Two strategies for multimodal fusion

Guillermo Pérez
University of Seville
Palos de la Frontera s/n
41004 Sevilla. Spain
+34955056614
gperez@us.es

Gabriel Amores
University of Seville
Palos de la Frontera s/n
41004 Sevilla. Spain
+34954551549
jgabriel@us.es

Pilar Manchón
University of Seville
Palos de la Frontera s/n
41004 Sevilla. Spain
+34955056614
pmanchon@us.es

ABSTRACT

This paper describes our transition from a speech-only dialogue system to a multimodal one. Our description focuses on the fusion of input modalities coming from different channels.

Two strategies have been implemented for comparison purposes: the first solution is largely based on Johnston's work [Johnston et al. 1997, Johnston 1998], and involves modifying our parser to cope with simultaneous multimodal inputs, and to include temporal constraints at unification level. The second implementation proposes an original solution to the problem, and involves combining inputs coming from different multimodal channels at dialogue level. This solution is based on an implementation of the ISU approach [Traum et al. 1999, Amores et al. 2001].

These two strategies have been implemented in an Information-State-Update-based system, combining both speech and graphical inputs. A multimodal "Smart House" scenario where the user interacts with the system using a microphone and a touch-screen has been chosen.

The paper includes a high-level description of the algorithms implemented and concludes with a theoretical analysis of the advantages and drawbacks of both approaches.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and presentation]:** User Interfaces - *Theory and methods, User interface management systems, Voice I/O.*

General Terms

Algorithms, Performance, Theory.

Keywords

Multimodality, fusion.

1. INTRODUCTION

Multimodal interfaces allow for more flexible and natural interactions between human users and computer systems. They benefit from a variety of communication channels such as speech, text, gesture, handwriting, etc.

Multimodal systems have been largely studied since the appearance of the "Put-That-There" system [Bolt, 1980]. Oviatt´s results [Oviatt et al. 1997] showed the potential gain of multimodal systems compared to uni-modal ones in terms of user preferences and the possibility of mutual dissambiguation.

The fusion of multimodal inputs has also evolved since Bolt's proposal [Bolt 1980], which suffered from lack of generality, defining rules that could only apply to speech-driven systems.

Johnston proposed a new aproach [Johnston 1998] using a unification based multidimensional parsing of typed feature structures that partially overcame the limitations previously mentioned.

Johnston himself [Johnston et al. 2000] found that this solution could be improved both at parsing level, because of its inherent computational complexity, and at natural language understanding level because it did not allow a tight-coupling of parsing and input recognition (speech or gesture). He proposed an alternative approach using finite-state multimodal grammars.

From our point of view this last approach can be improved significantly with a new approach, fusing multimodal inputs not at grammar level, but at dialogue level and within a richer (non-finite-state) model of dialogue context: the Information State Update (ISU) approach [Traum et al. 1999].

Our system can be described as a collaborative dialogue manager linked to a Natural Language Understanding Module, which allows dialogues driven by the semantic information provided by the user and by the dialogue expectations generated by the dialgue manager.

The kernel of our system is then composed by :

- A Natural Language Understanding (NLU) module: which is in charge of the lexical and syntactic analysis and produces the Information States..
- A Dialogue Manager which manipulates Information States (or Dialogue Moves) through the application of dialogue update rules.

The Information States that we have configured for this scenario are based on the DTAC protocol [Siridus Deliverable 3.2, Quesada et al. 2000], A DTAC consists of a feature-value structure with four main features: DMOVE, TYPE, ARG and CONT. The following figure illustrates the DTAC obtained for the command "*Turn on the kitchen light*" in our scenario:

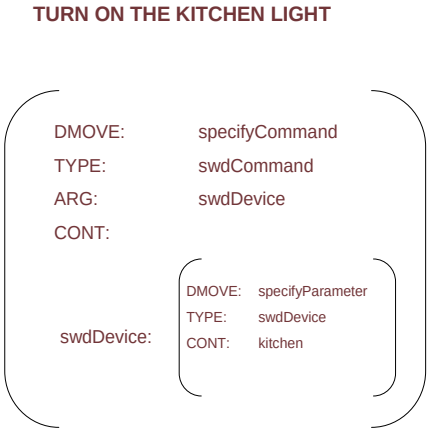

FIGURE 1

More examples of Information States applied to the Smart House Scenario using the DTAC protocol can be found in [Quesada et al. 2001].

Dialogue Update Rules take the following form in our system:

```
( RuleID:    MAKECALL;
   PriorityLevel:    15;
   TriggeringCondition:
      (DMOVE:specifyCommand,TYPE:MakeCall);
   DeclareExpectations: {
      Dest <= (DMOVE:specifyParameter,TYPE:Name|PhoneNumber);
   }
   SetExpectations: {
      Confirm <= (DMOVE:answerYN);
   }
   ActionsExpectations: {
      [Dest] => {
                        ExecuteDMFunction(MakeCallDest);
      }
      [Confirm] => {
                        ExecuteDMFunction(MakeCallDisam);
   }
   PostActions: {
            @if ((@is-MAKECALL.Confirm.TYPE == "YES") {
            ExecuteDMFunction(MakeCallDest);
                }
            }
   }
)
```

The item "Triggering Conditions" describes the Dialogue Move (Dmove) that must arrive for the rule to be activated.

"DeclareExpectations" defines additional information needed for the rule to be fulfilled. This information could have been provided previously in the dialogue history, or during the same interaction.

The "SetExpectations" section defines additional Dialogue Moves (DMoves) needed to successfully execute the rule, such as an explicit confirmation before executing a command.

As its name indicates, "ActionExpectations" defines the actions to be carried out when either the "DeclareExpectations" have not been fulfilled by the current input nor within the Dialogue History, or when some "SetExpectations" have been defined.

Finally the "PostActions" section describes what should be done once the rule is active and all the expectations have been fulfilled.

The whole system kernel has been wrapped as an OAA [Martin el al. 1999] agent and used to provide dialogue management services within different scenarios [Quesada et al. 2001, Quesada et al. 2000]

Section two describes the steps needed to move from a speech-only dialogue system to our new multimodal one.

Section three gives an overview on the current status of the project and describes the agents already implemented.

Section four deals with two strategies implemented to fuse multimodal inputs: a first one using the unification module of our parser, and a second one using our implementation of the ISU approach and the dialogue expectations generated by the dialogue manager.

In section five both strategies are compared.

Section six gives an example of how these strategies could be used for managing a multimodal scientific visualization software.

Sections seven and eight summarize the conclusions and future work.

2. From speech-only to multimodal interaction

Before any further considerations, some preliminary steps had to be taken in order to make the system work multimodally.

The first step involved moving from a synchronous, system-driven, turn taking approach to an asynchrounous, mixed-initiative model. We faced this evolution by means of an intermediate (input pool) layer whose role is to store all inputs coming from the user at any time and make them available to the system when requested. The input pool was implemented as an independent OAA agent.

The second step involved modifying the GUI interface [Quesada et al. 2001], which was originally just a floor plan representation of the house designed to configure the distribution of devices and functionalities. The new extended version of the GUI allows the user to refer to parts of the house by clicking on them with the pen.

The third step was to make the speech-only input pool a multimodal input pool. This goal was achieved by allowing different kinds of inputs and storing them in a simple FIFO queue (see fig.2). Namely, the multimodal input pool accepts two kinds of inputs:

- SPEECH, including the following fields: **init_time**, **end_time**, **sentece_score**, **list[word. word_score]**.

- CLICKs, including the **icon** and **time** fields.

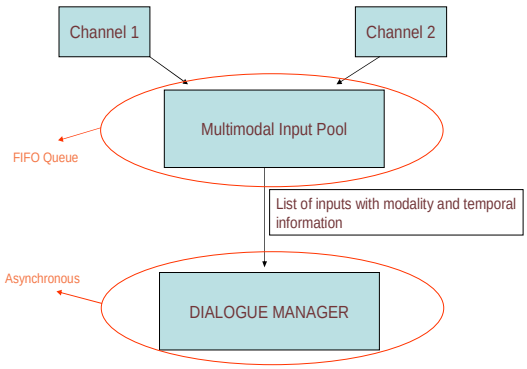

FIGURE 2

For multimodal information rendering we have implemented at this stage a basic heuristic-based presentation layer which is out of the scope of this paper.

A global view of how the system interacts with the user is then as follows (fig. 3):

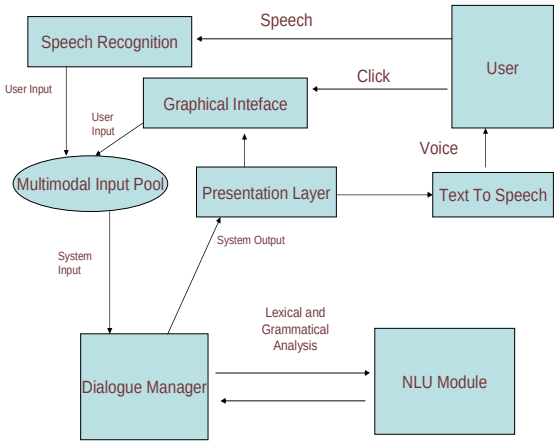

FIGURE 3

3. Current Implementation Status

As mentioned above, we are applying our system to a Smart House scenario. Four specific agents connected through the OAA facilitator have been added:

- Home Setup: This agent allows the installation of new devices and their configuration.
- Action Manager: This agent sends the actual commands which turn on/off the devices.
- Knowledge manager: This agent contains the structure and general ontology of the house.
- Display Agent: An agent which displays the system output graphically.

The last agent has been especially developed for the new multimodal architecture, and acts as an alternative output modality to the TTS.

The first three agents were already implemented and described elsewhere [Quesada et al. 2001]. Nevertheless further improvements have been made for the following agents:

- The Home Setup now allows the user to click on the icons (as mentioned in section 2)

- The Knowledge Manager is now linked with OWL using RDQL queries (http://www.w3.org/2004/OWL/).

For further information on these new features, please refer to the TALK project deliverable 2.1 [Milward et al. 2005 –to appear-].

The user interfaces with the system by means of a Tablet PC using both speech an the Tablet PC pen as input modalities, and gets feedback by speech (TTS) and graphically from the system.

In the screenshot below (fig. 4) we can appreciate the Display Agent, the Home Setup and the TTS Agent, which is actually an OAA wrapper for the Microsoft animated agent (http://www.microsoft.com/msagent/default.asp).

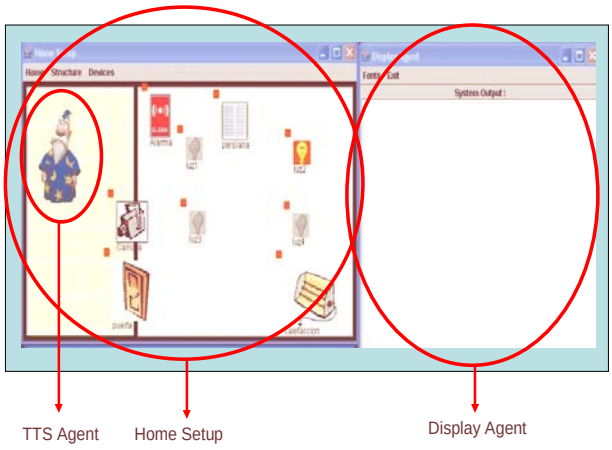

Figure 4

We would like to mention that as a part of the system design we are currently carrying out a set of Wizard of Oz experiments with disabled people. These experiments will hopefully help us to better identify the user preferences and improve our interfaces and the system. overall A detailed description of the platform used for these experiments can be found in [Manchón et al. 2005]

4. Multimodal fusion: two strategies

4.1 Strategy 1

The first strategy implemented follows Johnston's proposal [Johnston et al. 1997, Johnston 1998], by using a unification based parser and including modality and temporal constraints at unification level. Our implementation differs from Johnston´s in that we add a higher level of flexibility.

The main motivation behind this strategy is that multimodality is conceived of as a single communicative act between two participants, and as such should be treated by a single grammar which is capable of accepting input coming from different modalities. As expected, our system permits that the communicative act may range from speech-only to clicks-only or hybrid inputs, and all are considered equal as far as the grammar is concerned. Obviously, as described below, this is an advantage as long as we consider single-task interactions and not multiple task interactions. The pragmatic ambiguity which may result in multimodal multi-tasking cannot be resolved by a single grammar.

Graphically, this strategy fuses inputs at our NLU module (fig. 5):

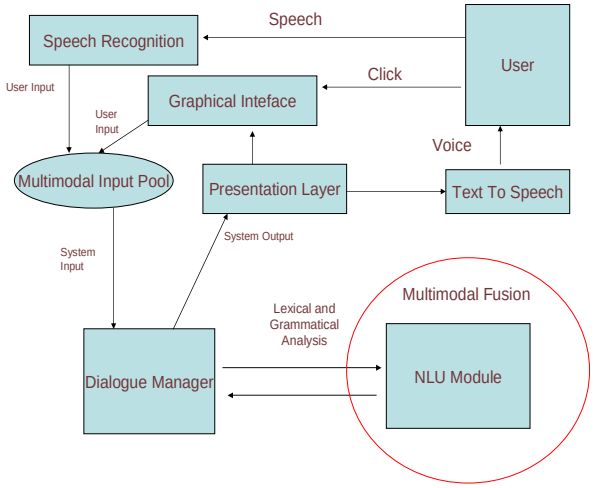

FIGURE 5

We use our own lexical and grammatical analyser [Quesada et al. 1995, Amores et al., 2000] where each input is described by a set of feature-value pairs. When the parser receives an input sentece (either speech-only, click-only or mixed), it calls the lexical analyser adding three new ad-hoc feature-value pairs: MODALITY, TIME_INIT and TIME_END.

These features are then used in conjunction with a set of logical operators to define complex expressions in order to enforce modality and temporal constraints.

Imagine that we want to define a grammar rule for an input as "switch on *the light*", where light can be either specified by voice or clicked. Imagine also that we know that when using the mixed modality input (that is to say: when clicking on the light icon, actually the user clicks before saying "switch on".

In this case, we could specify a rule for the voice only inputs (therefore with natural *command + parameter* order), and another one that only applies to mixed inputs where we accept an inverse order *parameter + command*.

The unification rule will look like the following one:

```
(Rule 1 : Command -> CommandOn DeviceSpecifier)
  { @up = @self-1;}
(Rule 2 : Command -> DeviceSpecifier CommandOn)
  @up.DeviceSpecifier =a @self-1;
  @if(((@self-1.MODALITY == CLICK) && (@self-2.MODALITY == VOICE))
  @then {
    @if ((@self-1.TIME_INIT - @self-2.TIME_INIT <= 5) && (@self-1.TIME_INIT - @self-2.TIME_INIT <= -5))
      @then { @break();}
      @else { @up.MODALITY =a [VOICE,CLICK];
        @if((@self-1.TIME_INIT <= @self-2.TIME_INIT))
          @then { @up.TIME_INIT =a @self-1.TIME_INIT;}
          @else { @up.TIME_INIT =a @self-2.TIME_INIT;}
        @if((@self-1.TIME_END >= @self-2.TIME_END))
          @then { @up.TIME_END =a @self-1.TIME_END;}
          @else { @up.TIME_END =a @self-2.TIME_END;}
      }
    }
    @else {
      @break();
      }
}
```

Remark that, in addition to the modality constraint, we have defined an overlap timeframe (5 time units) within which this inputs have to occur. These timeframes could be configured independently (rule by rule) if our data was accurate enough.

These rules describe under what conditions the right-hand symbols can unify and, if the conditions are met, how the unification has to be done. Notice that we are not using only temporal data as subcategorization edges but actually letting the user configure the constraints case by case.

However we feel that this flexibility is not always needed, so we have implemented a set of macros to be used at unification level that, from our point of view, cover a number of cases:

1) @assign_modality(@self-1,@self-2,@self-n)
   a. check if the modality of all the constituents is the same, otherwise, assign MODALITY:[MIXED] to the mother node.
2) @assign_time_init(@self-1,@self-2,@self-n)
   a. Get the lowest time init and assign it to the mother node
3) @assign_time_end(@self-1,@self-2,@self-n)
   a. Get the highest time end and assign it to the mother node

4.2 Strategy 2

The second strategy combines simultaneous inputs coming from different channels (modalities) at Dialogue Level. The idea is to check the multimodal input pool before launching the actions expectations waiting an "inter-modality" time.

Obviously, this strategy assumes that each individual input can be considered as an independent Dialogue Move.

Graphically, this strategy fuses the multimodal inputs at dialogue level (fig. 6):

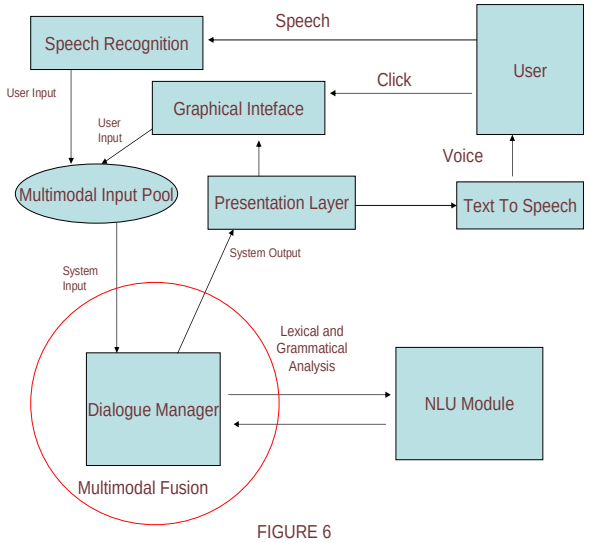

FIGURE 6

In this approach, the multimodal input pool receives and stores all inputs including information such as time and and modality. The Dialogue Manager checks the input pool regularly to retrieve the corresponding input. If more than one input is received during a certain timeframe, they are considered simultaneous or pseudo-simultaneous. In this case, further analysis is needed in order to determine whether those independent multimodal inputs are truly related or not.

If the inputs timing with respect to each other is deemed to be within the plausible time range to consider them a potential multimodal combination, then additional information will be taken into account to decide whether these independent DMoves are complementary or not:

- If one is TriggeringCondition of a Dialogue Rule, and the other one is part of the expectations
- If both are expectations of an already active Dialogue Rule.
- If there is no other parallel dialogue history whose active Dialogue Rules may conflict with the previously identified one.

When all indicates that the DMoves are related and complementary, they merge into a unique Information State. Otherwise, different paths may be taken depending on the situation:

- One of them may complete an already active Dialogue Rule whereas the other may trigger a new unrelated TriggeringCondition and therefore a new parallel dialogue history.
- Each of them may complete already active Dialogue Rules in parallel dialogue histories unambiguously.
- Both of them may complete already active Dialogue Rules in parallel dialogue histories in an ambiguous manner, in which case disambiguation subdialogues will be needed.
- They are unrelated and not compatible with any active Dialogue Rule, so two new tasks with their respective dialogue histories will be initiated.

Our approach can be described by this high-level algorithm:

- Receive uni-modal input A (DMove)
- Receive uni-modal input B (DMove)
- IF A & B are complementary
  & contextually appropriate
  & within a predefined timeframe
  - THEN Create new IS from these DMoves + Dialogue History
- ELSE
  - store the & disambiguate

This algorithm takes into account:

1. Dialogue Moves generated
2. Modality
3. Inter-Input timing
4. Dialogue Move order
5. Existing Dialogue Moves
6. Existing Dialogue Histories
7. Scenario and contextual factors

Dialogue Rules may also be configured with the same logical operators mentioned within the Strategy 1, since the Dialogue Manager actually uses the unification module of the parser. Similar rules to the one detailed for Strategy 1could be configured within the Dialogue Manager.

The difference is *where* we are applying these rules: for Strategy one the coverage is composed by the symbols (terminals and not terminals) within the grammar rules, meanwhile the coverage for Strategy 2 are the DTACs structures that describe the DMoves.

Although taking into account a considerable number of factors may not appear as a very appealing solution, this innovative approach enables the system to cope with "Multimodal Multitasking", which would not be possible within the implementation of Strategy 1.

By Multimodal Multitasking we imply the possibility of accomplishing independent unrelated tasks simultaneously, sparing continuous system disambiguation. Humans have often proven to be able and even prefer to accomplish several tasks at once, as long as they are familiar with the tools and/or environment and none of the tasks imply too heavy a cognitive load.

With this approach, multimodal systems have taken a step forward towards more intelligent, flexible and collaborative systems.

5. Comparison of strategies

Computational efficiency: The first strategy is much heavier from a computational point of view since tasks are added at unification level which represents 80% of the parsing time [Amores et al. 2000]. On the other hand, the additional computational complexity added by the second strategy is of no consequence.

Dependency on time measures: The first strategy is highly dependent on the precision of the time data. The overlapping times fixed at unification rules assume that the init_time and end_time features are accurate, which is not always the case. The second strategy however allows for a certain degree of flexibility.

Background data: In order to define the appropriate time ranges for multimodal complementary inputs, real user data is required. The more precise this time ranges need to be, the more important it becomes to collect large amounts of data, especially considering the possibility of tuning the thresholds rule by rule.

Multimodal multitasking: The multimodal multitasking is the ability to carry independent tasks at the same time by means of different multimodal channels. The notion of task only exists at dialogue level, therefore strategy one cannot be applied if dealing with multimodal multitasking.

Inter-modality dissambiguation: When dealing with more complex modalities (i.e. voice and gesture recognition) we may expect not only pairs item-time, but full lattices coming from both channels. The mutual disambiguation could be more easily dealt with the first strategy. The second strategy would become considerably more complex.

Dialogue Acts: At theoretical level, a potential problem of the second strategy could arise from the assumption that any uni-modal input generates always a Dialogue Move. Although we have been unable to find any example or situation where this assumption is false, it could possibly be the case with more sophisticated not speech-driven systems.

Number of Modalities: We believe that as the number of modalities increases, the best choice would be the second strategy, since the first strategy implies a high computational overload which would become unbearable with a higher number of modalities.

6. ShowCase: Scientific Visualization

Imagine the following futuristic scenario: there is a highly skilled scientist; let's say that within the Meteorology domain. He has an embedded PDA-like device which is programmed as a general purpose personal assistant (P.A.). This device has also a specific program for Meteorology scientific visualization. The device admits inputs by voice and pen at any time.

Now consider the following interaction:

- **Scientist:** *Please load the "Meteo SciVis" program*. (Unimodal voice input)
- **P.A:** *Here it is, sir. Would you like to load a specific graph?* (Unimodal voice output)
- **Scientist:** *Yes, load the "Spain_2090" graph.* . (Unimodal voice input)
- **Scientist**: *Now change this label to "Seville" –click on a label-* (Multimodal voice and pen input).

Nothing new so far. Our future device understands everything and accomplishes the tasks properly. Both strategies are suitable for this.

But now imagine that the scientist wants to switch on the light in his office. Probably, he is not going to get up and manually switch it on (this is an old habit from the XXth century), he will just ask the P.A. to do it. But in the future he won't need to stop working on his graph, so he will change a particular line of the graph at the same time. Therefore, the P.A. will receive two simultaneous inputs:

- Switch on the light (voice)
- Move this line from here to here (pen)

This short multimodal-multitasking example shows the potential gain of the second strategy:

The first strategy will always try to fuse both inputs because at grammar level the concept of "task" doesn't exist. Therefore, a P.A. ruled by the first strategy would not accomplish the task correctly.

On the other hand, the second strategy would identify that these two inputs correspond to different dialogue branches, so the P.A. will not try to fuse the independent inputs.

7. Conclusions

This paper describes the evolution from a speech-only system to a multimodal one, implementing an intermediate layer called "multimodal input pool" whose role is to allow for asynchronous behaviour.

The general steps taken to cope with both speech and clicking inputs have been described and two strategies to fuse multimodal entries explained and compared.

Comparing the advantages and drawbacks of both strategies we can conclude that strategy 2 suits better the needs of our naïve voice-and-click scenario. Although strategy 1 is certainly more powerful (we can tune the fusion rule by rule in greater detail) and may have some theoretical advantages over the second strategy, it requires an enormous amount of data, and the potential advantages remain to be proven in a real environment. However, strategy 2 seems to provide a suitable solution for the dialogue issues mentioned, without increasing the computational complexity significantly.

8. FUTURE WORK

Future research includes the study and comparison of both strategies in a scenario with different and more complex multimodal channels.

It would also be interesting to study the possibility of using both strategies at the same time, automatically deciding when to apply each one.

9. ACKNOWLEDGMENTS

This work was done under the "TALK" research project, funded by EU´s FP6 [ref. 507802] and the "Multilingual Management of Spoken Dialogues" project, funded by the Spanish Ministry of Education under grant TIC2002-00526.

10. REFERENCES

[1] Gabriel Amores y José Francisco Quesada (1997) "*Episteme*" Procesamiento del Lenguaje Natural 21. pp. 1-16.

[2] Gabriel Amores, José Francisco Quesada (2000) "*Diseño e Implementación de Sistemas de Traducción Automática*" Ed. Universidad de Sevilla (Book, ISBN: 84-472-0585-1).

[3] Richard A. Bolt (1980), "Put-that-there": Voice and gesture at the graphics interface, ACM SIGGRAPH Computer Graphics, v.14 n.3, p.262-270, July 1980

[4] Michael Johnston, Philip R. Cohen, David McGee, Sharon L. Oviatt, James A. Pittman, Ira A. Smith (1997): "*Unification-based Multimodal Integration*". ACL 1997: 281-288

[5] Michael Johnston (1998): "*Unification-based Multimodal Parsing*". COLING-ACL 1998, 624-630.

[6] Michael Johnston, Srinivas Bangalore (2000). *"Finite State Multimodal Parsing and Understanding"*. Proceedings of the 18th conference on Computational linguistics - Volume 1. pp 369-375.

[7] Michael Johnston, Srinivas Bangalore (2001). "*Finite-state Methods for Multimodal Parsing and Integration*", Finite State Methods in Natural Language Processing, August 2001.

[8] Pilar Manchón, Guillermo Pérez, Gabriel Amores (2005). "*WOZ experiments in Multimodal Dialogue Systems*". Proceedings of the ninth workshop on the semantics and pragmatics of dialogue, 2005, 131-135

[9] David Martin, Adam Cheyer. And Doug Moran (1999). *"The Open Agent Architecture: A framework for building distributed software systems*" Applied Artificial Intelligence: An International Journal. Volume 13, Number 1-2, January-March 1999. pp 91-128.

[10] David Milward, Gabriel Amores, Tilman Becker, Nate Blaylock, Malte Gabslid, Staffan Larsson, Oliver Lemon, Pilar Manchón, Guillermo Pérez, Jan Schehl (2005 –to appear). "*Integration of ontological knowledge with the ISU approach*". Deliverable 2.1, Talk Project.

[11] Sharon Oviatt (1999). *"Ten myths of multimodal interaction",* Communications of the ACM, Vol. 42, No. 11, November, 1999, pp. 74-81.

[12] Sharon Oviatt (2003). "*Multimodal interfaces*". In The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications, J. JACKO AND A. SEARS, Eds. Lawrence Erlbaum Assoc., Mahwah, NJ, 2003, chap.14, 286-304

[13] José Francisco Quesada, Gabriel Amores (1995). *"A computational Model for the Efficient Retrieval of Very Large Structure-Based Knowledge Bases."* Proceedings of Knowledge Representation, Use and Storage for Efficiency (KRUSE95) International Symposium, pp. 86-96 .

[14] José Francisco Quesada, Doroteo Torre, Gabriel Amores (2000). "*Design of a Natural Command Language Dialogue System*". Deliverable 3.2, Siridus Project

[15] David Traum, Johan Bos, Robin Cooper, Staffan Larsson, Ian Lewin, Colin Matheson and Massimo Poesio. (1999). *"A model of Dialogue Moves and Information State Revision"*. Technical Report D2.1, Trindi Project.