

Role-plays for CALL: System Architecture and Resources

Sabrina Wilske, Magdalena Wolska

Computational Linguistics, Saarland University, Saarbrücken, Germany

Key words: *e-learning, CALL, system architecture, resources, authoring*

Abstract:

We present a system architecture for roleplay-like dialogue exercises for computer-assisted language learning (CALL). Our ultimate goal is to (i) provide learners with flexible dialogue exercises which facilitate practising interactive skills in a variety of typical communicative scenarios, and (ii) provide teachers and CALL content developers with an environment which would allow them to create and maintain resources needed for such exercises in an easy way. The modularity of our dialogue exercise architecture is motivated by the envisioned distribution of labour in resources authoring between language experts (linguistic resources) and knowledge engineering experts (domain models for the dialogue scenarios).

1 Motivation

Practicing communicative skills is one of the most important aspects of foreign language learning, stressed, in particular, in the communication approach [1]. One of the techniques which encourage verbal interaction are role-plays, interactive activities grounded in a well-defined authentic “real world” situation in which a language learner is given a specific task to perform or a problem to solve along with the context in which this is to be done. The communicative scenarios to which a learner is typically exposed range from basic social interactions (such as greetings or introductions) to everyday situations in which learners could find themselves in a foreign country (such as booking a hotel room, ordering a meal in a restaurant or shopping). In a classroom, a learner and a tutor or peer learner take on situation-specific roles and enact them in a dialogue. In the absence of a dialogue partner, however, language learners do not have a chance of putting their conversational skills into practice. With the aim of facilitating language learning through verbal interaction without a human partner, we have been investigating methods of applying techniques from dialogue systems technology to develop a system for authoring and executing role-play dialogues.

The idea of building a system for dialogue-based exercises for CALL is not new. A number of academic and industrial projects developed variants of computer-based simulations of roleplay-like activities. Previous systems include Herr Kommissar [2], FLUENT [3], Sampras [4], Dreistadt [5], and Te Kaitito [6]. Dialogues in CALL are typically set within a virtual microworld presented, for example, as a simulated encounter with a virtual character in an everyday situation. Alternative approaches to realizing a dialogue activity explore adapting existing spoken dialogue systems for the purpose of language learning, for instance, by recasting the setting as a translation game [7] or adding language learning oriented corrective feedback [8]. Perhaps the most comprehensive system with spoken dialogue exercises is the Tactical Language Training System [9] which consists of a number of 3D animated games assessing learners’ language mastery and providing tailored assistance.

Building on the insights from the above projects we have been developing an architecture for authoring and executing language learning dialogues which (i) allows learners to exercise dialogues in various scenarios and their different variants within one scenario, (ii) allows teachers without technical background to author the dialogues and the necessary linguistic resources, (iii) is modular in a way that facilitates division of labour between content authors with and without technical background.

In this paper, we present the architecture of our system whose core components include: linguistic resources for dialogue utterances relevant in a given scenario, an ontology-based domain-model, and a generic dialogue model that allows to conduct dialogues according to the ontology specification. The paper is structured as follows: In Section 2 we specify the properties of the dialogue activity and outline the architecture. In Section 3 we present the details of the present implementation. We show a walk-through example in Section 4 and conclude with a discussion in Section 5.

2 Approach

We start by specifying the properties of dialogue exercises at which we are aiming. Then, we present the overall architecture and point at the interactions between the modules that provide for variation in the dialogue activity.

2.1 *The dialogue activity*

In order to identify desirable features of role-play dialogues as practiced in foreign language courses, we recorded four pairs of students in a beginner's English language course who participated in a conversation class.¹ In the beginning of the session, the students were presented with the topic of the exercise (an „In a restaurant“ scenario), a list of communicative acts relevant in the situation (e.g. "ask for a recommendation", "say what you would like to order") as well as common phrasings used in this setting. They were, moreover, given a transcript of an example dialogue in a restaurant setting and listened to a recording of the dialogue. Then they were asked by the instructor to re-enact similar conversations by taking the roles of a waiter and a restaurant patron.

We transcribed the students' dialogues and analysed them in terms of their variability. Our analysis showed that students tended to diverge considerably from the patterns and examples given in the textbook for the course and from the example dialogue. In particular, students referred to restaurant dishes different from the ones with which they were presented, used alternative phrasings, extended the prescribed dialogues by introducing new utterances or even sub-topics, or simply altered the order of the suggested sub-topics.

Motivated by these findings, in computer-simulated dialogues for language learning, much like in the classroom activity, we would like to give learners an opportunity to practice and repeat the dialogue exercise (i) in different scenarios; e.g. "In a restaurant" or "In a hotel", (ii) in different configurations of a particular scenario; e.g. refer to different restaurant menus, (iii) by following different paths in a dialogue; e.g. start with a starter/not order a starter, and (iv) using different linguistic realizations of the communicative goals. Thus, our goal is a system which we can parametrize with respect to (A) the underlying domain model (different scenario domains and their instances) and (B) the variation in the dialogue flow (e.g. alternative ordering of the dialogue states) and whose linguistic resources support variation.² Two dialogue excerpts from an "In a restaurant" scenario, presented in Table 1, illustrate the kinds of dialogues that our system supports; the waiter's utterances are marked with W and the guest's utterances with G:

1 All the students agreed to participate in our pilot study and they knew that they were being recorded.

2 We eventually also want to include parameters for pedagogical criteria such as the type of language task and level of proficiency or register.

	Dialogue 1	Dialogue 2
W1:	Good evening, Sir.	Good evening. Are you ready to order?
G1:	Good evening.	Sure. What would you recommend for a starter?
W2:	What would you like as a starter?	Our Cesar Salad is very good.
G2:	Can you bring me a green salad, please?	I'll have the Cesar Salad then.
W3a:	Sure.	Okay.
W3b:	What would you like for main course?	And what would you like for the main course?
G3:	I'd like the Marinated Flank Steak, please.	Could I have the Beef Stroganoff?
W4:	How would you like your steak: rare, medium, or well done?	Sure. Would you like potatoes or rice with that?
G4:	Well done, please.	I'll have the potatoes, please.
W5:	Alright.	Okay.

Table 1: Example dialogues

In order to realise such dialogues a system needs knowledge about the scenario itself, the flow of dialogues in the given scenario, the range of utterances relevant in the given scenario, and the configuration of the particular scenario instance to which the learner is referring. In the following section we outline an architecture whose components encode this information and interact in executing the dialogue.

2.2 The system architecture

Our system comprises three components specifying information for the target dialogues: the domain model (a representation of domain knowledge relevant to a given scenario), the language model (a specification of the relevant utterances), and the dialogue model (possible sequences of utterances). A *microworld* is a representation of the state of a fictional world in which learners find themselves, i.e. a particular configuration of the scenario for the given dialogue exercise. The microworld is generated based on the domain model and presented to the learner in a textual form.³ Figure 1 shows the architecture of the system. Below we give a short description of the specification modules and explain how they interact.

Specification Modules

The **Dialogue Model** is a state-based representation of dialogue structures (utterance sequences) in the given scenario. Depending on the desired flexibility for the specific learning setting this can be a linear sequence, where one utterance follows another utterance, or a branching specification, where one utterance can be followed by a set of different utterances.

The **Language Model** defines the range of utterances that are relevant in a given scenario. It consists of a set of utterance templates, together with their types, whose slots can be filled with values depending on the entities available in the microworld.

The **Domain Model** encodes knowledge about the domain of the given scenario by specifying relevant concepts, entities, and relations. In our current implementation it also encodes infor-

³ While a textual setting is perhaps not as attractive as graphics and animations, as e.g. in [9], with our setup more variability between exercises can be introduced without involving programming experts and therefore at a lower development cost.

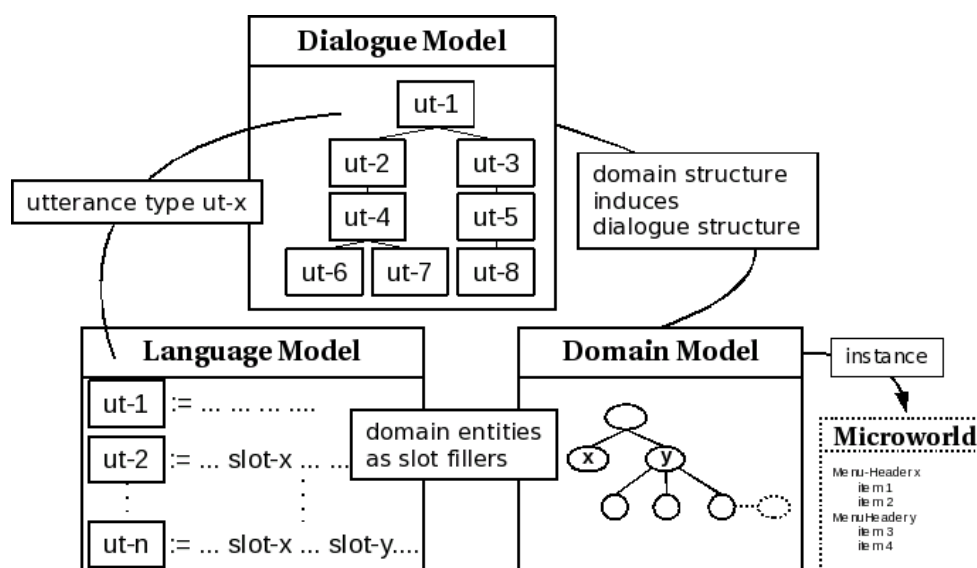


Figure 1: Architecture

mation relevant to the dialogue flow (see Section 3). The dialogue refers to entities in the domain and the domain can determine the structure of the dialogue.

The *microworld* is generated from the domain model. It is a specific instantiation of a subset of the domain model. As the frame of reference for the dialogue, it provides the context of the task that the language learner performs during the dialogue, in particular, the entities to which the learner (and the system) will refer. Different microworlds resulting from different instantiations from the domain model are one way of achieving variability in the dialogue exercise.

Interaction between the modules

The three modules interact with each other in different ways. See again Figure 1 for an illustration of the links that we explain in more detail below.

Dialogue Model and Language Model The structure of the dialogue is defined using utterances as building blocks. In the simplest case we define an utterance as a specific string of words, however, to gain more flexibility and more variation we define utterance templates and mark them with their types. Utterance types correspond to dialogue moves as defined, for instance, by the DAMSL annotation scheme [10]. The language model thus contains a mapping between utterance types and the respective set of strings that can realize them. The separation of the definition of the dialogue model and the language model facilitates independent development and relatively easy adaptation to other languages, as long as the cultural context is similar enough to keep the dialogue structure the same.

Language Model and Domain Model When defining realisations of utterance types, we abstract from specific content words and specify them as slots to be filled with appropriate items from the microworld. In principle then, by changing the domain model we can change the resulting dialogue without ever touching the dialogue model or the language model; this, of course, provided that the slots specified in language model's templates can be filled based on the new domain model. Again, such resource separation facilitates independent development.

Domain model and dialogue model The information in the domain model has an impact on the overall dialogue structure as well as on local substructures. First, the sequence of utterances in the overall dialogue structure is informed by the domain model; while executing the

dialogue, the system refers directly to concepts and relations between them specified in the domain model. Second, features of entities in the domain model can trigger sub-dialogues, as we will show in the example in Section 4.

3 Implementation

The current implementation is based on a three components: an existing parser and a custom generation module which shared a grammar, a manually built ontology and an existing dialogue management environment with a graphical user interface. The dialogues in the current system are conducted in a typewritten mode, however, we anticipate integrating a speech recognition module to enable spoken interaction. The following sections describe the current implementation in more detail.

3.1 The language model

The language model is defined as a context-free grammar in the Backus-Naur Form (BNF). More specifically, we use the Java Speech Grammar Format (JSGF) to define the set of possible utterance templates. In the grammar we allow to indicate slots that can be filled with values instantiated based on the microworld. Note that we use this grammar specification for parsing as well as for generating utterances.

Generating system utterances The generation process in our implementation is based on utterance templates specified by the language model. It takes an utterance type as input and returns a possible realisation of that utterance type. At the moment, the target realisation is picked at random from the entire set of possible realisations, though other choice policies could be implemented. If the utterance type is defined using one or more slots, the corresponding slot fillers have to be given as input. Each slot in the utterance is replaced by the given filler value. For example, the request G3 from the example in Section 2.1, reproduced below:

G3: I would like the Marinated Flank Steak, please.

is represented as a template as follows:

REQUEST(X_o) := I would like X_o please.

and needs a filler for the slot X_o (the requested object).

Understanding user utterances JSGF provides so called tags to define the semantics of the utterances. As multiple tags can be placed at each rule expansion step, one parse can yield more than one tag. For each parse we collect the set of tags, but ignore the complex parse derivation. The dialogue proceeds based on the collected tags. Take as an example the same utterance G3 above with added tags in curly brackets:

REQUEST(X_o) := I would like X_o { X_o } please request }

In this case the parser's output will be: {request, X_o }.

3.2 The domain model and the microworld

In the current implementation, the domain knowledge relevant for the scenario is represented in a form of an ontology, which we built using Protege,⁴ whose internal representation is in the Web Ontology Language (OWL).⁵ Aside from domain concepts and relations the ontology also encodes certain conversational knowledge related to the dialogue structure. In particular it encodes the conventional sequence of topics (here: concepts) raised in the given scenario.

We assume an agreement on how the microworld is generated from the domain model: We declare a distinct concept in the domain model to serve as the root for building up the micro-

⁴ <http://protege.stanford.edu/>

⁵ <http://www.w3.org/TR/owl-features/>

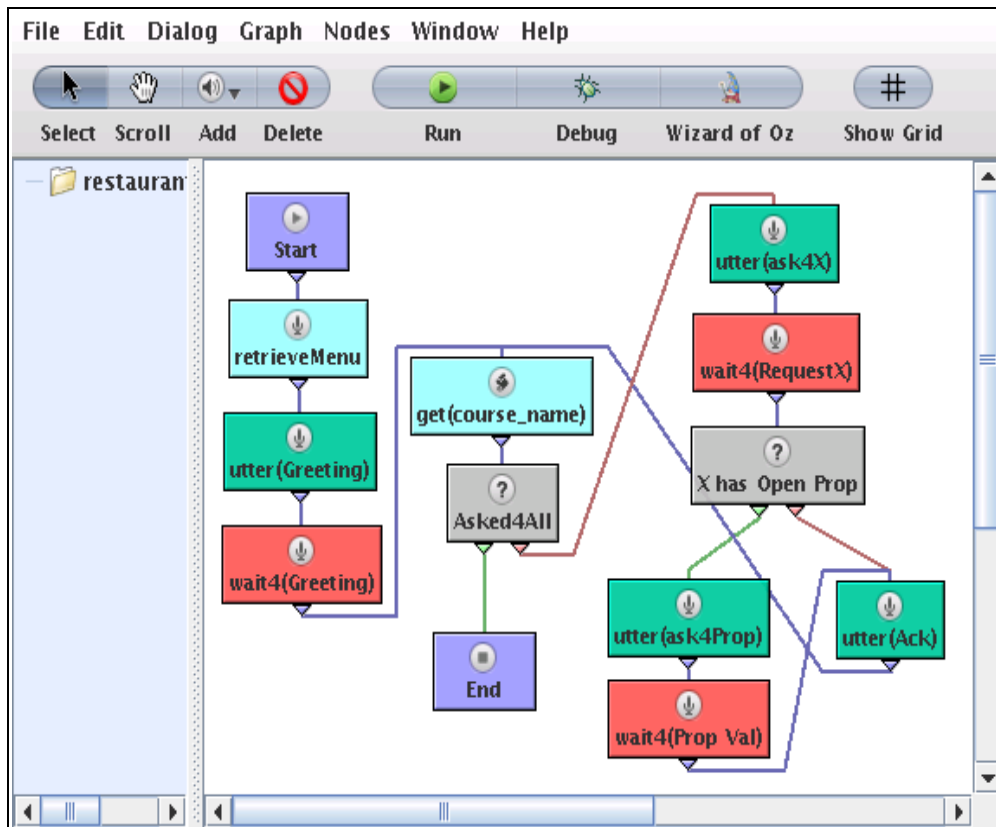


Figure 2: Screenshot of the DialogOS GUI

world, and we (randomly) select a subset of the root's instances (or subclasses; depending on the domain) to represent the microworld which we render as text.

3.3 The dialogue model

The dialogue model in our current system is formulated using the state-based dialogue management tool DialogOS⁶. DialogOS is an extensible platform that allows to connect arbitrary components and exchange information with them.⁷ We connect an open source parser⁸ and a custom-built generation module which both accept the language model, as described above, as input. We also connect an interface for the domain model.⁹

In DialogOS the dialogue model is an extended final state automaton. Its nodes have different functions: They can output information to the connected modules or receive input. They can also test and manipulate the values of internal state variables. Figure 2 shows a fragment of the restaurant dialogue in the DialogOS graphical user interface. The nodes labeled `utter(X)` call the generation module and return the result as output to the user via a text console. The nodes labeled with `wait4(X)` call the parser to analyze the user input and wait for an utterance with specific expected semantics.¹⁰

4 An example

Let us illustrate our approach in a walk-through example. Consider the restaurant situation introduced in Section 2, where the task is to communicate with the waiter to order dinner.

⁶ <http://www.clt-st.de/dialogos>

⁷ The exchange is implemented via a Java-based API

⁸ The Java Speech API implementation of CMU Sphinx' parser available at <http://cmusphinx.sourceforge.net>

⁹ The interface to OWL ontology is implemented in Jena (<http://jena.sourceforge.net/>).

¹⁰ At the moment, we do not implement any clarification strategies for input that was not expected, but we are planning to extend the model in the future.

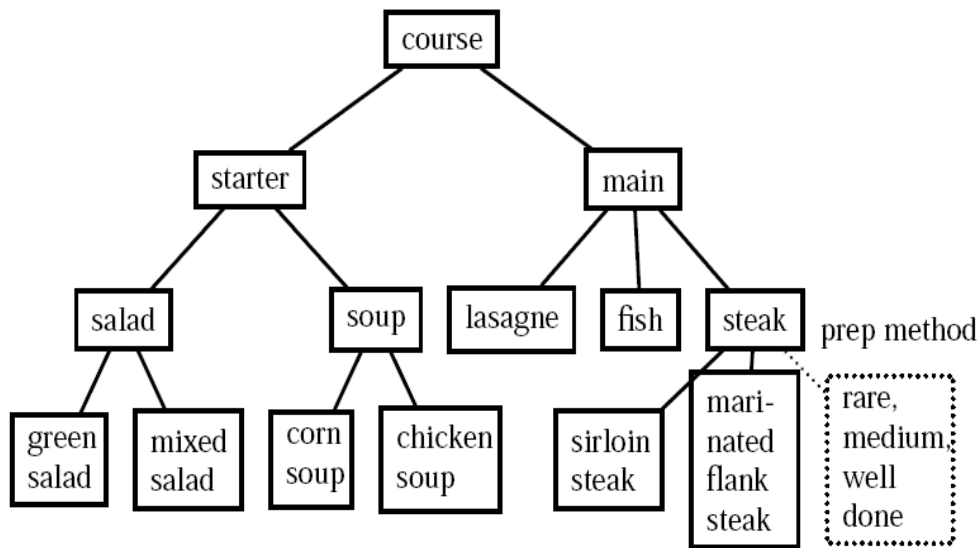


Figure 3: Part of the domain model

The domain model for this scenario contains relevant knowledge about restaurants, in particular that they have some sort of a menu, i.e. a list of meals and drinks that the guest can order from. The “menu” class of the ontology is dedicated to generate microworlds for this scenario. It also includes a classification of restaurant dishes/menu items and imposes a partial ordering on the classes (this corresponds to dialogue relevant knowledge of the facts that, e.g., the starter is ordered before the main dish and that the last course is the dessert). The domain model also contains knowledge about features of certain classes. The microworld is built by randomly selecting subclasses and instances of the dedicated class (here: “menu”) in the domain model, resulting in a specific menu presented to the learner.

Figure 3 shows a small fragment of a possible domain model for the example restaurant dialogue. This fragment is used to generate “The Menu“ microworld (classes “starter” and “main” inherit from the dedicated microword class “menu”). The fragment contains two kinds of courses: starters and main courses and for each of them a couple of instances. For the main course *steak* there is a property *preparation method* with values: *rare*, *medium*, and *well done*. This property triggers the utterance W4: “How would you like your steak: rare, medium, or well done?” in the dialogue example.

The language model for this dialogue contains utterance templates for the restaurant domain, examples of which are shown below:

- GREETING := Hello | Good evening
- ASKfor(X_o) := What (would you like | do you want) (for | as) X_o ?
- REQUEST(X_o) := [I would like | Can you bring me] X_o [please]?
- ASKhow(X_o , LIST $_p$) := How [do you want | would you like] X_o , LIST $_p$?
- REQUEST(X_o , Y_p) := (I would like (it| X_o) Y_p | Y_p) [please].
- ACKNOWLEDGE := sure | okay | alright

X_o stands for an object, Y_p for a property and LIST $_p$ for a list of properties.¹¹

The dialogue model is defined as depicted in Figure 4. The boxes framed by dotted lines are the learner’s utterances, those in solid frames are the system’s utterances. After the system initiates with a greeting, the learner is expected to answer with a greeting. The system then checks the domain model and proceeds with the dialogue according to the specified ordering of the concepts; in our example a starter and a main course realized by utterances W2 and W3b in Dialogue1. The expected response is to choose a meal within the appropriate course

11 The vertical bar | connects alternatives, square brackets [] indicate optional parts. We omitted the semantic tags here.

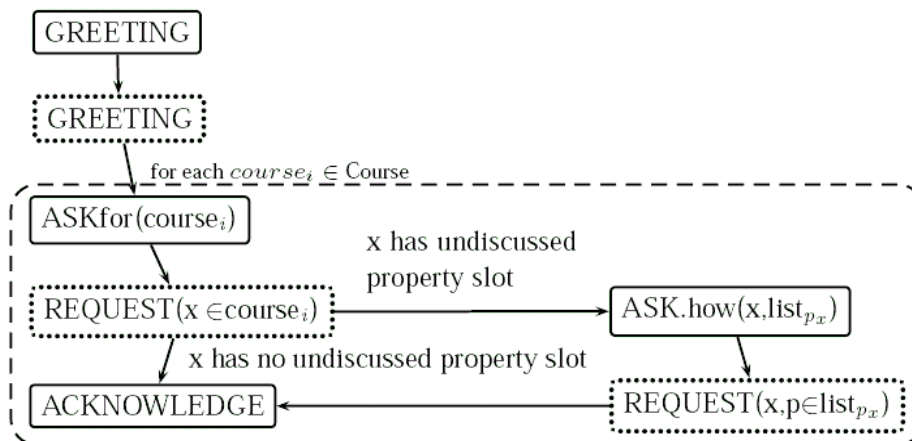


Figure 4: Dialogue model

(utterances G2 and G3). For each meal the domain model is checked to see if it contains any properties that need to be discussed. If so, the system initiates a sub-dialogue that discusses the property by asking a question about it and expecting an appropriate answer. In both cases the system acknowledges the learner's response.

The screenshot of DialogOS in Figure 2 shows a fragment of the dialogue model. For each system utterance there is an `utter` node and for each expected user utterances a `wait4` node. The node labeled `retrieveMenu` queries the domain model based on the dedicated microworld-specific class and receives a copy of the menu for internal use. It uses the menu for retrieving the names of the courses (node `get(course name)`) in order to obtain slot fillers for the utterance template `ASKfor(Xo)`. The node `X has open Prop` queries the domain model to check if the ordered menu item has any properties that need to be determined. If so, the dialogue proceeds with the system's question about that property.

Note that this example is rather simple as it only expects and can deal with one type of utterance at a time. It cannot handle any unexpected answers either. However, our architecture allows to specify far more complex dialogue models, and we have implemented those in two different domains: the restaurant domain illustrated here, and shopping.

5 Conclusion

In this paper we presented an architecture which supports executing language learning dialogues. The key aspect of our approach is the separation of the domain knowledge from the dialogue model and the language model, which facilitates the division of the authoring task. Furthermore, our architecture supports alternative dialogue variants within one scenario.

There are many ways to optimize and further develop our system. Right now we have a rather ad-hoc way to generate a microworld representation from a given domain model; we want to find a more principled approach. Related to this is the fact that it is not clear that an ontology is always the appropriate way to represent knowledge about a scenario: On the one hand, there may be domains that can be represented with less complex data structures, on the other hand, there might be domains that require a more complex ontology than those we developed so far, and thus require different ways to instantiate the microworld.

Given that one of our goals is an easy authoring process for non-experts, we have to think of ways of making the authoring process of the domain model easier and more intuitive than it is now. We also want to include a representation of parameters related such criteria as the proficiency level, the register, and the socio-cultural context.

References:

- [1] Canale, Michael and Swain, Merrill: Theoretical bases of communicative approaches to second language teaching and testing. (1980)
- [2] DeSmedt, W.H.: Herr Kommissar: An ICALL conversation simulator for intermediate German. In Intelligent language tutors: Theory shaping technology. (1995)
- [3] Hamburger, H.: Tutorial Tools for Language Learning by Two-Medium Dialogue. In Intelligent language tutors: Theory shaping technology. (1995)
- [4] Lehuen, J.: A Dialog-based Architecture for Computer Aided Language Learning. In AAAI Fall Symposium on Building Dialog Systems for Tutorial Applications. (2000)
- [5] Lech, T.C. and de Smedt, K.: Dreistadt: A language enabled MOO for language learning. In Proceedings of the ECAI-06 Workshop on Language-enabled Educational Technology. (2006)
- [6] Vlugter, P. and Knott, A.: A multi-speaker dialogue system for computer-aided language learning. In Proceedings of Brandial-06. (2006)
- [7] Wang, C. and Seneff, S.: A Spoken Translation Game for Second Language Learning. In Proceedings of AIED-07. (2007)
- [8] Raux, A. and Eskenazi, M.: Using Task-Oriented Spoken Dialogue Systems for Language Learning: Potential, Practical Applications and Challenges. In Proceedings of InSTIL/ICALL2004 - NLP and Speech Technologies in Advanced Language Learning Systems. (2004)
- [9] Johnson, W. L. and Choi, S. and Marsella, S. and Mote, N. and Narayanan, S. and Vilhjálmsón, H.: Tactical Language Training System: Supporting the Rapid Acquisition of Foreign Language and Cultural Skills. In Proceedings of InSTIL/ICALL - NLP and Speech Technologies in Advanced Language Learning Systems. (2004)
- [10] James Allen and Mark Core: Draft of DAMSL: Dialog Act Markup in Several Layers. (1997)

Authors:

Sabrina Wilske Diplom-Linguist and Magdalena Wolska, MPhil
Saarland University, Computational Linguistics
Universität des Saarlandes,
FR 4.7 Computerlinguistik,
Postfach 15 11 50,
66041 Saarbrücken, Germany
{sw,magda}@coli.uni-saarland.de