Multimodal Maps: An Agent-Based Approach

Adam Cheyer and Luc Julia

SRI International
333 Ravenswood Ave
Menlo Park, CA 94025 - USA

Abstract. In this paper, we discuss how multiple input modalities may be combined to produce more natural user interfaces. To illustrate this technique, we present a prototype map-based application for a travel planning domain. The application is distinguished by a synergistic combination of handwriting, gesture and speech modalities; access to existing data sources including the World Wide Web; and a mobile handheld interface. To implement the described application, a hierarchical distributed network of heterogeneous software agents was augmented by appropriate functionality for developing synergistic multimodal applications.

1 Introduction

As computer systems become more powerful and complex, efforts to make computer interfaces more simple and natural become increasingly important. Natural interfaces should be designed to facilitate communication in ways people are already accustomed to using. Such interfaces allow users to concentrate on the tasks they are trying to accomplish, not worry about what they must do to control the interface.

In this paper, we begin by discussing what input modalities humans are comfortable using when interacting with computers, and how these modalities should best be combined in order to produce natural interfaces. In Sect. 3, we present a prototype map-based application for the travel planning domain which uses a synergistic combination of several input modalities. Section 4 describes the agent-based approach we used to implement the application and the work on which it is based. In Sect. 5, we summarize our conclusions and future directions.

2 Natural Input

2.1 Input Modalities

Direct manipulation interface technologies are currently the most widely used techniques for creating user interfaces. Through the use of menus and a graphical user interface, users are presented with sets of discrete actions and the objects on which to perform them. Pointing devices such as a mouse facilitate selection of an object or action, and drag and drop techniques allow items to be moved or combined with other entities or actions.

With the addition of electronic pen devices, gestural drawings add a new dimension direct manipulation interfaces. Gestures allow users to communicate a surprisingly wide range of meaningful requests with a few simple strokes. Research has shown that multiple gestures can be combined to form dialog, with rules of temporal grouping overriding temporal sequencing (Rhyne, 1987). Gestural commands are particularly applicable to graphical or editing type tasks.

Direct manipulation interactions possess many desirable qualities: communication is generally fast and concise; input techniques are easy to learn and remember; the user has a good idea about what can be accomplished, as the visual presentation of the available actions is generally easily accessible. However, direct manipulation suffers from limitations when trying to access or describe entities which are not or can not be visualized by the user.

Limitations of direct manipulation style interfaces can be addressed by another interface technology, that of natural language interfaces. Natural language interfaces excel in describing entities that are not currently displayed on the monitor, in specifying temporal relations between entities or actions, and in identifying members of sets. These strengths are exactly the weaknesses of direct manipulation interfaces, and concurrently, the weaknesses of natural language interfaces (ambiguity, conceptual coverage, etc.) can be overcome by the strengths of direct manipulation.

Natural language content can be entered through different input modalities, including typing, handwriting, and speech. It is important to note that, while the same textual content can be provided by the three modalities, each modality has widely varying properties.

- Spoken language is the modality used first and foremost in human-human interactive problem solving (Cohen et al., 1990). Speech is an extremely fast medium, several times faster than typing or handwriting. In addition, speech input contains content that is not present in other forms of natural language input, such as prosidy, tone and characteristics of the speaker (age, sex, accent).
- Typing is the most common way of entering information into a computer, because it is reasonably fast, very accurate, and requires no computational resources.
- Handwriting has been shown to be useful for certain types of tasks, such as performing numerical calculations and manipulating names which are difficult to pronounce (Oviatt, 1994; Oviatt and Olson, 1994). Because of its relatively slow production rate, handwriting may induce users to produce different types of input than is generated by spoken language; abbreviations, symbols and non-grammatical patterns may be expected to be more prevalent amid written input.

2.2 Combination of Modalities

As noted in the previous section, direct manipulation and natural language seem to be very complementary modalities. It is therefore not surprising that a number of multimodal systems combine the two.

Notable among such systems is the Cohen's Shoptalk system (Cohen, 1992), a prototype manufacturing and decision-support system that aids in tasks such as quality assurance monitoring, and production scheduling. The natural language module of Shoptalk is based on the Chat-85 natural language system (Warren and Perreira, 1982) and is particularly good at handling time, tense, and temporal reasoning.

A number of systems have focused on combining the speed of speech with the reference provided by direct manipulation of a mouse pointer. Such systems include the XTRA system (Allegayer et al, 1989), CUBRICON (Neal and Shapiro, 1991), the PAC-Amodeus model (Nigay and Coutaz, 1993), and TAPAGE (Faure and Julia, 1994).

XTRA and CUBRICON are both systems that combine complex spoken input with mouse clicks, using several knowledge sources for reference identification. CUBRICON's domain is a map-based task, making it similar to the application developed in this paper. However, the two are different in that CUBRICON can only use direct manipulation to indicate a specific item, whereas our system produces a richer mixing of modalities by adding both gestural and written language as input modalities.

The PAC-Amodeus systems such as VoicePaint and Notebook allow the user to synergistically combine vocal or mouse-click commands when interacting with notes or graphical objects. However, due to the selected domains, the natural language input is very simple, generally of the style "Insert a note here".

TAPAGE is another system that allows true synergistic combination of spoken input with direct manipulation. Like PAC-Amodeus, TAPAGE's domain provides only simple linguistic input. However, TAPAGE uses a pen-based interface instead of a mouse, allowing gestural commands. TAPAGE, selected as a building block for our map application, will be described more in detail in Sect. 4.2.

Other interesting work regarding the simultaneous combination of handgestures and gaze can be found in Bolt (1980) and Koons, Sparrell and Thorisson (1993).

3 A Multimodal Map Application

In this section, we will describe a prototype map-based application for a travel planning domain. In order to provide the most natural user interface possible, the system permits the user to simultaneously combine direct manipulation, gestural drawings, handwritten, typed and spoken natural language. When designing the system, other criteria were considered as well:

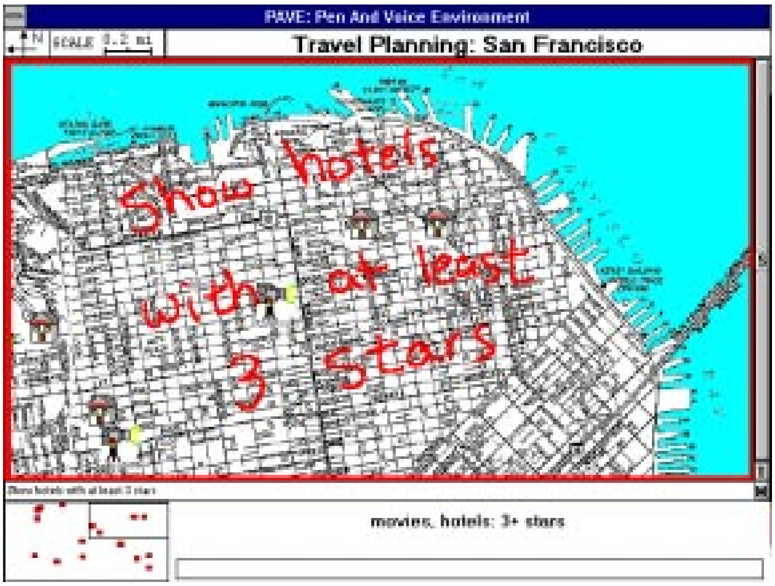
PAVE: Pen And Voice Environment
Travel Planning: San Francisco
SCALE 0.2 mi
Show hotels with at least 3 stars
movies, hotels: 3+ stars

Fig. 1. Multimodal application for travel planning

- The user interface must be light and fast enough to run on a handheld PDA while able to access applications and data that may require a more powerful machine.
- Existing commercial or research natural language and speech recognition systems should be used.
- Through the multimodal interface, a user must be able to transparently access a wide variety of data sources, including information stored in HTML form on the World Wide Web.

As illustrated in Fig. 1, the user is presented with a pen sensitive map display on which drawn gestures and written natural language statements may be combined with spoken input. As opposed to a static paper map, the location, resolution, and content presented by the map change, according to the requests of the user. Objects of interest, such as restaurants, movie theaters, hotels, tourist sites, municipal buildings, etc. are displayed as icons. The user may ask the map to perform various actions. For example :

- *distance calculation* : e.g. "How far is the hotel from Fisherman's Wharf?"
- *object location* : e.g. "Where is the nearest post office?"
- *filtering* : e.g. "Display the French restaurants within 1 mile of this hotel."
- *information retrieval* : e.g. "Show me all available information about Alcatraz."

The application also makes use of multimodal (multimedia) output as well as input: video, text, sound and voice can all be combined when presenting an answer to a query.

During input, requests can be entered using gestures (see Fig. 2 for sample gestures), handwriting, voice, or a combination of pen and voice. For instance, in order to calculate the distance between two points on the map, a command may be issued using the following:

– *gesture*, by simply drawing a line between the two points of interest.
– *voice*, by speaking "What is the distance from the post office to the hotel?".
– *handwriting*, by writing "dist p.o. to hotel?"
– *synergistic combination of pen and voice*, by speaking "What is the distance from here to this hotel?" while simultaneously indicating the specified locations by pointing or circling.

Notice that in our example of synergistic combination of pen and voice, the arguments to the verb "distance" can be specified before, at the same time, or shortly after the vocalization of the request to calculate the distance. If a user's request is ambiguous or underspecified, the system will wait several seconds and then issue a prompt requesting additional information.

The user interface runs on pen-equipped PC's or a Dauphin handheld PDA (Dauphin, DTR-1 User's Manual) using either a microphone or a telephone for voice input. The interface is connected either by modem or ethernet to a server machine which will manage database access, natural language processing and speech recognition for the application. The result is a mobile system that provides a synergistic pen/voice interface to remote databases.

In general, the speed of the system is quite acceptable. For gestural commands, which are handled locally on the user interface machine, a response is produced in less than one second. For handwritten commands, the time to recognize the handwriting, process the English query, access a database and begin to display the results on the user interface is less than three seconds (assuming an ethernet connection, and good network and database response). Solutions to verbal commands are displayed in three to five seconds after the end of speech has been detected; partial feedback indicating the current status of the speech recognition is provided earlier.

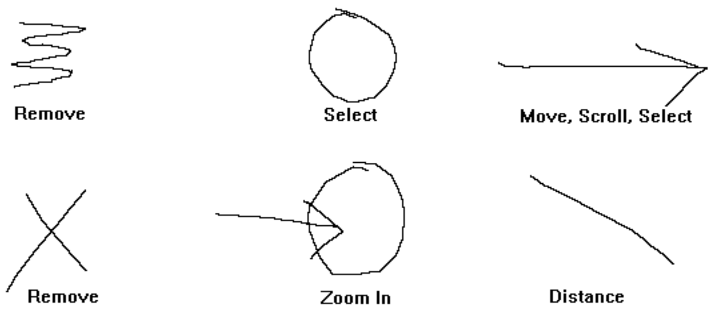

Fig. 2. Sample gestures

4 Approach

In order to implement the application described in the previous section, we chose to augment a proven agent- based architecture with functionalities developed for a synergistically multimodal application. The result is a flexible methodology for designing and implementing distributed multimodal applications.

4.1 Building Blocks

Open Agent Architecture. The Open Agent Architecture (OAA) (Cohen et al., 1994) provides a framework for coordinating a society of agents which interact to solve problems for the user. Through the use of agents, the OAA provides distributed access to commercial applications, such as mail systems, calendar programs, databases, etc.

The Open Agent Architecture possesses several properties which make it a good candidate for our needs:

- An Interagent Communication Language (ICL) and Query Protocol have been developed, allowing agents to communicate among themselves. Agents can run on different platforms and be implemented in a variety of programming languages.
- Several natural language systems have been integrated into the OAA which convert English into the Interagent Communication Language. In addition, a speech recognition agent has been developed to provide transparent access to the Corona speech recognition system.
- The agent architecture has been used to provide natural language and agent access to various heterogeneous data and knowledge sources.
- Agent interaction is very fine-grained. The architecture was designed so that a number of agents can work together, when appropriate in parallel, to produce fast responses to queries.

The architecture for the OAA, based loosely on Schwartz's FLiPSiDE system (Schwartz, 1993), uses a hierarchical configuration where client agents connect to a "facilitator" server. Facilitators provide content-based message routing, global data management, and process coordination for their set of connected agents. Facilitators can, in turn, be connected as clients of other facilitators. Each facilitator records the published functionality of their sub-agents, and when queries arrive in Interagent Communication Language form, they are responsible for breaking apart any complex queries and for distributing goals to the appropriate agents. An agent solving a goal may require supporting information and the agent architecture provides numerous means of requesting data from other agents or from the user.

Among the assortment of agent architectures, the Open Agent Architecture can be most closely compared to work by the ARPA knowledge sharing community (Genesereth and Singh, 1994). The OAA's query protocol, Interagent Communication Language and Facilitator mechanisms have similar instantiations in the SHADE project, in the form of KQML, KIF and various independent capability matchmakers. Other agent architectures, such as General Magic's Telescript (General Magic, 1995), MASCOS (Park et al, submitted), or the CORBA distributed object approach (Object Management Group, 1991) do not provide as fully developed mechanisms for interagent communication and delegation.

The Open Agent Architecture provides capability for accessing distributed knowledge sources through natural language and voice, but it is lacking integration with a synergistic multimodal interface.

TAPAGE. TAPAGE (edition de Tableaux par la Parole et la Geste) is a synergistic pen/voice system for designing and correcting tables.

To capture signals emitted during a user's interaction, TAPAGE integrates a set of modality agents, each responsible for a very specialized kind of signal (Faure and Julia, 1994). The modality agents are connected to an 'interpret agent' which is responsible for combining the inputs across all modalities to form a valid command for the application. The interpret agent receives filtered results from the modality agents, sorts the information into the correct fields, performs type-checking on the arguments, and prompts the user for any missing information, according to the model of the interaction. The interpret agent is also responsible for merging the data streams sent by the modality agents, and for resolving ambiguities among them, based on its knowledge of the application's internal state. Another function of the interpret agent is to produce reflexes: reflexes are actions output at the interface level without involving the functional core of the application.

The TAPAGE system can accept multimodal input, but it is not a distributed system; its functional core is fixed. In TAPAGE, the set of linguistic input is limited to a *verb object argument* format.

4.2 Synthesis

In the Open Agent Architecture, agents are distributed entities that can run on different machines, and communicate together to solve a task for the user. In TAPAGE, agents are used to provide streams of input to a central interpret process, responsible for merging incoming data. A generalization of these two types of agents could be:

Macro Agents: contain some knowledge and ability to reason about a domain, and can answer or make queries to other macro agents using the Interagent Communication Language.

Micro Agents: are responsible for handling a single input or output data stream, either filtering the signal to or from a hierarchically superior 'interpret' agent.

The network architecture that we used was hierarchical at two resolutions: micro agents are connected to a superior macro agent, and macro agents are connected in turn to a facilitator agent. In both cases, a server is responsible for the supervision of its client sub-agents.

In order to describe our implementation, we will first give a description of each agent used in our application and then illustrate the flow of communication among agents produced by a user's request.

Speech Recognition (SR) Agent: The SR agent provides a mapping from the Interagent Communication Language to the API for the Decipher (Corona) speech recognition system (Cohen et al., 1990), a continuous speech speaker independent recognizer based on Hidden Markov Model technology. This macro agent is also responsible for supervising a child micro agent whose task is to control the speech data stream. The SR agent can provide feedback to an interface agent about the current status and progress of the micro agent (e.g. "listening", "end of speech detected", etc.) This agent is written in C.

Natural Language (NL) Parser Agent: translates English expressions into the Interagent Communication Language (ICL). For a more complete description of the ICL, see Cohen et al. (Cohen et al., 1994). The NL agent we selected for our application is the simplest of those integrated into the OAA. It is written in Prolog using Definite Clause Grammars, and supports a distributed vocabulary; each agent dynamically adds word definitions as it connects to the network. A current project is underway to integrate the Gemini natural language system (Cohen et al., 1990), a robust bottom up parser and semantic interpreter specifically designed for use in Spoken Language Understanding projects.

Database Agents: Database agents can reside at local or remote locations and can be grouped hierarchically according to content. Micro agents can be connected to database agents to monitor relevant positions or events in real time. In our travel planning application, database agents provide maps for each city, as well as icons, vocabulary and information about available hotels, restaurants, movies, theaters, municipal buildings and tourist attractions. Three types of databases were used: Prolog databases, X.500 hierarchical databases, and data loaded automatically by scanning HTML pages from the World Wide Web (WWW). In one instance, a local newspaper provides weekly updates to its Mosaic-accessible list of current movie times and reviews, as well as adding several new restaurant reviews to a growing collection; this information is extracted by an HTML reading database agent and made accessible to the agent architecture. Descriptions and addresses of new restaurants are presented to the user on request, and the user can choose to add them to the permanent database by specifying positional coordinates on the map (e.g. "add this new restaurant here"), information lacking in the WWW database.

Reference Resolution Agent: This agent is responsible for merging requests arriving in parallel from different modalities, and for controlling interactions between the user interface agent, database agents and modality agents. In this implementation, the reference resolution agent is domain specific: knowledge is encoded as to what actions must be performed to resolve each possible type of ICL request in its particular domain. For a given ICL logical form, the agent can verify argument types, supply default values, and resolve argument references. Some argument references are descriptive ("How far is it to the hotel on Emerson Street?"); in this case, a domain agent will try to resolve the definite reference by sending database agent requests. Other references, particularly when contextual or deictic, are resolved by the user interface agent ("What are the rates for this hotel?"). Once arguments to a query have been resolved, this agent coordinates the actions and calculations necessary to produce the result of the request.

Interface Agent: This macro agent is responsible for managing what is currently being displayed to the user, and for accepting the user's multimodal input. The Interface Agent also coordinates client modality agents and resolves ambiguities among them : handwriting and gestures are interpreted locally by micro agents and combined with results from the speech recognition agent, running on a remote speech server. The handwriting micro-agent interfaces with the Microsoft PenWindows API and accesses a handwriting recognizer by CIC Corporation. The gesture micro- agent accesses recognition algorithms developed for TAPAGE.

An important task for the interface agent is to record which objects of each type are currently salient, in order to resolve contextual references such as "the hotel" or "where I was before." Deictic references are resolved by gestural or direct manipulation commands. If no such indication is currently specified, the user interface agent waits long enough to give the user an opportunity to supply the value, and then prompts the user for it.

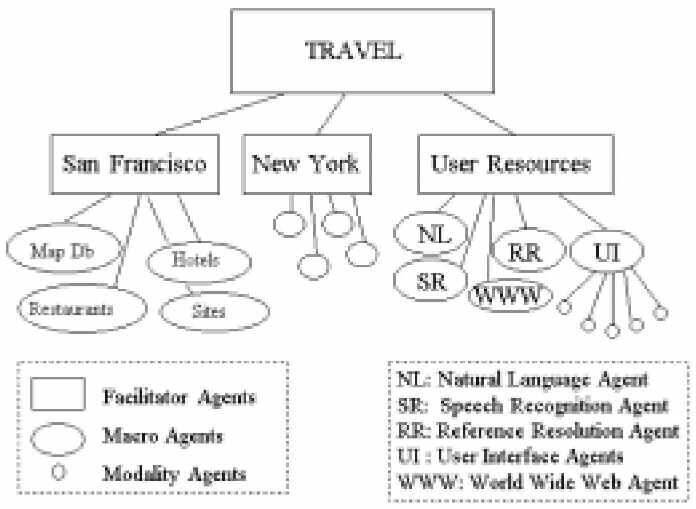
TRAVEL
San Francisco
New York
User Resources
Map Db
Hotels
Restaurants
Sites
NL
SR
RR
UI
WWW
Facilitator Agents
Macro Agents
Modality Agents
NL: Natural Language Agent
SR: Speech Recognition Agent
RR: Reference Resolution Agent
UI : User Interface Agents
WWW: World Wide Web Agent

Fig. 3. Agent Architecture for Map Application

We shall now give an example of the distributed interaction of agents for a specific query. In the following example, all communication among agents passes transparently through a facilitator agent in an undirected fashion; this process is left out of the description for brevity.

1. A user speaks: “How far is the restaurant from this hotel?”
2. The speech recognition agent monitors the status and results from its micro agent, sending feedback received by the user interface agent. When the string is recognized, a translation is requested.
3. The English request is received by the NL agent and translated into ICL form.
4. The reference resolution agent (RR) receives the ICL distance request containing one definite and one deictic reference and asks for resolution of these references.
5. The interface agent uses contextual structures to find what “the restaurant” refers to, and waits for the user to make a gesture indicating “the hotel”, issuing prompts if necessary.
6. When the references have been resolved, the domain agent (RR) sends database requests asking for the coordinates of the items in question. It then calculates the distance according to the scale of the currently displayed map, and requests the user interface to produce output displaying the result of the calculation.

5 Conclusions

By augmenting an existing agent-based architecture with concepts necessary for synergistic multimodal input, we were able to rapidly develop a map-based application for a travel planning task. The resulting application has met our initial requirements: a mobile, synergistic pen/voice interface providing good natural language access to heterogeneous distributed knowledge sources. The approach used was general and should provide a for developing synergistic multimodal applications for other domains.

The system described here is one of the first that accepts commands made of synergistic combinations of spoken language, handwriting and gestural input. This fusion of modalities can produce more complex interactions than in many systems and the prototype application will serve as a testbed for acquiring a better understanding of multimodal input.

In the near future, we will continue to verify and extend our approach by building other multimodal applications. We are interested in generalizing the methodology even further; work has already begun on an agent-building tool which will simplify and automate many of the details of developing new agents and domains.

References

[1989]Allegayer, J., Jansen-Winkeln, R., Reddig, C. and Reithinger, N. (1989) Bidirectional use of knowledge in the multi-modal NL access system XTRA. In *Proceedings of IJCAI-89*, Detroit, pp. 1492-1497.

[1980]Bolt, R. (1980) Put that there: Voice and Gesture at the Graphic Interface, *Computer Graphics, 14(3)*, pp. 262-270.

[1990]Cohen, M., Murveit, H., Bernstein, J., Price, P., and Weintraub, M. (1990) The DECIPHER Speech Recognition System. In *1990 IEEE ICASSP*, pp. 77-80.

[1992]Cohen, P. (1992) The role of natural language in a multimodal interface. In *Proceedings of UIST'92*, pp. 143-149.

[1994]Cohen, P.R., Cheyer, A., Wang, M. and Baeg, S.C. (1994) An Open Agent Architecture. In *Proceedings AAAI'94 – SA*, Stanford, pp. 1-8.

[DTR-1 User's Manual]*Dauphin DTR-1 User's Manual*, Dauphin Technology, Inc., Lombard, Ill 60148.

[1994]Faure, C. and Julia, L. (1994) An Agent-Based Architecture for a Multimodal Interface. In *Proceedings AAAI'94 – IM4S*, Stanford, pp. 82-86.

[1994]Genesereth, M. and Singh, N.P. (1994) *A knowledge sharing approach to software interoperation*, unpublished manuscript, Computer Science Department, Stanford University.

[1995]*Telescript Product Documentation* (1995), General Magic Inc.

[1993]Koons, D.B., Sparrell, C.J., and Thorisson, K.R. (1993) Integrating Simultaneous Input from Speech, Gaze and Hand Gestures. In *Intelligent Multimedia Interfaces*, Maybury, M.T. (ed.), Menlo Park: AAAI Press/MIT Press.

[1993]Maybury, M.T. (ed.) (1993) *Intelligent Multimedia Interfaces*, Menlo Park: AAAI Press/MIT Press.

[1991]Neal, J.G., and Shapiro, S.C. (1991) Intelligent Multi-media Interface Technology. In *Intelligent User Interfaces*, Sullivan, J.W. and Tyler, S.W. (eds.), Reading: Addison-Wesley Pub. Co., pp. 11-43.

[1993]Nigay, L. and Coutaz, J. (1993) A Design Space for Multimodal Systems: Concurrent Processing and Data Fusion. In *Proceedings InterCHI'93*, Amsterdam, ACM Press, pp. 172-178.

[1991]Object Management Group (1991) *The Common Object Request Broker: Architecture and Specification*, OMG Document Number 91.12.1.

[1994]Oviatt, S. (1994) Toward Empirically-Based Design of Multimodal Dialogue Systems. In *Proceedings of AAAI'94 – IM4S*, Stanford, pp. 30-36.

[1994]Oviatt, S. and Olsen, E. (1994) Integration Themes in Multimodal Human-Computer Interaction. In *Proceedings of ICSLP'94*, Yokohama, pp. 551-554.

[submitted]Park, S.K., Choi J.M., Myeong-Wuk J., Lee G.L., and Lim Y.H. (submitted for publication), *MASCOS : A Multi-Agent System as the Computer Secretary.*

[1987]Rhyne J. (1987) Dialogue Management for Gestural Interfaces, *Computer Graphics, 21(2)*, pp. 137-142.

[1993]Schwartz, D.G. (1993) *Cooperating heterogeneous systems: A blackboard-based meta approach*, Technical Report 93-112, Center for Automation and Intelligent Systems Research, Case Western Reserve University, Cleveland Ohio, (unpublished PhD. thesis).

[1991]Sullivan, J. and Tyler, S. (eds.) (1991) *Intelligent User Interfaces*, Reading: Addison-Wesley Pub. Co.

[1982]Warren, D. and Pereira, F. (1982) An Efficient Easily Adaptable System for Interpreting Natural Language Queries, *American Journal of Computational Linguistics, 8(3)*, pp. 110-123.