An Integration Framework for a Mobile Multimodal Dialogue System Accessing the Semantic Web

Norbert Reithinger, Daniel Sonntag

DFKI GmbH – German Research Center for Artificial Intelligence
Stuhlsatzenhausweg 3
D-66123 Saarbrücken, Germany
{bert,sonntag}@dfki.de

Abstract

Advanced intelligent multimodal interface systems usually comprise many sub-systems. For the integration of already existing software components in the SMARTWEB[1] system we developed an integration framework, the IHUB. It allows us to reuse already existing components for interpretation and processing of multimodal user interactions. The framework facilitates the integration of the user in the interpretation loop by controlling the message flow in the system which is important in our domain, the multimodal access to the Semantic Web. A technical evaluation of the framework shows the efficient routing of messages to make real-time interactive editing of semantic queries possible.

1. Introduction

Advanced intelligent multimodal interface systems usually comprise many sub-systems. Needed functionalities are, e.g. modality recognition for speech or gestures, modality interpretation and fusion, intention processing, modality fission, and finally, result rendering for graphics on a screen or synthesis of speech. For many of these sub-tasks software modules from academia or industry can be used off the shelf. Furthermore, in many projects integration frameworks for this types of systems have been developed and exist, like the Galaxy Communicator, Open Agent Architecture, Multiplatform, the General Architecture for Text Engineering or Psyclone [1, 2, 3, 4, 5]. The W3C consortium also proposes inter-module communication standards like the Voice Extensible Markup Language VoiceXML[2] or the Extensible MultiModal Annotation markup language EMMA[3], with products from industry supporting these standards[4].

Against this rich background of available knowledge, the process to start a new project should be straightforward: you write down your list of needed functionalities and technical requirements, fill your shopping cart with all the necessary technology, and build your system in a straightforward way.

However, if one looks closer to the actual project's requirements, this idealistic vision begins to blur. In the project SMARTWEB [6] we develop a context-aware, mobile multimodal user interface to the Semantic Web. The user carries a smartphone as interaction device. He poses questions via speech and gestures on the device. The multi-modal input is transmitted using UMTS or wireless LAN to a back end server system. There is where the server-based multi-modal recognisers, the dialogue system, and the Semantic Web access subsystems run. Whereas the computer-telephony platform and the recognisers are based on commercial products from Sympalog GmbH[5], the core dialogue engine and the access functions to the Semantic Web are the main focus of our research.

In this article, we will first analyse the requirements for dialogue processing in SMARTWEB and will then introduce the component architecture we developed. Attention will be drawn to implementation issues of the information hub (IHUB) and its performance to enable real-time interactive queries to the Semantic Web.

2. Requirements

Dialogue processing in SMARTWEB differs from traditional dialogue systems in various ways. Roughly speaking, a traditional NLP dialog system realises the recognise - analyse - react - generate - synthesise pipeline [7]. Once the recogniser has started, the information in the pipeline is propagated until the end, which means, that the user/system interaction is reduced to user and result messages (Figure 1). If the user uses "barge-in" to stop or modify his input, it might interrupt the output of the synthesised reaction. However, it is unclear whether or not the internal state of the system reflects this interruption. In more or less stateless environments like VoiceXML based solutions this is a viable way to go.

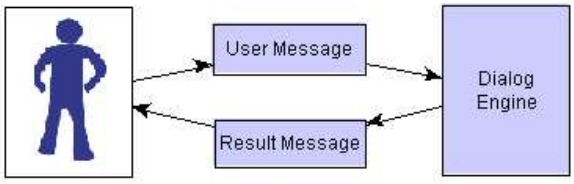

Figure 1: Traditional Dialog Engine architecture. The user is presented with final result at the end of the processing pipeline.

The dialogue engine of SMARTWEB realises the analyse - react - generate steps from parsing the output of the speech recogniser over the connection to the Semantic Web interface and finally to the rendering of the graphics and generation of

[1] http://www.smartweb-project.de
[2] http://www.w3.org/TR/voicexml20
[3] http://www.w3.org/TR/emma
[4] http://www.voicexml.org
[5] http://www.sympalog.de

the input to the speech synthesiser. The I/O interfaces use the W3C recommendations EMMA for multimodal I/O representations, RDF for the interaction with the Semantic Web interface and the Speech Synthesis Markup Language SSML to connect to the speech synthesis subsystem. In our highly interactive, context-sensitive and multimodal system environment we have to find advanced solutions to control the flow of information in the dialogue system. SMARTWEB accesses the open domain of the Semantic Web. This requires a large language vocabulary for recognition and processing as basis. Also, the user has to have the opportunity to intervene early in the processing to correct and clarify the input.

Formal queries to the Semantic Web are posed in an explicit query language which are ontology instances serialised as EMMA/RDF XML-files. The user interaction is focused on deriving this representation from the user input, which should be voice input as standard, despite its interpretation difficulties. In combination with the mobile use with its varying acoustic conditions, this requires immediate multimodal feedback on the smartphone.

In general, situations where the user can intervene are the direct manipulation of interpretation results with the pen and the indirect manipulation with speech. The pen can be used to correct the best recognition result. It is immediately presented on the screen and the user is able to edit, delete or insert words. In the next step, he can correct the understood semantic content: the output of the dialogue speech interpretation module will be paraphrased and realised on the screen. The user can again edit, delete or insert concepts. This representation is intentionally similar to the Semantic Web query being posed, but the user can still intervene, before the request is sent to the Semantic Mediator. This module is the gateway to the Semantic Web and addresses different strands of search technology, such as a semantic Q&A-system, or an ontology database. Of course, the user can cancel a request if the answer is not important anymore - the reasons might vary, e.g., the system is searching too long. A CANCEL button is visible permanently on the interface.

Using speech, the user can correct an utterance as soon as he realises the error. An example like

> "I did not mean Fuji but Fiji."

can be uttered during the processing of the sentence or after the system delivered an answer to the original utterance.

All these interventions can be viewed as barge-in. The direct manipulation actions on the interface are clearly defined and can be processed in a (relatively) straightforward way in the processing chain of the dialogue module. However, if the user corrects items on the screen, a re-interpretation has to take place. The underlying architecture must support this fast-and-easy re-interpretation. Figure 2 outlines the user in the interpretation loop, where REAPR is the acronym for the reaction and presentation module. Speech based barge-in differs from that processing mode. While the direct manipulation takes place inside an interpretation step, a new utterance from the user requires, for first, a complete interpretation of the utterance and, for second, the initiation of the intended activities [8].

Though we need the flexibility to cope with direct user interactions, we still want and need to reuse well-tested off the shelf components available at our lab that were developed over the years in various projects like Verbmobil[6] [9], Smartkom[7] [10], Miamm[8] [11], and Comic[9] [12], and for various integration platforms. All these components are written in Java.

Another requirement of our integration platform is that it must be fairly easy to replicate a complete dialogue module: SMARTWEB will be demonstrated in a realistic environment where multiple users are able to connect to the access system. As a consequence the systems structure must ease the start of new dialogue module instances to process the requests of new users.

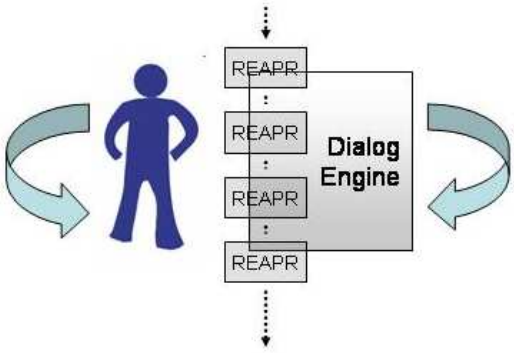

Figure 2: User in the Interpretation Loop.

3. Dialog system components

Even though we have to integrate different, self-contained processing modules, we benefit from the fact that all are written in Java. The architecture we developed follows the hub-and-spoke architecture, and is topologically similar to the Galaxy Communicator and the Open Agent Architecture (OAA). In contrast to OAA the hub in our system – which we named IHUB – does not reason about the content of a message. It only routes the messages between the components and controls the validity of the messages. Figure 3 shows all modules in our dialogue system.

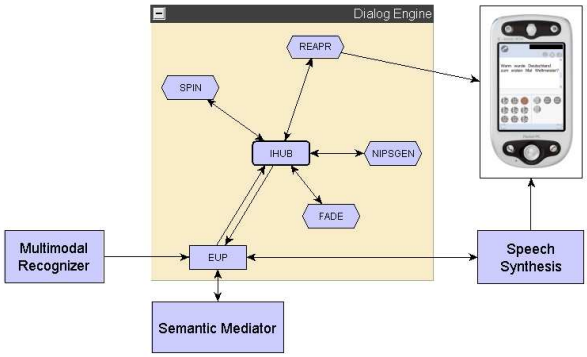

Figure 3: Server-based Dialogue System Components

The dialogue module employs simple and clean interfaces to external modules such as the speech recogniser. The EMMA Unpacker and Packer (EUP) component provides the communication with the external world and communicates with the other

[6] http://verbmobil.dfki.de
[7] http://www.smartkom.org
[8] http://www.miamm.org
[9] http://www.hcrc.ed.ac.uk/comic

modules in the system, namely the Multimodal Recogniser, the Semantic Mediator which provides the interfaces to the Semantic Web, and the Speech Synthesis.

The actual processing starts with the speech interpretation component (SPIN). Its input is mainly the N-best word chains, the output is a parsed word chain containing an RDF/S description for the analysable parts of the chain.

The modality fusion and discourse component (FADE) keeps track of the ongoing discourse, completes different types of anaphora, and merges input from different modalities. Input is mainly a parsed word chain, and the gesture description on an ontological level with geometric position. The output is the enhanced word chain (e.g. resolved co-references) and the completed ontological description of the user's utterance.

The system reaction and presentation component REAPR decides what to do with the interpretation of the user's input. It gets all input from SPIN (the processed best word sequence) and FADE (the query's words and completed ontological description). These intermediate results are presented on the screen and can be edited by the user. If the user changes the paraphrase or original input, the message is sent back to the originating module for reinterpretation. Finally it sends the user's request to the Semantic Mediator. The results obtained from the Semantic Mediator are prepared for presentation and rendered on the mobile device.

If there is a semantic representation as answer, the text generation module NIPSGEN generates a text, which is sent to the speech synthesis.

All messages are routed through the IHUB. It knows about the scheduling requirements between subcomponents and the addressees of particular messages. This knowledge is the centre piece of the hub-and-spoke architecture and is embedded in IHUB's rule base. The internal communication between components is based technically on the exchange of objects containing ontological information, using the Jena Semantic Web framework[10] for Java. We would like to place emphasis on the ontology instances that are communicated between the components. Every component adds its processing output in ontological form. The result being sent to the Semantic Mediator is the ontological semantic description of the user query.

Additionally, the IHUB has to block messages which are no longer valid. Consider a SPIN result that was passed both to FADE for discourse processing and to REAPR for immediate control by the user. If the user changes a word, REAPR sends back the new message to SPIN. FADE, in the meantime, did already process the previous interpretation and passes it on to the IHUB for further distribution.

To control the messages we employ a simple mechanism: All messages communicated have a *turn* (major) and a *job* (minor) number. If the user corrects any input, the job number of the resulting message is increased. All jobs with a lower job number are invalidated and blocked by the IHUB. This correction mainly takes place at two processing stages during interaction: Firstly, the voice input must be interpreted as natural language input. Secondly, the natural language input (text) must be semantically parsed and transformed to a Semantic Web Query. The IHUB's message blocking behaviour is designed for allowing flexible corrections and reinterpretations at these stages.

Our message blocking design is beneficial in that we do not have to change the internal logic of the components of the dialogue system. They terminate processing all incoming messages, the produced output may be discarded then. This is safe for all components without a memory. i.e., for the majority. Modules like FADE which contain the dialogue memory, amongst other things, have to augment the standard modus operandi. We therefore introduced two additional signals (messages) in the architecture.

- COMMIT: if REAPR sends the final interpretation of the user's input to the Semantic Web, it sends this signal, including the turn and job number, to the IHUB, which distributes it to all components. These modules then know which interpretation is the "final" one and can update the respective persistent memories.
- CANCEL: if the user pushes the CANCEL button or says something like "Stop it", the respective turn/job number combination is also distributed throughout the system. If a component with memory is currently in the process of interpreting this input, all intermediate representations can be erased, or, if a COMMIT was already sent, the entries must be marked in the dialogue memory as cancelled by the user. Since the user can refer to his own, albeit cancelled, input the information in the dialogue memory can still be useful.

4. Implementation

Following the general object-oriented software development framework, we implemented the basic IHUB functionality with a main controller class and interfaces for the components which are realised as threads in one Java environment. The interface contains the basic send/receive methods and functionality for basic error handling. Control functionality comprises the controlled start and termination of components and the restart of threads in the case they have thrown an error previously. Also, a basic control GUI is available to see "inside" the system and to support development and debugging.

One dialogue engine for one user therefore comprises of an IHUB controller as master process and one thread for each component in figure 3. The interface to the outside world, the EMMA unpacker and packer EUP, also runs as independent thread. If it gets a new EMMA message from the multimodal recogniser, it creates the internal, Jena based representation of that message. All control information, like turn number, and start and end time, are copied from the corresponding EMMA slots and inserted to the Jena message. EUP then sends the message to the IHUB with job number 1 for further processing.

In the IHUB, a straightforward rule based mechanism decides on the addressees of a message based on the sender. The base work flow is guaranteed by implication rules that at first take sender information, including time, turn and job number, as antecedent. Implication rules are expressions like $X \Rightarrow Y$, where X and Y are disjoint sets of items. Our implications are

- $EUP \Rightarrow REAPR, SPIN$ (immediate representation, interpretation)
- $SPIN \Rightarrow REAPR, FADE$ (immediate representation, discourse processing)
- $FADE \Rightarrow REAPR$ (immediate representation)
- $REAPR \Rightarrow FADE, SPIN$ (commit rendering, increase job number)

These rules are crafted by hand and represent the flow of the dialogue system along the principal design of the components. Since a message may have several addressees, which may synchronously update a message, we need a criterion to

[10] http://jena.sourceforge.net

discard older messages. Therefore a component must increase the job number if the message was *altered* in the sense that the *user* changed the content through intervention in the interpretation loop. The IHUB then knows that messages with lower job numbers for the same turn must be discarded. Note that components that *enrich* the description of a message, like FADE, do not increase the job number, since the content as stated by the user remains the same basically.

The rules in the IHUB don't analyse the content of a message beyond the sender and the turn and job numbers. Therefore, the logic within the hub is fast and does not delay processing significantly longer than a direct module-to-module communication path.

In a first performance evaluation we measured whether there is any significant communication overhead, whether the passing of structured objects i.e. instances of an ontology does not hamper the speed of message passing and how message blocking for a new interpretation performs.

The exchange for messages containing between 100 and 10000 characters in a realistic message exchange pattern between the components needed between 5.22 ms to 5.72 ms. This time does not vary even if you repeat the experiments up to 10000 times. Medium message run-times between subcomponents over the IHUB for structured data using the Jena data structures are also in the 5ms range on a standard laptop (Pentium 4, 2.66 GHz) In a further experiment where we utilised the message blocking feature the processing times show almost no measurable differences to the processing times as presented in both previous tests.

5. Conclusion

In this paper we motivated and presented the implementation of an integration framework for a mobile multimodal dialogue system accessing the Semantic Web. The IHUB allows us to reuse already existing components for interpretation and processing of multimodal user interactions. The framework was developed with two major requirements. On the one hand it has to facilitate the integration of the user in the interpretation loop which is important in our domain, the multimodal access to the Semantic Web. The Semantic Web query must be as explicit and as accurate as possible, therefore we give the user the power to control and enhance the interpretation process of his query. On the other hand, the integration framework must support that the resulting dialogue system can be started in multiple incarnations to enable an efficient multi-user operation of the SMARTWEB access system. Since existing frameworks like OAA or Galaxy Communicator support these requirements only partially, if at all, we came up with a new realisation of the hub-and-spoke architecture for dialogue systems.

In the future we may need further intelligence to the IHUB in terms of the rule base and the expressivity of the rule language. The use of Jena provides us with a rule engine which is able to encompass more of processing workflow intelligence. We would then be able to resort to our domain and discourse ontologies in order to create rules based on the information content of a message. However, we strive for simplicity in the IHUB and will add these functionalities only if really needed.

6. Acknowledgments

We would like to thank Tim Gehrmann and Alexander Pfalzgraf for fruitful discussions and for their help during implementation and evaluation of the system, and Markus Löckelt for useful input on earlier versions of this paper.

This research was funded by the German Federal Ministry for Education and Research under grant number 01IMD01A. The views expressed are the responsibility of the authors. Points of view or opinions do not, therefore, necessarily represent official Ministry for Education and Research position or policy.

7. References

[1] S. Seneff, R. Lau, and J. Polifroni, "Organization, Communication, and Control in the Galaxy-II Conversational System," in *Proc. of Eurospeech'99*, Budapest, Hungary, 1999, pp. 1271–1274.

[2] A. J. Cheyer and D. L. Martin, "The Open Agent Architecture," *Autonomous Agents and Multi-Agent Systems*, vol. 4, no. 1–2, pp. 143–148, 2001.

[3] K. Bontcheva, V. Tablan, D. Maynard, and H. Cunningham, "Evolving GATE to Meet New Challenges in Language Engineering," *Natural Language Engineering*, vol. 10, 2004, special issue on Software Architecture for Language Engineering.

[4] G. Herzog, A. Ndiaye, S. Merten, H. Kirchmann, T. Becker, and P. Poller, "Large-scale Software Integration for Spoken Language and Multimodal Dialog Systems," *Natural Language Engineering*, vol. 10, 2004, special issue on Software Architecture for Language Engineering.

[5] K. R. Thorisson, C. Pennock, T. List, and J. DiPirro, "Artificial intelligence in computer graphics: A constructionist approach," *Computer Graphics*, pp. 26–30, February 2004.

[6] W. Wahlster, "Smartweb: Mobile applications of the semantic web," in *GI Jahrestagung 2004*, P. Dadam and M. Reichert, Eds. Springer, 2004, pp. 26–27.

[7] J. Allen, D. Byron, M. Dzikovska, G. Ferguson, L. Galescu, and A. Stent, "An Architecture for a Generic Dialogue Shell," *Natural Language Engineering*, vol. 6, no. 3, pp. 1–16, 2000.

[8] R. Carlson, J. Hirschberg, M. Swerts, and G. Skantze, Eds., *ISCA Tutorial and Research Workshop on Error Handling in Spoken Dialogue Systems*, 2003.

[9] W. Wahlster, Ed., *VERBMOBIL: Foundations of Speech-to-Speech Translation*. Springer, 2000.

[10] ——, *SmartKom: Foundations of Multimodal Dialogue Systems*. Berlin: Springer, 2005.

[11] N. Reithinger, D. Fedeler, A. Kumar, C. Lauer, E. Pecourt, and L. Romary, "MIAMM - A Multimodal Dialogue System Using Haptics," in *Advances in Natural Multimodal Dialogue Systems*, J. van Kuppevelt, L. Dybkjaer, and N. O. Bersen, Eds. Kluwer Academic Publishers, 2005.

[12] N. Pfleger, "Context based multimodal fusion," in *ICMI '04: Proceedings of the 6th international conference on Multimodal interfaces*. ACM Press, 2004, pp. 265–272.