

# Using the CSLU toolkit for practicals in spoken dialogue technology

*Michael F. McTear*

School of Information and Software Engineering, University of Ulster, Newtownabbey, Northern Ireland  
mf.mctear@ulst.ac.uk

## **Abstract**

Spoken language interaction with computers has become a practical possibility as a result of recent technological developments in the speech sciences. This paper reports on the use of CSLU's RAD (Rapid Application Developer) to provide practical experience for undergraduate students in the specification and development of spoken dialogue systems. Two groups of students were involved. The first group included students of linguistics, speech and language therapy, and communication; the second group included students of computational linguistics who had taken several courses in computing. The paper describes the use of the toolkit for students with these different degrees of competence in computing and reports on plans for future work with the toolkit.

## **1. Introduction**

Spoken language interaction with computers has become a practical possibility as a result of recent technological developments in the speech sciences. However, the development of a spoken dialogue system is a complex process involving the integration of the various components of spoken language technology, including speech recognition, natural language processing, dialogue modelling, and speech synthesis. Fortunately, various toolkits and authoring environments have been produced that provide assistance with this process. This paper reports on the use of CSLU's RAD (Rapid Application Developer) to provide practical experience for undergraduate linguistics and computational linguistics students in the specification and development of spoken dialogue systems. The paper concludes with an overview of ongoing and future projects involving the toolkit.

## **2. The CSLU toolkit: a brief overview**

The CSLU toolkit, which is available free-of-charge under a license agreement for educational, research, personal, or evaluation purposes, was developed at the Center for Spoken Language Understanding (CSLU) at the Oregon Graduate Institute of Science and Technology to support speech-related research and development activities [1]. The toolkit includes core technologies for speech recognition and text-to-speech

synthesis, as well as a graphically-based authoring environment (RAD) for designing and implementing spoken dialogue systems. Building a dialogue system with RAD involves selecting and linking graphical dialogue objects into a finite-state dialogue model, which may include branching decisions, loops, jumps, and sub-dialogues. Each object can be used for functions such as generating prompts, recording and recognizing speech, and performing actions. As far as speech recognition is concerned, the input can be in the form of single words, for which a tree-based recognizer is used, or as phrases or sentences that are specified using a finite state grammar, which also enables keyword spotting. There are additional built-in facilities for digit and alpha-digit recognition. It is also possible to implement dynamic recognition, in which case a list of words to be recognized is obtained from some external source, such as a Web page, and pronunciation models for the words are generated dynamically at run-time. Prompts can be specified in textual form which are output using the Festival TTS (text-to-speech) system, or they can be pre-recorded, and, with some additional effort, spliced together at run-time. The use of the sub-dialogue states permits a more modular dialogue design, as sub-tasks, such as eliciting an account number, can be implemented in a subdialogue that is potentially re-usable. Repair dialogues are a special case of sub-dialogue. A default repair sub-dialogue is included that is activated if the recognition score for the user's input falls below a given threshold, but it is also relatively easy to design and implement customized repair subdialogues and one example of how to do this is provided in the tutorial manual. Finally, there are facilities to support external communications. An interface with MS Excel was provided in an earlier release that used dynamic link libraries (DLLs) to communicate with the back-end system. This enabled simple applications such as catalogue ordering to be developed, although the interface provided was not particularly stable. Functions are provided for voice-based Web access using sockets for Internet communication. In this way it is possible to download and parse a Web document – for example, for football scores – and to develop a dialogue that provides answers to the user's questions about how a team performed. The functions that are included enable a given URL to be accessed and the HTML document to be read and parsed, relevant

strings to be identified, tags to be removed, and the required information to be output using text-to-speech.

### 3. Use of the toolkit for courses in spoken dialogue technology

The toolkit has been used at CSLU to support courses for a wide variety of students, including high school pupils. Several interesting applications have been developed, including Airline Reservations, Directory Assistance, Bank-By-Phone, and a Homework Help-Line [2]. The toolkit has also been used at CSLU to develop interactive learning tools for language training with profoundly children and for children with speech disabilities [3].

This paper reports on the use of the toolkit at the University of Ulster. Initially the toolkit was used for final year projects that included applications such as the simulation of dialogues with household equipment such as a microwave oven and a VCR [4] and a system for directory assistance [5]. These applications involved the development of a fairly complex system by a single student over the course of a year. The toolkit is now being used additionally to provide practicals in a course on spoken dialogue technology. Previously the technologies for spoken dialogue systems were presented in lectures, using examples of systems reported in the literature and with practicals for discussion on ways of specifying and designing such systems. Now, with the use of the toolkit, it is possible for students to put the theory into practice. The course consists of lectures, in which the theoretical background is presented, and laboratory classes, in which relevant techniques involving the toolkit are taught and practised. The students then specify and develop a system of their choice. Assessment includes both the program developed using the toolkit as well as a paper integrating the theoretical contents of the lectures with what was achieved and achievable using the toolkit. The course has been taught to two groups of students. The first group involves students of linguistics, speech and language therapy, and communication studies. These students have a background in theories of language, dialogue and interaction, but have limited computing skills and no previous programming experience. The second group is from computational linguistics. These students have studied computing science as well as linguistics and can thus be expected to cope more easily with applications that require some programming competence. The next sections report on what could be achieved with these two groups of students.

#### 3.1 Using the toolkit with non-computing students

Students were introduced to the basics of the software development lifecycle for spoken dialogue systems, involving requirements analysis, specification, design, implementation, and testing. Thus they were able to

motivate the potential applicability of spoken dialogue technology for various transactional dialogues, such as ordering meals from a student meal service, placing orders, establishing delivery dates, or connecting to a salesperson in a furniture store, and booking in animals for a cattle market. These applications had in common a well-structured task that could be decomposed into orderly sub-tasks. Based on a detailed task analysis the students were able to specify the dialogue flow using flowcharts, which in turn could be easily implemented in RAD using a finite-state dialogue, as shown in the following example.



Figure 1. Using RAD to implement a simple spoken dialogue system for a student meals service.

This dialogue gives a simple choice of meal types, each of which allowed a limited number of choices. A sub-dialogue was implemented for drinks, as this sub-dialogue could be used for either meal type.

The furniture store application involved a more complex dialogue structure as three different types of transaction were involved, each with more complex sub-tasks, such as requesting a particular delivery date. However, the basic design principle is the same as for the simpler application.

Designing dialogues such as these enabled the students to appreciate the need for a clear analysis and specification of the system and to understand the advantages as well as the limitations of a state-based dialogue control. For example: providing confirmations and permitting corrections of misrecognised items required careful design to take account of the limitations of the dialogue control while at the same time ensuring that information had been accurately elicited and allowing for an acceptable dialogue flow. Additional tasks involved a limited amount of coding, for example, to return variables holding the words recognised at a particular dialogue state, to provide spliced prompts, or to enable dynamic recognition. Students from the Speech and Language Therapy degree also used the facility for hand-crafted

pronunciations to develop an interface that would recognise the speech of a user with a speech disability.

### **3.2 Using the toolkit with computing students**

The second group of students were in the final year of a degree programme in Computing and Linguistics, having taken a number of modules in computing, linguistics, and computational linguistics. Thus it could be assumed that these students would feel more confident with any programming required for applications with the toolkit, although none of the students had any previous experience with TCL, the language used for application development with the toolkit.

One of the main limitations of the applications developed by the non-computing students was that, although their systems elicited a wide range of information through the dialogues, there was no facility to communicate with an external information source such as a database, spreadsheet, or Web page. The second group of students was in a position to develop applications with such a facility.

Interfacing with a Web page involved the use of the built-in functions supplied with the toolkit. One application that was developed made use of the Faculty of Informatics Web pages that had a standard set of pages holding information about members of staff, such as their emails, room numbers, and research interests. Given that the structure of these pages was not likely to change, functions could be implemented to parse the pages to find the relevant information to answer the user's queries as elicited in the spoken dialogue. For example: the system would acquire the name of the person for whom the information was required as well as the item of information to be supplied. The name was appended to a URL template so that the appropriate page could be accessed and parsed. Using the facility for dynamic recognition, items of information that were changed, such as titles of new research projects, could be added to the recognition vocabulary, as required.

Other types of file access are not currently supported within the toolkit and had to be simulated. For example: a suitably structured text file could be used as a rudimentary database which could be accessed using various functions specified in TCL to read in the file and find relevant records (in the form of separate lines in the file) and fields (in the form of TCL lists) that retrieved the information specified in the query as elicited in the user-system dialogue.

A final enhancement was to develop additional functions such as logging the dialogue flow and validating account numbers. Logging the dialogue flow was useful as a means of debugging programs under development. Validation routines were an application of techniques acquired by the students for the implementation of standard routines in computing. For

these functions and for communication with external information sources some degree of competence in TCL was required.

### **3.3 Evaluation**

Although no formal evaluation has been carried out as yet on the use of the toolkit to support the practical sessions as described here, it is useful to report on some informal observations. The non-computing group consisted of 25 students with access to a lab in which the toolkit was installed on each individual PC. The computing group consisted of 14 students. In this case the toolkit was accessed from a server running on a Novell network. Both groups of students experienced teething problems with the system hanging up, returning incomprehensible error messages, or generally performing at a sub-optimal level. Some of these problems were due to the inexperience of the students who made predictable errors that led to system failure. In other cases the hardware was simply not powerful and fast enough for the software. In the case of the server-based system, some functions, such as the alpha-digit and digit recognisers that rely on DLLs, did not work. Despite these problems, the students reported a great sense of achievement and excitement at being able to specify and implement a working spoken dialogue system. These encouraging results provide a good foundation for the extension of the course to students in other degree programmes and, in the longer term, to schoolchildren and adolescents.

## **4. Future developments**

There are several ways in which the ideas presented could be further developed. Some of these developments involve implementing additional functions using the facilities provided in the current version of the toolkit, while other developments depend on facilities to be provided in future releases [1].

### **4.1 More complex repair and validation**

In the current version of the toolkit there is a default repair sub-dialogue that is activated if the recognition score for the user's input falls below a given threshold. The RAD manual presents an alternative repair strategy in which the sub-dialogue prompts for a clarification if the score of the 1-Best word is close to the threshold or if on mis-recognition the same word is listed as the 1-Best word two times in a row.

There is scope for a range of additional repair strategies, some of which can be used to handle generic repair situations such as low confidence recognition, while others apply to particular dialogue functions, such as eliciting a name or account number. These different types of repair sub-dialogue are currently being developed in a number of student projects.

There is also a need for specialised sub-dialogues for functions such as eliciting a series of digits, as in an account number, or eliciting expressions of time and

dates. Problems can occur in the elicitation of strings of digits. For example: the system may recognise more or fewer digits than were actually spoken by the user. There is a need for clearly specified sub-dialogues that can successfully elicit the correct string of digits. Similarly, sub-dialogues for time and date expressions, making use of grammar-based recognition specified using finite-state grammars, would provide a set of reusable components for a wide range of applications. In each case the subdialogues would require robust error handling routines.

#### 4.2 Interface to information sources

A clear requirement for dialogue systems is the ability to communicate with external information sources. Work is currently ongoing, both at CSLU and the University of Ulster, to develop a robust interface to a database system such as MS Access. With such an interface students will be able to develop dialogues that retrieve information from a database and possibly update the database, as required.

#### 4.3 Multilingual dialogue

The toolkit has been developed in a number of languages, including Mexican Spanish [6]. There is currently a proposal to extend the languages to include some minority languages, such as Irish. This work would introduce a number of research challenges, such as the issue of the generalizability of the technologies currently used in the toolkit to minority languages. There would also be a number of practical advantages, such as the application of the toolkit to the teaching of minority languages.

#### 4.4 Technical developments at CSLU

The toolkit is constantly under development at CSLU. Two projects are of particular significance for spoken dialogue technology. The first involves the addition of a robust parsing component to provide for linguistic processing of the user's input. Currently the system can recognise either single words or strings determined by a finite state grammar. More complex input, including the phenomena of spontaneous speech such as hesitations and restarts, cannot be handled, and the recognised strings are not subject to further linguistic analysis. A new component PROFER (Predictive ROBust Finite-State parsER) that is to be integrated into a future version of the toolkit will enable robust semantic parsing with output in the form of case-frame representations of the concepts extracted from the recognised input [7].

A second development involves more advanced dialogue management. The limitations of finite state dialogue control are well-known [8]. At the same time, the structured control provided by the toolkit has the advantage of ease of development. Work in progress at CSLU is aimed at providing more flexible dialogue

control while at the same time preserving the advantages of a structured dialogue design. [9]

### References

- [1] Sutton S et al. (1998). Universal Speech Tools: The CSLU Toolkit, *Proc 5th International Conference on Spoken Language Processing*, Dec 1998, Sydney, Australia, 3221-3224.
- [2] Colton D, Cole R, Novick D and Sutton S (1996). A laboratory course for designing and testing spoken dialogue systems. *Proc International Conference on Acoustics, Speech and Signal Processing*, May 1996, Atlanta, GA, 1129-1132.
- [3] Cole R. et al. (1998). Intelligent Animated Agents for Interactive Language Learning, *STiLL: ESCA Workshop on Speech Technology in Language Learning*, May 1998, Stockholm, Sweden.
- [4] McTear M, Downey S, Behzadafshar H, McCarthy N and McLaughlin A (1997). Going Slurpping - Initial Experiences with the CSLU Rapid Prototyping Environment for Spoken Dialogue Systems (CSLUrp). *Informatics Research Reports* No. 12, Faculty of Informatics, University of Ulster, 13-22.
- [5] Greenan K and McTear M (1997). A Rapid Prototyping Approach to Spoken Dialogue System Development: The Directory Assistance Project, *Eighth Irish Conference on Artificial Intelligence (AI-97)*, Sep 1997 University of Ulster, Vol. 2: 160-167.
- [6] Serridge B, Cole R, Barbosa A, Munive N and Vargas A (1998). Creating a Mexican Spanish version of the CSLU toolkit, *Proc 5th International Conference on Spoken Language Processing*, Dec 1998, Sydney, Australia, 3225-3228.
- [7] Kaiser E, Johnston M and Heeman P (1999). PROFER: Predictive, Robust Finite-State Parsing for Spoken Language, *Proc ICASSP*, Mar 1999, Phoenix, Arizona.
- [8] McTear M (1998). Modelling spoken dialogues with state transition diagrams: experiences with the CSLU toolkit. *Proc 5th International Conference on Spoken Language Processing*, Dec 1998, Sydney, Australia, 1223-1226.
- [9] Heeman P, Johnston M, Denney J and Kaiser E (1998). Beyond structured dialogues: Factoring out grounding. *Proc 5th International Conference on Spoken Language Processing*, Dec 1998, Sydney, Australia, 863-866.