THE REWARD SERVICE CREATION ENVIRONMENT, AN OVERVIEW

Tom Brøndsted, Bo Nygaard Bai, Jesper Østergaard Olsen

Center for PersonKommunikation
Fredrik Bajers Vej 7A-5
Institute for Electronic Systems,
Aalborg University
DK-9220, Aalborg Ø, Denmark
{tb,bai,jo}@cpk.auc.dk

ABSTRACT

The present paper describes the platform for building spoken language systems being designed and implemented within the EU-language engineering project REWARD. The platform collects and streamlines a set of software tools such that they together constitute the basic modules needed to enable dialogue developers to establish new dialogue applications with only minimal knowledge outside their own field of experience and within a minimum amount of time. The system differs from other platforms, as non-expert users have been strongly involved in the design phase.

Keywords: spoken language dialogues, non-expert design tools, resource management, reusability, rapid prototyping, abstraction, and extendibility.

1. INTRODUCTION

The three year EU-Language Engineering project REWARD ("Real World Applications of Robust Dialogue" LE1-2632) addresses the needs of organisations which do business over the telephone (i) to automate certain telephone services using spoken language dialogue technology and (ii) to automate the process of creating such services. The project brings together two technology suppliers, Vocalis Ltd. and CPK (Center for PersonKommunikation), along with a group of user organisations consisting of: (a) The British telemarketing and market research organisation Taylor Nelson AGB, (b) the Dutch market research organisation NIPO (Nederlands Instituut voor de Publieke Opinie en het Marktunderzoek), (c) the Danish business travel agency DanTransport and (d) the Spanish hardware maintenance company MADE (Manufacturing and Development SA).

In the spring of 1997, the REWARD project underwent a major revision prompted by the realisation that a simple merger of the technology suppliers' existing tools would not achieve the stated goals of the project in terms of both functionality and ease of use. As a result, CPK was given the primary role of developing a new suite of dialogue creation tools. These tools should significantly improve the productivity of a dialogue designer and bring down the cost of developing and deploying spoken dialogue teleservices. During the design and implementation of the new tools, the user group should design dialogues using existing dialogue building tools developed at Vocalis and, in parallel, participate as advisers in the design phase of the new platform. Thus, the platform described in this paper is the result of design decisions largely initiated by the users (cf. [8], [9]).

2. THE PLATFORM

The REWARD platform consists of two major software components (Figure 1): (i) The Dialogue Creation Environment and (ii) the Runtime System. The Dialogue Creation Environment is being built entirely by CPK. The runtime system is primarily being built by Vocalis Ltd., using natural language technology developed at CPK.

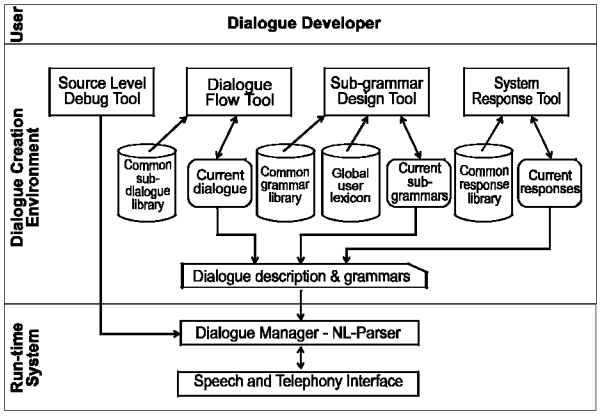

Figure 1: REWARD Platform architecture.

The Dialogue Creation Environment being focussed on in this paper provides users with a number of window-based tools (implemented in JAVA). The tools allow them to implement, debug, and maintain dialogue systems in a dedicated dialogue description language and to maintain reusable dialogue resources. Tools for describing dialogue flows and defining sub languages make up the main components of the environment.

2.1. Dialogue flow structure.

In the beginning of the project, two approaches to flow chart based dialogue design were evaluated. The first one based on the Generic Dialogue System (gds) described in [2] and the second one on a dialogue building system developed at Vocalis Ltd. As a simple merger of the best sub components of the two systems turned out to be unrealisable, the development of an

entire new tool, DFT (Dialogue Flow Tool), based on the gained experiences was initiated.

The DFT is the primary entry point to the Dialogue Creation Environment. Dialogues are described and implemented in a graphical environment using directed flow chart structures. The DFT works with two formalisms for the representation of dialogue descriptions:

1. A Graphical Programming Language (GPL).
2. A textual scripting Dialogue Specification Language (DSL).

The DFT compiles GPL into the equivalent DSL for execution by the dialogue manager of the runtime system. The largest proportion of DSL is taken up with commands corresponding to the states of the flow chart. Currently, round fifty commands have been defined including the very basic ones like recognising, answering, recording, and dialling. It is possible for the user to use the DSL scripting language directly, but this is not encouraged. The scripting language is primarily intended for allowing integration of other dialogue development tools with the DFT. The DFT can import any legal and complete sub-dialogue written in the DSL scripting language.

The user edits the flow of control in the dialogue description by the method of direct manipulation. The user can directly refer to and manipulate the commands and variables in the dialogue description trough a simple point and click interface. This is opposed to the name binding used in the textual scripting language where names are used to bind a transit from one node to the label of the next command to be executed.

The flow graph consists of nodes connected by arrows. Each node represents some action to be taken at that point in the dialogue. The arrows specify the possible ways the different actions can follow each other.

An extensive verification process helps to catch errors early in implementation process. The user interface of the DFT is intended to support the work of the dialogue designer through the entire dialogue creation process. To aid this, the validation process can be configured to check only certain aspects of the dialogue design. In the early stages of the design phase, the dialogue designer may want to merely sketch the overall flow of the dialogue and not to be bothered with filling in the details of the individual dialogue actions. At this stage the designer may want to know only that the flow graph contains no dead ends or unreachable nodes. The DFT allows editing of such incomplete dialogue descriptions but will not attempt to compile and run them.

A. Frames.

The dialogue flow graph is contained in a frame. A frame is comparable to a function in C or a procedure in Pascal. A frame can take any number of parameters and return any number of values. In addition, a frame can hold a number of local variables for its own internal calculations. The flow graph of a large and complex dialogue will typically be split into a number of frames, each corresponding to a sub-dialogue. Frames can be nested to any depth.

It is intended that frames can be used to represent reusable sub-dialogues.

B. Nodes.

Each node takes a number of parameters and can return a number of values. Results are passed back to the enclosing frame after completion of the node's action. For each returned value the enclosing frame can chose to ignore the value or assign it to a variable. Additionally, each node has an associated inspector that provides a specialised graphical interface for that particular type of node. The inspector is used to bind variables and values to the node's parameters. Typical parameters may be prompts to be played or grammars to be used for recognition and parsing.

C. Expressions.

The GPL includes a special type of node that implements a simple expression syntax. This means that calculations and simple data manipulations can be implemented directly in the dialogue description. This reduces the need to extend the runtime system with external C-functions for many common operations.

D. Extensions.

Although many commonly used operations can be accomplished in the dialogue description by the use of expressions, it is not intended to have the power of a general purpose programming language. Complex application specific processing must be implemented in an external programming language like C, and added to the set of available dialogue commands.

E. Procedures.

The procedure frames appear as nodes in the node library. The DFT automatically builds and maintains a default inspector for new procedure nodes. This means that a procedure node can immediately be used in the dialogue description and that the process of calling a procedure node is equivalent to that of calling one of the basic nodes. Procedure nodes can also be added to the common sub-dialogue library (see section 2.3), which effectively makes them an extension of the dialogue language.

2.2. Sub Language Definition.

Two approaches to sub language definition and sub grammar design have been examined and elaborated:

- In the beginning of the project, the users evaluated a flow chart based tool for drawing label based recursive transition network grammars (RTNs) in a graphical environment. The tool was a sub component of the Generic Dialogue System described in [2] and similar to the concept of the GrapHvite developer kit [10] with an additional possibility of attaching "semantic actions" to word transitions.

- Since the revision of the project (cf. section 1), a radical different approach has been implemented based on the initially somewhat vague idea that the Service Creation Environment should be able to generate the necessary sub grammars based on a few word, phrase, or sentence examples typed in by the user.

The users rejected the first approach primarily because the flow chart based tool was too heavy to use when dealing with simple keyword or phrase ("concept") spotting tasks. Further, the tool provided no facility for resource management or reuse of grammars.

The second approach has led to the unification based tool described in greater detail in [5], [6], [7]. The tool presupposes a Global User Lexicon where lexical items are coded in a compound feature based format. The Global User Lexicon allows expressions like *{category=city, country=UK}* to be deduced from word lists like "*London, Leeds*" and subsequent expansion of the word lists based on the deduced expressions: "*London, Leeds, Manchester, Liverpool etc.*" (cities in the United Kingdom).

Global User Lexicons are not necessarily "general" in a linguistic sense. In the REWARD project, it is presupposed that they are implemented and maintained by the users themselves and that they reflect the class of domains relevant to the users' organisation. It is expected that the global lexicon largely can be generated using attributed databases like staff lists, lists of articles or customers etc. available in the user organisation. Thus, the Global User Lexicon is an important resource management facility of the Service Creation Environment (see section 2.3).

The unification based concept described above, presupposes a more powerful compound feature based grammar formalism than the initially evaluated RTNs. The formalism for sub language definition is an Augmented Phrase Structure Grammar (APSG) format. APSGs are used both for parsing (extracting semantics from) spoken input and, in a converted finite state approximation format, for constraining speech recognition. Vocabularies are phonemically transcribed using a dedicated transcription tool. Semantics generated by the parser is represented in nested lisp-like frame structures that are interpreted by the dialogue manager of the run-time system. For a more detailed description of the grammar format, parser, converter, and semantic frames, refer to [4], [6]. The concepts behind the phonetic transcription tool are described in [1].

As the user organisations mostly deal with very system-directed dialogues, they will normally implement one named sub grammar for each state associated with a "system prompt" in the dialogue. However, more than one sub grammar can be active at any time such that the dialogue control can branch on the name of the sub grammar recognising and parsing input. The system is capable of analysing grammars and deducing the possible values for each semantic frame. The DFT uses this facility to validate branches on semantic values returned by the natural language parser of the Run Time System.

2.3. Reusability in Spoken Dialogues.

A spoken dialogue system can be viewed as a program. The task of creating a program becomes much easier when the programmer has at his disposal a suitable library of functions that fits the application domain. Also, when he makes several programs within the same application domain, he can often reuse parts of earlier programs. An obvious way to speed up the dialogue creation process would be to use suitable prefabricated elements. Unfortunately no great store of reusable dialogue components currently exists.

The users in the REWARD project need to use spoken dialogues systems within the fields of telemarketing and market research. For this to be viable, it must be possible to quickly and cheaply create and deploy spoken dialogues systems. These systems will typically have a limited life span but will be very similar in nature. It must be expected that there will be a large degree of commonality between such dialogues. This presents an obvious case for reuse.

A spoken dialogue system is a collection of many different types of resources like common sub grammars for parsing dates, parsing passwords etc. Common sub grammars may be embedded in common sub dialogues e.g. describing clarification dialogues in the context of date prompting, password prompting etc.

This complicates the process of collecting reusable spoken dialogue components. The flow graph description of a sub-dialogue has little meaning without the underlying resources on which it depends: sub grammars, word-models and prompts. An important function of the Service Creation Environment is managing resources and their interdependencies to simplify the extraction of reusable sub-dialogues.

When a sub-dialogue is found to be reusable, it should be placed in a frame and put in the library of common sub-dialogues. When a sub-dialogue is moved to the common sub-dialogue library, the DFT first validates it to ensure that it is complete, It then makes a complete dependency analysis to locate all external resources referenced by the sub-dialogue. Finally, the DFT makes a copy of all the resources referenced by the sub-dialogue and attaches it to the sub-dialogue. This effectively turns the sub-dialogue into a fully self-contained and reusable dialogue unit.

It is the intention that user organisations over time will be able to build their own library of customised dialogue components.

2.4. Wizards

Wizards are a programmatic representation of pieces of domain specific knowledge. The Dialogue Creation Environment employs wizards to encapsulate knowledge about the design and implementation of spoken dialogue systems. By using wizards the dialogue designer can let the Dialogue Creation Environment take an active part in the design and implementation of the dialogue. This can greatly speed up the implementation of common spoken dialogue idioms and may also help beginners to make better dialogues.

The DFT supports the use of two types of wizards:

- Task-wizards are pre-programmed actions that semi-automates common dialogue design tasks and idioms. A task-wizard must be activated explicitly by the dialogue designer to aid him with a certain task. An example could be a wizard for building the skeleton dialogue for a standard information retrieval task. The wizard would first ask the designer to answer a number of questions about the dialogue e.g. should it have an opening prompt? Should it continue in a question-answer loop allowing multiple inquiries? Etc. Based on the answers, the wizard will perform the actions needed to build a standard implementation of such a dialogue. Afterwards the designer can work on from the template dialogue created by the wizard.
- Guardian-wizards are active agents that are looking over the shoulder of the dialogue designer, warning him about problematic constructs and recommending alternatives. Guardian-wizards may be triggered indirectly by actions performed by the dialogue designer. An example could be a wizard warning that a particular recognition vocabulary is stressing the speech recognition and is likely to yield bad recognition performance. The wizard may then advise the designer to use an extra turn to reduce the perplexity of the individual turns.

Wizards for use in spoken dialogue design systems is still an unexplored area. The Dialogue Creation Environment will initially have very few wizards available. It is anticipated that tasks for which useful wizards can be made will become apparent through the process of building dialogues using the tools.

3. CONCLUSION

A common requirement for the user organisations in REWARD is the ability both to implement and deploy new spoken dialogue applications within a very short time frame and to be able to continuously update existing spoken dialogue services. An important step for the organisations in achieving this goal is to develop and maintain their spoken dialogue services in house. The REWARD Service Creation Environment aims at simplifying the process of constructing a spoken dialogue system to the level where non-specialists given a reasonable amount of training can learn how to implement and deploy spoken dialogue systems within their domain of expertise.

Due to the revision of the project mentioned in the introduction, the users have not yet had time for a proper evaluation of the Service Creation Environment. However, the strong user involvement in the specification and design phases gives reason to believe that the environment achieves the stated goals in terms of both functionality and ease of use.

A release of the Dialogue Creation Environment for educational and scientific use is planned. A more portable runtime system needs to be developed for this release. The NLP modules will be made available also as an independent distribution.

4. REFERENCES

1. O. Andersen, R. Kuhn, A. Lazarides, P. Dalsgaard, J. Haas, E. Nöth: "Comparison of two tree-structured approaches for grapheme-to-phoneme conversion". *Proc. of ICSLP 1996, pp. 1808-11.*
2. A. Baekgaard: "A Generic Dialogue System". *Spoken Language Dialogue Systems 10. R 96-101, CPK, Aalborg University 1996.*
3. T. Brøndsted, L.B., M. Manthey, P. Mc Kevitt, T. Moeslund, K.G. Olesen: "The Intellimedia WorkBench - an environment for building multimodal systems". *Second International Conference on Cooperative Multimodal Communication, Theory and Applications. Tilburg 1998, pp. 66-70.*
4. T. Brøndsted, L.B., P. Dalsgaard, M. Manthey, P. Mc Kevitt, T. Moeslund, K.G. Olesen: *A platform for developing Intelligent MultiMedia Applications. Technical Report R-98-1004. CPK, Aalborg University 1998.*
5. T. Brøndsted: "The Linguistic Components of the REWARD Dialogue Creation Environment and Run Time System". *4th IEEE WS on IVTTA. Turin 1998. In press.*
6. T. Brøndsted: "The Natural Language Parsing Modules in REWARD and IntelliMedia 2000+". *S. Kirchmeier-Andersen, H.E. Thomsen (eds.): Proceedings from the Danish Society for Computational Linguistics (DALF), Copenhagen Business School, Dep. of Computational Linguistics, 1998. In press.*
7. Tom Brøndsted: "Non-Expert Access to Unification based Speech Understanding". *These Proceedings.*
8. K. Failenschmid: "Spoken Dialogue System Design – The Influence or the Organisational Context on the Design Process". *4th IEEE WS on IVTTA. Turin 1998. In press.*
9. K. Failenschmid, S. Thornton: "End-user driven Dialogue System Design". *These Proceedings.*
10. K. Power, C. Matheson, D. Ollason, R. Morton: *The grapHvite Book. For grapHvite v1.0. Entropic, Cambridge Research Laboratory. Cambridge 1996.*