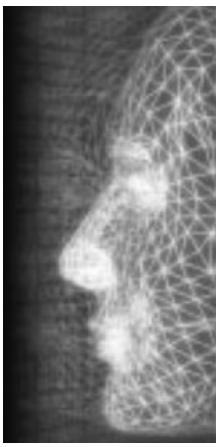
COMPUTER ANIMATION AND VIRTUAL WORLDS
*Comp. Anim. Virtual Worlds* 2008; **19**: 605–619
Published online 6 October 2008 in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/cav.265

# Language learning with interactive virtual agent scenarios and speech recognition: Lessons learned



*By James N. Anderson*, Nancie Davidson, Hazel Morton and Mervyn A. Jack*

***The SPELL (Spoken Electronic Language Learning) system is a self-access computer-assisted language learning (CALL) package that integrates speaker-independent continuous speech recognition technology with virtual worlds and embodied virtual agents to create an environment in which learners can converse in the target language within meaningful contextualized scenarios. In this paper we provide an overview of the functionality, architecture, and implementation of the SPELL system. We also describe four phases of usability evaluation conducted with the system and summarize the main results of these user assessments. Finally, we discuss the most significant lessons learned in the development and evaluation of the system. The paper focuses on the technological aspects of the system and its evaluation for usability and robustness, rather than its pedagogical methodology. Copyright © 2008 John Wiley & Sons, Ltd.***

*Received: 16 May 2008; Revised: 26 August 2008; Accepted: 28 August 2008*

KEY WORDS: computer-assisted language learning; speech recognition; virtual worlds; embodied virtual agents; usability evaluation

## Introduction

As desktop computer technology has increased in sophistication and fallen in price, the potential for computer-assisted language learning (CALL) applications has risen accordingly. CALL packages can focus on different aspects of language learning (vocabulary, pronunciation, grammar, etc.) and can employ a range of technologies and methodologies. Until recently, very few commercial CALL packages have attempted to simulate real-life conversational scenarios in a foreign language context using an integration of virtual worlds, animated human-like virtual agents, and automated speech recognition technology.

It has been shown that conversational interaction in the target language is important for language learners, particularly when that interaction involves confirmative or corrective feedback (whether explicit or implicit), since it encourages the learners to test and refine their utterances in a meaningful goal-driven participatory context.[1–6] A virtual world can offer a highly contextualized environment for language learning, where learners can either observe or participate in conversational scenarios without the costs incurred by equivalent real-life scenarios (e.g., field trips). Furthermore, users of virtual worlds can experience ''presence'', that is, the subjective sense of ''being there'' in the virtual world.[7–9] Presence increases users' engagement with the activities they conduct in the world and encourages them to behave in ways similar to the way they would behave in an equivalent real-world situation. Animated virtual agents are increasingly used in computer interfaces, and research has shown that adult users prefer applications with such agents to similar applications without them.[10–16] In particular, animated agents have been used in pedagogical applications to good effect.[17–21]

As for speech recognition technology in CALL applications, its value and relevance has been debated,

*Correspondence to: J. N. Anderson, Centre for Communication Interface Research, The University of Edinburgh, Alexander Graham Bell Building, King's Buildings, Edinburgh EH9 3JL, UK. E-mail: jad@ccir.ed.ac.uk

with critics focusing primarily on poor recognition rates for non-native speakers and the difficulties encountered in using such technology to improve learner pronunciation.[22–27] Nevertheless, it is universally recognized that in principle automated speech recognition offers tremendous potential for CALL packages, provided the technological limitations can be resolved or at least circumvented.

The SPELL (Spoken Electronic Language Learning) system has been designed and implemented with precisely these considerations in mind. SPELL is a self-access CALL package that integrates speaker-independent continuous speech recognition technology with virtual worlds and embodied virtual agents to create an environment in which learners can converse in the target language within meaningful contextualized scenarios. To our knowledge, the only comparable system to SPELL is the Tactical Language Training System developed at the Information Sciences Institute (University of Southern California), which has been designed to assist U.S. servicemen in the rapid acquisition of communicative competence in Arabic and other relevant languages.[28] As will be explained below, however, the two systems differ in a number of significant respects.

In this paper we provide an overview of the functionality, architecture, and implementation of the SPELL system. We also describe four phases of usability evaluation conducted with the system and summarize the main results of these user assessments. Finally, we discuss the most significant lessons learned in the development and evaluation of the system. The focus of the paper is on the technological aspects of the system and its evaluation for usability and robustness, rather than its pedagogical methodology (details of which have been published elsewhere).[29]

## The SPELL Application

At the heart of the SPELL system is an integrated language learning application which uses human-like virtual agents equipped with speech recognition and presented within a virtual world to simulate conversational scenarios, thus allowing language learners to observe and participate in spoken interactions with a view to accomplishing real-life goals such as ordering from a menu or buying a train ticket. The use of embodied virtual agents within a simulated 3D environment allows for a substantial degree of user involvement and presence, which can aid the learning process and increases users' engagement with the application.

*Figure 1. One-to-one scenario (''Kissaten De'' Japanese lesson).*

The SPELL application is designed for language learners at the beginner level. It makes use of three types of language-learning scenario: *observational* scenarios, *one-to-one* scenarios, and *interactive* scenarios. In an observational scenario, the user participates merely as a spectator, observing a spoken interaction between two or more virtual agents (e.g., two diners discussing menu choices and then giving their orders to a waiter). In a one-to-one scenario (Figure 1), the user participates in a question–answer dialog with a single virtual agent, with a view to consolidating what has been learned from the preceding observational scenario (e.g., the user is asked ''What food does John like?'' and ''What food do you like?''). In an interactive scenario (Figure 2), the user enters fully into the context, viewing the scene from an immersive first-person perspective and participating in a spoken interaction with two or more virtual agents (e.g., the user takes the role of one of the diners, discussing her menu choices with the other diner, and then giving her final order to the waiter).[1]

A single lesson typically consists of one observational scenario, two or three one-to-one scenarios (covering different conversational elements), and one interactive scenario. One of the agents plays the role of a ''friend'' who features in all the lesson scenarios with a view to establishing continuity and relationship with the user. In the observational scenario, several agents (one of whom is the ''friend'') illustrate the use of key phrases

[1]Video clips illustrating the three types of scenario are available on the SPELL website (http://www.ccir.ed.ac.uk/SPELL).



DOI: 10.1002/cav


*Figure 2. Interactive scenario (''At the Café'' English lesson).*

and constructions. In the one-to-one and interactive scenarios, the agents direct the learning process by inviting the user to speak and providing informative feedback. (In the one-to-one scenarios, the ''friend'' agent does this alone; in the interactive scenario, other agents may also do this.) Where appropriate, the agents assist the user by reformulating questions and recasting the user's own utterances so as to implicitly correct any grammatical errors. The only language spoken in all three scenario types is the target language.

A typical learning session using the SPELL application proceeds as follows. On starting the application, the user first selects their native language (L1) and enters their username and password (used for logging user-specific information such as learning progress and spoken error rates). The user then selects the target language (L2). From this point on, all application text, button labels, etc., are presented in L2, although L1 translations are available via tool-tips. The user is next presented with a menu of available lessons, where each lesson focuses on a particular real-world situation (e.g., ''At the café'' and ''At the station''). After choosing a lesson, the user is invited to choose a language learning scenario (e.g., ''Watch and listen'' for the observational scenario, ''About drinks'' for the one-to-one scenarios, and ''Go to the café'' for the interactive scenario). Representative thumbnail snapshots of the virtual world are used in the lesson menus and scenario menus, to aid user comprehension.

The user normally begins with the observational scenario, before proceeding through each of the one-to-one scenarios, and concluding with the interactive scenario. After each scenario (except the last), the user is invited to proceed directly to the next logical scenario in the lesson. However, users have complete control over which scenarios to access and when. They can pause a scenario at any point and then resume it. They can stop a scenario and either restart it from the beginning or start a new scenario. They can switch subtitles on or off at any time (subtitles are displayed in L2 and deactivated by default). A number of supplementary resources are also available at all times from a right-hand button menu. These resources appear in the main application window as pages of formatted text. They include a vocabulary list, grammatical information, cultural information, and a complete transcription of the observational scenario. The resource pages are presented in L2 by default, but L1 translations can be accessed via an icon at the top of each page. The right-hand button menu is hidden when the scenario is playing (to maximize screen usage for the 3D view) and reappears when the scenario is paused or stopped (Figure 3). The menu provides a ''back'' button, to take the user to the previously accessed scenario or resource, as well as buttons to return the user to the scenario menu or the lesson menu.

Since the agents use recorded natural speech (see later section on content creation) it might be asked why the system uses a virtual world with animated agents rather than video clips with human actors. There are several reasons for this design choice. First, the language-learning scenarios are designed to allow the learners alternative paths through their interactions with the agents; thus the agents' reactions, both verbal and physical, may differ depending on the response from the learner. Using animated agents allows the necessary flexibility in displaying such reactions. If videos clips were used, multiple scenes would need to be recorded

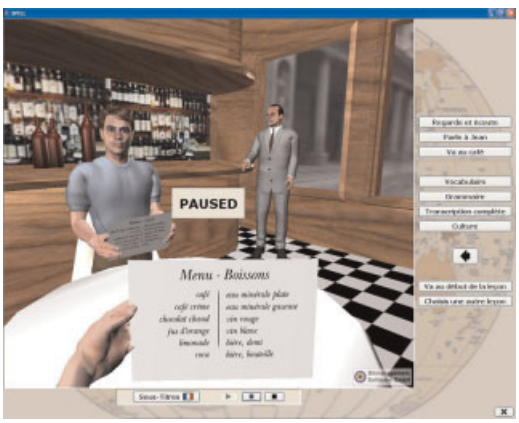



*Figure 3. Paused scenario showing right-hand button menu (''Au Café'' French lesson).*



DOI: 10.1002/cav

and stored so that agents could react dynamically to the learner's input, and minor changes to the ''screenplay'' would require entire clips to be re-recorded. Secondly, since the SPELL system offers lessons in multiple languages (e.g., café lesson in English, French, Italian, and Japanese), development time can be minimized by re-using the agent animations. Finally, the use of a virtual world can induce a more marked sense of ''presence''; in the café scenarios, for example, the learner can see his/her own virtual hand reaching out to take the menu (Figure 2).

At this point it will be appropriate to delineate the differences between the SPELL system and the Tactical Language Training System.[28] SPELL is designed to be used as a classroom tool for standard language courses in schools and colleges. As such, the application incorporates supplementary resources (vocabulary list, cultural information, etc.) and provides the student with access to them throughout the teaching sessions. SPELL focuses on ''everyday'' conversational scenarios, rather than specialized contexts such as military scenarios, and does not train in the use of gestures and other non-verbal communication. A distinctive three-stage methodology (observational, one-to-one, and interactive scenarios) is employed that allows beginners to ''ease into'' a conversation in a foreign language. Finally, and most significantly, the SPELL system uses a virtual tutor, playing the role of a ''friend'', who guides the student through the conversational scenarios and provides implicit feedback on the student's utterances.

## Software Implementation

We will now briefly describe the technology used to implement the SPELL system.

The main SPELL application is coded in Java. It is compiled and run using Sun's Java Platform Standard Edition on Windows XP. The virtual world is displayed within the application using an ActiveX component (Bitmanagement's BS Contact VRML/X3D Player).[30] All audio within the application is rendered using the Java Sound API.

The speech recognition and natural language interpretation is implemented using technology developed by Nuance Communications, which includes a Java API for application development. The speech recognition engine employs acoustic models (which are trained on data obtained from a large sample of native speakers), a pronunciation dictionary, and a language model in the form of one or more user-defined recognition ''grammars''. A grammar is a syntactical definition that specifies which utterances can be recognized when the grammar is ''active''. Grammars are defined with a typical Backus-Naur formalism (i.e., one grammar can be defined in terms of other grammars, but must ultimately resolve into a string of terminals and operators) and may also ascribe semantic tags for natural language interpretation. A sample grammar is provided with commentary in Appendix A.

The supplementary resources (vocabulary pages, etc.) are coded in XHTML and rendered within the SPELL application using standard Java GUI components. The agents' avatars (i.e., their graphical representations within the virtual environment) are coded in VRML 2.0 and conform to the H-Anim 1.1 specification for humanoid models.[31] The scenes for the lessons (e.g., café, train station) are also specified in VRML 2.0 format, as are the ''props'' manipulated by the agents (e.g., menus, wine glasses, tickets, money). All other application content is specified in XML format (e.g., lesson descriptions, scenario descriptions, agent dialogs).

The SPELL system also includes a number of custom tools used for creating content for the main application, all of which have been coded in Java. The main SPELL application and lesson development tools can run comfortably on a modest desktop PC platform (e.g., 3.2 GHz Pentium 4, 1024 MB RAM, mid-range graphics card with 3D acceleration).

## System Architecture

The SPELL application is based on the ARMADA system (Adaptable Real-time Multiple Agent Dialogue Architecture) to execute its conversational scenarios. ARMADA has the following features:

- *Agent independence*: virtual agents run independently of one another, each with its own processing thread and dialog manager.
- *Event-driven execution*: the actions of agents are executed in response to events that occur within the virtual environment (such as the user speaking or an agent signaling to another agent).
- *Modular architecture*: agents are comprised of a set of discrete modules, centered on a dialog manager module that determines how the agent should respond to specific events within its environment.
- *Extensibility*: the characteristics and capabilities of agents can be adapted and extended to meet the demands of the application.



DOI: 10.1002/cav

- *Observational or interactive operation*: scenarios can be constructed that feature agents conversing only with one another or interacting with a human user.
- *Versatile high-level dialog language*: the dialog scripts for agents are written in a high-level JavaScript-like programming language that allows complex dialogs to be implemented with relative ease and efficiency.

An ARMADA-based virtual agent is comprised of a dialog manager module (the agent's ''brain''), which determines the response of the agent to events within a scenario, in conjunction with a number of auxiliary modules that determine (i) the range of events to which the agent can respond, and (ii) the capabilities of the agent in response to events. These auxiliary modules implement the agent's sensory and motional faculties (the agent's ''eyes'', ''ears'', ''hands'', ''feet'', etc.).

Each auxiliary module is associated with a set of *events* (or more precisely, *event types*), a set of *functions*, and a set of *actions*. *Events* are generated by the auxiliary module and passed to the dialog manager for handling. *Functions* are implemented by the auxiliary module and accessed by the dialog manager in order to obtain or manipulate data relevant to that module. *Actions* are implemented by the auxiliary module and executed by the dialog manager in order to carry out the agent's response to events. In the SPELL application, for example, an agent's speech recognition module generates an *event* when spoken input is received from the user; it implements a *function* to return the interpreted content of the input; and it implements an *action* to activate a new recognition grammar for future spoken input.

Some auxiliary modules are specific to an agent (e.g., its animation module) while others are shared among agents (e.g., a scene manager handling props that can be manipulated either by agents or by the user). Whether a particular module is shared depends on the function of the module along with any implementation constraints that apply (e.g., only one audio input channel available for speech recognition). In some cases, where issues of efficiency or synchronization arise, auxiliary modules can be designed to interact directly with one another (e.g., the animation and audio modules, for lip synchronization and other facial animations).

Agents within the SPELL application have the following auxiliary modules in addition to the dialog manager module (see Figure 4):

- *Animation Manager*: handles gestures, facial expressions, lip movements, etc.
- *Audio Manager*: handles speech prompt playback.


*Figure 4. ARMADA-based SPELL agent.*

- *Database Manager* (shared): handles general information storage and retrieval (e.g., user progress, subtitles for prompts).
- *Environment Manager* (shared): provides access to runtime parameters (e.g., user name).
- *Message Manager* (shared): handles ''back-channel'' communication and co-ordination between agents.
- *Navigation Manager*: handles movement between locations within the virtual scene.
- *Scene Manager* (shared): handles manipulation of props within the virtual scene.
- *Speech Recognition Manager* (shared): handles speech input from user and natural language interpretation.
- *Speech Synthesis Manager*: generates synthesized speech prompts (used for initial dialog design and debugging).
- *Subtitle Manager* (shared): handles display of subtitles.

We will now explain the architecture of the dialog manager. In essence, an agent's dialog manager executes a finite state machine, the states and transitions of which are defined by a dialog script. The dialog script is loaded by the dialog manager when the agent is created and initialized for a particular scenario. This script specifies a set of *states*, a set of *conditions*, and a set of *results*. A unique ID number is assigned to each state, condition, and result. Every state specified by a dialog script has associated with it a text description (e.g., ''waiting for customer to order a drink'') and a *condition–result pair list*. The condition–result pair list is an ordered list (possibly empty) of *condition–result pairs*, where a



DOI: 10.1002/cav

condition–result pair is an ordered pair of ID numbers that identify, respectively, a condition and a result (both of which must be specified elsewhere in the dialog script). A typical condition–result pair list might look like this:

```
[101,101][102,102](110,120)
```

The meaning of a condition–result pair amounts to this: *if the condition is fulfilled then the result should be executed*. A condition–result pair can be either *blocking* (denoted by square brackets surrounding the ID numbers, e.g., `[101,101]`) or *non-blocking* (denoted by round brackets surrounding the ID numbers, e.g., `(101,101)`). A blocking pair prevents the evaluation of any subsequent condition–result pairs in the list, where as a non-blocking pair does not.

The conditions and results specified in a dialog script are expressed in a high-level JavaScript-like programming language, which supports the evaluation of complex Boolean, numerical, and string expressions, along with global variable assignment and access. A condition must take the form of a Boolean expression, i.e., an expression that evaluates to *true* or *false*. A result is essentially a series of *actions* to be performed, but may also involve conditional flow structures (such as the familiar *if-else* structures and *while* loops supported by other programming languages).

Both conditions and results may access the *functions* implemented by auxiliary modules, but only results may access the *actions* implemented by those modules. In addition, the dialog manager supplies a number of *built-in* functions and actions, to support data manipulation (e.g., string-to-integer conversion) and dialog execution handling (e.g., changing the current dialog state).

When the dialog manager receives an event (from one of the auxiliary modules), it proceeds as follows:

1. Obtain (from the dialog script) the condition–result pair list for the current dialog state.
2. Take the first (or next) condition–result pair in the list, that is, an ordered pair of ID numbers $\langle C,R \rangle$. (If there is no such pair, then finish event handling.)
3. Evaluate the condition with ID number $C$. If the condition evaluates to *true*, then execute the result with ID number $R$.
4. If the condition evaluated to *true*, and the condition–result pair is a *non-blocking* pair, then loop to 2. Otherwise, finish event handling.

The result with ID number 0 has special significance, since it is reserved for any initialization that the agent needs to perform (e.g., settings its location in the scene, loading animation scripts, loading recognition grammars). This result is executed immediately when an agent's dialog manager is signaled to start running its dialog script.

The dialog manager also extends the structure of a finite state machine with the concept of a *superstate*. Superstates are defined in exactly the same way as states, each with its own condition–result pair list, but transitions from one superstate to another typically occur far less frequently (if at all) in an agent's execution of its dialog. At any point in the dialog execution, the agent can be in one state and also one superstate; thus, in effect, the dialog manager runs two finite state machines rather than one. Superstates allow for easy handling of top-level events that can occur throughout the dialog (or throughout a particular stage in the dialog) where those events should be handled in the same way regardless of the agent's current state; for example, in a scenario where the user is able to say ''stop'' or ''help'' at any time. Without the superstate feature, the same condition–result pairs would need to be assigned to all of the states in the dialog in order to handle these top-level events, but with this feature the specification of the dialogs can often be simplified, leaving less room for omissions and redundancies when adding new states or editing existing ones.

## Content Creation

We will now give an overview of the software tools used to create content for the SPELL application.

The agents' dialog scripts are written using a custom-built *Dialog Editor* application. At any one time, the application displays the information for one state (ID number, description, and condition–result pair list), one condition (ID number and condition code), and one result (ID number and result code). The editor has simple text-editing features (undo, search, replace) and allows dialog code to be parsed for syntactical correctness. The dialog scripts are stored as text files in XML format.

The agents' avatars are designed using Curious Labs' *Poser* software. This software allows the physical characteristics of the human models to be customized (body proportions, facial features, skin color, hair style, etc.) as well as clothing. The models are exported from *Poser* in VRML 2.0 format. The models are then ''tweaked'' for efficiency (e.g., by removing hidden faces) using a custom-built 3D editor, which exports the



DOI: 10.1002/cav

finished models to H-Anim 1.1 format with additional joint-skinning information. (The avatars' joints are animated using a custom ''skinning'' algorithm.) The body animations for the agents are created using a custom-built key-frame-based animation editor and the animations are stored as text files using a custom scripting language.

The audio prompts for the agents are recorded from human speakers using a standard audio editor. The SPELL system uses recorded natural speech because, although more costly and time-consuming to create, it offers the most realistic guide in terms of native speaker pronunciation and intonation for language learners. Synthesized speech is a more flexible solution for audio output, but its naturalness is limited and while usable for native speaker applications it would not be the optimum solution for an application for non-native speaker language learners. As noted earlier, however, the SPELL system does include support for synthesized speech, which can be used for initial dialog design and debugging. The recordings are only made once the final wording of the prompts has been established.

The mouth animations for lip synchronization are generated from the audio prompts using a custom-built tool which inputs the audio to an English-language speech recognition engine (loaded with a dictation grammar) and maps the recognized phonemes to corresponding visemes (i.e., facial poses). Mouth animation scripts for each set of prompts (including the prompts for non-English lessons) are created off-line in this fashion. Although the recognition engine is only moderately accurate (i.e., it often produces inaccurate transcriptions) the resultant mouth animations are typically very similar to those that *would* have been generated if the input *had* been perfectly recognized, because the recognized sentences sound similar to the actual spoken sentences (and similar sounds are mapped to similar or identical visemes). Even though the transcriptions of the prompts are readily available, the engine is loaded with a dictation grammar rather than a rule-based grammar defined to recognize the exact transcription of a prompt (and only that transcription) for two reasons: first, using a rule-based grammar would only be suitable for English-language prompts; and second, we have found that using a rule-based grammar more often than not results in a rejection by the engine rather than a perfect recognition. (The lip-sync tool uses the U.S. English dictation engine included in the Microsoft Speech SDK v5.1.) The mouth animations scripts generated in this way are augmented with other appropriate facial animations (blinking, smiling, frowning, eyebrow-raising, etc.) to indicate emotions that correlate with the content of the audio prompts. Similarly, the body animations are crafted to include gesticulations that naturally support the speech of the virtual characters.

The virtual scenes (café, train station, etc.) and props (menus, tickets, etc.) are designed using Newtek's *LightWave 3D* and exported to VRML 2.0 format. The image textures for the scenes are created using Adobe's *Photoshop*.

The speech recognition grammars, lesson specifications, scenario specifications, and supplementary resources are all created and edited using a standard text editor.

## User Assessments

The SPELL application and content has been evaluated for robustness, usability, and effectiveness through four distinct phases of user assessments, the procedure and results of which are described below. Due to logistical constraints, the evaluations focused on the performance of the system and immediate user responses to it, rather than long-term effects on learning.

### Evaluation Phase I

A fully functional prototype was used and assessed by two professional language teachers who were native speakers of the initial target languages (Italian and Japanese). The application interface and the lesson content were then improved on the basis of feedback from these teachers.

### Evaluation Phase 2

52 high-school language students (34 students of Italian and 18 students of Japanese), aged between 13 and 17, from five different schools in Scotland, were invited to use the SPELL application in a classroom context. 26 of the students had been learning their target language for less than a year; 12 had been learning for 1 year, 11 for 2 years, and 3 for over 2 years. The participants tried a SPELL lesson concerned with ordering food and drink from a menu in a restaurant. The evaluation sought to investigate various aspects of the SPELL application:

- the usability of the interface design
- user attitudes to using the application



DOI: 10.1002/cav

- user attitudes to interacting with the characters in the lesson
- user behavior in accessing the supplementary resources
- user responses types (e.g., single word or sentence)
- the speech recognition accuracy of the system

This evaluation was performed using a combination of objective measures (log files and transcriptions of spoken user input) and subjective measures (user attitude questionnaires and interviews). The questionnaires used a seven-point Likert scale[32] to investigate user attitudes toward the SPELL application in general and toward interactions with the virtual characters in the lessons more specifically.

The most significant results of this second phase of evaluation were as follows. The user attitude questionnaires revealed high levels of engagement and enjoyment with using the application, and strong inclinations to use it again. The speech recognition feature of the system was considered particularly appealing.

The recognition accuracy results showed that *word-for-word* recognition was not robust enough to reliably detect errors by the language learners (e.g., grammatical mistakes). The overall word-for-word accuracy was 56.4% for the Italian group and 72.4% for the Japanese group. These results were due in large part to the fact that the acoustic models employed in the speech recognition system were generated from native speakers (models generated from non-native speakers are currently unavailable). Nevertheless, this shortcoming did not detract from users' experiences for two reasons. First, the recognition errors tended to be masked by the design of the agent dialogs; in most cases either the recognized sentence was semantically equivalent to the spoken sentence, resulting in the same response from the virtual tutor, or the utterance was rejected altogether, resulting in the virtual tutor simply repeating the question. Second, the impact of the errors was outweighed by the users' high degree of engagement with the system.

## Evaluation Phase 3

A similar user experiment was conducted in the same five schools in Scotland, this time with 41 students (24 students of Italian and 17 students of Japanese). Most of the students had been learning their target language for less than a year; five had been learning for 1-2 years, three for 2–3 years, and two for over 3 years. In this phase, the participants tried a SPELL lesson set at a railway station, which involved tasks such as inquiring about train times and ordering tickets. The evaluation in this user assessment focused primarily on lesson completion rates, user attitudes, and speech recognition accuracy. Once again, the questionnaire results showed very positive user attitudes toward the SPELL application, with participants commenting especially on its immersive nature, its uniqueness in comparison to other language learning tools, and the level of interactivity involved in speaking to the agents. In particular, 97% of the students felt that the system was a useful learning tool, remarking that the application gave them valuable opportunity to practice their language skills and thus helped to motivate them in the overall learning process. The findings regarding recognition accuracy were comparable to those of the second phase.

## Evaluation Phase 4

The final evaluation phase was designed to investigate cross-cultural differences in the use of the SPELL application. Two user groups were involved in the research: students of French in Scotland and students of English as a Foreign Language (EFL) in Beijing, China. The first group consisted of 28 students, who had been studying French for an average 4.7 years. The second group consisted of 48 students, who had been studying English for an average 6.8 years. Both groups used versions of the SPELL lesson used in the third phase (Figure 5).

In this phase the evaluation focused not only on user attitudes (assessed via usability questionnaires and verbal interviews) and speech recognition accuracy, but

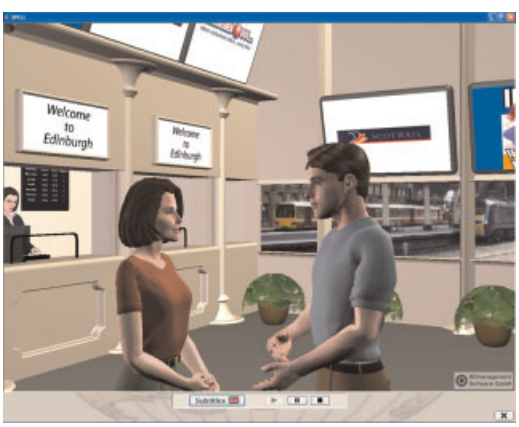



*Figure 5. Observational scenario (''At the Station'' English lesson).*



DOI: 10.1002/cav

also on user motivation. Using Likert-style questionnaires designed to assess various types of motivation, the students' attitudes toward the task of language learning were measured before and after using the SPELL application.

Overall, both groups of students found using the application enjoyable, engaging, and useful. They particularly valued the ability to converse with virtual characters, with many remarking that it made them more relaxed than when interacting with a human teacher. In the EFL group, however, a majority of students indicated that the level of the lesson was too easy for them. This is not wholly surprising given the length of time these students had already been studying English. With respect to motivation, it was found that the use of the SPELL system increased the motivation levels of the EFL group but did not do so for the French group. It was also found, however, that the French group were more motivated by external rewards (e.g., getting a good grade) and the EFL group more by intrinsic rewards (e.g., a personal sense of satisfaction and accomplishment). This suggests that the SPELL system has more potential to boost motivation among intrinsically motivated students than those who are externally motivated.

Even though this fourth phase involved native Chinese speakers rather than English speakers, the recognition accuracy results were comparable to those of the previous two evaluations. As before, it was found that the recognition errors did not significantly detriment the users' experiences.

Further details and discussion of the user attitude measurements and speech recognition analyses for these system evaluations are available elsewhere.[33–35]

# Lessons Learned

We will now discuss some of the lessons learned from the implementation and evaluation of the SPELL language learning application. As noted previously, the focus here will be on architecture and technical implementation of the system, rather than the language learning methodology or lesson content.

## System Architecture

The ARMADA system, with its finite state machine model for agent interactions, proved to be more than adequate for this application. This is primarily because the flow of the agent dialogs needed to be linear and relatively constrained for the purposes of language learning (at least at the beginner/intermediate level). In the observational scenarios, the course of the interaction was entirely pre-determined, since there was no user interaction. In the one-to-one scenarios and interactive scenarios, the user was expected to proceed through an orderly series of agent questions. The highly contextualized conversational forms featured in the SPELL application would contrast with, for example, an ''open conversation'' with no specified goal or path. The ARMADA system is capable of supporting more complex, non-linear forms of conversational interaction, but this would require more sophisticated agent dialog scripts. In the context of the SPELL project, the event-driven condition–result structure of the agent dialogs made it relatively straightforward to implement the scenarios (even for developers with no previous experience of programming).

## Single Agents Versus Multiple Agents

Writing agent dialog scripts for single-agent (i.e., one-to-one) scenarios proved to be straightforward. Substantially more thought and time was required to write the scripts for multiple-agent (i.e., observational and interactive) scenarios, because of the need to coordinate the actions and responses of the agents with one another.

The main means of coordinating agents was by messaging. For example, when one agent had finished playing the audio prompt of a question to another agent, it would send a message to the other agent to signal that the other agent should now play the audio prompt of its answer. (These inter-agent messages were invisible to the user, of course.) This messaging needed to be explicitly written into the agent dialog scripts. The most difficult cases arose in the interactive scenarios, due to the need for both agents to be appropriately responsive to user input and to coordinate their actions in natural ways (e.g., when the user is unsuccessful in communicating with the waiter in a café, the ''friend'' agent has to ''step in'' to the conversation and assist the user by temporarily responding on their behalf, before ''stepping out'' of the conversation again).



DOI: 10.1002/cav

One notable challenge involved the handling of props by agents, particularly in cases where one agent would hand a prop to another agent (e.g., a waiter character handing a menu to a customer character). This required the agent animations to be accurately time-synchronized, given that each agent would be autonomously executing its own dialog. The agent animations were ''statically'' scripted in the sense that one agent would move its hand (with prop) to a predetermined location with the expectation that another agent would have moved its hand to the same location at the same time in order to receive the prop. This demanded careful coordination between the agents. One option for eliminating these difficulties would be to extend the agents' animation modules to allow for *dynamic* animations, e.g., for an agent to move its hand to the current location of a prop held by another agent, *whatever location that may be*. This would require the use of inverse kinetic algorithms.

The difficulty of coordinating multiple agents arose in significant part from the approach taken to writing their dialog scripts. Each script was written individually, in a separate instance of the editor application. Moreover, the scripts were programmed at a relatively low level, by directly coding the dialog states, conditions, and results. One approach less liable to lead to confusion would be to use an integrated dialog editor that allowed the scripts for multiple agents to be coded simultaneously, with a GUI that enabled developers to visualize the agent dialogs in a parallel fashion (e.g., as discrete flow diagrams with interconnections to indicate message-passing, hand-shaking, or some other form of agent coordination). An integrated editor such as this might allow agent dialogs to be coded at a higher level (e.g., graphically as flow diagrams) and compiled into low-level dialog scripts. Another useful feature would be to offer simulations of the agent scenario without the full graphical front-end of the main SPELL application, perhaps also providing standard debugging features (breakpoints, step-by-step execution, etc.). In any event, we concluded that a considerable amount of coding time could be saved by introducing a more sophisticated development environment that facilitates visualization and implementation of dialogs for multiple agent scenarios.

## Agent Animations

The implementation of agent animations proved to be one of the most time-consuming aspects of lesson creation. As described previously, the animations were created with a custom tool called *Avatar Poser*, which involved manipulating the joints of avatars into ''poses'' and assembling animation sequences using ''poses'' as key frames. This tool supported only forward kinematics: in order to position an avatar's hand on a table, for example, the rotation of each of the joints of the avatar's arm (shoulder, elbow, and wrist) had to be individually adjusted. This method of avatar manipulation takes time and skill, without which it results in unnatural-looking animations. A more efficient approach would be to extend the animation tool to support inverse kinematics. In this way, the extremities of an avatar can be dragged directly into the required position and the joint rotations adjusted accordingly (even observing the natural rotation limits of human joints). Another time-saving approach would be to provide a library of common poses, accessible from within the animation tool, which could be imported, ''tweaked'' as necessary, and assembled into animations.

It was also observed that *animation re-use* could have been better planned when creating similar lessons in multiple languages. For example, in the original versions of the ''At the café'' lesson, the physical layout of the scenes (in terms of furniture shape, size, location, and orientation) was markedly different between the Italian and Japanese versions of the lesson. This meant that separate agent animations for handling items such as menus and glasses had to be created for each set of scenarios. Furthermore, since the Italian and Japanese avatars had different bodily proportions, animations designed for one avatar could not be directly re-used by another (e.g., separate animations had to be created for the Italian and Japanese customer characters, even though many of the animations were of the same type).

It is therefore clear that considerable time and effort could be saved by planning for maximal re-use of animations between different language versions. This would involve designing the scenes and avatars so that as many animations as possible can be used across versions. For example, scenes would have furniture and props with similar locations and proportions. Likewise, avatars would have exactly the same bodily proportions despite differing appearances (clothing, skin tone, ethnic facial characteristics, etc.). Although some animations would inevitably have to be designed for particular language versions (e.g., culturally appropriate forms of greeting) with sufficient planning a large proportion could be re-used across versions. Indeed, these obser-



DOI: 10.1002/cav

vations suggest that the optimal strategy would be to design prototypical scenes that could be used across all languages and then compose language-specific dialogs to fit these scenes.

## Graphical Quality and "Presence"

The graphical quality of the virtual agents and scenes, which can best be described as semi-realistic, was found to be acceptable for the purposes of the application. The graphics were not photorealistic in quality, but neither were they cartoon-like. The level of detail was not so high that the animation and rendering drew substantial processing power away from the speech recognition engine, but neither was it so low that it distracted users and interfered with their engagement with the application. Comments from users regarding the graphics ranged from ''as real to life as possible'', ''very good'', and ''quite realistic'', to ''not too bad'' and ''a bit fake''. We have speculated that users' attitudes were shaped partly by expectations based on their exposure to state-of-the-art computer game consoles. In the educational setting, however, the SPELL system also benefited from comparisons with other teaching tools that are far less sophisticated.

In the third phase of evaluation, a number of users made comments indicating that they had experienced a significant degree of physical and social presence when interacting with the virtual characters:

- ''It's realistic. As close to being in the situation as possible.''
- ''Felt like a real life interaction.''
- ''It felt very personal.''
- ''Interacting with the characters feels very personal.''
- ''Almost felt as if I was there.''
- ''Looks realistic. Feels like I was there.''
- ''In the interactive it felt like I was walking to the counter.''
- ''Everything is there. It feels real.''

In a minority of cases, the sense of presence functioned negatively; for example, one student remarked that he felt under pressure whenever the virtual tutor turned to look at him.

In light of the user responses to the SPELL application, we concluded that the graphical detail was pitched at a level appropriate for the purposes of the application and for the technology with which it was implemented.

## Lip Synchronization

The method used to generate lip-sync animations, as described above, was relatively crude in its approach. However, our own observations suggested that the visual results were surprisingly good for the modest amount of effort required to produce them. Although participants in the user assessments were not asked about it directly, none of the users commented critically about any aspect of the mouth animations. It is plausible to suppose that inaccuracies in the lip-sync animations were less noticeable to the users because they were not native language speakers. Nevertheless, it seems that the animations were not so unnatural that they distracted, irritated, or confused the users. We therefore concluded that, for a language-learning application of this kind, the method adopted was an effective and time-efficient means of generating lip-sync animations for a large number of audio prompts. The accuracy of lip-sync could be improved by using same-language recognition engines, where those recognition engines are available and affordable, without increasing the time required to generate the animations.

## Speech Recognition Technology

As we have noted, the acoustic models used by the speech recognition engine in the SPELL application were trained with *native* speakers, but the nature of the application required it to recognize utterances from *non-native* speakers. Inevitably this meant that the numbers of rejections and misrecognitions were higher than would normally be expected for a leading technology recognition engine.

User utterances are categorized as either *in-grammar* or *out-of-grammar* depending on whether or not the response has been predicted by the developers and therefore specified in the recognition grammar. Out-of-grammar responses cannot be recognized and are rejected by the recognizer, although false acceptance of such input can occur (usually when the out-of-grammar utterance is very similar to an allowable phrase). Although the grammars for the SPELL lessons were designed to pick up grammatical errors commonly made by learners of the relevant language (i.e., the recognition grammars allowed for some linguistically ungrammatical constructions), there was still a significant proportion of out-of-grammar utterances (anything from 5 to 75% of utterances, depending on the length and complexity of the response required from the user at



DOI: 10.1002/cav

a particular point in the scenario). Out-of-grammar utterances are commonly encountered by speech-input applications designed for native speakers due to normal speech disfluencies: repetitions, pauses, coughs, ''ums'', and so forth. Because the users of the SPELL application were learners, the proportion of out-of-grammar utterances encountered was considerably higher than it would otherwise have been. Nevertheless, around 25% of these out-of-grammar utterances were accepted and interpreted with the user's intended meaning.

In-grammar utterances can be recognized word-for-word (the ideal result), recognized with the same semantic value (i.e., the user's intended meaning), misrecognized (i.e., recognized but with a different semantic value), or rejected as unrecognized. In the three evaluation phases with real students, *accurate word-for-word recognition* occurred for between 50% (at worst) and 75% (at best) of in-grammar utterances, and *semantically equivalent recognition* occurred for between 65% (at worst) and 80% (at best). The word-for-word recognition rates are not high enough to support precise analysis of learner errors within the SPELL application. Nevertheless, the recognition rates are adequate to support realistic, intelligible conversations between the users and the virtual characters, and to enable the agents to identify and respond appropriately to some of the most common grammatical errors committed by language learners. Moreover, in those cases where in-grammar utterances were not recognized either word-for-word or with semantic equivalence, the majority are *rejected* rather than *misrecognized*. In such cases, the virtual tutor simply repeats or reformulates the question put to the user (rather than giving a response based on something the user did not actually say) and so it is less obvious to the user that the application has performed sub-optimally. One could argue that a non-native speaker using an application to practice speaking a foreign language is less likely to be frustrated by this outcome than a native-speaker using an application for some other purpose (e.g., an automated banking service).

Despite shortcomings in the recognition accuracy, the results of the user assessments are promising for applications of this kind, not least because the recognition technology employed was not designed to cater for non-native speakers. Several strategies could be applied for improving recognition rates in future versions of the SPELL application. One option would be to use acoustic models trained with speech from non-native speakers. Another would be to adopt newer techniques aimed at improving recognition results for non-native speakers.[36–38] Since it was observed by researchers that a significant proportion of misrecognitions or rejections were due to mispronunciations, a third option would be to use a customized dictionary that incorporates the most common mispronunciations made by language learners. (This would have the additional advantage of allowing for some pronunciation errors, as well as grammatical errors, to be detected and corrected by the virtual agents.) A fourth option would be to use less ''open'' grammars, by removing some of the less common learner utterances. Since this would reduce the number of possible recognition paths, it would have the effect of boosting recognition rates for the remaining utterances. These options are not mutually exclusive and could be used in combination to greatest effect.

## Conclusion

The SPELL system is, to our knowledge, the first instance of a user-tested CALL package designed for classroom use which simulates real-life everyday conversational scenarios that a learner may encounter in a foreign language context using an integration of virtual worlds, animated human-like virtual agents, and automated speech recognition technology. We have shown that such a system, which is technologically feasible as a commercial product designed to run on a typical desktop PC platform, can support high levels of user acceptability and engagement. A number of lessons were learned in the development of the system, which we have documented in this paper. In particular, we have noted how many of the time-consuming aspects of content creation and some of the technological limitations of the current system could be mitigated by careful planning at the content design stage.

## Appendix A

The following is a simplified sample of a grammar taken from the one-to-one scenario in the EFL version of the ''At the café'' lesson.

The grammar is designed to capture the learner's answer to the question, ''What food does Katie like?'' The top-level grammar is divided into two sub-grammars, one for grammatically correct utterances (''LikeOK'') and one for grammatically incorrect utterances (''LikeError''). These sub-grammars refer in turn to another grammar (''FoodList'', defined elsewhere)



DOI: 10.1002/cav

```
.Like  [
      LikeOK
      LikeError
]
LikeOK [
      ([katie she] likes FoodList:f)
]     {<food $f>}
LikeError  [
      (?it's the FoodList:f)                {<ErrorType1 SET>}
      ([katie she] like FoodList:f)         {<ErrorType2 SET>}
      ([katie she] is liking FoodList:f)    {<ErrorType3 SET>}
]     {<food $f><command recast>}
```

that matches any of the items on the menu in the scenario. The menu item uttered by the learner is returned in the semantic tag ''food'' (so that the agent can determine whether to express agreement or disagreement in its response). The sub-grammar ''LikeError'' is designed to recognize three common types of grammatical error for learners of English: (1) article insertion for uncountable nouns; (2) omission of present tense third singular (-s); (3) present progressive (-ing) used for present tense. The frequencies of these error types are logged by the SPELL system and can be used in a number of ways to provide feedback to learners. Utterances matching the ''LikeError'' grammar are also tagged as requiring the agent to recast the learner's utterance in its response, as in the following sample dialog:

```
Agent: What food does Katie like?
Learner: Katie like pizza.
Agent: That's right. Katie likes pizza. What food do you like?
```

**ACKNOWLEDGEMENTS**

The SPELL project was made possible due to financial support from the Scottish Enterprise Proof of Concept Fund. We also note the generous support of Nuance Communications Inc. in this work.

## References

1. Long MH. The role of the linguistic environment in second language acquisition. In *Handbook of Second Language Acquisition*, Richie WC, Bhatia TK (eds). Academic Press: New York, 1996; 413–468.
2. Long MH, Inagaki S, Ortega L. The role of implicit negative feedback in SLA: models and recasts in Japanese and Spanish. *The Modern Language Journal* 1998; **82**(3): 357–371.
3. Mackey A, Philp J. Conversational interaction and second language development: recasts, responses, and red herrings? *The Modern Language Journal* 1998; **82**(3): 338–356.
4. Mackey A. Input, interaction, and second language development: an empirical study of question formation in ESL. *Studies in Second Language Acquisition* 1999; **21**: 557–587.
5. Swain M. Communicative competence: some roles of comprehensible input and comprehensible output in its development. In *Input in Second Language Acquisition*, Gass SM, Madden C (eds). Newbury House Press: Rowley, MA, 1985; 235–253.
6. Swain M. Three functions of output in second language learning. In *Principle & Practice in Applied Linguistics: Studies in Honour of H.G. Widdowson*, Cook G, Seidlhofer B (eds). Oxford University Press: Oxford, 1995; 125–144.
7. Slater M, Usoh M, Steed A. Depth of presence in virtual environments. *Presence: Teleoperators and Virtual Environments* 1994; **3**(2): 130–144.
8. Lombard M, Ditton TB. At the heart of it all: the concept of presence. *Journal of Computer-Mediated Communication* 1997; **3**(2). http://jcmc.indiana.edu/vol3/issue2/lombard.html [accessed on 22 September 2008].
9. IJsselsteijn WA, Riva G. Being there: the experience of presence in mediated environments. In *Being There: Concepts, Effects and Measurement of User Presence in Synthetic Environments*, Riva G, Davide F, IJsselsteijn WA, (eds). IOS Press: Amsterdam, 2003; 3–16.
10. Noma T, Badler NI. A virtual human presenter. *Proceedings of IJCAI-97 Workshop on Animated Interface Agents: Making Them Intelligent*, Nagoya, Japan, 1997; 45–51.
11. Noma T, Zhao L, Badler NI. Design of a virtual human presenter. *IEEE Computer Graphics and Applications* 2000; **20**(4): 79–85.
12. Cassell J, Bickmore T, Billinghurst M, *et al.* Embodiment in conversational interfaces: Rea. *Proceedings of SIGCHI Conference on Human Factors in Computing Systems*, Pittsburgh, PA, 1999; 520–527.
13. Marsella S, Gratch J, Rickel J. Expressive behaviors for virtual worlds. In *Life-like Characters: Tools, Affective Func-*



DOI: 10.1002/cav

*tions and Applications*, Prendinger H, Ishizuka M (eds). Springer: New York, 2003; 317–360.
14. McBreen HM, Jack MA. Evaluating humanoid synthetic agents in e-retail applications. *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans* 2001; **31**(5): 394–405.
15. McBreen HM. Embodied conversational agents in e-commerce applications. In *Socially Intelligent Agents: Creating Relationships with Computers and Robots*, Dautenhahn K, Bond AH, Cañamero L, Edmonds B (eds). Kluwer Academic: Boston, MA, 2002; 267–274.
16. André E, Rist T, Müller J. Employing AI methods to control the behavior of animated interface agents. *Applied Artificial Intelligence* 1999; **13**(4): 415–448.
17. Lester JC, Converse SA, Kahler SE, Barlow ST, Stone BA, Bhogal RS. Animated pedagogical agents and problem-solving effectiveness: a large-scale empirical evaluation. *Proceedings of 8th World Conference on Artificial Intelligence in Education*, Kobe, Japan, 1997; 23–30.
18. Lester JC, Converse SA, Kahler SE, Barlow ST, Stone BA, Bhogal RS. The persona effect: affective impact of animated pedagogical agents. *Proceedings of SIGCHI Conference on Human Factors in Computing Systems*, Atlanta, GA, 1997; 359–366.
19. Massaro DW. *Perceiving Talking Faces: From Speech Perception to a Behavioral Principle*. MIT Press: Cambridge, MA, 1998.
20. Shaw E, Johnson WL, Ganeshan R. Pedagogical agents on the web. *Proceedings of 3rd Annual Conference on Autonomous Agents*, Seattle, WA, 1999; 283–290.
21. Johnson WL, Rickel J, Lester JC. Animated pedagogical agents: face-to-face interaction in interactive learning environments. *International Journal of Artificial Intelligence in Education* 2000; **11**: 47–78.
22. Ehsani F, Knodt E. Speech technology in computer-assisted language learning: strengths and limitations of a new CALL paradigm. *Language Learning & Technology* 1998; **2**(1): 45–60.
23. Derwing TM, Munro MJ, Carbonaro M. Does popular speech recognition software work with ESL speech? *TESOL Quarterly* 2000; **34**(3): 592–603.
24. Neri A, Cucchiarini C, Strik W. Automatic speech recognition for second language learning: how and why it actually works. *Proceedings of 15th International Conference on Phonetic Sciences*, Barcelona, Spain, 2003; 1157–1160.
25. Hincks R. Speech recognition for language teaching and evaluating: a study of existing commercial products. *Proceedings of 7th International Conference on Spoken Language Processing*, Denver, CO, 2002; 733–736.
26. Eskenazi M. Using a computer in foreign language pronunciation training: what advantages? *CALICO Journal* 1999; **16**(3): 447–469.
27. Holland VM, Kaplan JD, Sabol MA. Preliminary tests of language learning in a speech-interactive graphics microworld. *CALICO Journal* 1999; **16**(3): 339–359.
28. Johnson WL, Marsella S, Mote N, Vilhjálmsson H, Narayanan S, Choi S. Tactical language training system: supporting the rapid acquisition of foreign language and cultural skills. *Proceedings of InSTIL/ICALL 2004 Symposium on Computer Assisted Learning*, Venice, Italy, 2004.
29. Morton H, Jack MA. Scenario-based spoken interaction with virtual agents. *Computer Assisted Language Learning* 2005; **18**(3): 171–191.
30. Bitmanagement BS Contact VRML/X3D Player. Available at http://www.bitmanagement.com/products/bs_contact_vrml.en.html Accessed on 23 April 2008.
31. H-Anim 1.1 Specification. Available at http://www.h-anim.org/Specifications/H-Anim1.1/ Accessed on 14 April 2008.
32. Likert R. A technique for the measurement of attitudes. *Archives of Psychology* 1932; **22**(140): 1–55.
33. Morton H, Davidson N, Jack MA. Evaluation of a speech interactive CALL system. In *Handbook of Research on Computer-Enhanced Language Acquisition and Learning*, Zhang F, Barber B (eds). Information Science Research: Hershey, PA, 2008; 219–239.
34. Morton H. Spoken electronic language learning (SPELL): Results of User Trials in Scottish Schools. Centre for Communication Interface Research, The University of Edinburgh, 2005.
35. Morton H, Jack MA. Speech interactive CALL: A Cross-Cultural Evaluation. Under review for *Computer Assisted Language Learning* 2008.
36. Morgan JJ. Making a speech recognizer tolerate non-native speech through Gaussian mixture merging. *Proceedings of In-STIL/ICALL 2004*, 2004.
37. Bouselmi G, Fohr D, Illina I, Haton J-P. Fully automated non-native speech recognition using confusion-based acoustic model integration and graphemic constraints. *Proceedings of ICASSP 2006*, 2006.
38. Bouselmi G, Fohr D, Illina I, Haton J-P. Multilingual non-native speech recognition using phonetic confusion-based acoustic model modification and graphemic constraints. *Proceedings of INTERSPEECH 2006*, Pittsburgh, PA, 2006.

*Authors' biographies:*

**Dr James Anderson** is a research fellow at the Centre for Communication Interface Research at the University of Edinburgh. His research at CCIR has focused on technology development and usability assessment in the areas of telepresence shopping services, virtual mannequins and garment modeling, shared-space technology for virtual conferencing, and the use of synthetic human-like agents in both financial services and spoken language learning. He holds PhDs in computer simulation and philosophical theology.



DOI: 10.1002/cav

**Dr Nancie Davidson** is a senior researcher at the Centre for Communication Interface Research at the University of Edinburgh. She holds a BEng (Hons) in Electronics and Electrical Engineering and a PhD in Human-Computer Interaction. Her research focuses on the use of speech recognition in computer interfaces, in both automated telephony and multimodal systems incorporating embodied conversational agents.

**Dr Hazel Morton** holds Masters degrees in English and in Applied Linguistics and a PhD in Computer Assisted Language Learning. Her main academic interests are in the areas of speech recognition technology and embodied conversational agents for eCommerce and eLearning applications. Her thesis explored, through a series of empirical evaluations, the design and evaluation of a speech-interactive CALL program which utilizes speech recognition and animated agent technologies.

**Professor Mervyn Jack** is Professor of Electronic Systems at the University of Edinburgh. A Fellow of the Royal Society of Edinburgh, he leads a multi-disciplinary team of researchers investigating usability engineering of eCommerce services. His main research interests are dialog engineering and virtual reality systems design for advanced eCommerce and consumer applications. He is author of some 250 scientific papers and five textbooks.



DOI: 10.1002/cav