Human-Computer Interaction -- INTERACT'03
M. Rauterberg et al. (Eds.)
Published by IOS Press, (c) IFIP, 2003, pp. 753-756

Rapid prototyping collaborative dialogue interfaces

E. DeKoven, M.P.A.J. de Hoogh, D.V. Keyson

T.U. Delft, Landbergstraat 15, Delft, 2628CE, The Netherlands

{e.dekoven, m.p.a.j.dehoogh, d.keyson}@io.tudelft.nl

Abstract: An increasing number of consumer products for everyday use offer the user advanced programming capabilities. Existing technologies offer the potential to design human-product interaction supported by collaborative dialogue. Collaboration can afford a rich level of conversation between product and user. However, there are few methods and tools for designers, requiring a minimum of software engineering experience, that support the rapid creation and exploration of possibilities afforded by collaborative interfaces. The current paper presents a design process and tool for the early design and rapid prototyping of collaborative dialogue interfaces.

Keywords: rapid prototyping, dialogue management, tools, collaborative dialogue

1 Introduction

Microwaves, VCRs, and thermostats are examples of an increasing number of end-user programmable home appliances (PHA). By interacting with the user at the device feature level, current programmable appliances often cannot help the user explicitly communicate a goal to the product, nor can they suggest to the user which product features are best to use to achieve a given goal. Instead, the product should be designed to collaborate with the user on the process of achieving the user's goals.

Model-based development architectures, such as Collagen (Rich and Sidner, 1998), typically use an explicit embedded task model to support the creation of task-aware collaborative agents. Using plan recognition, a collaborative agent can compare observed user actions to the model, guess the user's current intentions, and determine efficient plans for achieving the user's goals. The agent can then generate task-specific advice, for example suggesting things the user may want to do or say next. If properly designed, the advice should alleviate some of the user's communication challenges and increase the user's efficiency in using the product.

However, the mere presence of advice-generating capabilities does not automatically lead to a more usable interface design. Moreover, model-based development platforms such as Collagen require too much development effort to build a reliable enough agent for early stages of design and usability testing. Instead, the designer should quickly be able to try out new dialogue sequences or agent responses, and in particular, expected user statements to which the system must respond.

The general goal of our research is to develop an integrated approach to the design, implementation, and evaluation of PHAs. In particular, our interest is in the development of interfaces for PHAs that support the user in communicating with the system about which task or strategy to pursue, as well as what to say or do next. In this paper, we present a dialogue design method and tool aimed at supporting rapid design and prototyping of collaborative interaction for PHAs.

2 A case study

Before describing our dialogue prototyping method, it is useful to consider an example of a interface with collaborative dialogue. An interface design supporting mixed user-product initiative was developed (see Figure 1) to consider how users could begin to program a home thermostat via a collaborative user-product dialogue (Freudenthal, et al. 2001).

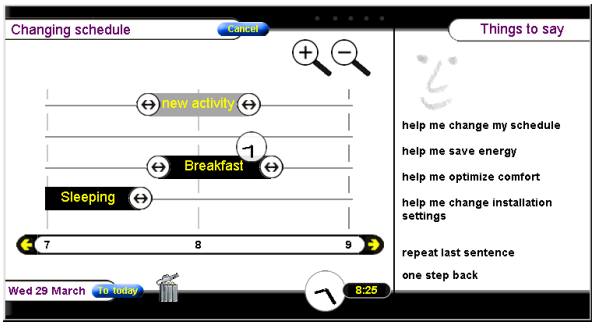

Figure 1: Thermostat schedule (left) and Things To Say menu (right).

The thermostat interface supports collaborative human-product dialogue in two unique ways. First, the title bar at the top-left corner of the screen shows a textual description of the user's current task. In addition, the right-hand side of the screen contains a task navigator, called the Things To Say (TTSay) menu. The TTSay contents are based on the current state of the user's task, including all of the relevant next tasks the user may want to consider. Selecting one of these items (through speech or touch) declares that goal as the user's current intention. The TTSay thus supports mixed-initiative negotiation and planning.

3 Dialogue prototyping

Dialogue design is an iterative process. Given the initial idea of programming a thermostat with reusable activities, it is not immediately obvious what to put in the TTSay, and what spoken commands to accept.

At very early stages of design, dialogue development can begin with having test subjects act out scenarios, one subject being the thermostat and the other subject the user. Such dialogues give some initial sense of the types of capabilities users may expect of the product. However, it is difficult to generalize from these dialogues to reusable agent behaviors. In particular, the test subject acting as the product cannot give consistent responses or useful advice without having clear bounds for what the product can do or say.

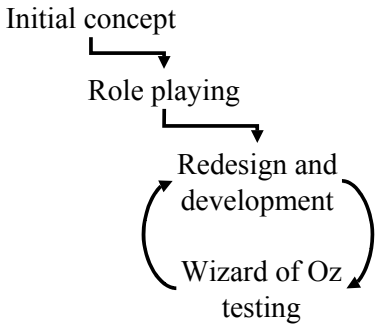

Figure 2: A typical dialogue design process.

Given a first design of the interaction, more detailed evaluation studies are typically done through Wizard of Oz (WOz) testing, in which a human wizard (usually one of the designers) plays the part of the dialogue agent. Results of testing will then be used to redesign the system to use more appropriate feedback or feedforward, and to identify specific key phrases and dialogue strategies that would most efficiently assist the user in achieving their task. The wizard needs a list of pre-designed responses from which to choose, along with a set of rules for when each response is applicable.

Whereas role-playing, and even early WOz testing, can be done with a minimum of materials, later design stages should be done with a running prototype of the target interface design. To support rapid design exploration, it should be simple to develop a dialogue, a WOz interface and a TTSay design from scratch with minimal programming requirements. As mentioned above, existing model-based systems require significant programming efforts. Similarly, existing dialogue prototyping tools, such as CSLUrp (McTear, 1999), are not designed to support WOz

testing, nor adaptive interfaces with TTSay level interaction. In order to enable rapid prototyoing of interfaces with TTSay collaborative dialogue functionality with task models we developed a method and design platform.

4 Dialogue Prototyping Platform

The dialogues supported by our Dialogue Prototyping Platform (DPP) are completely determined by data provided by the dialogue designer: frames, menus and scripts. A **frame** identifies a specific dialogue state, and is typically associated with a specific set of options the user can select grouped in a **menu**, each item associated with a set of commands to execute. To prevent unnecessary replication, **scripts** can take arguments (parameters) for execution of a set of commands. The dialogue state is maintained in a set of properties, which are name-value pairs (e.g. room:kitchen), and a stack of activated frames. Two types of properties are distinguished: application properties holding application data relevant to the dialogue, and dialogue properties holding dialogue data (possibly unrelated to the application).

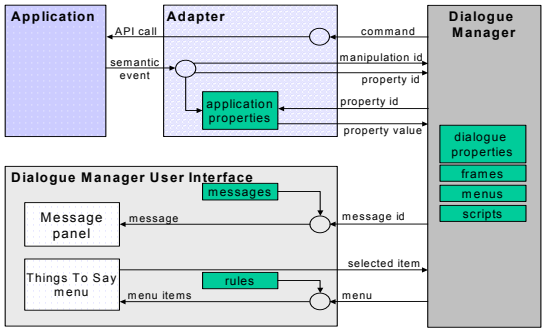

Figure 3: Architecture of our prototyping platform

Figure 3 shows a diagram of the basic architecture used by our dialogue prototyping platform. Four major parts can be distinguished:

Application: The Application is a prototype of the product interface. At least, the application needs to report user actions relevant to the dialogue – what are called 'semantic events' and process API calls.

Adapter: The adapter maintains the set of application properties. For each semantic event it determines what application properties should change and which manipulation was performed. The names of the updated properties and the manipulation are reported to the Dialogue Manager. It also executes commands, some of which return a result, by translating them into appropriate API calls.

Dialogue Manager (DM): The DM is initialized the frames, menus and scripts. These are used to process events from the Adapter and the DMUI, executing commands to change dialogue properties, activate a frame, and passing them to the Adapter for execution as specified in the menus and scripts. The DM also allows run-time inspection of dialogue properties.

Dialogue Manager User Interface (DMUI): The DMUI controls any task-adaptive user interfaces desired. The current implementation supports two interfaces, one for displaying any agent messages, visually and/or via speech to the user, and another to display a visual list of suggestions, such as the Things To Say menu used by Freudenthal et al. (2001) and Sidner and Forlines (2002). In the latter case, the DMUI also reports the selection of menu items to the DM.

The DPP also contains a default Wizard of Oz (WOz) implementation that will allow a wizard (typically the experimenter) to simulate speech recognition capabilities.

5 Using the platform

The Application, Adapter and Dialogue Manager User Interface need to be written by a programmer. The strength of this architecture is that it supports a designer to focus exclusively on the collaborative dialogue, in terms of predetermined states.

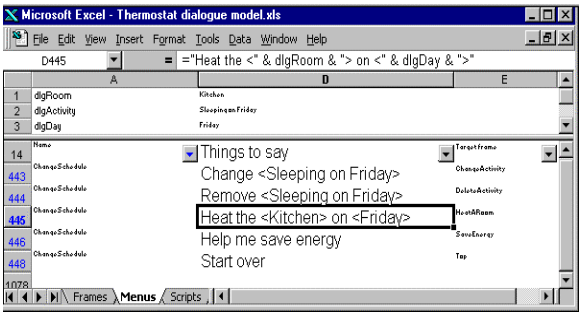

Figure 4: Dialogue design using Microsoft Excel.

At an early design stage, it is possible to engage test participants in a collaborative effort to determine how the dialogues should progress. New dialogue properties, frames and menus can quickly be defined and changed in the Excel file, with all participants seeing the results in real time. The DPP automatically turns this into a running dialogue, for use in the next round of design.

6 Discussion

So far, design students have developed three distinct expansions of the initial design using our method and tool and are currently conducting experiments using the designs. Whereas the design students do not have a programming background, they are able to use our tool in participatory design and evaluation sessions. This paper thus contributes to existing dialogue design tools by providing a simple prototyping technique for early stages of design.

An important next step would be to develop graphical tools to assist the designer in creating the frames, menus and scripts. In the future, we would also like to see how we could integrate our prototyping tool with more powerful dialogue management systems such as Collagen.

References

Freudenthal, A., Keyson, D.V., DeKoven, E., de Hoogh, M.P.A.J. (2001), Communicating extensive smart home functionality to users of all ages: the design of a mixed-initiative multimodal thermostat interface, in *OIKOS 2001 Workshop: Methodological Issues in the Design of Household Technologies*, Mol-slaboratoriet, Denmark, pp. 34-39.

McTear, M. (1999), Software to Support Research and Development of Spoken Dialogue Systems. *Eurospeech*, Budapest, Romania, September.

Rich, C., Sidner, C. L. (1998). COLLAGEN: A Collaboration Manager for Software Interface Agents, in *User Modeling and User-Adapted Interaction*, 8, pp. 315-350.

Sidner, C.L., Forlines, C. (2002), Subset Languages for Conversing with Collaborative Interface Agents, *International Conference on Spoken Language Processing (ICSLP)*, September.