Decalog 2007: Proceedings of the 11th Workshop on the Semantics and Pragmatics of Dialogue, pages 159–160.
Trento, Italy, 30 May – 1 June 2007. Edited by Ron Artstein and Laure Vieu.

Dialog OS: an extensible platform for teaching spoken dialogue systems

Daniel Bobbert
CLT Sprachtechnologie GmbH
Science Park Saar
66123 Saarbrücken, Germany
bobbert@clt-st.de

Magdalena Wolska
Computational Linguistics
Universität des Saarlandes
66041 Saarbrücken, Germany
magda@coli.uni-sb.de

1 Introduction

With the area of spoken dialogue systems rapidly developing, educational resources for teaching basic concepts of dialogue systems design in Language Technology and Computational Linguistics courses are becoming of growing importance. Dialog OS[1] is an extensible platform for developing (spoken) dialogue systems that is intended, among others, as an educational tool.[2] It allows students to quickly grasp the main ideas of finite-state-based modelling and to develop relatively complex applications with flexible dialogue strategies. Thanks to Dialog OS' intuitive interface and extensibility, system implementation tasks can be distributed among non-technically- and technically-oriented students making the tool suitable for a variety of courses with participants of different backgrounds and interests. Below, we give a brief overview of the framework and outline some of the student projects in which it was used as a basis for dialogue management and modelling.

2 Dialog OS: a brief overview

Dialog OS is an extensible platform for managing and modelling (spoken) dialogue systems. It comprises an intuitive Graphical User Interface (GUI), default dialogue components, and a communications API to build new components. Dialog OS is written in Java and operates in a client-server mode. The central component can handle connections with an arbitrary number of client components (or "Devices", in Dialog OS terminology) via TCP/IP sockets. Technical requirements for Dialog OS are: 1 GHz Pentium, 512 MB RAM, Windows 2000/XP, Java Runtime 1.5 or newer.

[1]Dialog OS is a registered trademark of CLT Sprachtechnologie GmbH. Other product and company names listed are trademarks or trade names of their respective owners.

[2]Dialog OS is developed and distributed by CLT Sprachtechnologie GmbH: http://www.clt-st.de/dialogos

Default components Dialog OS comes with built-in modules for professional quality speech input and output using technology from Nuance and AT&T. As part of the platform, Dialog OS provides a number of default input/output device clients that can be directly connected without extra programming. Among those are: a simple text console for text-based input and output, a sound player, and a default client for a connection to an SQL database. CLT can also provide built-in connections to a number of other research and commercial Automatic Speech Recognition (ASR) and Text-To-Speech (TTS) systems.

Extensibility Dialog OS can be extended to work with an arbitrary number of clients through a Java-based API. The low-level communication between Dialog OS and the clients is handled by a dedicated internal protocol and remains invisible to the user. Programming a new client involves a Java implementation of a high-level functional protocol for the given client, without having to deal with the details of network connection with the dialogue engine itself.

FSA-based dialogue modelling The central part of the dialogue system is the dialogue model. Dialog OS offers an intuitive way of modelling dialogues using Finite State Automata (McTear, 2002). Building a dialogue model consists of adding and linking dialogue graph nodes represented as icons on a GUI workspace. Those include input/output nodes and internal nodes, for example, to execute scripts, set and test variables, enter a sub-graph (i.e. execute a sub-automaton).[3] The dialogue model is stored in an XML format.

Dialog OS builds on the functionality of its predecessor, DiaMant (Fliedner and Bobbert, 2003). Below, we list some of the features taken over, extended or enhanced in Dialog OS:

[3]The expressive power of the dialogue models is effectively that of push-down automata.

User input The input nodes for text-based or spoken interaction allow to specify a list of expected input values; outgoing edges are created automatically. User input may be matched directly against the list, or against a regular expression. For spoken input via default ASR components, both the recognised string and the recognition confidences can be accessed.

Built-in data types Global variables can be of simple types (e.g. String, Integer, etc.) as well as more complex data structures of key-value pairs.

Scripting language Dialog OS includes an interpreter of a JavaScript-like scripting language for simple data manipulation functions, e.g., to match input against a regular expression. These can be integrated through a *Script* node.

Sub-automata The *Procedure* node allows for flexible and modular dialogue modelling. Recurring parts of the dialogue can be saved as individual parameterisable sub-automata, direct counterparts of sub-routines in programming languages.

Wizard-of-Oz (WOz) mode Dialog OS can be run in WOz mode (Fraser and Gilbert, 1991) in which one or more of the "Devices" are simulated and dialogue execution details are saved in logfiles; this allows to set up small-scale WOz experiments.

3 Dialog OS in the classroom

We have been using Dialog OS and its predecessor at Saarbrücken in a number of courses involving spoken dialogue systems. Notable features that make it suitable for educational purposes include:

Intuitive interface: Learning to use Dialog OS takes very little time. Thanks to the GUI, even non-computational students can easily configure a functional system with little (or even no) knowledge of programming. The low learning overhead allows to concentrate on modelling interesting dialogue phenomena rather than technical details.

High-level language for building new components: A Java-based API makes the development process efficient and allows for the final system to be built on a single programming platform and kept highly modular.[4]

Below we briefly outline larger spoken dialogue systems developed as part of software projects using the Dialog OS framework.

Talking Robots with LEGO MindStorms® Within two runs of the course, students built various speech-enabled mobile robots using LEGO and Dialog OS as dialogue framework (Koller and Kruijff, 2004). Integration involved writing a client to control the MindStorms RCX (Dialog OS provides built-in support for MindStorms NXT). Luigi Legonelli, the Shell Game robot, and a modified version of Mico, the bar-keeper,[5] have been presented at CeBIT '03 and '06, respectively.

Campus information system A group of three students built a spoken information system for Saarland University campus. The system can answer questions on employee's offices, telephone numbers, office locations, etc. The highlights of the system are modularity[6] and an adaptive clarification model needed to handle many foreign names and foreign user accents.

Talking elevator In two editions of this course, students built speech interfaces to the elevators in the institute's buildings. In the first course, a simple mono-lingual system was developed. In an ongoing project, students are building a trilingual system with speaker identification, using their own version of a Nuance client and an elevator client that communicates with the elevator hardware via a serial protocol.

References

G. Fliedner and D. Bobbert. 2003. DiaMant: A Tool for Rapidly Developing Spoken Dialogue Systems. In *Proc. of DiaBruck*, pages 177–178, Wallerfangen, Germany.

N. M. Fraser and G. N. Gilbert. 1991. Simulating speech systems. *Computer Speech and Language*, 5:81–99.

A. Koller and G-J. M. Kruijff. 2004. Talking robots with lego mindstorms. In *Proc. of COLING-04*, pages 336–342.

O. Lemon and X. Liu. 2006. DUDE: A dialogue and understanding development environment, mapping business process models to information state update dialogue systems. In *Proc. of EACL-06*, pages 99–102, Trento, Italy.

M. F. McTear. 1999. Using the CSLU toolkit for practicals in spoken dialogue technology. In *Proc. of ESCA/SOCRATES Workshop on Method and Tool Innovations for Speech Science Education*, pages 113–116, UK.

M. F. McTear. 2002. Spoken dialogue technology: enabling the conversational user interface. *ACM Computing Surveys (CSUR)*, 34(1):90–169.

[4] A GUI is also part of CSLU (McTear, 1999) and DUDE (Lemon and Liu, 2006) dialogue toolkits. However, DUDE has not yet been tested with novice users, while extending CSLU Toolkit involves programming in C, rather than in a higher-level language such as Java.

[5] Mico's mechanics were substantially re-designed by CLT, however, the dialogue model was, for the most part, taken over from the student project.

[6] Currently, the system supports only German, but the modular design was motivated by anticipated extensions for English and French.