Press.
- Takeda K.; Chin, D. N.; and Miyamoto, I. 1992. MERA: Meta Language for Software Engineering. In Proceedings of the 4th International Conference on Software Engineering and Knowledge Engineering, 495–501. Capri, Italy.
- Wahlster, W. 1991. User and Discourse Models for Multimodal Communication. In Sullivan, J. W. and Tyler, S. W. (eds.), *Intelligent User Interfaces*. Reading, MA: Addison-Wesley, 45–67.
- Wahlster, W.; André, E.; Finkler, W.; Profitlich, H.-J.; and Rist, T. 1993. Plan-based integration of natural language and graphics generation. *Artificial Intelligence* 63, 387–427.
- R. Wilensky; Chin, D. N.; Luria, M.; Martin, J.; Mayfield, J.; and Wu, D. 1988. “The Berkeley UNIX Consultant Project,” *Computational Linguistics* 14(4): 35–84.