

VERBMOBIL: The Evolution of a Complex Large Speech-to-Speech Translation System

Thomas Bub, Johannes Schwinn

German Research Center for Artificial Intelligence GmbH (DFKI)

Postfach 2080, D-67663 Kaiserslautern

{bub, schwinn}@dfki.uni-kl.de

ABSTRACT

The ambitious goal of the project *VerbMobil*¹ is the development of a portable speech-to-speech translation system dealing with face-to-face dialogs. It constitutes a new generation of translation systems in which spontaneously spoken language, speaker independence and speaker adaptability are among the main features. *VerbMobil* brings together researchers from the fields of speech processing, computational linguistics and artificial intelligence and goes beyond the state of the art in these areas.

Besides the speech and language processing issues, the specific constraints of the project represent an extreme challenge on the part of project management, software engineering and test and evaluation of the system:

- size and complexity: 150 researchers from 29 organizations at different sites on three continents are involved in the software development,
- integration of heterogeneous software: in order to reuse existing software, hardware and know-how, only a few restrictions were given to the partners.

In the following article we describe the *VerbMobil* scenario, the system architecture and the system evolution.

1. INTRODUCTION

The objective of *VerbMobil* is to develop a portable simultaneous speech-to-speech translation system for face-to-face negotiation dialogs [1]. The *VerbMobil* scenario assumes a Japanese and a German manager discussing the date of their next meeting. As usual in similar situations both partners speak English. According to the scenario their English understanding competence is higher than their speaking performance. In case the active knowledge of English turns out to be insufficient, the dialog partners are allowed to use their native language. The *VerbMobil* system supports them by translating from their mother tongue, i.e. Japanese or German, into English (figure 1).

The project involves about 150 researchers and engineers from 29 organizations at different sites in Germany, USA and Japan. These partners develop modules which are centrally integrated to build an

operational *VerbMobil* system. The software is developed in different environments (Solaris, UX, OSF/1, AIX, Windows) and implemented in different programming languages (C, C++, Lisp, Prolog, tcl/tk, Fortran).

Because of the large number of organizations involved in the project and the complexity and heterogeneity of the produced system pieces, system integration and evaluation is, besides the functional objectives of the project, an extremely challenging task.

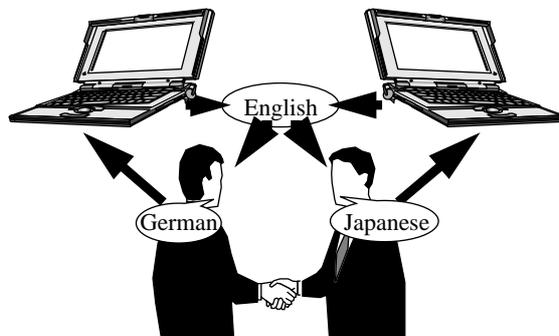


Figure 1: *VerbMobil* supports the translation of negotiation dialogs.

2. FUNCTIONAL ARCHITECTURE OF THE VERBMOBIL PROTOTYPE

VerbMobil translates face-to-face negotiation dialogs. While the dialog partners speak English, the system spots for keywords in order to compute a dialog history. From this passive mode the system may turn to the active mode when a speaker wishes to continue in his mother tongue. The output of the continuous German or Japanese speech recognizers is then analyzed with respect to syntactic, semantic and pragmatic aspects, transferred into English and, after generation of suitable English phrases, passed to an English speech synthesizer.

In addition to the passive, *keyword spotting mode* and the active, *deep analysis and transfer mode*, additional features include *shallow analysis and transfer* and *single word transfer*. The latter allows the user to use the system as a speech driven dictionary. In all modes, the system operates on the same domain model and dialog history.

2.1. The Modularized Functional Architecture

The fundamental concept of the functional architecture of *VerbMobil* is that of interactively communicating modules, i.e. the functionality of the

1. This work was funded by the German Federal Ministry of Education, Science, Research and Technology (BMBF) in the framework of the *VerbMobil* Project under Grant 413-4001-01 IV 301 4. The responsibility for the contents of this paper lies with the authors

In cases of system faults and clarification dialogs, the *German generator* produces prosodically annotated German text that will be synthesized by the *German synthesis*.

Besides the above mentioned mainstream modules, there are several modules for *shallow analysis and transfer* to produce faster, but less reliable translations. Like the deep analysis modules, these modules take word lattices as input and produce text or annotated text. Because different translations of the same turn may be computed, it is the task of the module *selection* to select a suitable result to be synthesized. Different rule sets allow for different selection criteria, e.g. system performance versus translation quality.

During first acceptance tests of the system it turned out that users expect a complex and powerful system such as *Verbmobil* to be able to translate single words as well. We therefore introduced the module *single word transfer* which maps source language words onto target language words.

All in all, there are 24 functional modules resulting in about 60 communication interfaces. Only a small set of data formats is used: pcm-coded acoustic data, word lattices, *vit*-terms and text strings.

3. SOFTWARE ARCHITECTURE OF THE VERBMOBIL PROTOTYPE

The software architecture of *Verbmobil* reflects the functional architecture by mapping the interactively communicating functional modules to message-passing software-components.

The following concepts are used to fulfill the software technical requirements.

- **Breadboard:** The basic software-technical concept is the *breadboard*. The fundamental idea of this concept is that the *Verbmobil* system consists at the beginning of dummy modules, simulating the mapping of input to the corresponding output data. Hence, system integration is nothing else but successive replacement of modules by modules with increased functionality.

A further advantage of this concept is that, once the integration framework exists, an operational system is available and can be used for the evaluation and testing of single modules in the context of the integrated system.

- **Communication:** The concept of communication between heterogeneous modules is based on three layers of abstraction from low level communication protocols to an easy-to-handle process communication environment that is available from within all used programming languages.

- **User Interface:** The main window of the user interface displays the system's functional architecture and the state of the individual modules. Components are individually configurable via dialog boxes. If the component provides an internal visualization, it can be switched on or off. The *user interface manager* maintains a history of sent and received messages and allows the simulation of all messages between components. In this way, the complete system behavior can be simulated for test and debug purposes. The

graphical user interface can be adapted to the needs of different users.

- **Visualization of Exchanged Data:** The task of the *visualization manager* is presentation of complex acoustic and linguistic data structures that are exchanged between modules so that the user can verify and edit them. Depending on the data type a suitable visualization tool is selected. If needed, all data can be archived for further analysis.

- **Test Environment:** An automatic test environment is easily obtained by simulating the pressed *speech button* and by automatically loading prerecorded dialogs into the system. To achieve this, only changes of the *user interface manager* were necessary. All results are automatically archived.

4. SOFTWARE-TECHNICAL REALIZATION

- **Components:** Functional modules are reflected by components, which in turn are realized by at least one process which may recursively start and control further processes. To meet the software-technical requirements additional technical components were introduced, namely the components *testbed manager (tbm)*, *user interface manager (gui)* and *visualization manager (vim)*. The realization of the technical components does not differ from that of functional components, i.e. they are interactively communicating processes.

- **Breadboard:** The breadboard is realized as a technical component, called *testbed manager*, and the configuration file which describes the actual system configuration. The *testbed manager* starts, controls and stops the processes that realize the components and supervises their communication.

- **Communication:** The lowest layer of the communication environment is realized by PVM (Parallel Virtual Machine) [3]. It is used by ICE (Intarc Communication Environment) [4] to implement the second layer which offers the needed subset of functionality in all used programming languages. The third communication layer is programming language dependent. It reduces the programming effort for component implementers to the handling of received messages.

5. MODULE INTEGRATION AND SYSTEM EVOLUTION

The continuous evolution of modules leads to the evolutionary enhancement of the whole system. It must be kept in mind that this evolutionary integration process requires the existence of a first complete "bootstrapping" system.

Whereas the breadboard concept seems to be clear and straightforward, we faced a lot of practical problems that we will illustrate by examples.

- **Non-monotonic System Evolution:** Whereas the evolutionary model of integration should assure that the system functionality increases monotonically, this was not always the case. The reasons were either technical, e.g. components were not upward compatible with earlier versions, or were due to the research character of *Verbmobil* when recent insights required a fundamental change of the applied methodology.

- **Global Consequences of Local Decisions:** Because of the complexity of the system and the mostly local point of view of the involved partners, it may happen that the side-effects of local changes have not been foreseen by

the responsible partners. It is the task of the *system integration group* to recognize such situations and to discuss and find suitable solutions.

• **Cyclic Dependencies:** In order to provide a certain functionality, module A needs input from module B which in turn requires prior information from module A. Development and testing of cyclically dependent modules is done either by isolating single functions and sequentializing the dependencies or by using manually generated test data.

6. MODULE AND SYSTEM EVALUATION

The *Verbmobil* breadboard architecture allows the exchange of modules against new module versions. This local growth of performance leads to the continuous evolution of the *Verbmobil* System. In order to judge the effect on the overall system performance, single modules as well as the complete integrated system are continuously evaluated.

Test Data

We distinguish test and evaluation data. Test data have been purposefully selected to evaluate the processing of specific phenomena. Evaluation data are used to demonstrate module and system performance on unseen realistic data. Whereas test data may be public at module development time, evaluation data are only published after the evaluation. As the recording and transcription of dialogues is expensive, both types of data will become part of the training corpus afterwards.

Evaluation Process

Recorded spontaneous dialogues are automatically turnwise fed into the system. Input and output of all modules, i.e. the module interfaces, are archived for further evaluation.

During end-to-end evaluation the transcription of the audio data is compared with the generated translation. This is done for the different, deep and shallow, analyses and transfers. We use interactive world wide web pages for the manual judgement of the results by competent interpreters. Translations are considered approximately correct, if the translation is understandable and if the original information content has been preserved.

Selected modules or groups of modules are evaluated by further analyzing the archived input-output data. Wherever possible this process is either automated or supported by software tools. For example, the recognizer output is automatically compared to the transcription of the audio signal. As metric for the acoustic fault rate, the minimal Levensthein distance of all possible word chains in the word lattice is used. Since the beginning of the project, the word accuracy rate has continuously increased. The actual word accuracy rate amounts to 70 %.

For most other interfaces, e.g. the syntactic-semantic analysis, manual judgement is necessary, which is supported by suitable visualization tools.

Use of Test Data for Module Development

Because of the parallel development of modules there do not always exist consistent versions of all concerned modules. Thus the interfaces of modules have to be simulated with test data during development. The breadboard architecture allows the use of dummy modules mapping given input data to hand-coded or semi-automatically generated output.

Problems Encountered

As the *Verbmobil* architecture is not strictly sequential, the evaluation of isolated module functionality becomes difficult. The output of a module may depend on the quality of the input of several others or may depend on the quality of requested information from other modules. E.g. if ambiguities appear *transfer* will request resolution by *semantic evaluation*, thus the quality of the transfer result depends strongly on the quality and robustness of *semantic evaluation*.

Other aspects concern the operational architecture, e.g. if a module is allowed to interrupt another module when certain preconditions are fulfilled. These phenomena can hardly be tested.

7. CONCLUSION

The evolutionary process of system integration and the architectural concept have proven suitable. The implemented breadboard architecture and the transparent interface to the communication mechanisms provide a flexible framework for the realization of *Verbmobil* and its ongoing evolution. They assure that it is possible to integrate a set of completely new components within only a few days. Moreover, the architecture of *Verbmobil* supports the easy reuse of the concerned modules in other contexts.

This framework is also used for the continuous evaluation of the modules and system evolution. The production of test results has been completely automated. The evaluation of the test results has been partially automated, where suitable machine-processable metrics exist, e.g. for the acoustic recognition, but requires human intervention, where the results are not deterministic, e.g. for the end-to-end evaluation of the translation quality.

8. REFERENCES

1. Wahlster, W.: "Verbmobil: Translation of Face-to-Face Dialogs". Proceedings of MT Summit IV, Kobe, Japan, July 1993.
2. Dorna, M.: "The ADT-Package for the Verbmobil Interface Term". To appear as Verbmobil-Report, 1996, DFKI, Saarbrücken
3. Geist, A. / Benuelin, A. / Dongarra, J. / Jiang, W. / Manchek, R. / Sunderam, V.: "PVM 3 User's Guide and Reference Manual". Oak Ridge National Laboratory, Oak Ridge, Tennessee 1994.
4. Amtrup, J. W.: "ICE INTARC Communication Environment Design and Specification". Verbmobil Memo 48, University of Hamburg, 1994.