# DialogOS: Simple and extensible dialog modeling

*Alexander Koller*[1], *Timo Baumann*[2], *Arne Köhn*[3]

[1]Saarland University [2]Carnegie Mellon University [3]Universität Hamburg

koller@coli.uni-saarland.de, tbaumann@cs.cmu.edu, koehn@informatik.uni-hamburg.de

## Abstract

We present the open-source extensible dialog manager DialogOS. DialogOS features simple finite-state based dialog management (which can be expanded to more complex DM strategies via a full-fledged scripting language) in combination with integrated speech recognition and synthesis in multiple languages. DialogOS runs on all major platforms, provides a simple-to-use graphical interface and can easily be extended via well-defined plugin and client interfaces, or can be integrated server-side into larger existing software infrastructures. We hope that DialogOS will help foster research and teaching given that it lowers the bar of entry into building and testing spoken dialog systems and provides paths to extend one's system as development progresses.

## 1. Introduction

With the popularity of spoken dialog systems in everyday digital assistants, there is an increasing interest both in academia and industry in tools that make it simple and quick to develop such dialog systems. However, while there is no shortage of dialog platforms of varying levels of complexity and robustness, most of these systems require writing dialog specifications in specialized languages such as VoiceXML. This raises the barrier of entry for getting started with such systems, and restricts their use to professionals who can invest the time into learning how to use them.

In this paper, we describe DialogOS[1], a spoken dialog system which is simple to get started with, but also offers advanced features and can be easily extended and integrated with external tools. DialogOS is a finite-state dialog system in which the dialog graph is "drawn" in a GUI (see Figure 1) by placing nodes (e.g. for speech recognition and synthesis) on a canvas. DialogOS was originally a commercial system [1], developed by CLT Sprachtechnologie GmbH, and was popular as a teaching tool both in academia [2] and in high schools[2]. Here we describe a modernized version of DialogOS, which was just released as an open-source project, after the rights to DialogOS were acquired by Saarland University. DialogOS is written in Java. In contrast to the commercial version, which was limited to Windows, the new open-source version runs on Windows, MacOS, and Linux.

Below, we sketch the core functions of DialogOS and then discuss advanced use cases and the integration with external software.

## 2. DialogOS

At its core, DialogOS is a finite-state dialog system. A dialog consists of *nodes*, each of which performs one specific operation, such as speaking one utterance with a TTS system or processing one user utterance with an ASR system. Nodes can be connected with *edges*; for instance, an ASR node may have multiple outgoing edges for the different recognition outcomes. These nodes and edges can be placed on the canvas using drag and drop. While the dialog is executed, DialogOS walks through nodes one at a time, following the edges from one node to another. The dialog terminates when an End node is reached. It can be further structured into subgraphs.

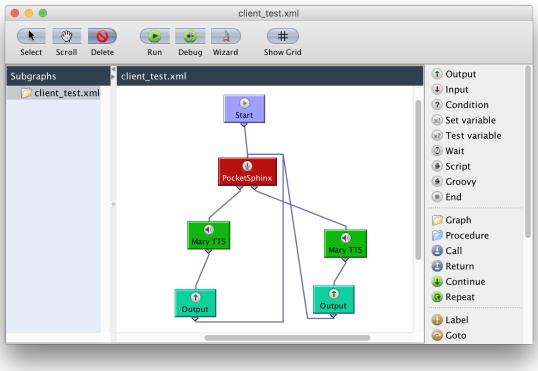


Figure 1: *The main DialogOS window.*

DialogOS goes beyond a purely finite-state model in that it can store arbitrary values in *variables*, which are global to the dialog. Variables can be set in ASR and other input nodes, based on matching regular expressions or speech recognizer grammars. They can then be processed in Script nodes, and Conditional nodes can branch to different parts of the dialog depending on the values of variables and they can be used as parts of expressions in TTS nodes. DialogOS variables support all datatypes available in the Java standard library, including lists and dictionaries, and Script nodes can be programmed either with a DialogOS-specific scripting language or with the general-purpose language Groovy,[3] a dynamically typed language with syntax similar to Java.

Because DialogOS is meant to make it easy to get started, it comes bundled with a number of key components, including MaryTTS [3] for speech synthesis and CMU Sphinx-4 [4] for speech recognition. Synthesis and recognition models can simply be installed over the Internet from within the GUI. DialogOS also has builtin support for Lego Mindstorms robots, using nodes for controlling motors, reading sensor values, and starting and stopping programs on a Mindstorms robot. Thus the DialogOS distribution contains all the parts, out of the box, for building "talking Lego robots".

## 3. Extending DialogOS

While DialogOS aims to make it simple to develop simple dialogs, it is also designed to connect to existing software, using

[1]https://www.dialogos.app/

[2]http://www.debacher.de/wiki/DialogOS

[3]http://www.groovy-lang.org

*plugins*, *clients*, or *Groovy*. In this way, DialogOS can be extended to support a wide variety of domain-specific functionality.

*Plugins* are modules which provide new node types to DialogOS. The MaryTTS, Sphinx, and Lego nodes that are bundled with DialogOS are implemented as plugins. In addition, users of DialogOS can develop their own plugins, which are Java libraries that implement the DialogOS plugin API. The node types of these plugins become visible within the DialogOS GUI when these libraries are added to the classpath. This makes it possible to integrate different speech recognizers and TTS systems into DialogOS, or to develop entirely new node types, such as for database queries or custom backend integrations.

If developing a plugin incurs too much overhead, e.g. because an extension is only needed for a single dialog, *Groovy* nodes can be used to extend the functionality from within the dialog; DialogOS provides an editor with syntax highlighting. Because Groovy nodes are able to import and make use of all libraries added to the classpath, any JVM-based library can be used to extend the dialog's functionality.

*Clients* are standalone programs which connect to DialogOS over TCP/IP sockets. DialogOS can send messages to a client using an Output node, and receive messages from the client using an Input node, which – like a speech recognition node – may branch to different output edges depending on the message. Clients written in Java can simply implement the DialogOS client API, which hides the burden of managing the inter-process communication from the developer. In principle, clients could also be written in other languages, such as Python.

## 4. DialogOS in practice

DialogOS has been used extensively by students at the middle school, high school, and academic levels. It is ideally suited for an educational setting because its learning curve is gentle and no programming skills are required to build a nontrivial spoken dialog system; the builtin Lego interface helps generate excitement among students.

Almost as a side effect, dialog development with DialogOS teaches students about fundamental concepts of computer science, including finite-state automata (and their limitations that can be overcome by using variables) and context-free grammars (used for speech recognition). The concept of abstraction is prevalent in DialogOS, e. g. when using sub-dialogs or when deciding on how to implement a system's capabilities: in the dialog graph, as a script, or as an external client.

However, DialogOS is not limited to toy applications in the educational domain: Because of its extensible design, DialogOS is also suitable for research on dialog systems. In such a setting, DialogOS is often used to develop the dialog in the GUI, but execute the dialog in a windowless mode which permits deployment on a server or an embedded system. External clients can perform arbitrarily complex tasks, including running a whole separate dialog system, as in [5], where incremental speech processing was performed for parts of the dialog and DialogOS was used for the conventional, non-research parts of the system. Since its release as open-source, DialogOS has also been used as the integrated dialog manager in a larger (and proprietary) software environment [6]. In that case, DialogOS was used as a frame-based DM and used externally provided speech recognition, synthesis, understanding and generation components; the integration was performed by implementing a plugin that provided its own node types for interaction with the environment. The graphical interface of DialogOS allowed to rapidly prototype and change dialog plans during testing with minimal required expertise.

## 5. User studies with DialogOS

DialogOS supports researchers in conducting empirical user studies in two ways: Dialogs can be logged for further analysis, and parts of the dialog can be implemented by a Wizard-of-Oz.

The logging facility creates an XML-based log of each run of a dialog: It records the initial settings as well as an XML structure for every node visited, including the ID of the node and timestamps for entering and exiting. Each node adds relevant meta-data such as the recognized input for recognition nodes, the generated output for TTS nodes, or changes to variable assignments for script nodes. Sub-dialogs are nested within the XML structure to represent the dialog flow. Additional nodes introduced by plugins can add their own metadata.

For Wizard-of-Oz experiments, the behavior of certain nodes can be controlled by a researcher: when entered, the possible outcomes are shown and the intended one can be selected. This allows researchers to replace dialog components with human-in-the-loop states imitating the component's behavior. While the dialog is running, the wizard is shown the dialog graph (as in Figure 1) with the currently active node highlighted.

## 6. Conclusion

We have presented DialogOS, a recently open-sourced spoken dialog system that is built with simplicity and extensibility in mind. With DialogOS it is easy to build one's own dialog models within minutes and with only a minimum of expertise required, making it an ideal tool for teaching. Yet, DialogOS provides all features necessary to build full-fledged dialog systems for research and application.

In the future, we plan to actively improve and extend DialogOS, e.g. by adding new plugins and features for collaboration, dialog version control, database and professional robotics integration (via ROS). We cordially invite contributions from the community.

**Acknowledgments.** We are indebted to Manfred Pinkal and Diana Steffen for their support in making DialogOS an open-source project. We are also grateful to the students at the Universität Hamburg who helped clean up, modernize, and extend the DialogOS source code: Bri Burr, Vincent Dahmen, Max Friedrich, Dorothee Geiser, Otis Juliusson, Annika Kolaska, Till Kollenda, Nicolas Schroh, Phil Sehlmeyer, and André Simon.

## 7. References

[1] D. Bobbert and M. Wolska, "Dialog OS: an extensible platform for teaching spoken dialogue systems," in *Decalog 2007: Proceedings of the 11th Workshop on the Semantics of Dialogue*, R. Artstein and L. Vieu, Eds., Trento, Italy, 2007, pp. 159–160.

[2] A. Koller and G.-J. Kruijff, "Talking robots with LEGO Mindstorms," in *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*, Geneva, 2004.

[3] M. Schröder and J. Trouvain, "The german text-to-speech synthesis system mary: A tool for research, development and teaching," in *Speech Synthesis Workshop*, Perthshire, 2001.

[4] P. Lamere, P. Kwok, E. Gouvea, B. Raj, R. Singh, W. Walker, M. Warmuth, and P. Wolf, "The cmu sphinx-4 speech recognition system," in *Proceedings of ICASSP*, 2003.

[5] T. Baumann, M. Paetzel, P. Schlesinger, and W. Menzel, "Using affordances to shape the interaction in a hybrid spoken dialogue system," in *Proceedings of ESSV*, P. Wagner, Ed., 2013, pp. 12–19.

[6] V. Tsai, T. Baumann, F. Pecune, and J. Cassell, "Faster responses are better responses: Introducing incrementality into sociable virtual personal assistants," in *Proceedings of IWSDS*, 2018.