

# Mobile Architecture for Distributed Multimodal Dialogues

*Markku Turunen, Esa-Pekka Salonen, Jaakko Hakulinen, Juuso Kanner and Anssi Kainulainen*

Speech-based and Pervasive Interaction group, Tampere Unit for Computer-Human Interaction,

Department of Computer Sciences, University of Tampere, Finland

{Markku.Turunen, Esa-Pekka.Salonen, Jaakko.Hakulinen, Juuso.Kanner,  
Anssi.Kainulainen}@cs.uta.fi

## Abstract

There is an increasing need for mobile spoken dialogue systems. Mobile devices, such as smartphones and personal digital assistants, can be used to implement efficient speech-based and multimodal interfaces. Currently, the development of such applications lacks suitable tools. We introduce general system architecture for mobile spoken dialogue systems. The architecture is a mobile version of the existing Jaspis architecture. The architecture makes it possible to implement distributed dialogue systems using compact software agents in flexible ways both on the server and mobile devices. We present how an existing server-based timetable application is turned into a multimodal mobile application. The server handles the overall coordination of the dialogue by generating dialogue task descriptions using VoiceXML, and the mobile device realizes the dialogue descriptions with available resources.

## 1. Introduction

Most contemporary spoken and multimodal dialogue systems are either desktop applications or server-based systems. In the past few years the interest towards mobile spoken and multimodal dialogue applications has been increasing. New devices, such as smartphones, have been used mainly as terminal devices. However, resources of mobile devices can be used to handle more versatile tasks than just acting as simple terminals. Furthermore, when parts of the processing takes place in the mobile devices they can use multimodality (e.g. graphics and haptics) to support speech interfaces when available. Finally, dialogues can be distributed between multiple servers and mobile devices to create truly distributed multimodal interfaces.

In this paper we introduce a general system architecture for mobile spoken dialogue systems. The architecture is a mobile version of the agent-based Jaspis architecture used for many desktop and server-based spoken dialogue systems [13]. With the unified base architecture and compact software agents we are able to create versatile applications where the tasks can be distributed efficiently between servers and mobile devices. In addition, the mobile architecture is capable of running complete spoken and multimodal dialogue applications in the mobile device. This allows iterative construction of mobile applications starting from the situation in which most of the functionality is performed on the server, but gradually more functionality can be transferred from the server to the mobile device in the form of compact agents.

As an example, we present two versions of the same timetable system. The speech-only version is fully server-based, and is accessed with fixed-line or mobile phones. The multimodal version of the system is distributed between a server and a smartphone. In both versions the server generates high-

level dialogue descriptions that are carried out by the smartphone or the server with available resources. Mostly the same components are used both on the server and the mobile device.

## 2. Jaspis architecture

The Jaspis architecture is a generic system architecture for adaptive speech-based applications. It has been used to construct several spoken dialogue systems for domains such as e-mail and bus timetables. The architecture addresses challenges of speech-based systems, such as multilinguality, error management, user guidance and feedback, robustness, and accessibility. An overview of the architecture and its applications can be found from [13].

In addition to traditional spoken dialogue systems, the Jaspis architecture has been extended to support applications for pervasive computing [6]. The resulting Jaspis<sup>2</sup> architecture [11] has support for concurrent distributed dialogues and complex distributed systems. In this paper we present an implementation of the core architecture for mobile devices running J2ME (Micro Edition of the Java 2 Platform) environments, such as Series 60 mobile phones. The resulting architecture has been used to implement distributed multimodal spoken dialogue applications for smartphones in the bus timetable domain [10]. Next we present the architecture basis.

The Jaspis architecture offers building blocks for speech systems in the form of three kinds of software components called agents, evaluators and managers. The functionality of the system is distributed across numerous compact agents, and multiple evaluators are used for dynamic reasoning of how to select between the agents. Managers are used to coordinate agents and evaluators. A more comprehensive description of these components follows.

There are several definitions for software agents in the literature. In contrast to most software agents, Jaspis agents are compact and handle well-defined, highly specialized, and preferably atomic tasks. In practise this means that Jaspis agents are quite small (most agents contain no more than tens of lines of code) and Jaspis-based systems contain rather many agents (some implemented Jaspis-based applications consist of hundreds of agents). These highly modular and small components are ideally suitable for mobile devices with limited resources.

Agents can work together in two ways in Jaspis-based systems. First, the functionality of the system is distributed between numerous agents, as mentioned previously. Each agent is highly specialized, and the execution of an application consists of a sequence of agents selected after each one. In this way, the task at hand is constructed iteratively by a set of agents, each one contributing to the overall solution. In many cases, a task is performed by selecting one agent from

each system module, such as first selecting a speech recognizer agent, then a natural language understanding agent, followed by a dialogue agent, then a domain agent, a response generation agent, and finally a synthesizer agent. However, in practical spoken dialogue systems the interaction between agents can be more complicated, for example there can be multiple dialogue and domain agents interacting in non-trivial ways before a suitable answer to the user question can be generated.

The second way that Jaspis agents can work together is the use of multiple agents for the same purpose. The agents are able to carry out the same task, but in different ways. The agents may perform closely similar functionality, such as two response generation agents producing the same content with different words, or they may carry out the task with more varying functionality, such as two dialogue agents using different error correction techniques (e.g., implicit or explicit confirmations).

Agents are organized into modules specialized for certain tasks. Typical spoken dialogue systems include modules for tasks such as dialogue management and natural language generation. Figure 1 illustrates different modules of the Stopman bus timetable system [14]. Every module contains two other types of components beside the agents. Managers coordinate other components, and evaluators determine how suitable the agents are for specific tasks. Together, they provide a system-level adaptation mechanism to be presented next.

In the system-level adaptation mechanism an evaluation process is applied for every system task to select one of the available agents to carry out the task. In practise, the evaluation process is performed by a set of evaluators. When evaluation is applied, every evaluator gives an estimate for each agent in the module. The estimates tell how suitable the agents are for the current task. Since every evaluator estimates only specific aspects of the agents, the adaptation process is distributed and thus easily extendable by adding new evaluators. In addition, each agent has its own self-adaptation mechanism. The details of the adaptation mechanism can be found from previous Jaspis publications [13].

Compact software agents and the system-level adaptation mechanism make it possible to construct highly distributed and dynamic multimodal spoken dialogue systems. Compact reusable agents can be used across different devices, customized efficiently when necessary, and selected dynamically (e.g., based on the modalities available in the mobile devices) with the build-in adaptation mechanism. For example, the task at the hand can be handled with one set of agents when the user interacts with a speech-only device, while another set of agents can be used when the user switches to a device capable for multimodal interaction.

Next, we present the mobile version of the Jaspis architecture, and after that an example application that utilizes the architecture will be presented.

### 3. Mobile Jaspis

The first implementation of the mobile version of the Jaspis architecture is targeted for devices supporting J2ME (Java 2 Micro Edition), such as smartphones on Symbian platforms. The design of the mobile version followed closely the design of the original architecture. The highly modular approach of the base architecture made the conversion quite straightforward, and the mobile version of the architecture contains

mostly the same modules and components as the original architecture. In addition, the mobile version has additional and extended components to handle tasks specific to mobile interaction. Specialized software agents can be constructed efficiently using the object-oriented software development approach and inheritance in particular [12]. Similar approaches are used in other systems (e.g. [1] , [7] and [9]). Next, we present the issues specific to the mobile version.

While regular Jaspis applications are running as stand-alone processes, mobile applications are running as *midlets*. In midlets, all graphical user interface elements and their events are accessed via the shared display object. A very similar approach is used in Java applets. In order to create a generic solution, a new concept, *running context*, was introduced. The running context interface defines a way to access the required features of the underlying execution platform. For midlets and applets this is the shared display for GUI objects. In architectural means this is realized so that the running context specific components are in their own package, and applications run on top of this package.

In comparison to J2SE, the J2ME platform contains several limitations. J2ME, and in more precisely the Mobile Information Device Profile (MIDP) 2.0 combined with the Connected Limited Device Configuration (CLDC) 1.0, supports only a subset of J2SE's libraries and APIs. Because of that, some components, such as socket connections had to be redesigned and rewritten. The most fundamental limitations are the lack of floating points and XML classes. Third party solutions were used to overcome these problems.

Since the Jaspis architecture relies heavily on XML technology, the biggest task during the porting was the building of XML support. A third party open source XML DOM parser, kXML [<http://kxml.sourceforge.net/>], was taken into use. We were able to create a completely symmetrical implementation of the Jaspis XML classes with the kXML package, and in this way there was no need to change the way that the existing Jaspis components use XML.

For floating point support an open source floating point math library *dfp* (<http://dfp.sourceforge.net/>) was taken into use. The usage of the class was somewhat more complicated than using normal float or double types, but it offered all required features and accuracy. J2ME is also missing file access, which forced us to abandon the Jaspis logging framework. This was a set back, as the logging is very handy for debugging terminal side defects. A workaround would be to send the logging data to a server via network.

Devices supporting the J2ME platform are usually constrained with limited performance and memory. With the current Series 60 mobile phones (e.g., Nokia 6600 and 6630), the performance of the implementation was a good enough for executing the Jaspis with limited number of managers and classes. The first implemented J2ME application, a limited VoiceXML browser was usable for practical purposes, and has been used to implement a multimodal version of an existing telephone-based timetable system. This will be presented in the following section in more detail.

### 4. An example application: a multimodal distributed bus timetable system

We have developed several bus timetable systems with varying functionality and approaches, one of them being the Stopman system [14]. The functionality of the Stopman sys-

tem covers queries about timetables of bus stops. The system is targeted for regular bus travelers, and it provides timetables for each of the approximately 1200 bus stops in Tampere area. This telephone-based speech-only system has been publicly available since August 2003.

The aim of Stopman is to satisfy most of the callers with couple of dialogue turns. At the beginning of the call, the system requests the user to give a bus stop name. After listing the busses going through the requested stop, the rest of the system functions are available. Functionality includes navigation in the timetables, selection of specific bus lines, and specifying a certain time. For example, the user may request timetables for the bus line 23 passing the main square on Tuesday after 08.27. Recently, we have extended the system to support multimodal interaction with smartphones.

#### 4.1. Application architecture

The Stopman system is constructed using the Jaspis architecture, as illustrated in Figure 1. The system contains standard Jaspis managers (Communication, Dialogue, Presentation and Input managers), and an additional Database Manager for communication with the timetable database. In the original telephone system all components are running on the same server machine, and the user access the system by calling the system with his/her fixed-line or mobile phone.

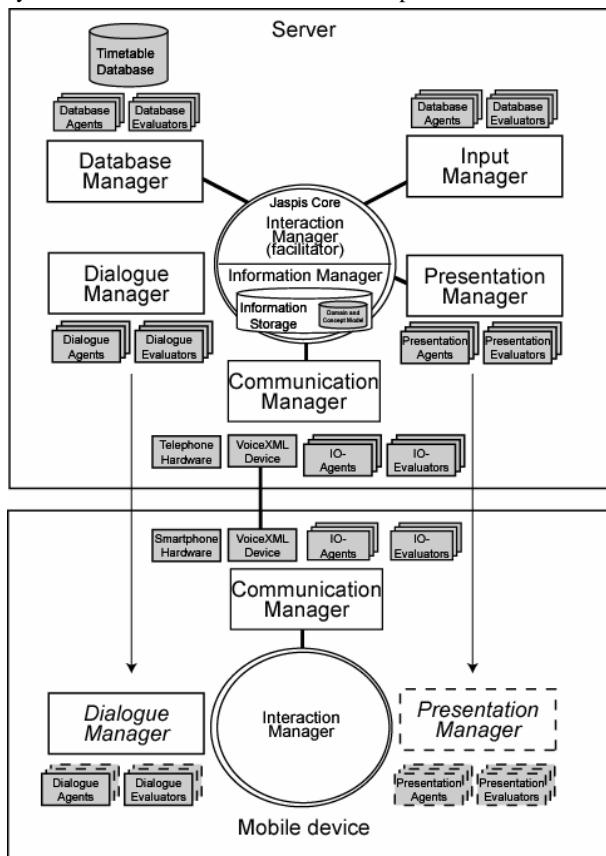


Figure 1: Distributed bus timetable application architecture.

In the smartphone version the components are distributed between the server and the mobile device. In the current version the mobile device contains a VoiceXML handler capable of executing dialogue descriptions that the server system gen-

erates. In the future versions some of the components running on the server side will be transferred to the mobile side, as illustrated in Figure 1 with dotted lines. Next we present in more detail how the distribution works in practice.

#### 4.2. Distribution of dialogue tasks

We have previously presented a generic model for distribution of dialogue management tasks to available terminal devices [10]. The role of the server in this model is to handle the high-level dialogue management by generating extended VoiceXML descriptions that provide all the information needed to execute small multimodal dialogue tasks (e.g., menus) in the mobile devices. The tasks are realized with available resources, such as speech recognizers and synthesizers. This way the resources available in each environment can be used optimally as devices that know their own resources make the decisions how the descriptions are actualized.

The generation of VoiceXML descriptions is realized using a multi-level distributed structure, where specialized agents contribute in generating the complete dialogue description. Different agents are selected to enrich the final task description to match the situation at hand. The evaluators use the information from the shared Information Storage to make the decision which agents are most suitable to be selected for each task. For example, if the mobile device is not capable of displaying graphics, no agents will be selected to generate graphic elements to the dialogue description. Furthermore, the decision on which parts of the description should be generated can be made regarding to the user context and preferences, among other factors.

Distribution of tasks takes place in the communication management subsystem, as illustrated in Figure 2. The Communication Manager contains virtual devices that represent abstractions of resources for input and output tasks. There are different devices for different software and hardware resources, such as speech recognizers and telephony hardware. In the Stopman application two VoiceXML handler devices carry out the dialogue descriptions that the application generates. Based on the available resources, the handlers carry out the dialogue tasks, such as speech recognition tasks. For example, if the mobile device contains speech recognizer and speech synthesizer devices, the entire dialogue task can be completed in the mobile device. Otherwise, shared technology resources are utilized.

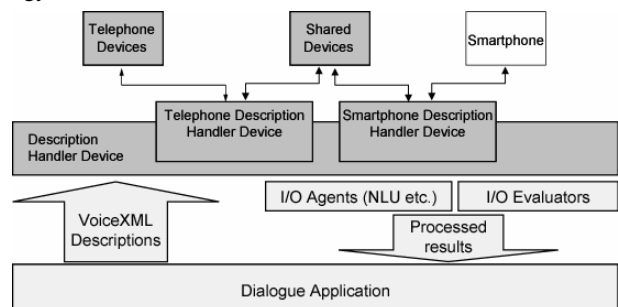


Figure 2: Distribution process.

Our distribution approach has similarities with other distributed multimodal systems. For example, in [5] different output realizations for different devices are generated utilizing a markup that describes the semantics of the outputs.

VoiceXML-based approaches have been used successfully in various dialogue systems in non-trivial ways (e.g., [3] and [8]). An interesting approach is the compilation of VoiceXML descriptions to be used with embedded devices in ubiquitous computing applications [2].

### 4.3. Speech-only and multimodal interfaces

The resulting interfaces are illustrated in Figure 3. With the telephony interface speech and touch tones can be used to interact with the system. The dialogue advances in system-initiated fashion until the bus line listing is spoken to the user, after which there are a few alternative functions available for the users. Forms are interactive so that users can use speech or navigate between options by the telephone keypad. The active item is selected and spoken to the user. Selection of the items is done using touch tones.

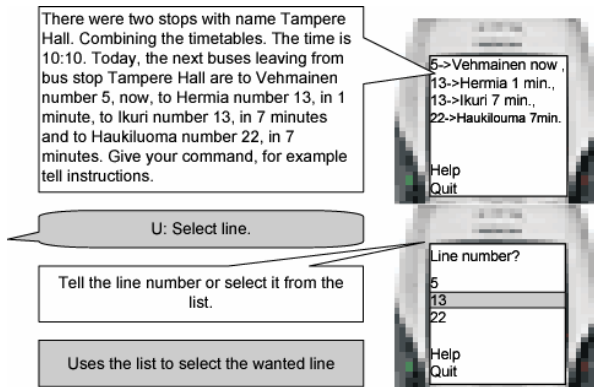


Figure 3: Mobile device interface.

The smartphone interface differs from the speech-only telephony interface so that the display, joystick and keypad are used as a supporting medium to speech in both inputs and outputs. In addition to reading the prompts the supporting information is presented on the screen, and menus are presented graphically. Items can be selected by using the joystick or the keypad like in the telephony version. When an option is selected it is highlighted and its value is spoken.

As shown in Figure 3, the spoken prompts and the displayed contents are not literally the same. It is more useful to show only the essential information on the screen whereas more words are needed in order to make the prompt intelligible and pleasant. However, the conveyed information is the same. The use of display could be made even more efficient by using richer forms of graphics, such as embodied conversational agents. We have developed multimodal integrated tutors [4] that can be used efficiently in mobile interaction.

## 5. Conclusions and future work

We have presented an architecture that enables the building of mobile multimodal dialogue applications. An example application that uses distribution of dialogue management tasks to mobile devices was presented. We have shown that the architecture is feasible for building highly distributed applications with its compact software agents and system-level adaptation mechanism.

Pervasive computing environments raise new challenges for spoken and multimodal dialogue systems. On these environments the dialogue can occur in different places and can be temporally discontinuous because the user is moving in the

environment. This means that the devices used for communication might also change, even within the dialogue. The distribution of dialogue management tasks gives the devices a more autonomous role. For example, decisions such as the selection of modalities can be left for the devices. This allows the dialogue management to concentrate on higher level tasks, such as continuing a temporally discontinued dialogue that might require restoration of its state using some contextual information from the interrupted dialogue. These are examples of the challenges that need to be addressed in the further development of spoken and multimodal dialogue system architectures.

## 6. References

- [1] Bohus, D. & Rudnicky, A. "RavenClaw: Dialog Management Using Hierarchical Task Decomposition and an Expectation Agenda", *Proc. Eurospeech 2003*: 597-600.
- [2] Buhler, D. & Hamerich, S. "Towards VoiceXML Compilation for Portable Embedded Applications in Ubiquitous Environments", *Proc. Interspeech 2005*: 3397-3400.
- [3] Di Fabbrizio, G. & Lewis, C. "Florence: a Dialogue Manager Framework for Spoken Dialogue Systems", *Proc. ICSLP 2004*: 3065-3068.
- [4] Hakulinen, J., Turunen, M. & Salonen, E-P. "Visualization of Spoken Dialogue Systems for Demonstration, Debugging and Tutoring", *Proc. Interspeech 2005*: 853-856.
- [5] Edlund, J., Beskow, J. & Nordstrand, M. "GESOM - A Model for Describing and generating Multi-modal Output", *Proc. ISCA Tutorial and Research Workshop on Multi-Modal Dialogue in Mobile Environments*, 2002.
- [6] Kainulainen, A., Turunen, M., Hakulinen, J., Salonen, E-P., Prusi, P. & Helin, L. "A Speech-based and Auditory Ubiquitous Office Environment", *Proc. SPECOM 2005*: 231-234.
- [7] O'Neill, I. M. & McTear, M. F. "Object-Oriented Modelling of Spoken Language Dialogue Systems", *Natural Language Engineering*, 6, 3, 2000.
- [8] Pakucs, B. "VoiceXML-based dynamic plug and play dialogue management for mobile environments", *Proc. ISCA Tutorial and Research Workshop on Multi-Modal Dialogue in Mobile Environments*, 2002.
- [9] Pakucs, B. "Towards Dynamic Multi-Domain Dialogue Processing", *Proc. Eurospeech 2003*: 741-744.
- [10] Salonen, E-P., Turunen, M., Hakulinen, J., Helin, L., Prusi, P. & Kainulainen, A. "Distributed Dialogue Management for Smart Terminal Devices", *Proc. Interspeech 2005*: 849-852.
- [11] Turunen, M. & Hakulinen, J. "Jaspis<sup>2</sup> - An Architecture For Supporting Distributed Spoken Dialogues", *Proc. Eurospeech 2003*: 1913-1916.
- [12] Turunen, M., Salonen, E-P., Hartikainen, M. & Hakulinen, J. "Robust and Adaptive Architecture for Multilingual Spoken Dialogue Systems", *Proc. ICSLP 2004*: 3081-3084.
- [13] Turunen, M., Hakulinen, J., Rähkä, K-J., Salonen, E-P., Kainulainen, A. & Prusi, P. "An architecture and applications for speech-based accessibility systems", *IBM Systems Journal*, Vol. 44, No 3: 485-504, 2005.
- [14] Turunen, M., Hakulinen, J., Salonen, E-P., Kainulainen, A. & Helin, L. "Spoken and Multimodal Bus Timetable Systems: Design, Development and Evaluation", *Proc. SPECOM 2005*: 389-392.