Interaction Modes and Initiative in Human-Computer Collaboration Should be Based on Task Requirements and Responsibilities

David N. Chin
Department of Information and Computer Sciences
University of Hawaii at Manoa
2565 The Mall
Honolulu, HI 96822 U.S.A.
email: Chin@Hawaii.Edu

1 Previous Arguments for Agents as Intermediaries

At the 1988 Workshop on Architectures for Intelligent Interfaces [Chin, 1992b], I was a strong proponent of the "intermediary" model of human-computer interaction. At that time, I argued that an intelligent interface such as UC (the UNIX Consultant) [Wilensky *et al.*, 1988] cannot just respond passively to the user's commands and queries. It must be able to take the initiative in order to volunteer information, correct user misconceptions, or reject unethical requests. An interface based solely on the "model world" interaction mode would not be flexible enough to provide these types of behavior in which the computer takes the initiative. Most users do not expect a model world to volunteer suggestions—only agents do that and the only agents familiar to people are intermediaries. In other words, users are extremely surprised when programs that they have been directly manipulating for a long time suddenly tell them, "You are using me incorrectly." Users can much more easily accept suggestions when volunteered by intermediary agents.

2 The Need for User Models

A computational agent only needs to take the initiative when it has greater knowledge than the user. If the system is assumed to know less than the user, then the system cannot differentiate potential user mistakes from reasonable user actions that the system does not understand. This requirement was the impetus for my work on modeling of the user's domain knowledge and automatic user model acquisition [Chin, 1988; Chin, 1989; Chin, 1992a; Chin, 1993]. I argued that a computational agent should maintain knowledge of the domain expertise of its users, otherwise it cannot volunteer information properly. For example, suppose someone were to ask for a telephone number and the respondent did not know the answer. If the respondent were to suggest the alternative plan of looking in the phone book, not only would this suggestion be unhelpful to the asker, but the asker might reasonably feel insulted by the suggestion (which implies that the asker was too lazy to look in the phone book in the first place). On the other hand, if the asker is a new immigrant to the US from a much less-developed culture that does not have phone books, then such a suggestion would be quite helpful when combined with an explanation of phone books. In order to distinguish between such cases, the respondent must maintain a model of the user's domain knowledge.

3 Complex Views on Intermediary vs. Model World Interaction Modes

Since the 1988 Workshop on Architectures for Intelligent Interfaces, I have completely revised my position on intermediary vs. model world interaction modes, in part due to my experience in designing three new collaborative systems: MC (the Maintenance Consultant), MOANA (MOdel Acquisition by NAtural language) [Chin, 1991; Takeda *et al.*, 1990; Chin *et al.*, 1988], and LEI (Location and Elevation Interpreter) [Futch *et al.*, 1992]. MC is a natural language consultant system (similar to UC) to help users of the SMA system (Software Maintenance Assistant) in their task of maintaining COBOL software. MC is based on the intermediary interaction model, whereas SMA is based on the model-world interaction mode. MC monitors all of the user's actions within the SMA model world and allows mixed-initiative dialogs, although normally the user holds the initiative and MC only interferes in exceptional cases. On the other hand, the MOANA system only allows system-directed dialog. The user is forced to respond to the system's queries that are directed toward gathering information from the user about the formal requirements of the user's new software system. Finally the LEI system, which understands English descriptions of locations and translates these into geodesic coordinates with the help of digitized maps and spatial reasoning, also controls the dialog initiative, but provides a model-world interface for users to create and manipulate data requested by the system.

These three collaborative systems have radically different mixes of interaction modes and different techniques for controlling initiative. MC/SMA provides a main interaction mode based on a model-world interaction (SMA), within which one window provides an intermediary interaction mode with mixed initiative (MC). MOANA provides a pure intermediary interaction mode with no user user initiative allowed. Finally, LEI controls the overall initiative but allows the user to manipulate a model-world interface in order to satisfy system requests.

Each of these collaborative systems manages its multimedia interaction using different strategies. In MOANA, the user is allowed to view, but not edit, the changing formal spec-

ifications (which are represented in a graphical representation language [Takeda *et al.*, 1992]) as they are acquired from the user in a graphical window. In LEI, the user's model-world interface includes a color map for the user to manipulate. Finally, in MC/SMA, information about the maintained software is kept in a series of graphical models, so MC is able to combine textual and graphical presentation modes as appropriate for the requested information and as appropriate to the user's level of familiarity with the particular graphical models used to represent the requested information. Likewise, the user can refer to objects within the graphical model using linguistic terms such as "this relation" (referring to the relation highlighted by the user) or "the entity on the right."

3.1 Choosing Interaction Modes to Satisfy Task Requirements

An analysis of the interaction modes of three collaborative systems shows that the most important considerations in the design of the human-computer interface are satisfying the requirements of the particular collaborative task and determining which subtasks will be handled by the user or by the program. In general, the choice of interaction modes is based on the the number of unstructured possibilities available within the task. If there are too many unstructured possibilities, then it is often simpler (for the user, not the programmer) to use a natural language interface. With only a few possibilities, a direct manipulation interface is simpler, faster, and easier to use. When natural language is involved, users tend to assume an intermediary because users are predisposed to believe that model-world interfaces do not understand natural language.

MC uses natural language because there are many possible types of queries that the user can make. Providing menus for these query types would result in unmanageable lengths and/or depths of menus. Providing direct-manipulation icons for these queries would likewise result in too many icons. For the same reason, the MOANA system uses natural language because of the unbounded number of possible replies to many of its queries. On the other hand, there are a limited number of possible actions that a user can perform on a map, so the LEI system provides a direct-manipulation environment for its user.

This rule for choosing an interaction model should only serve as a general guideline. One can still use natural language to handle limited possibilities or use direct manipulation to handle large numbers of unstructured possibilities. For example, in MOANA, many queries have only a few possible replies. MOANA lists these possibilities and asks the user to select among them. Likewise, one can still provide a direct-manipulation interface for large numbers of possible actions. This forces users to memorize large numbers of manipulations (e.g. the emacs editor and the TeX document formatting system) or to search through large numbers of objects/menus (e.g. many WYSIWYG editors). However, such systems are far from ideal and some combination of interaction models such as found in MC/SMA may work better.

3.2 Choosing Control of Initiative Based on Task Responsibilities

In general, control of initiative should be given to the agent (user or program) that is primarily responsible for a particular task. For example, LEI is mainly responsible for interpreting the English locality descriptions, so it maintains control of the initiative until it runs into unknown words, whereupon LEI requests clarification from the user of the geographical meaning of the unknown words. At this point LEI passes initiative to the user to manipulate its model-world-based map interface. After the user has finished this task, control of initiative returns to LEI. In MOANA, the system has complete responsibility for eliciting the required information from the user, so it never relinquishes control of dialog initiative (any queries from the user are ignored). Finally in MC/SMA, the user has the primary responsibility for software maintenance, so the user holds the initiative in manipulating the main SMA model-world graphical interface. The user is free to pass initiative to MC by providing MC with subtasks. MC relinquishes control of the initiative as soon as it has completed its assigned task. Very infrequently, MC may seize the initiative temporarily when it detects problems.

4 Conclusions

By analyzing the three systems described above, we find that my earlier arguments that intelligent interfaces should be based on the intermediary interaction model were incorrect. A system can still volunteer information without providing an intermediary interaction model. For example, one can imagine an SMA system that has a "suggestions" window rather than an MC window. The suggestions window volunteers information just like MC, but does not accept any natural language input from the user. Additional help can take the form of direct manipulation of the suggestion text (e.g. by highlighting key words and then invoking an additional help button).

However, my earlier analysis of what knowledge is needed by a collaborative system still holds: the system must model the user's domain knowledge not only to avoid making inappropriate suggestions, but also to select the proper mix of multi-media formats (one should select only formats that the user can understand). A model of the user's domain knowledge also allows the system to augment a multi-media presentation that is only partially understandable to a particular user with additional explanatory text. Besides modeling user knowledge, MC also models the user's task structure and keeps track of the user's current progress within the task structure in order to properly interpret ambiguous requests (often a particular interpretation of a request only makes sense for some specific task). Other knowledge needed by MC for integrating multi-media include a context model that includes not only the linguistic context, but also the visual context (positions and status of objects in the graphical editor and other windows) and the history of user manipulations of objects within SMA's model-world interface.

Naturally, user modeling has a rather high cost in complexity. In order to keep costs down in MC, only information about user knowledge at a course grain is kept between sessions.

For example, MC keeps track of the frequency of usage of SMA commands (this is used to infer familiarity and expertise based on the assumption that heavier usage leads to expertise), but not of particular options of the commands. Even at this course grain, there are hundreds of objects in MC's hierarchy for which usage frequency statistics are kept. In order to maintain this user model and the context model, there is considerable traffic between MC and SMA's direct-manipulation interfaces. For example, whenever the user manipulates anything in SMA's graphical editor, this must be passed to MC so that it can update its context model. Otherwise, linguistic references to objects and actions performed by the user in the editor would not be understandable to MC.

The bottom line for more intelligent human-computer collaboration is much higher complexity in the computer program. Just as better graphical interfaces today often consist of more code and processing power than the actual application portion, in the future, intelligent components will require more code and processing power than the rest of the program, including any graphical interfaces and actual application code.

References

[Chin, 1988] David N. Chin. Exploiting User Expertise in Answer Expression. In the *Proceedings of the Seventh National Conference on Artificial Intelligence*, Saint Paul, MN, August, pages 756–760.

[Chin *et al.*, 1988] David N. Chin., Koji Takeda, and Isao Miyamoto. Using Natural Language and Stereotypical Knowledge for Acquisition of Software Models. In the IEEE International Workshop on *Tools for Artificial Intelligence*, Fairfax, VA, October, IEEE Computer Society Press, Washington D.C., pages 290–295.

[Chin, 1989] David N. Chin. KNOME: Modeling What the User Knows in UC. In A. Kobsa and W. Wahlster (Eds.), *User Models in Dialog Systems*, Springer-Verlag, Berlin, pages 74–107.

[Chin, 1991] David N. Chin. Natural Language Dialogs for Knowledge Acquisition. In M. McTear and N. Creany (Eds.), *AI and Cognitive Science '90*, Springer-Verlag, London, pages 233–243.

[Chin, 1992a] David N. Chin. Applying User Models to Produce Cooperative Multimedia Explanation. In the *AAAI Spring Symposium Series: Producing Cooperative Explanations*, Stanford, CA, March, pages 26–30.

[Chin, 1992b] David N. Chin. Intelligent Interfaces as Agents. In J. W. Sullivan and S. W. Tyler (Eds.), *Intelligent User Interfaces*, Addison-Wesley, Reading, MA, pages 177–206.

[Chin, 1993] David N. Chin. Acquiring User Models. To appear in *Artificial Intelligence Review*.

[Futch *et al.*, 1992] Shaun Futch, David N. Chin., Matthew McGranaghan, and Jinn-Guey Lay. Spatial-Linguistic Reasoning in LEI (Locality and Elevation Interpreter). In A. U. Frank, I. Campari, and U. Formentini (Eds.), *Theories and Methods of Spatio-Temporal Reasoning in Geographical Space.* Springer-Verlag, Berlin, pages 318–327.

[Takeda *et al.*, 1990] Koji Takeda, David N. Chin, Isao Miyamoto. MOANA, a Software Engineering Tool with Natural Language User Interface. In the *Proceedings of Pacific Rim International Conference on Artificial Intelligence*, Nagoya, Japan, November, 1990, pages 897–902.

[Takeda *et al.*, 1992] Koji Takeda, David N. Chin, Isao Miyamoto. MERA: Meta Language for Software Engineering. In the *Proceedings of the 4th International Conference on Software Engineering and Knowledge Engineering*, Capri, Italy, June, 1992, pages 495–502.

[Wilensky *et al.*, 1988] Robert Wilensky, David N. Chin, Marc Luria, James Martin, James Mayfield, and Dekai Wu. The Berkeley UNIX Consultant Project. In *Computational Linguistics*, 14(4):35–84, December.