Dynamic Language Change in MIMUS

Carmen Del Solar, *Guillermo Pérez*, *Eva Florencio*, *David Moral*, *Gabriel Amores*, *Pilar Manchón*

Julietta Research Group, University of Seville, Seville, Spain

carsolval@alum.us.es, {gperez, dmoral, evaflorencio, jgabriel, pmanchon}@us.es

Abstract

One of the most widely pursued goals in dialogue system development is the improvement of usability, which is mainly achieved by providing users with both friendly and manageable interfaces. The MIMUS dialogue system supports multimodal interactions, allowing the user to interact not only verbally but also graphically. In addition to this, MIMUS (MultIModal, University of Seville) is a multilingual system that enables the user to communicate (dynamically) in Spanish and English. The present paper describes the MIMUS architecture, the components that entail direct relationship with multilinguality, and the way in which languages can be dynamically switched within a single dialogue.

Index Terms: dialogue systems, multimodal, multilingual, dynamic language change

10.21437/Interspeech.2007-578

1. Introduction

One of the main challenges for dialogue systems these days is multilingual environments, which require that they be able to operate in more than one language. A number of systems have already engaged in significant efforts at solving the numerous difficulties that multilinguality poses: Voyager [1] was ported from English to other languages by separating language dependent information from the system kernel, the FAME [2] multimodal demonstrator, which allows cross-lingual document retrieval, employs a database containing multilingual information for all stored objects, and SQEL [3], which manages questions about train connections in Czech, German, Slovak and Slovenian, uses a multilingual recogniser that identifies the language while running, and selects the semantic parser accordingly. In plurilingual societies like Spain, systems have been developed that handle dialogues in at least two of the official languages of the state (the E-MATTER system [4] supports dialogues in both Castilian Spanish and Catalan and "identifies, pre-processes and reads e-mails in Castilian Spanish, Catalan, Galician, Basque, English and French").

The study of multilinguality as a global phenomenon embracing different modules has been shown before in the context of dialogue systems. Holzapfel [5] and Jonson [6] emphasise the need for differentiating the Dialogue Manager module from the Natural Language Understanding (NLU) module, which becomes the transmission component between the speech recogniser and the Dialogue Manager itself.

In MIMUS the NLU module computes linguistic information by generating language-independent content representations (i.e., feature structures) of the user's productions, thus isolating the Dialogue Manager from multilinguality problems. Also, the Generation module follows the reverse path: it creates derivation trees for the aforementioned abstract semantic representations, in order to generate a string of words in the working language. As far as the Knowledge Manager module goes, the fact that it is separated from the NLU module allows for the existing domain knowledge (i.e., databases and OWL ontologies –OWL Web Ontology Language: http://www.w3.org/TR/owl-features/) to be reused across languages. Finally, both the Automatic Speech Recognition (henceforth, ASR) Manager and the Text-to-Speech (TTS) Manager support language change at run time.

Strategies for managing interactions in multiple languages differ vastly. MIMUS is not only portable to other languages. Its rule-based approach also allows the user to be able to interchange languages in the middle of a dialogue, and to continue without restarting the system.

2. MIMUS architecture

MIMUS is a multimodal and multilingual dialogue system, optimised for disabled users in the in-home domain. Users may control devices; ask about their status and/or quantity; switch applications (telephone, MP3, etc.); and change operating language environments.

MIMUS offers a symmetric architecture: graphical and voice modalities are available both at input and output. MIMUS still goes a step further in this multimodal symmetry, since any functionality can be achieved by either mixing modalities, using only voice, or just graphically. This is particularly advantageous for disabled users, who may have the system adapted to their particular circumstances. Moreover, the user can switch directly to another language, just by producing an utterance like "Switch to Spanish".

Like most of the modern systems, such as FAME [2], SMARTKOM [7], MATCH [8], GoDIS [9] and many others, MIMUS consists of a set of collaborative agents. The framework used for the communication and cooperation among the agents is the Open Agent Architecture (OAA) [10]. Figure 1 shows the agents which compose MIMUS with their configuration files and the common knowledge resource (i.e., an OWL ontology).

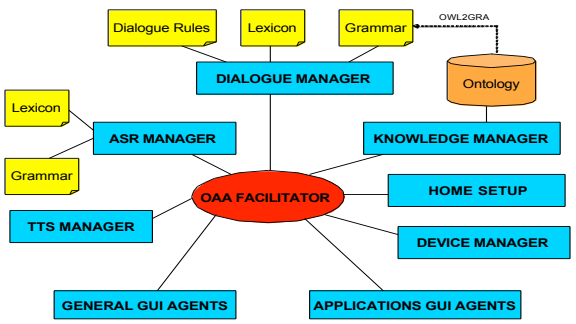

Figure 1: *MIMUS architecture.*

2.1. Dialogue Manager (DM)

This is MIMUS' main module; it may be described as a collaborative dialogue manager, linked to a NLU module on the one hand and to a Generation module on the other. This allows dialogues driven from the semantic information provided by the user, and through the dialogue expectations generated by the dialogue manager. The kernel is composed by:

- A Natural Language Understanding (NLU) module, in charge of the lexical and syntactic analysis and generator of the Information States.
- A Natural Language Generation module, which elaborates utterances from abstract content representations.
- An ISU-based [11] Dialogue Manager, which handles and modifies Information States by applying dialogue update rules. The Information States configured for this scenario are based on the DTAC protocol [12].

2.2. Knowledge Manager (KM)

This is an interface with the knowledge resource; it allows agents to do queries over the ontology, which is updated at run time. MIMUS integrates the OWL2GRA tool (http://www.grupo.us.es/julietta/tools.php), whose goal is to capture the linguistic information behind OWL ontologies, generating productions for context-free grammars by using a rule-based configuration file [13].

2.3. Home Setup (HS)

This agent offers a 3D view (Figure 2) of the house. The HS is clickable, enabling the user to refer graphically to both the devices and the rooms in the house.

Figure 2: *Home Setup*.

2.4. Device Manager

This agent controls physical devices through either the X10 (http://www.x10.com) or the Lonworks (http://www.echelon.com) protocols.

2.5. ASR Manager

This is an OAA wrapper for ASRs. MIMUS' current version is integrated with Nuance (http://www.nuance.com).

2.6. TTS Manager

This is an OAA wrapper for TTS engines. MIMUS' present implementation includes Loquendo (http://www.loquendo.com).

2.7. General GUI Agents

These agents are domain-independent components whose role is to provide different kinds of graphical output (i.e., plain text or options lists).

2.8. Application GUI Agents

These are domain-specific agents and graphical interfaces for devices such as TV applications, MP3 player, telephone, and so on.

3. The role of the Natural Language Understanding module

As we mentioned before, the NLU module goes through an analysis phase that consists of two stages: the lexical analysis and the grammatical analysis. During the lexical analysis phase the NLU module receives a string of words (given by the recogniser) as input and its goal is to output a list of the syntactic categories associated with each word. The syntactic analysis provides an information structure representation of the original input consisting of one or more Dialogue Moves (DMoves) [12]. Information structures are represented through the DTAC protocol in MIMUS. In this fashion, the input sentence "Turn on the kitchen" would be parsed into the following f-structure [14]:

```
(DMOVE:specifyCommand,
 TYPE:CommandOn,
 ARGS:[DeviceSpecifier],
 DeviceSpecifier:
        DMOVE:specifyParameter,
        TYPE:DeviceSpecifier,
        ARGS:[Location],
        Location:
                DMOVE:specifyParameter,
                TYPE:Location,
                CONT:Kitchen)))
```

In MIMUS the NLU module can switch from a language configuration to another during an ongoing interaction, without having to interrupt the dialogue between the system and the user. This is achieved by means of the "LoadNLU()" function, called from the dialogue management file, and whose goal is to unload the working NLU configuration and load the target language NLU configuration after that. This is in turn specified in an NL file where language-dependent settings (such as the grammar and the lexicon) are defined. Hence, the function "LoadNLU(English.nl)" first releases the NLU configuration for Spanish (the default one) and then activates the NLU settings for English.

4. The role of the Natural Language Generation module

Once the Information State (that is, an augmented DTAC structure) contains all the information necessary to produce the desired output message, the Generation module takes this DTAC structure and returns a feature structure with linguistic features instead, by means of predefined *sentence planning* templates [15]. Given the information state, a function is called in the dialogue management file (the "NLG()" function) which specifies both the sentence planning template to be invoked and the current information state (where this applies). After template application, a deep syntactic feature structure is obtained.

The representation thus yielded is language-independent at this point. How each conceptual feature translates into different languages is specified in the lexical transfer module, which ensures that multilingual language generation is much simpler than having to develop independent templates for each target language. The generation process in MIMUS is, therefore, subdivided into four sub-processes: sentence planning, lexicalisation, syntactic generation, and morphological generation. The syntactic representation attained after lexicalisation (lexical transfer) can then be passed on to the surface morpho-syntactic realisation module. In this phase a generation grammar will be used. Finally, the morphological module returns inflected forms of terminal symbols.

```
Template : ReplyQuantity {
 cltype =a decl;
 obj.head =a DeviceSpecifier.DeviceType.CONT;
 obj.quant =a RefRes.Quantity;
   @if (RefRes.Quantity > 1)
   @then { obj.agr.num =a pl;
   @else { obj.agr.num =a sing;
 v_pred =a there_be_pres_3sg;
 obj.pmod =a
ApplyTemplate(LocativePP,DevSpec.Location);
 obj.mod.a_pred =a DevSpec.Descriptor.CONT;
 obj.state.a_pred =a DevSpec.State.CONT;}
Template : LocativePP {
 spec =a def;
 head =a CONT;
 pcase =a loc;}
```

The NLG module can also switch configurations at run-time. Through the "LoadNLG()" function, defined in the dialogue management file, the working NLG configuration will be unloaded and the configuration file where the pertinent lexical transfer module is selected will be loaded instead. The function "LoadNLG(English.nl)" will release the NLG configuration for Spanish and load the NLG configuration for English.

5. The roles of the ASR Manager and the TTS Manager

The ASR grammar and the TTS language are controlled from Dialogue Rules, in the dialogue management configuration file. The initial language configuration, both for the Nuance ASR and the Loquendo TTS, can be set either in Spanish or in English. The ASR is configured by specifying the recognition grammar that is to be loaded (the appropriate language settings being automatically activated accordingly). In MIMUS the ASR language and the recognition grammar can be activated in the abovementioned dialogue management file, through the "SetGrammar()" function.

Switching the language in which the TTS operates is straightforward, since all of the languages can be loaded simultaneously at start time and then used at convenience throughout the dialogue. This is done through the function "SetLanguage()", also called from the Dialogue Manager module.

The ASR configuration, however, requires a more elaborate implementation. Consequently, the ASR Manager handles a pool of active grammars created for this purpose, which consists of the main grammar and a number of secondary grammars. The former covers all valid utterances in the application, and the latter only apply to language switching utterances such as "Cambia a español". This strategy allows for language switching commands in both the active and the target languages.

6. The role of the Dialogue Manager

The core of our system can be described as a collaborative Dialogue Manager linked to the previously described NLU module. This allows dialogues driven by the dialogue expectations generated by the Dialogue Manager.

As mentioned before, the Dialogue Manager is in charge of the analysis of DMoves, the application of update rules, and the storage and retrieval of information states [12].

Language change in MIMUS is achieved by means of a specific dialogue rule. This dialogue rule is activated by the user issuing a request to switch languages. Such a dialogue rule is implemented as follows (for clarification purposes, the following is a simplified version of the original rule):

```
( RuleID: SWITCH;
   TriggeringCondition:
      (DMOVE:specifyCommand,
      TYPE:SwitchLang);
   DeclareExpectations: {
     LanguageSpecifier <=
        (DMOVE:specifyParameter,
        TYPE:LanguageSpecifier);}
   SetExpectations: {
     Confirm <= (DMOVE:answerYN);}
   ActionsExpectations: {
     [LanguageSpecifier] =>
       {NLG(WhichLanguage,@is-SWITCH);}
     [Confirm] =>
       {NLG(AskConfirmation,@is-SWITCH);}}
   PostActions:  {
   @if(@is-SWITCH.LanguageSpecifier="Spanish")
      &&(@is-SWITCH.Confirm.TYPE="YES")
   @then {
         LoadNLU("../spanish/Spanish.nl");
         LoadNLG("../spanish/Spanish.nl");
         SetGrammar(".SPANISH");
         SetLanguage(Spanish);
         NLG(SwitchConfirm,@is-SWITCH);}}
})
```

"TriggeringCondition" describes the DMove that must be satisfied for the rule to be activated. In this case, any of the language switching commands will function as triggers of this rule.

"DeclareExpectations" and "SetExpectations" both define additional information needed for the rule to be fully completed. In this implementation of the SWITCH dialogue rule, the target language fulfils the "LanguageSpecifier" expectation.

"ActionsExpectations" define the actions to be taken when "DeclareExpectations" are neither part of the input nor within the dialogue history, or when other "SetExpectations" have been defined.

Finally, the "PostActions" section includes the actions to be carried out after all the expectations have been fulfilled. In the example above, the NLU and NLG configurations for Spanish will be loaded. The ASR grammar will be set to Spanish. And the TTS will start to operate in Spanish. Finally, the "NLG()" function will call the generation template "SwitchConfirm" in the just-loaded NLG configuration.

7. Advantages of this strategy

MIMUS' architecture allows for a dialogue management configuration file to be implemented either

separately (one dialogue management file per language) or jointly (a single dialogue management file for all of the languages involved). A shared dialogue management file means that a single dialogue can be carried out in two different languages. A direct benefit from this strategy is the fact that dialogue history is maintained across language change. A dialogue rule whose expectations have not been fulfilled will continue to be incomplete even after language switching has taken place:

```
U: Turn on the light.
S: Which light are you referring to?
U: Switch to Spanish, please.
S: Muy bien. Hablemos español a partir de
   ahora.
S: ¿A qué luz se refiere?
```

Independent dialogue management files, on the other hand, set the ground for a framework in which different dialogues, one for each one of the participating languages, can be held. This is, indeed, a desirable feature for multilingual dialogue systems, as Jonson [6] discusses, since interactions in different languages (entailing different cultural environments) may imply the development of clearly differentiated dialogue strategies.

8. Conclusions and future work

This paper describes the implementation of a successful rule-based approach towards dynamic language change in a multilingual dialogue system. MIMUS' reconfigurable framework is described, constituting a major enhancement with respect to the system's usability conditions.

In the present implementation of the system, however, dynamic language change can be performed by voice only. MIMUS' multimodal character demands that only graphical and mixed modality inputs be offered as well. Likewise, in the current Home Setup agent (Figure 2), the labels showing the name of the different areas where the system operates are in English, and they are static. In future versions of the system, labels should be updated in accordance with the language selected by the user. Also an extension is the implementation of the language-dependant configuration files for German (as well as the development of an ASR grammar in this language), which are ongoing tasks at the moment.

9. Acknowledgements

The work described in this paper has been partially funded by the EU project TALK (contract no. 507802), and by the Spanish Ministry of Science and Technology, under project TIN2006-14433-C02-02.

10. References

[1] J. Glass, G. Flammia, D. Goodine, M. Phillips, J. Polifroni, S. Sakai, S. Seneff, and V. Zue, "Multilingual Spoken-Language Understanding in the MIT Voyager System", Speech Communication, vol. 17, pp. 1-18, 1995.

[2] F. Metze, P. Gieselmann, H. Holzapfel, T. Kluge, I. Rogina, A. Waibel, and M. Wölfel, "The FAME Interactive Space", Proceedings of the 2nd Joint Workshop on Multimodal Interaction and Related Machine Learning Algorithms, MLMI2005 (Edinburgh, U.K.), 2005.

[3] I. Ipsic, N. Pavesic, F. Mihelic, and E. Nth. "Multilingual Spoken Dialogue System (SQEL)", ISIE'99: Proceedings of the IEEE International Symposium on Industrial Electronics (Bled, Slovenija, July 12-16, 1999), IEEE, vol. 1, str. 183-187, 1999.

[4] R. Jonson, "System Functionality Specification", Deliverable 1-1, EMATTER project, 2001.

[5] H. Holzapfel, "Towards Development of Multilingual Spoken Dialogue Systems", 2nd Language and Technology Conference, L&T2005 (Annandale-on-Hudson, N.Y., U.S.A), 2005.

[6] R. Jonson, "Multilingual NLP Methods for Multilingual Dialogue Systems", 2002.

[7] J. Alexanderson and T. Becker, "Overlay as the Basic Operation for Discourse Processing in a Multimodal Dialogue System", SMARTKOM, Report Nr. 4, 2001.

[8] M. Johnston, S. Bangalore, G. Vasireddy, A. Stent, P. Ehlen, M. Walker, S. Whittaker, and P. Maloor, "MATCH: An Architecture for Multimodal Dialogue Systems", Proceedings of the Association for Computational Linguistics, ACL (Philadelphia, U.S.A., July, 2002), pp. 376-383, 2002.

[9] S. Larsson, P. Ljunglöf, R. Cooper, E. Engdahl, and S. Ericsson, "GoDis: An Accomodating Dialogue System", ANLP-NAACL Workshop: Conversational Systems, 2000.

[10] D.L. Martin, A.J. Cheyer, and D.B. Moran, "The Open Agent Architecture: A Framework for Building Distributed Software Systems", Applied Artificial Intelligence, Vol. 13, Nr. 1-2, pp. 91-128, 1999.

[11] S. Larsson, R. Jonson, J.G. Amores, C. García, and J.F. Quesada, "Evaluation of Contribution of the Information State Based View of Dialogue", SIRIDUS, Deliverable D3.4, 2002.

[12] J.F. Quesada, D. Torre, and J.G. Amores, "Design of a Natural Command Language Dialogue System", SIRIDUS, Deliverable D3.2, 2000.

[13] G. Pérez, J.G. Amores, P. Manchón, F. Gómez, and J. González, "Integrating OWL Ontologies with a Dialogue Manager", XXII Congreso de la SEPLN (Zaragoza, Spain), 2006.

[14] J. Bresnan and R. Kaplan. Lexical-functional grammar: A formal system for grammatical representation. In J. Bresnan (ed.), *The Mental Representation of Grammatical Relations*, pp. 173-281, MIT Press, Cambridge, MA, 1982.

[15] J.G. Amores, G. Pérez, and P. Manchón, "Reusing MT Components in Natural Language Generation for Dialogue Systems", Procesamiento del Lenguaje Natural 37, pp 215-221, 2006.