DialogDesigner – A Tool for Rapid System Design and Evaluation

Hans Dybkjær
Prolog Development Center A/S
H. J. Holst Vej 3C-5C
2605 Brøndby, Denmark
dybkjaer@pdc.dk

Laila Dybkjær
Natural Interactive Systems Laboratory
University of Southern Denmark
Campusvej 55, 5230 Odense M
laila@nis.sdu.dk

Abstract

As spoken dialogue systems mature, the need for rapid development tools increases. We describe such a tool that is currently being used for commercial design, specification and evaluation, and that is in the process of being developed into a complete case tool.

1 Introduction

Improved recognition and understanding of spoken interaction facilitate the development of higher level tools that may enhance the clarity of spoken dialogue systems (SDSs) and reduce their development time and cost. This paper describes a tool – named DialogDesigner[1] – which supports SDS developers in rapidly designing and evaluating a dialogue model. In the following Section 2 provides an overall description of DialogDesigner. Sections 3, 4, 5 and 6 present different aspects of the tool functionality in terms of how to model the dialogue, get various graphical views, run a Wizard-of-Oz (WOZ) simulation session, and extract different presentations in HTML. Sections 7 and 8 describe related work on design and evaluation tools and development tools, respectively. Section 9 concludes the paper.

2 DialogDesigner

The basis in DialogDesigner is the design window where one can enter and browse a dialogue model, including prompts, conditions, and state transitions. Having entered a dialogue model there are various presentation possibilities.

One option is to view a graphical presentation of the dialogue model. This presentation can be made more or less detailed depending on what the designer wants to see. A second option is to run a WOZ simulation. This can be done with users or as part of presentations to and discussions with customers. The simulation is logged and can be saved for later analysis and commenting. The simulation log can also be used normatively to generate test scripts for use in a systematic functionality test. A third option is to extract HTML versions of the entire dialogue as well as of prompt and phrase lists.

In the following we explain the design window and the three mentioned main options, and illustrate the tool via the early design of a pizza application.

3 Dialogue Structure and Prompts

The design window (Figure 1) has at its top three fields for administrative purposes (name of application, version and note) (1). The rest of the window concerns application design. The designer starts by entering a new group (2). A group consists of one or more dialogue states which conceptually belong together and are described by the group. A group or a state can be moved up or down in the emerging dialogue structure (3) using the arrow buttons (2). New states are entered at (4). Here one can also indicate if there is any priority condition (conditions are numbers, not Booleans) for entering the state, grammars needed for this state, and parameters that can be tested in conditions on states or transitions. No grammars are needed if the state does not take input from the user but continues directly to another state.

A state usually has one or more prompts attached. These are entered by clicking “edit” at (5). This leads to a window (not shown) listing all phrases already entered. New phrases can be added and one can compose a prompt for the state by selecting one or more phrases or named sets of indexed phrases and storing them. The resulting text is then shown at (5) when one returns to the design window.

To get from one state to another, transitions (6) are needed. Some transitions are globally enabled when input from the user is expected. These may include e.g. request for repetition and no input registered. When

[1] See also www.spokendialogue.dk/DialogDesigner.

there are several such global transitions it may pay off to group them together as done in Figure 1 under the group StandardReactions. Here (Commands) contain user-initiated meta-communication commands, such as help and repeat, while (Events) contain system triggers for meta-communication, such as no input and nothing understood. (Standard) contains default domain value reactions such as price information which the user may request at any time during the dialogue. A state may have several possible transitions leading to different new states (targets) where the choice of transition depends on the user's immediate input or on which information has been achieved so far. Transitions may target states or groups of states. In the latter case state conditions will determine which state to enter. Conditions on transitions express what must be fulfilled in order to select them. Transitions may also be accompanied by a prompt e.g. to provide feedback on the user's input or bridging to the output for the next state.

Transition information is entered at (7) where clicking on clone will enable the designer to enter a new transition. Transition prompt texts are entered in the same way as state prompt texts, as explained above.

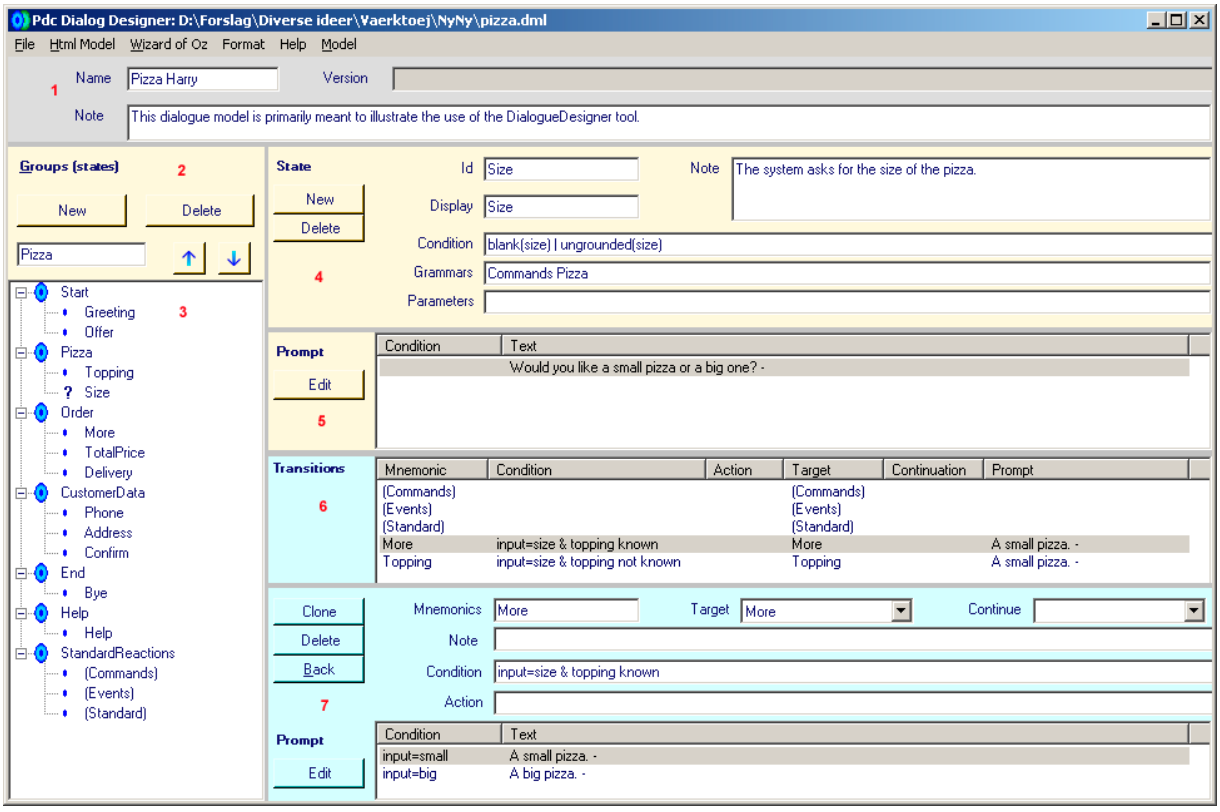

Figure 1. The design window. Red numbers are referenced in the text.

4 Graphical View

Clicking Model in the top menu bar in the design window (Figure 1) opens a new window which allows the designer to see various graphical views of the dialogue (Figure 2). The graph part (7) is empty when the designer opens the window. To the left (1) are the groups and states specified in the design window. To the right (4) the designer can choose what he wants the graph to show. This should be done before he starts drawing the graph. Ticking Domain will enable all domain, i.e. task-related, transitions to be drawn. Ticking Command and System, respectively, will enable meta-transitions to be shown where System covers meta-transitions triggered by system events and Command covers user-initiated meta-transitions. Incoming and Outgoing allow the designer to see incoming and outgoing transitions, respectively, for a group or a state. Local shows transitions going out of and coming into the same state. Via shows transitions to a state that by default continues to some other state. Whenever the designer ticks one of the options Via, Incoming, Outgoing and Local, and selects a group or a state, the Outgoing (5) and Incoming (6) lists will show the transitions that will be drawn, if any.

To draw a group or a state in the graph part of the window (7) one must double-click the group or state at (1). Groups are shown in a double ellipsis to indicate

that they can be further expanded, while states are drawn in a single ellipsis. The ellipsis of a selected group or state is shown in red. To expand a selected group or a state and see its transitions as specified at (4) one must click the expand button at (2). To collapse a group again one must double-click the group at (1). Domain transition labels are green while system transitions are red and command transition are yellow.

The graphical view is well-suited to get an overview of the dialogue structure and see connections at a more or less fine-grained level.

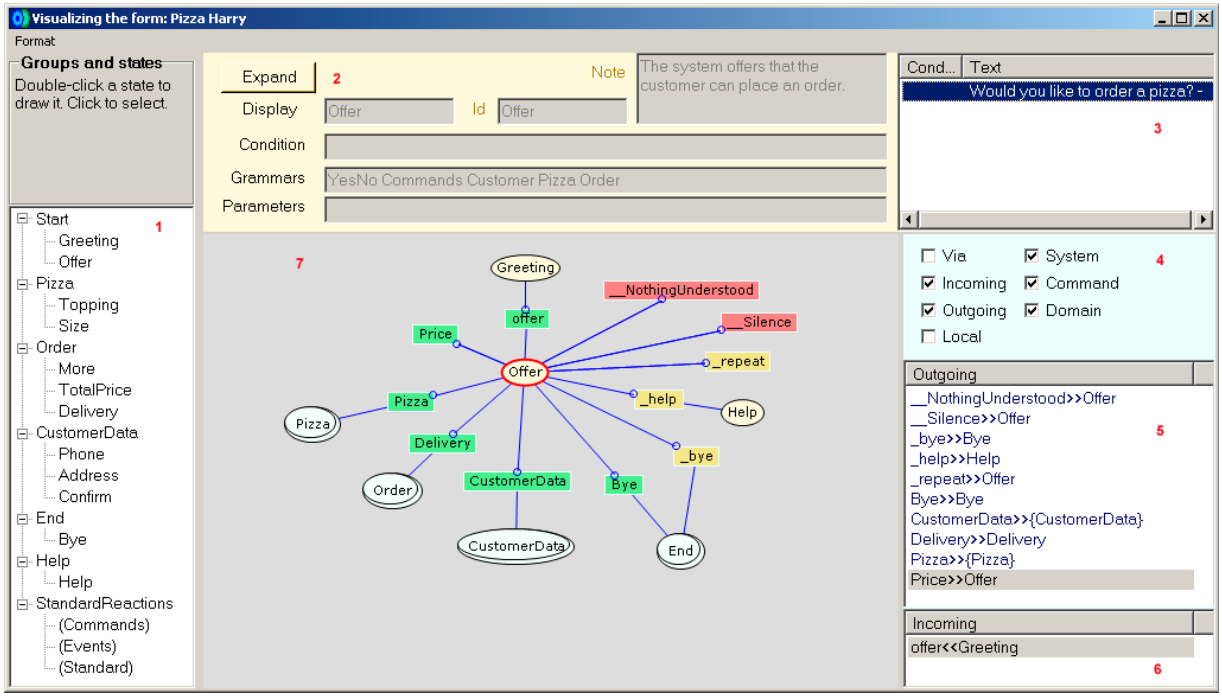

Figure 2. The graphical view. Red numbers are referenced in the text.

5 Wizard of Oz

In the design window (Figure 1) one may select “Wizard of Oz” -> “Woz” from the menu bar. Doing this opens a new window as shown in Figure 3. This window enables the designer to simulate a user-system interaction using the designed dialogue model.

The designer starts a dialogue by clicking Start (1, where the button now is labelled Stop because a dialogue is ongoing). This will cause the system utterance for the initial state to be displayed in the Prompt field (2). At the same time all possible transitions from this state are shown in the Next field (4). Which one to choose depends on the user’s input which is entered at (3). Entering the user’s input does not automatically cause a selection of a transition. This must be done manually. But writing down the user’s input means that the log eventually will contain a full dialogue with both system and user utterances. Such dialogues may later be used for testing the application and for further analysis. At (3) it is also possible to write notes to the current dialogue state, user input or transition.

The designer selects a transition by double-clicking on it. In doing this the previous system and user turn will be displayed in the log field at (5). At the same time the next system prompt is shown in the Prompt field and the new transition possibilities are shown in the Next field. The designer may copy and save a log for later inspection in the analysis window.

The analysis window is opened from the design windows menu bar “Wizard of Oz” -> “Edit logs”. This window looks quite similar to the Woz window but supports the designer in inspecting, editing and commenting a previously saved log from a simulated interaction.

6 HTML Presentations

The HTML menu in the design window (Figure 1) gives access to a number of options for HTML presentations.

Phrase and prompt lists and a presentation of the dialogue model may be extracted in HTML. These are helpful for communicating with customers and phrase speakers. The HTML dialogue model can be used for navigating the dialogue via links, cf. Figure 4, without having access to the DialogDesigner.

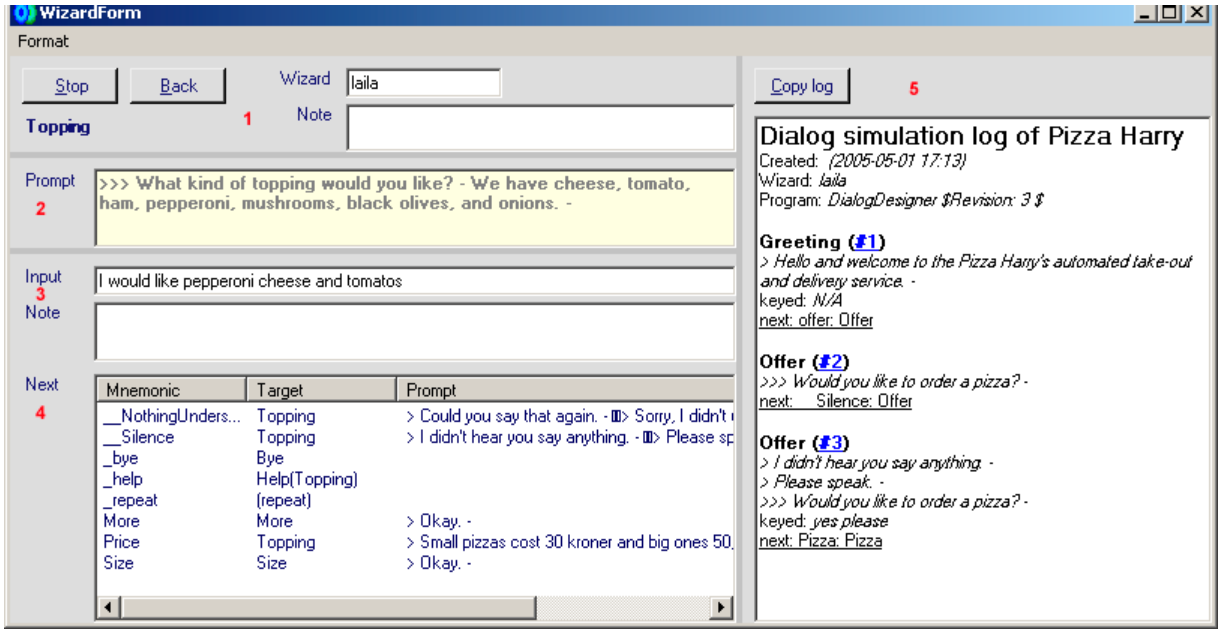

Figure 3. The simulation window. The log is stored in XML and may later be analysed in a similar window, or the RTF-format in the right-most pane may be copied to another document. Red numbers are referenced in the text.

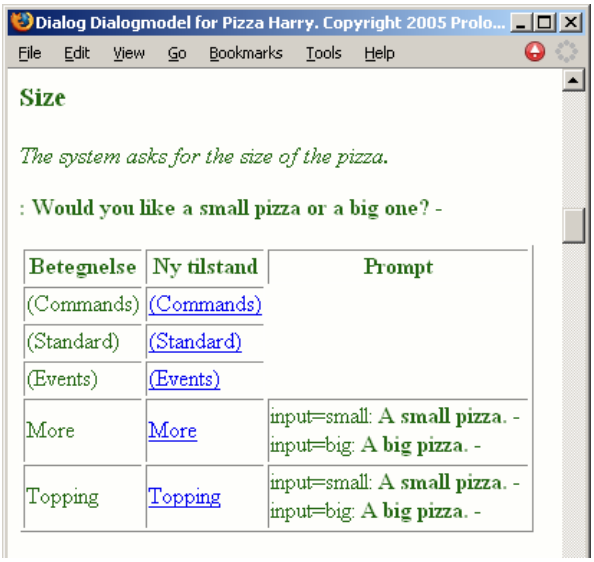

Figure 4. Excerpt of HTML presentation.

7 Related Design and Evaluation Tools

Other tools than DialogDesigner exist which are meant to support the design and evaluation of SDSs and which support WOZ. Two such tools are Suede [Klemmer et al. 2000], developed at the University of Washington, and the WOZ tool developed by Richard Breuer [WOZ tool] as a by-product of his work at Scansoft.

Suede offers an interface for each of the three main activities of design, test, and analysis. The design interface allows the designer to create example dialogue scripts and a design graph representing the general design solution. For each prompt the audio output may be played if it has been recorded. The test mode enables WOZ simulation. The designer selects a prompt from a list of available prompts given the present state. The selected prompt is played to the user. Based on the user's answer the designer selects again one among the now available prompts, etc. Simulation of recognition errors is supported. The analysis interface is similar to the design interface except for the top of the window which contains user audio input from the last session. Moreover the design graph is annotated with test data which can be played.

The WOZ tool developed by Richard *Breuer* offers interfaces for the three main activities of design, WOZ simulation and export. In the design mode the designer can specify the dialogue design in terms of prompts, questions and concepts. Like in DialogDesigner but contrary to Suede this interface is textual and not graphical. However, one has - like in DialogDesigner - the option to view a graphical version of the designed dialogue model. In WOZ mode the designer chooses the output to the user from a list of possible next prompts or questions depending on the user's input. The export activity is facilitated from a menu point in the design window. There are several export possibilities, including export to XML, HTML or HDDL (a proprietary programming language used by the SpeechMania platform [Aust et al. 1995]).

Figure 5 gives a rough comparison of which features are included in DialogueDesigner, Suede and Woz tool.

8 Related Development Tools

IVR tools extended with recognition facilities, such as HotVoice from Dolphin and Edify, may also be seen as related work. Both these examples offer a graphical interface for dialogue flow design. In addition HotVoice also offers the possibility to edit the program text generated via the graphical interface or write the design directly in the HotVoice language. The language used by HotVoice as well as the one used by Edify are proprietary languages just like HDDL. A major difference between DialogDesigner and the IVR tools is that the possibilities for designing a dialogue using an IVR tool are fairly low-level. IVR tools are fine for specifying dialogues as a flow diagram. However, it would be difficult to use them for the design of complex dialogues.

Spoken dialogue platforms such as SpeechMania, Envox 6 VoiceXML Studio (both also support IVR), OpenSpeech, and the CSLU Toolkit are more aimed at implementation. To different extents they offer tools like "standard dialogues" for "best practices" in user interface design, such as entering a pin code.

However, common to these tools is that they focus on the implementation rather than on the modelling and evaluation – they are not case tools. And they do not focus on presentation to customers and users.

9 Conclusion and Future Work

We have described DialogDesigner which is a tool in support of SDS dialogue design and evaluation. It focuses on communication and modelling flexibility as argued in [Dybkjær and Dybkjær 2004]. The HTML extracts, graph views and simulation mode provide strong support for communication with customers and domain experts which is important in real-life projects. The ability to place conditions on states, transitions and prompts provides a useful flexibility in dialogue modelling.

Three next tool development and extension steps are planned. They include features for enhanced design process support (cf. Figure 5) as well as implementation support (code generation), transcription, and synthesis. Code generation will allow the automatic generation of VoiceXML code based on the design description presented above. Automatic code generation has the potential to save considerable effort. However, it will be a challenge to flexibly support e.g. agent or problem solving approaches. For transcription we envision a tool comparable to the TranscriptionStation included in the SpeechMania platform. It requires that spoken input is recorded and that the recognised utterances are used as the basis for the transcription process. The synthesis extension must allow the user of DialogDesigner to either record output phrases for use in system simulations or use speech synthesis for the same purpose.

Feature	DialogDesigner	Suede	Breuer	HotVoice	SpeechMania
graph view	+	+	+	+	-
graph design	*	+	-	+	-
structured prompts	+	-	-	(+)[2]	+
record prompts	-	-	(+)[4]	-	+
play prompts	*	+	+	+	-
speech recognition	-	-	(+)[4]	(+)[3]	(+)[3]
log analysis	+	+	-	-	-
regression test	*	-?	(+)[4]		+
debug	-	-	(+)[4]	+	+
WOZ	+	+	+	-	-
make test scripts	+	(+) [5]	-	-	-
phrase list	+	-	(+)[4]	-	+
prompt list	+	-	-	-	-
code generation	*	-	+	+	+
standard dialogues	-	-	-	(+)	+
state conditions	+	-	-	-	+

Figure 5. Tool comparison. +: Has feature. -: Does not have feature. ?: Unknown, *: In pipeline

References

Harald Aust, Martin Oerder, F. Seide and V. Stenbiss: The Philips automatic train timetable information system. *Speech Communication* 17, 1995, 249-262.

CSLU Toolkit: http://cslu.cse.ogi.edu/toolkit/

Hans Dybkjær and Laila Dybkjær: Modeling Complex Spoken Dialog. IEEE Computer, August 2004, 32-40.

Edify: http://www.edify.com/

Envox: www.envox.com

HotVoice: www.dolphin.no

OpenSpeech: http://scansoft.com/products/

Scott R. Klemmer, Anoop K. Sinha, Jack Chen, James A. Landay, Nadeem Aboobaker, and Annie Wang: SUEDE: A Wizard of Oz Prototyping Tool for Speech User Interfaces. CHI Letters, The 13th Annual ACM Symposium on User Interface Software and Technology: UIST 2000. 2(2): 1-10.

WOZ tool: http://www.softdoc.de/body/home.htm

[2] Must be coded.

[3] Has recognition as part of the running system but recognition cannot be tested during simulation.

[4] By using SpeechMania tools on generated code.

[S] Sound must be transcribed.