

Distributed Dialogue Management for Smart Terminal Devices

Esa-Pekka Salonen, Markku Turunen, Jaakko Hakulinen, Leena Helin, Perttu Prusi and Anssi Kainulainen

Speech-based and Pervasive Interaction group, TAUCHI, Department of Computer Sciences
University of Tampere, Tampere, Finland

{Esa-Pekka.Salonen, Markku.Turunen, Jaakko.Hakulinen, Leena.Helin, Perttu.Prusi,
Anssi.Kainulainen}@cs.uta.fi

Abstract

Mobile devices, such as smartphones and personal digital assistants, can be used to implement efficient speech-based and multimodal interfaces. Most of the systems are server-based, and there is a need to distribute the dialogue management tasks between the terminal devices and the server. Since the technologies are not mature and the platforms are constantly changing, approaches that support generic and reusable components are needed. We describe a model where the tasks of the dialogue management are divided into two parts. The server handles the overall coordination of the dialogue by generating dialogue task descriptions. Smart terminal devices realize the low-level decisions by executing individual dialogue tasks. A prototype of multimodal dialogue application using the model is presented. It has two different user interface realizations using standard telephony equipment and smartphones.

1. Introduction

In the past few years the interest towards mobile spoken and multimodal dialogue applications has been increasing. Most of the research has concentrated on the use of speech as a solution to circumvent problems of small displays (e.g. [1]). We try to increase the usability of speech applications with optimal use of available resources. This can mean, for example, use of supporting modalities where applicable.

The use of speech in mobile user interfaces is no longer limited to standard telephony use. New devices, such as smartphones, can be used in more versatile ways as terminal devices. Different devices have different features that can be employed in interaction. Items such as menus and commands that are always available can be easily presented graphically, while this is a challenging task in a speech only environment. In general, resources of terminal devices can be used to support speech interfaces when available. Furthermore, dialogues can be distributed between the devices to create distributed multimodal interfaces.

Because of the diversity of devices, it would be very laborious to convert each application for each and every different platform. The need for generic ways to employ new technologies and devices is clear. Distribution of the responsibilities has already been introduced e.g. in [2], where different output realizations for different devices are generated utilizing a markup that describes the semantics of the outputs.

We introduce a general model to distribute spoken dialogues between servers and smart terminals. Our model covers both inputs and outputs. The server generates high-level descriptions of individual dialogue tasks that are carried out by terminal devices. A general description language is used. It

is important that the model is such that the processing makes optimal use of the terminals and is easily extendable for new terminal devices. This can be achieved so that the device dependent and device independent parts of processing are kept separate. It is important that the device independent components contain no references to specific devices and their capabilities. The device dependent parts, on the other hand, should be such that they contain only those parts of processing that are related to the device and its resources. This way it is easy to introduce new terminal devices and the device independent parts of the system can be used as they are.

In our previous work we have suggested distribution of tasks as a solution for bringing robustness and flexibility to the dialogue management [3, 4]. The model presented here uses distribution to achieve the optimal use of the available resources. We see the methods complementary and the previous methods can be used together with the presented model. When the capabilities of terminal devices are used optimally we can achieve better user interfaces. However, consistency should be taken into account when implementing systems with distributed structure [5]. Even if the dialogues are distributed over a period of time we should ensure continuity [6]. This speaks on behalf of central storing of information and coordination of the devices. This can be achieved by utilizing shared system knowledge, i.e. all the components must be aware of the overall situation.

Markup languages are successfully used as the output of dialogue management systems in various applications [7, 8]. Furthermore, there are various markup languages that could be used in distributed dialogue systems [9, 10, 11]. The success of VoiceXML [10] is one proof of the applicability of markup languages in describing dialogue tasks.

By using a markup language to describe the information needed in interaction tasks we can achieve both the optimal use of resources and consistency. Low-level tasks, e.g., selection of optimal output medium, can be distributed to devices that handle the tasks regarding the description as they see fit. Overall coordination of high level tasks is kept in the part of the system that generates the description. This also enables the system to be distributed physically and makes it possible to change the device used even during a single dialogue if needed.

Next we describe the model that is used to distribute dialogue management tasks to smart terminal devices. After this we present a timetable application that uses the model. We end the paper with discussion and conclusions.

2. Model for distributing dialogue tasks

We present a model that enables the distribution of the dialogue management tasks to different terminal devices. Here the main dialogue management system provides all the information needed to handle small dialogue tasks (e.g., menus) using a description language. Smart terminal devices use the available resources to actualize these tasks. 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.

We use VoiceXML to describe dialogue tasks. VoiceXML is well-known technology, and there are many software components available. VoiceXML, however, is designed for speech-based interaction. We support other modalities by including additional information to the VoiceXML descriptions as variables. In our current implementation, certain elements, such as speech recognition grammars are not included in the description. Instead, the recognition of spoken inputs is always handled with separate components at the server side. This way, we can handle limitations of the terminal devices. However, when technologies evolve the grammars can be added to enable speech recognition on the terminals. An example description follows.

```
<form>
  <field>
    <prompt>Do you want to retrieve the timetable for
      stop Tampere Hall?</prompt>
    <var name="display" value="Tampere Hall?" />
    <option>Yes</option>
    <option>No</option>
    <option>Help</option>
    <option>Quit</option>
  </field>
</form>
```

Example 1: An example description (Contents translated from Finnish).

Example 1 presents a form with four alternative options. Prompt field describes what is said to the user, display variable describes what should be presented visually and option fields describe some input options. Prompt and display elements can convey the information without the other being present. This is important, for example, if there is no display or the sounds are turned off. Since speech recognition is done with separate components in our current implementation, option fields may or may not cover all possible inputs.

The description in the Example 1 can be used with many devices. For example, in standard telephony environment the user can navigate between options and make selections using telephone keypad. If the terminal device has a display (e.g. smartphone or PDA) the value of the display element is shown and the options are presented graphically as well. On the other hand, if the terminal has only a speech channel for communication (e.g. loudspeaker and microphone in a ubiquitous computing environment) the options can be read out to the user and selected with speech recognition. It is noteworthy that the notation does not tie the implementation to any particular device. It is designed to convey the information needed in the dialogue task at hand and additional information can easily be conveyed by using variables. Therefore, this kind of notation could be used in any context where spoken or multimodal dialogue comes into question. We have

applied the model to standard telephony and smartphone contexts; however we see no restrictions in extending it to information kiosk or pervasive computing applications.

Depending on the capabilities of the used devices, the distribution of tasks between the server and the device are handled differently. This is because the resources in different devices are asymmetrical. The devices must distribute the tasks cleverly so that tasks are done where it is possible and what cannot be done is left for other components. For example, in telephony environment resources of the server are used to handle all tasks. In smartphone environment resources of the device are used. For example, speech inputs are recorded in the device and transferred to the server for recognition and further processing. Furthermore, an information kiosk application might have all mentioned resources in the device and the server would receive processed results from it. This means that the level of distribution depends on the used devices and their capabilities. This is different from standard VoiceXML architectures where the distribution is more or less static in means of responsibilities.

Next we will describe a bus timetable application that uses the presented model in distributing the dialogue management tasks to various terminal devices. We will also describe the underlying system architecture and explain how it is utilized.

3. Implementation

We have used the model to implement standard telephony and smartphone interfaces of a spoken dialogue bus timetable application. The smartphone interface uses speech and graphical elements, while the standard telephone interface uses speech and telephone keys. Both interfaces use the same spoken dialogue system for overall coordination and generation of dialogue descriptions.

3.1. System functionality

We have extended an existing timetable application to use the model described in the previous chapter. Stopman is a spoken dialogue application that answers to queries concerning timetables of bus stops. Elementary functionality is that the busses departing from a selected stop are listed. In addition, users can limit the query to some specific time or bus line. There are commands that are always available such as “Quit” and “Help”. The application features a few actions that can be handled by using a menu, such as the selection of a certain bus line. The system knows all the stops in Tampere region, which means about 1200 different bus stops and their names. Currently the system supports Finnish. Example interactions are found in the next chapters where the standard telephony and smartphone interfaces of the system are presented.

3.2. Underlying system architecture

Jaspis architecture [12] offers building blocks for distributed systems. It is based on a distributed set of managers containing evaluators and agents that share all the information via Information Storage. The shared Information Storage offers support to ensure consistency and continuity of distributed systems on architectural level. Managers co-ordinate sets of evaluators to select appropriate agents that actually handle the tasks. Agents in Jaspis-based applications are compact software components. Managers represent the high level organi-

zation of tasks that are handled within spoken dialogue applications.

The agents – managers – evaluators paradigm makes it possible to introduce new components for different tasks easily, without the need to alter the existing ones [13]. Inputs and outputs are handled within the Communication Management subsystem that is implemented as a modular structure where the devices are easily changeable without a need to alter the rest of the system. Distribution of tasks takes place in the Communication Management subsystem.

Communication Management subsystem in Jaspis contains abstract devices that are used to carry out tasks. Devices represent abstractions of resources that can handle tasks. Devices can also be combined, i.e. they can use other devices to complete the task. For example, a telephone synthesizer is a device that can handle a request to synthesize text and play the result via telephony card. Separate components, called engines, implement the interfaces to the actual resources such as telephony cards, speech recognizers, synthesizers, communication with smartphones etc. I/O Agents handle results the devices return, for example NLU components are implemented as I/O Agents. I/O Evaluators decide when the Communication Management is ready, i.e. when to continue. For example, based on results from one device an I/O Evaluator could reason that it is necessary to run more devices.

3.3. Distribution of tasks

There are different devices for handling the descriptions with different terminal devices. Figure 1 depicts the components used to distribute dialogue tasks in Stopman. Dialogue management system (1) produces descriptions of dialogue tasks (2). Each description is, e.g., a menu or a prompt that should be presented to the user (described in chapter 2). This description is handled with devices (3, 4) that use available resources to complete the task. There are different devices for telephony (3) and smartphone (4) use. Both devices are combining devices, i.e. they use other devices to complete the tasks. Everything else, including all NLU, dialogue, and NLG components are the same. In addition, we implemented the user interface components to a smartphone (6). The basic functionality of both combining devices is that they get VoiceXML description as the input and return speech recognition result or menu selection as a result.

In the case of standard telephony environment (5) all resources are on the server. The logic of handling the dialogue descriptions is implemented in a specific device (3). This device uses other devices that implement the functionality to core technologies (synthesizers, recognizers and telephony cards) to handle the tasks.

In the smartphone (6) interface the logic of handling the dialogue descriptions is divided between the terminal device and the server. Thus, the combining device (4) is different from the one in the telephony version. The description is handled with components placed in the smartphone. As a result the smartphone returns audio input or a menu selection. After this, speech recognition is done if needed in the

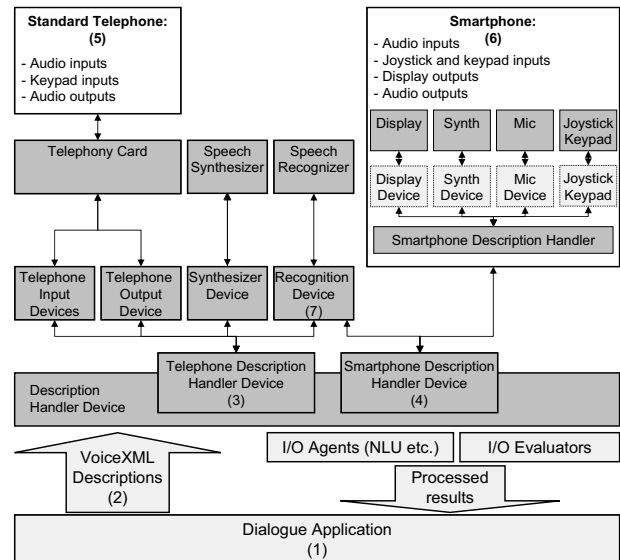
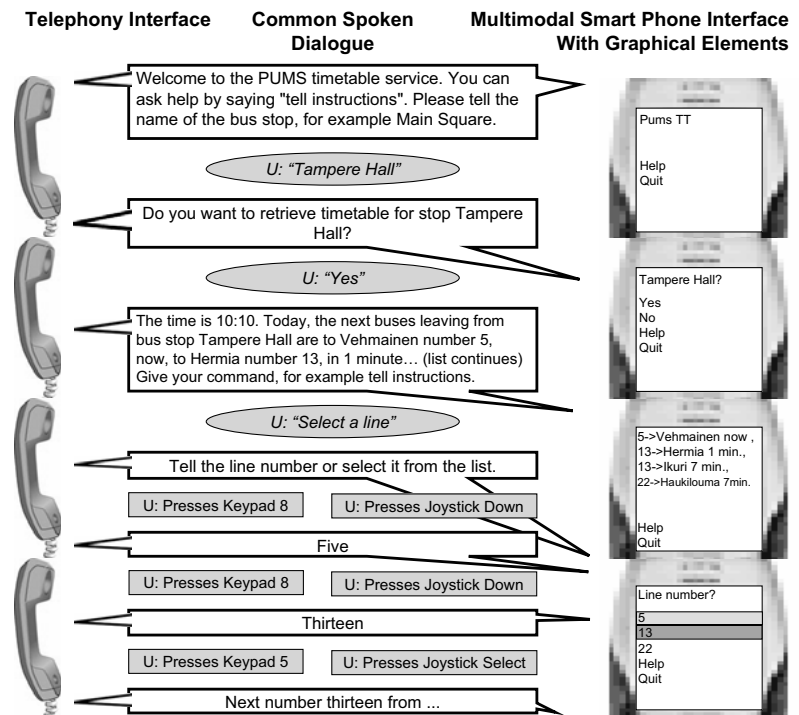


Figure 1: Components used for task distribution.

server using the same device as in the telephony version (7). This way the IOAgents that implement NLU functionality can be the same in both versions.

Smartphone interface is implemented in Java using MIDP 2.0 and has been tested on Nokia 6600. During the implementation phase we ported the core functionality of Jaspis to smartphone. This means that all processing done in the smartphone follows the same architecture as in the server side (as the insides of (6) in Figure 1 suggests). In practice this means that we could share code between the telephony and smartphone implementations.

Example 2 illustrates the two user interfaces of the timetable application that are utilizing the distributed model. The



Example 2: Two user interfaces of Stopman that are based on one description (Dialogue and contents translated from Finnish).

interaction is similar with both interfaces except that the smartphone interface uses the display to show possible menu options and recaps of system prompts. Even though VoiceXML is originally built for spoken interaction only; it is possible to use it for this kind of simple multimodal tasks.

3.4. Telephony interface

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. On the left hand side of Example 2 a short dialogue with the system including the use of touch tones to select options is presented.

3.5. Smartphone interface

The smartphone interface differs from the 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. Menus are presented graphically as well. 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. These features and differences to the telephony version are shown in right hand side of Example 2.

As shown in Example 2 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 understandable and speech synthesis fluent. However, the conveyed information is the same. The use of display could be made even more efficient by using graphics, animations etc. This discussion is, however, out of scope for this paper.

4. Discussion

With the current implementation of the task distributing model we are able to distribute tasks intelligently to different terminal devices. The dialogues are distributed so that the devices make the decision which parts it can handle and how they are carried out. If the device cannot complete the task remaining operations are handled on the server. This was demonstrated in the example interfaces to standard telephony and smartphone.

Ubiquitous computing environments raise new challenges to spoken dialogue management. 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 during one dialogue. The distribution of dialogue management tasks gives the devices a more autonomous role and releases the high level components from micro management. For example, low level decisions such as the selection of modalities can be left for the devices. The dialogue management concentrates on the task on a higher level, such as continuing a temporally discontinued dialogue that might require restoration of its state using some contextual information from the interrupted dialogue.

5. Conclusions

We have presented a model to distribute dialogue management tasks to smart devices and shown that the model is effective and useful in implementing spoken and multimodal dialogue applications to different environments. The model results in a manageable structure where consistency and continuity of dialogue is easily reached by using a suitable architecture, even if the devices used for interaction need to be changed. To proof the applicability of the model, a working prototype with two different interface realizations and task distribution levels was presented.

6. References

- [1] Hemsén, H. "Designing a Multimodal Dialogue System for Mobile Phones", *Proc. of Nordic Symposium on Multimodal Communications*, 2003.
- [2] Edlund, J., Beskow, J. & Nordstrand, M. "GESOM - A Model for Describing and generating Multi-modal Output", *Proc. of ISCA Tutorial and Research Workshop on Multi-Modal Dialogue in Mobile Environments*, 2002.
- [3] Turunen, M., Salonen, E-P., Hartikainen, M., & Hakulinen, J. "Robust and Adaptive Architecture for Multilingual Spoken Dialogue Systems", *Proc. of ICSLP 2004*, 2004.
- [4] Salonen, E-P., Hartikainen, M., Turunen, M., Hakulinen, J. & Funk, J. A. "Flexible Dialogue Management Using Distributed and Dynamic Dialogue Control", *Proc. of ICSLP 2004*, 2004.
- [5] Larsen, A. & Holmes, P. D. "An Architecture for Unified Dialogue in Distributed Object Systems", *Proc. of TOOLS 26 - Technology of Object-Oriented Languages*, 1998.
- [6] Savidis, A., Maou, N., Pachoulakis, I. & Stephanidis, C. "Continuity of Interaction in nomadic interfaces through migration and dynamic utilization of I/O resources", in *Universal Access in Information Society*, Springer Verlag, ISSN: 1615-5289, 2002.
- [7] Pakucs, B. "VoiceXML-based dynamic plug and play dialogue management for mobile environments", *Proc. of ISCA Tutorial and Research Workshop on Multi-Modal Dialogue in Mobile Environments*, 2002.
- [8] Di Fabbrizio, G. & Lewis, C. "Florence: a Dialogue Manager Framework for Spoken Dialogue Systems", *Proc. of ICSLP 2004*, 2004.
- [9] Katsurada, K., Nakamura, Y., Yamada, H. & Nitta, T. "XISL: A Language for Describing Multimodal Interaction Scenarios", *Proc. of ISCA Tutorial and Research Workshop on Multi-Modal Dialogue in Mobile Environments*, 2002.
- [10] VoiceXML specification: <http://www.w3c.org/TR/2004/REC-voicexml20-20040316/>
- [11] UIML specification: <http://www.uiml.org/specs/uiml3/DraftSpec.htm>
- [12] 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*. (to appear)
- [13] Hakulinen, J., Turunen, M. & Salonen, E-P. "Agents for Integrated Tutoring in Spoken Dialogue Systems", *Proc. of Eurospeech 2003*, 2003.