DUDE: a Dialogue and Understanding Development Environment, mapping Business Process Models to Information State Update dialogue systems

Oliver Lemon and **Xingkun Liu**
School of Informatics
University of Edinburgh
{olemon,xliu4}@inf.ed.ac.uk

Abstract

We demonstrate a new development environment[1] "Information State Update" dialogue systems which allows non-expert developers to produce complete spoken dialogue systems based only on a Business Process Model (BPM) describing their application (e.g. banking, cinema booking, shopping, restaurant information). The environment includes automatic generation of Grammatical Framework (GF) grammars for robust interpretation of spontaneous speech, and uses application databases to generate lexical entries and grammar rules. The GF grammar is compiled to an ATK or Nuance language model for speech recognition. The demonstration system allows users to create and modify spoken dialogue systems, starting with a definition of a Business Process Model and ending with a working system. This paper describes the environment, its main components, and some of the research issues involved in its development.

1 Introduction: Business Process Modelling and Contact Centres

Many companies use "business process models" (BPMs) to specify communicative (and many other) actions that must be performed in order to complete various tasks (e.g. verify customer identity, pay a bill). See for example BPEL4WS [2] (Andrews, 2003). These representations specify states of processes or tasks, transitions between the states, and conditions on transitions (see e.g. the cinema booking example in figure 1). Typically, a human telephone operator (using a presentation of a BPM on a GUI) will step through these states with a customer, during a telephone interaction (e.g. in a contact centre), in order to complete a business process. Note, however, that BPM representations do not traditionally model dialogue context, so that (as well as speech recognition, interpretation, and production) the human operator is responsible for:

- contextual interpretation of incoming speech
- maintaining and updating dialogue context
- dialogue strategy (e.g. implicit/explicit confirmation, initiative management).

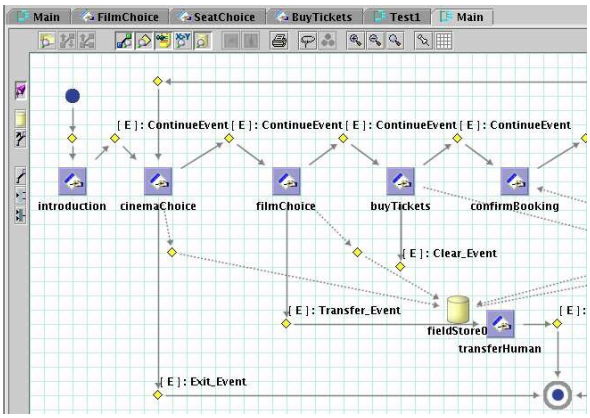

Figure 1: Part of an example Business Process Model (cinema booking) in the GT-X7 system (Graham Technology plc, 2005) (version 1.8.0).

A major advantage of current BPM systems (as well as their support for database access and enterprise system integration etc.) is their graphical development and authoring environments. See for example figure 1 from the GT-X7 system (Graham Technology plc, 2005), version 1.8.0. This shows part of a BPM for a cinema booking process. First (top left "introduction" node) the caller should hear an introduction, then (as long as there is a "ContinueEvent") they will be asked for the name of a cinema ("cinemaChoice"), and then for the name of a film ("filmChoice") and so on until the correct cinema tickets are payed for.

These systems allow non-experts to construct, modify, and rapidly deploy process models and the resulting interactions, including interactions with back-end

[1]This research is supported by Scottish Enterprise under the Edinburgh-Stanford Link programme. We thank Graham Technology for their collaboration.

[2]Business Process Execution Language for Web Services.

databases. For example, a manager may decide (after deployment of a banking application) that credit should now only be offered to customers with a credit rating of 5 or greater, and this change can be made simply by revising a condition on a state transition, presented as an arc in a process diagram. Thus the modelling environment allows for easy specification and revision of interactions. The process models are also hierarchical, so that complex processes can be built from nested combinations of simple interactions. By using these sorts of graphical tools, non-experts can deploy and manage complex business processes to be used by thousands of human contact centre operatives. However, many of these interactions are mundane and tedious for humans, and can easily be carried out by automated dialogue systems. We estimate that around 80% of contact-centre interactions involve simple information-gathering dialogues such as acquiring customer contact details. These can be handled robustly by Information State Update (ISU) dialogue systems (Larsson and Traum, 2000; Bos et al., 2003). Our contribution here is to allow non expert developers to build ISU systems using only the BPMs and databases that they are already familiar with, as shown in figure 2.

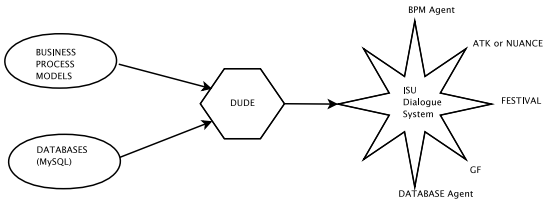

Figure 2: The DUDE development process

1.1 Automating Contact Centres with DUDE

Automation of contact centre interactions is a realistic aim only if state-of-the art dialogue management technology is employed. Currently, several companies are attempting to automate contact centers via simple speech-recognition-based interfaces using Voice XML. However, this is much like specification of dialogue managers using finite state networks, a technique which is known to be insufficient for flexible dialogues.

The main problem is that most traditional BPM systems lack a representation of dialogue context.[3] Here we show how to elaborate business process models with linguistic information of various types (e.g. how to generate appropriate clarification questions), and we show an ISU dialogue management component, which tracks dialogue context and takes standard BPMs as input to its discourse planner. Developers can now make use of the dialogue context (Information State) using DUDE to define process conditions that depend on IS features (e.g. user answer, dialogue-length, etc.).

[3]Footnote: The manufacturer of the GT-X7 system (Graham Technology plc, 2005) has independently created the agent247(TM) Dialogue Modelling component with dynamic prompt and Grammar generation for Natural Language Understanding.

Customers are now able to immediately declare their goals ("I want to change my address") rather than having to laboriously navigate a series of multiple-choice options. This sort of *"How may I help you?"* system is easily within current dialogue system expertise (Walker et al., 2000), but has not seen widespread commercial deployment. Another possibility opened up by the use of dialogue technology is the *personalization* of the dialogue with the customer. By interacting with a model of the customer's preferences a dialogue interface is able to recommend appropriate services for the customer (Moore et al., 2004), as well as modify its interaction style.

2 DUDE: a development environment

DUDE targets development of flexible and robust ISU dialogue systems from BPMs and databases. Its main components are:

- A graphical Business Process Modelling Tool (Graham Technology plc, 2005) (java)
- DIPPER generic dialogue manager (Bos et al., 2003) (java or prolog)
- MySQL databases
- a development GUI (java), see section 2.2

The spoken dialogue systems produced by DUDE all run using the Open Agent Architecture (OAA) (Cheyer and Martin, 2001) and employ the following agents in addition to DIPPER:

- Grammatical Framework (GF) parser (Ranta, 2004) (java)
- BPM agent (java) and Database agent (java)
- HTK speech recognizer (Young, 1995) using ATK (or alternatively Nuance)
- Festival2 speech synthesizer (Taylor et al., 1998)

We now highlight generic dialogue management, the DUDE developer GUI, and the use of GF.

2.1 DIPPER and generic dialogue management

Many sophisticated research systems are developed for specific applications and cannot be transferred to another, even very similar, task or domain. The problem of components being domain specific is especially severe in the core area of dialogue management. For example MIT's Pegasus and Mercury systems (Seneff, 2002) have dialogue managers which use approximately 350 domain-specific hand-coded rules each. The sheer amount of labor required to construct systems prevents them from being more widely and rapidly deployed. Using BPMs and related authoring tools to specify dialogue interactions addresses this problem and requires the development of domain-general dialogue managers, where BPMs represent application-specific information.

We have developed a generic dialogue manager (DM) using DIPPER. The core DM rules cover mixed initiative dialogue for multiple tasks (e.g. a BPM with several sub-processes), explicit and implicit confirmation, help, restart, repeat, and quit commands, and presentation and refinement of database query results. This is a domain-neutral abstraction of the ISU dialogue managers implemented for the FLIGHTS and TALK systems (Moore et al., 2004; Lemon et al., 2006).

The key point here is that the DM consults the BPM to determine what task-based steps to take next (e.g. ask for cinema name), when appropriate. Domain-general aspects of dialogue (e.g. confirmation and clarification strategies) are handled by the core DM. Values for constraints on transitions and branching in the BPM (e.g. present insurance option if the user is business-class) are compiled into domain-specific parts of the Information State. We use an XML format for BPMs, and compile them into finite state machines (the BPM agent) consulted by DIPPER for task-based dialogue control.

2.2 The DUDE developer GUI

Figures 3 to 5 show different screens from the DUDE GUI for dialogue system development. Figure 3 shows the developer associating "spotter" phrases with subtasks in the BPM. Here the developer is associating the phrases "hotels, hotel, stay, room, night, sleep" and "rooms" with the `hotels` task. This means that, for example, if the user says "I need a place to stay", the hotel-booking BPM will be triggered. (Note that multi-word phrases may also be defined). The defined spotters are automatically compiled into the GF grammar for parsing and speech recognition. By default all the lexical entries for answer-types for the subtasks will already be present as spotter phrases. DUDE checks for possible ambiguities (e.g. if "sushi" is a spotter for both cuisine_type for a restaurant subtask and food_type for a shopping process) and uses clarification subdialogues to resolve them at runtime.

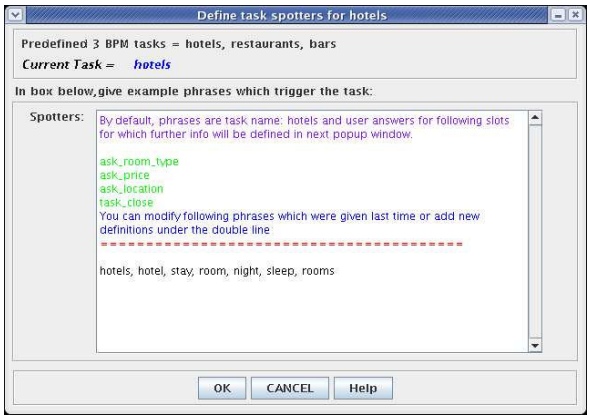

Figure 3: Example: using DUDE to define "spotter" phrases for different BPM subtasks

Figure 4 shows the developer's overview of the subtasks of a BPM (here, hotel information). The developer can navigate this representation and edit it to define prompts and manipulate the associated databases.

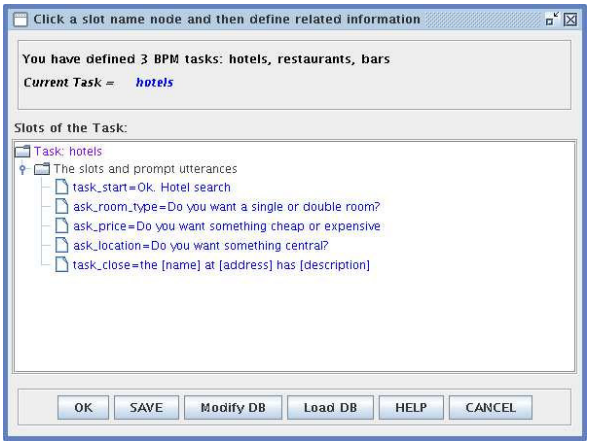

Figure 4: A Business Process Model viewed by DUDE

Figure 5 shows the developer specifying the required linguistic information to automate the "ask_price" subtask of the hotel-information BPM. Here the developer specifies the system prompt for the information ("Do you want something cheap or expensive?"), a phrase for implicit confirmation of provided values (here "a [X] hotel", where [X] is the semantics of the ASR hypothesis for the user input), and a clarifying phrase for this subtask (e.g. "Do you mean the hotel price?") for use when disambiguating between 2 or more tasks. The developer also specifies here the answer type that will resolve the system prompt. There are many predefined answer-types extracted from the databases associated with the BPMs, and the developer can select and/or edit these. They can also give additional (optional) example phrases that users might employ to answer the prompt, and these are automatically added to the GF grammar.

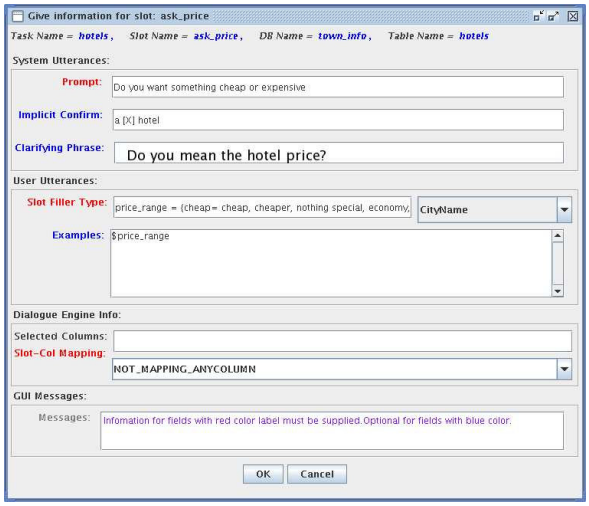

Figure 5: Example: using DUDE to define prompts, answer sets, and database mappings for the "ask_price" subtask of the BPM in figure 4

A similar GUI allows the developer to specify

database access and result presentation phases of the dialogue, if they are present in the BPM.

2.3 The Grammatical Framework: compiling grammars from BPMs, DBs, and example sets

GF (Ranta, 2004) is a language for writing multilingual grammars, on top of which various applications such as machine translation and human-machine interaction have been built. A GF grammar not only defines syntactic well-formedness, but also semantic content.

Using DUDE, system developers do not have to write a single line of GF grammar code. We have developed a core GF grammar for information-seeking dialogues (this supports a large fragment of spoken English, with utterances such as "Uh I think I think I want a less expensive X and uhhh a Y on DATE please" and so on). In addition, we compile all database entries and their properties into the appropriate "slot-filling" parts of the GF grammar for each specific BPM.

For example, a generated GF rule is:

```
Bpm_generalTypeRule_4:
town_info_hotels_name->Utt=->{ s = np.s}.
```

This means that all hotel names are valid utterances, and it is generated because "name" is a DB field for the subtask "hotels" in the "town_info" BPM.

Finally, we allow developers to give example sentences showing how users might respond to system prompts. If these are not already covered by the existing grammar we automatically generate rules to cover them. Finally GF, is a robust parser – it skips all disfluencies and unknown words to produce an interpretation of the user input if one exists. Note that the GF grammars developed by DUDE can be compiled to speech-recognition language models for both Nuance and HTK/ATK (Young, 1995).

2.4 Usability

We have built several demonstration systems using DUDE. We are able to build a new system in under an hour, but our planned evaluation will test the ability of novice users (with some knowledge of BPMs and databases) to iteratively develop their own ISU dialogue systems.

3 Summary

We demonstrate a development environment for "Information State Update" dialogue systems which allows non-expert developers to produce complete spoken dialogue systems based only on Business Process Models (BPM) describing their applications. The environment includes automatic generation of Grammatical Framework (GF) grammars for robust interpretation of spontaneous speech, and uses the application databases to generate lexical entries and grammar rules. The GF grammar is compiled to an ATK language model for speech recognition (Nuance is also supported). The demonstration system allows users to create and modify spoken dialogue systems, starting with a definition of a Business Process Model (e.g. banking, cinema booking, shopping, restaurant information) and ending with a working system. This paper describes the environment, its main components, and some of the research issues involved in its development.

References

Tony Andrews. 2003. Business process execution language for web services, version 1.1, http://www-106.ibm.com/developerworks/library/ws-bpel/. Technical report, IBM developer works.

Johan Bos, Ewan Klein, Oliver Lemon, and Tetsushi Oka. 2003. DIPPER: Description and Formalisation of an Information-State Update Dialogue System Architecture. In *4th SIGdial Workshop on Discourse and Dialogue*, pages 115–124, Sapporo.

Adam Cheyer and David Martin. 2001. The Open Agent Architecture. *Journal of Autonomous Agents and Multi-Agent Systems*, 4(1/2):143–148.

Graham Technology plc. 2005. GT-X7 v.1.8.0 from Graham Technology plc [without the agent247(TM) Dialogue and NLP Engine]. www.grahamtechnology.com.

Staffan Larsson and David Traum. 2000. Information state and dialogue management in the TRINDI Dialogue Move Engine Toolkit. *Natural Language Engineering*, 6(3-4):323–340.

Oliver Lemon, Kallirroi Georgila, James Henderson, and Matthew Stuttle. 2006. An ISU dialogue system exhibiting reinforcement learning of dialogue policies: generic slot-filling in the TALK in-car system. In *Proceedings of EACL*, page to appear.

Johanna Moore, Mary Ellen Foster, Oliver Lemon, and Michael White. 2004. Generating tailored, comparative descriptions in spoken dialogue. In *The 17th International FLAIRS Conference (Florida Artifical Intelligence Research Society)*.

A. Ranta. 2004. Grammatical framework. a type-theoretical grammar formalism. *Journal of Functional Programming*, 14(2):145–189.

Stephanie Seneff. 2002. Response Planning and Generation in the Mercury Flight Reservation System. *Computer Speech and Language*, 16.

P. Taylor, A. Black, and R. Caley. 1998. The architecture of the the Festival speech synthesis system. In *Third International Workshop on Speech Synthesis, Sydney, Australia*.

M. A. Walker, I. Langkilde, J. Wright, A. Gorin, and D. Litman. 2000. Learning to Predict Problematic Situations in a Spoken Dialogue System: Experiments with How May I Help You? In *Proceedings of the NAACL 2000*, Seattle.

Steve Young. 1995. Large vocabulary continuous speech recognition: A review. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 3–28.