Olympus: an open-source framework for conversational spoken language interface research

Dan Bohus, Antoine Raux, Thomas K. Harris,
Maxine Eskenazi, Alexander I. Rudnicky
School of Computer Science
Carnegie Mellon University
{dbohus, antoine, tkharris, max, air}@cs.cmu.edu

Abstract

We introduce Olympus, a freely available framework for research in conversational interfaces. Olympus' open, transparent, flexible, modular and scalable nature facilitates the development of large-scale, real-world systems, and enables research leading to technological and scientific advances in conversational spoken language interfaces. In this paper, we describe the overall architecture, several systems spanning different domains, and a number of current research efforts supported by Olympus.

1 Introduction

Spoken language interfaces developed in industrial and academic settings differ in terms of goals, the types of tasks and research questions addressed, and the kinds of resources available.

In order to be economically viable, most industry groups need to develop real-world applications that serve large and varied customer populations. As a result, they gain insight into the research questions that are truly significant for current-generation technologies. When needed, they are able to focus large resources (typically unavailable in academia) on addressing these questions. To protect their investments, companies do not generally disseminate new technologies and results.

In contrast, academia pursues long-term scientific research goals, which are not tied to immediate economic returns or customer populations. As a result, academic groups are free to explore a larger variety of research questions, even with a high risk of failure or a lack of immediate payoff. Academic groups also engage in a more open exchange of ideas and results. However, building spoken language interfaces requires significant investments that are sometimes beyond the reach of academic researchers. As a consequence, research in academia is oftentimes conducted with toy systems and skewed user populations. In turn, this raises questions about the validity of the results and hinders the research impact.

In an effort to address this problem and facilitate research on relevant, real-world questions, we have developed Olympus, a freely available framework for building and studying conversational spoken language interfaces. The Olympus architecture, described in Section 3, has its roots in the CMU Communicator project (Rudnicky et al., 1999). Based on that experience and subsequent projects, we have engineered Olympus into an open, transparent, flexible, modular, and scalable architecture.

To date, Olympus has been used to develop and deploy a number of spoken language interfaces spanning different domains and interaction types; these systems are presented in Section 4. They are currently supporting research on diverse aspects of spoken language interaction. Section 5 discusses three such efforts: error handling, multi-participant conversation, and turn-taking.

We believe that Olympus and other similar toolkits, discussed in Section 6, are essential in order to bridge the gap between industry and academia. Such frameworks lower the cost of entry for re-

search on practical conversational interfaces. They also promote technology transfer through the reuse of components, and support direct comparisons between systems and technologies.

2 Desired characteristics

While developing Olympus, we identified a number of characteristics that in our opinion are necessary to effectively support and foster research. The framework should be open, transparent, flexible, modular, and scalable.

Open. Complete source code should be available for all the components so that researchers and engineers can inspect and modify it towards their ends. Ideally, source code should be free for both research and commercial purposes and grow through contributions from the user community.

Transparent / Analytic. Open source code promotes transparency, but beyond that researchers must be able to analyze the system's behavior. To this end, every component should provide detailed accounts of their internal state. Furthermore, tools for data visualization and analysis should be an integral part of the framework.

Flexible. The framework should be able to accommodate a wide range of applications and research interests, and allow easy integration of new technologies.

Modular / Reusable. Specific functions (e.g. speech recognition, parsing) should be encapsulated in components with rich and well-defined interfaces, and an application-independent design. This will promote reusability, and will lessen the system development effort.

Scalable. While frameworks that rely on simple, well established approaches (e.g. finite-state dialogs in VoiceXML) allow the development of large-scale systems, this is usually not the case for frameworks that provide the flexibility and transparency needed for research. However, some research questions are not apparent until one moves from toy systems into large-scale applications. The framework should strive to not compromise scalability for the sake of flexibility or transparency.

3 Architecture

At the high level, a typical Olympus application consists of a series of components connected in a classical, pipeline architecture, as illustrated by the bold components in Figure 1. The audio signal for the user utterance is captured and passed through a speech recognition module that produces a recognition hypothesis (e.g., `two p.m.`). The recognition hypothesis is then forwarded to a language understanding component that extracts the relevant concepts (e.g., `[time=2p.m.]`), and then through a confidence annotation module that assigns a confidence score. Next, a dialog manager integrates this semantic input into the current context, and produces the next action to be taken by the system in the form of the semantic output (e.g., `{request end_time}`). A language generation module produces the corresponding surface form, which is subsequently passed to a speech synthesis module and rendered as audio.

Galaxy communication infrastructure. While the pipeline shown in bold in Figure 1 captures the logical flow of information in the system, in practice the system components communicate via a centralized message-passing infrastructure – Galaxy (Seneff et al., 1998). Each component is implemented as a separate process that connects to a traffic router – the Galaxy hub. The messages are sent through the hub, which forwards them to the appropriate destination. The routing logic is described via a configuration script.

Speech recognition. Olympus uses the Sphinx decoding engine (Huang et al., 1992). A recognition server captures the audio stream, forwards it to a set of parallel recognition engines, and collects the corresponding recognition results. The set of best hypotheses (one from each engine) is then forwarded to the language understanding component. The recognition engines can also generate n-best lists, but that process significantly slows down the systems and has not been used live. Interfaces to connect Sphinx-II and Sphinx-III engines, as well as a DTMF (touch-tone) decoder to the recognition server are currently available. The individual recognition engines can use either n-gram- or grammar-based language models. Dialog-state specific as well as class-based language models are supported, and tools for constructing language and acoustic models from data are readily available. Most of the Olympus systems described in the next section use two gender-specific Sphinx-II recognizers in parallel. Other parallel decoder configurations can also be created and used.

Language understanding is performed by Phoenix, a robust semantic parser (Ward and Issar,

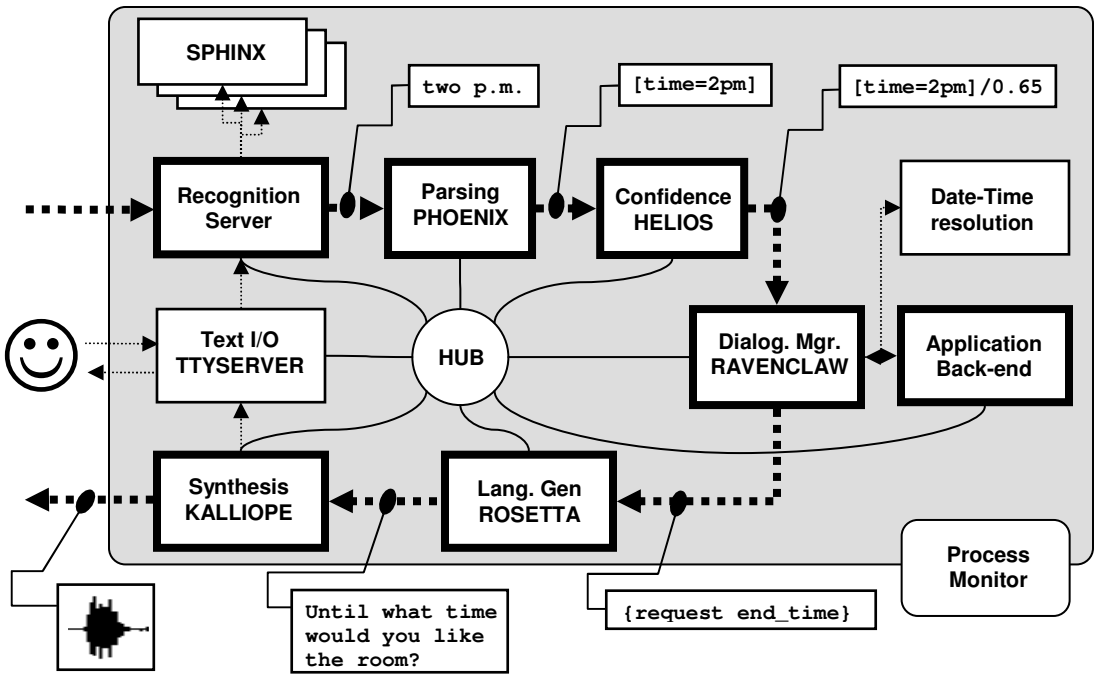

Figure 1. The Olympus dialog system reference architecture (a typical system)

1994). Phoenix uses a semantic grammar to parse the incoming set of recognition hypotheses. This grammar is assembled by concatenating a set of reusable grammar rules that capture domain-independent constructs like `[Yes]`, `[No]`, `[Help]`, `[Repeat]`, and `[Number]`, with a set of domain-specific grammar rules authored by the system developer. For each recognition hypothesis the output of the parser consists of a sequence of slots containing the concepts extracted from the utterance.

Confidence annotation. From Phoenix, the set of parsed hypotheses is passed to Helios, the confidence annotation component. Helios uses features from different knowledge sources in the system (e.g., recognition, understanding, dialog) to compute a confidence score for each hypothesis. This score reflects the probability of correct understanding, i.e. how much the system trusts that the current semantic interpretation corresponds to the user's intention. The hypothesis with the highest score is forwarded to the dialog manager.

Dialog management. Olympus uses the RavenClaw dialog management framework (Bohus and Rudnicky, 2003). In a RavenClaw-based dialog manager, the domain-specific dialog task is represented as a tree whose internal nodes capture the hierarchical structure of the dialog, and whose leaves encapsulate atomic dialog actions (e.g., asking a question, providing an answer, accessing a database). A domain-independent dialog engine executes this dialog task, interprets the input in the current dialog context and decides which action to engage next. In the process, the dialog manager may exchange information with other domain-specific agents (e.g., application back-end, database access, temporal reference resolution agent).

Language generation. The semantic output of the dialog manager is sent to the Rosetta template-based language generation component, which produces the corresponding surface form. Like the Phoenix grammar, the language generation templates are assembled by concatenating a set of predefined, domain-independent templates, with manually authored task-specific templates.

Speech synthesis. The prompts are synthesized by the Kalliope speech synthesis module. Kalliope can be currently configured to use Festival (Black and Lenzo, 2000), which is an open-source speech synthesis system, or Cepstral Swift (Cepstral 2005), a commercial engine. Finally, Kalliope also supports the SSML markup language.

Other components. The various components briefly described above form the core of the Olympus dialog system framework. Additional components have been created throughout the development of various systems, and, given the modularity of the architecture, can be easily reused. These include a telephony component, a text

input-and-output interface, and a temporal reference resolution agent that translates complex date-time expressions (including relative references, holidays, etc.) into a canonical form. Recently, a Jabber interface was implemented to support interactions via the popular GoogleTalk internet messaging system. A Skype speech client component is also available.

Data Analysis. Last but not least, a variety of tools for logging, data processing and data analytics are also available as part of the framework. These tools have been used for a wide variety of tasks ranging from system monitoring, to trends analysis, to training of internal models.

A key characteristic shared by all the Olympus components is the clear separation between domain-independent programs and domain-specific resources. This decoupling promotes reuse and lessens the system development effort. To build a new system, one can focus simply on developing resources (e.g., language model, grammar, dialog task specification, generation templates) without having to do any programming. On the other hand, since all components are open-source, any part of the system can be modified, for example to test new algorithms or compare approaches.

4 Systems

To date, the Olympus framework has been used to successfully build and deploy several spoken dialog systems spanning different domains and interaction types (see Table 1). Given the limited space, we discuss only three of these systems in a bit more detail: Let's Go!, LARRI, and TeamTalk. More information about the other systems can be found in (RavenClaw-Olympus, 2007).

4.1 Let's Go!

The Let's Go! Bus Information System (Raux et al 2005; 2006) is a telephone-based spoken dialog system that provides access to bus schedules. Interaction with the system starts with an open prompt, followed by a system-directed phase where the user is asked the missing information. Each of the three or four pieces of information provided (origin, destination, time of travel, and optional bus route) is explicitly confirmed. The system knows 12 bus routes, and about 1800 place names.

Originally developed as an in-lab research system, Let's Go! has been open to the general public since March, 2005. Outside of business hours, calls to the bus company are transferred to Let's Go!, providing a constant flow of genuine dialogs (about 40 calls per weeknight and 70 per weekend night). As of March, 2007, a corpus of about 30,000 calls to the system has been collected and partially transcribed and annotated. In itself, this publicly available corpus constitutes a unique resource for the community. In addition, the system itself has been modified for research experiments (e.g., Raux et al., 2005, Bohus et al., 2006). Between-system studies have been conducted by running several versions of the system in parallel and picking one at random for every call. We have recently opened this system to researchers from other groups who wish to conduct their own experiments.

4.2 LARRI

LARRI (Bohus and Rudnicky, 2002a) is a multimodal system for support of maintenance and repair activities for F/A-18 aircraft mechanics. The system implements an Interactive Electronic Technical Manual.

LARRI integrates a graphical user interface for easy visualization of dense technical information (e.g., instructions, schematics, video-streams) with a spoken dialog system that facilitates information access and offers assistance throughout the execution of procedural tasks. The GUI is accessible via a translucent head-worn display connected to a wearable client computer. A rotary mouse (dial) provides direct access to the GUI elements.

After an initial log-in phase, LARRI guides the user through the selected task, which consists of a sequence of steps containing instructions, optionally followed by verification questions. Basic steps can include animations or short video sequences that can be accessed by the user through the GUI or through spoken commands. The user can also take the initiative and access the documentation, either via the GUI or by simple commands such as "go to step 15" or "show me the figure".

The Olympus architecture was easily adapted for this mobile and multi-modal setting. The wearable computer hosts audio input and output clients, as well as the graphical user interface. The Galaxy hub architecture allows us to easily connect these

System name	Domain / Description	Genre
RoomLine (Bohus and Rudnicky 2005)	telephone-based system that provides support for conference room reservation and scheduling within the School of Computer Science at CMU.	information access (mixed initiative)
Let's Go! Public (Raux et al 2005)	telephone-based system that provides access to bus schedule information in the greater Pittsburgh area.	information access (system initiative)
LARRI (Bohus and Rudnicky 2002)	multi-modal system that provides assistance to F/A-18 aircraft personnel during maintenance tasks.	multi-modal task guidance and procedure browsing
Intelligent Procedure Assistant (Aist et al 2002)	early prototype for a multi-modal system aimed at providing guidance and support to the astronauts on the International Space Station during the execution of procedural tasks and checklists.	multi-modal task guidance and procedure browsing
TeamTalk (Harris et al 2005)	multi-participant spoken language command-and-control interface for a team of robots in the treasure-hunt domain.	multi-participant command-and-control
VERA	telephone-based taskable agent that can be instructed to deliver messages to a third party and make wake-up calls.	voice mail / message delivery
Madeleine	text-based dialog system for medical diagnosis.	diagnosis
ConQuest (Bohus et al 2007)	telephone-based spoken dialog system that provides conference schedule information.	information access (mixed-initiative)
RavenCalendar (Stenchikova et al 2007).	multimodal dialog system for managing personal calendar information, such as meetings, classes, appointments and reminders (uses Google Calendar as a back-end)	information access and scheduling

Table 1. Olympus-based spoken dialog systems (shaded cells indicate deployed systems)

components to the rest of the system, which runs on a separate server computer. The rotary-mouse events from the GUI are rendered as semantic inputs and are sent to Helios which in turn forwards the corresponding messages to the dialog manager.

4.3 TeamTalk

TeamTalk (Harris et al., 2005) is a multi-modal interface that facilitates communication between a human operator and a team of heterogeneous robots, and is designed for a multi-robot-assisted treasure-hunt domain. The human operator uses spoken language in concert with pen-gestures on a shared live map to elicit support from teams of robots. This support comes in the forms of mapping unexplored areas, searching explored areas for objects of interest, and leading the human to said objects. TeamTalk has been built as a fully functional interface to real robots, including the Pioneer P2DX and the Segway RMP. In addition, it can interface with virtual robots within the high-fidelity USARSim (Balakirsky et al., 2006) simulation environment. TeamTalk constitutes an excellent platform for multi-agent dialog research.

To build TeamTalk, we had to address two challenges to current architecture. The multi-participant nature of the interaction required multiple dialog managers; the live map with pen-gestured references required a multi-modal integration. Again, the flexibility and transparency of the Olympus framework allowed for relatively simple solutions to both of these challenges. To accommodate multi-participant dialog, each robot in the domain is associated with its own RavenClaw-based dialog manager, but all robots share the other Olympus components: speech recognition, language understanding, language generation and speech synthesis. To accommodate the live map GUI, a Galaxy server was built in Java that could send the user's inputs to Helios and receive outputs from RavenClaw.

5 Research

The Olympus framework, along with the systems developed using it, provides a robust basis for research in spoken language interfaces. In this section, we briefly outline three current research efforts supported by this architecture. Information about other supported research can be found in (RavenClaw-Olympus, 2007).

5.1 Error handling

A persistent and important problem in today's spoken language interfaces is their lack of robustness when faced with understanding errors. This problem stems from current limitations in speech recognition, and appears across most domains and interaction types. In the last three years, we conducted research aimed at improving robustness in spoken language interfaces by: (1) endowing them with the ability to accurately detect errors, (2) de-

veloping a rich repertoire of error recovery strategies and (3) developing scalable, data-driven approaches for building error recovery policies[1]. Two of the dialog systems from Table 1 (Let's Go! and RoomLine) have provided a realistic experimental platform for investigating these issues and evaluating the proposed solutions.

With respect to **error detection**, we have developed tools for learning confidence annotation models by integrating information from multiple knowledge sources in the system (Bohus and Rudnicky, 2002b). Additionally, Bohus and Rudnicky (2006) proposed a data-driven approach for constructing more accurate beliefs in spoken language interfaces by integrating information across multiple turns in the conversation. Experiments with the RoomLine system showed that the proposed belief updating models led to significant improvements (equivalent with a 13.5% absolute reduction in WER) in both the effectiveness and the efficiency of the interaction.

With respect to **error recovery strategies**, we have developed and evaluated a large set of strategies for handling misunderstandings and non-understandings (Bohus and Rudnicky, 2005). The strategies are implemented in a task-decoupled manner in the RavenClaw dialog management framework.

Finally, in (Bohus et al., 2006) we have proposed a novel online-learning based approach for building **error recovery policies** over a large set of non-understanding recovery strategies. An empirical evaluation conducted in the context of the Let's Go! system showed that the proposed approach led to a 12.5% increase in the non-understanding recovery rate; this improvement was attained in a relatively short (10-day) time period.

The models, tools and strategies developed throughout this research can and have been easily reused in other Olympus-based systems.

5.2 Multi-participant conversation

Conversational interfaces are generally built for one-on-one conversation. This has been a workable assumption for telephone-based systems, and a useful one for many single-purpose applications. However this assumption will soon become strained as a growing collection of always-available agents (e.g., personal trainers, pedestrian guides, or calendar systems) and embodied agents (e.g., appliances and robots) feature spoken language interfaces. When there are multiple active agents that wish to engage in spoken dialog, new issues arise. On the input side, the agents need to be able to identify the addressee of any given user utterance. On the output side, the agents need to address the problem of channel contention, i.e., multiple participants speaking over each other.

Two architectural solutions can be envisioned: (1) the agents share a single interface that understands multi-agent requirements, or (2) each agent uses its own interface and handles multi-participant behavior. Agents that provide different services have different dialog requirements, and we believe this makes a centralized interface problematic. Furthermore, the second solution better fits human communication behavior and therefore is likely to be more natural and habitable.

TeamTalk is a conversational system that follows the second approach: each robot has its own dialog manager. Processed user inputs are sent to all dialog managers in the system; each dialog manager decides based on a simple algorithm (Harris et al., 2004) whether or not the current input is addressed to it. If so, an action is taken. Otherwise the input is ignored; it will be processed and responded to by another robot. On the output side, to address the channel contention problem, each RavenClaw output message is augmented with information about the identity of the robot that generated it. The shared synthesis component queues the messages and plays them back sequentially with the corresponding voice.

We are currently looking into two additional challenges related to multi-participant dialog. We are interested in how to address groups and subgroups in addition to individuals of a group, and we are also interested in how to cope with multiple humans in addition to multiple agents. Some experiments investigating solutions to both of these issues have been conducted. Analysis of the results and refinements of these methods are ongoing.

5.3 Timing and turn-taking

While a lot of research has focused on higher levels of conversation such as natural language understanding and dialog planning, low-level interactional phenomena such as turn-taking have not

[1] A policy specifies how the system should choose between different recovery strategies at runtime.

received as much attention. As a result, current systems either constrain the interaction to a rigid one-speaker-at-a-time style or expose themselves to interactional problems such as inappropriate delays, spurious interruptions, or turn over-taking (Raux et al., 2006). To a large extent, these issues stem from the fact that in common dialog architectures, including Olympus, the dialog manager works asynchronously from the real world (i.e., utterances and actions that are planned are assumed to be executed instantaneously). This means that user barge-ins and backchannels are often interpreted in an incorrect context, which leads to confusion, unexpected user behavior and potential dialog breakdowns. Additionally, dialog systems' low-level interactional behavior is generally the result of ad-hoc rules encoded in different components that are not precisely coordinated.

In order to investigate and resolve these issues, we are currently developing version 2 of the Olympus framework. In addition to all the components described in this paper, Olympus 2 features an Interaction Manager which handles the precise timing of events perceived from the real world (e.g., user utterances) and of system actions (e.g., prompts). By providing an interface between the actual conversation and the asynchronous dialog manager, Olympus 2 allows a more reactive behavior without sacrificing the powerful dialog management features offered by RavenClaw. Olympus 2 is designed so that current Olympus-based systems can be upgraded with minimal effort.

This novel architecture, initially deployed in the Let's Go system, will enable research on turn-taking and other low-level conversational phenomena. Investigations within the context of other existing systems, such as LARRI and TeamTalk, will uncover novel challenges and research directions.

6 Discussion and conclusion

The primary goal of the Olympus framework is to enable research that leads to technological and scientific advances in spoken language interfaces.

Olympus is however by no means a singular effort. Several other toolkits for research and development are available to the community. They differ on a number of dimensions, such as objectives, scientific underpinnings, as well as technological and implementation aspects. Several toolkits, both commercial, e.g., TellMe, BeVocal, and academic, e.g., Ariadne (2007), SpeechBuilder (Glass et al., 2004), and the CSLU toolkit (Cole, 1999), are used for rapid development. Some, e.g., CSLU and SpeechBuilder, have also been used for educational purposes. And yet others, such as Olympus, GALATEEA (Kawamoto et al., 2002) and DIPPER (Bos et al., 2003) are primarily used for research. Different toolkits rely on different theories and dialog representations: finite-state, slot-filling, plan-based, information state-update. Each toolkit balances tradeoffs between complexity, ease-of-use, control, robustness, flexibility, etc.

We believe the strengths of the Olympus framework lie not only in its current components, but also in its open, transparent, and flexible nature. As we have seen in the previous sections, these characteristics have allowed us to develop and deploy practical, real-world systems operating in a broad spectrum of domains. Through these systems, Olympus provides an excellent basis for research on a wide variety of spoken dialog issues. The modular construction promotes the transfer and reuse of research contributions across systems.

While desirable, an in-depth understanding of the differences between all these toolkits remains an open question. We believe that an open exchange of experiences and resources across toolkits will create a better understanding of the current state-of-the-art, generate new ideas, and lead to better systems for everyone. Towards this end, we are making the Olympus framework, as well as a number of systems and dialog corpora, freely available to the community.

Acknowledgements

We would like to thank all those who have brought contributions to the components underlying the Olympus dialog system framework. Neither Olympus nor the dialog systems discussed in this paper would have been possible without their help. We particularly wish to thank Alan W Black for his continued support and advice. Work on Olympus components and systems was supported in part by DARPA, under contract NBCH-D-03-0010, Boeing, under contract CMU-BA-GTA-1, and the US National Science Foundation under grant number 0208835. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

References

Aist, G., Dowding, J., Hockey, B.A., Rayner, M., Hieronymus, J., Bohus, D., Boven, B., Blaylock, N., Campana, E., Early, S., Gorrell, G., and Phan, S., 2003. *Talking through procedures: An intelligent Space Station procedure assistant*, in Proc. of EACL-2003, Budapest, Hungary

Ariadne, 2007, *The Ariadne web-site,* as of January 2007, http://www.opendialog.org/.

Balakirsky, S., Scrapper, C., Carpin, S., and Lewis, M. 2006. *UsarSim: providing a framework for multi-robot performance evaluation,* in Proc. of PerMIS.

Black, A. and Lenzo, K., 2000. *Building Voices in the Festival Speech System*, http://festvox.org/bsv/, 2000.

Bohus, D., Grau Puerto, S., Huggins-Daines, D., Keri, V., Krishna, G., Kumar, K., Raux, A., Tomko, S., 2007. *Conquest – an Open-Source Dialog System for Conferences*, in Proc. of HLT 2007, Rochester, USA.

Bohus, D., Langner, B., Raux, A., Black, A., Eskenazi, M., Rudnicky, A. 2006. *Online Supervised Learning of Non-understanding Recovery Policies*, in Proc. of SLT-2006, Aruba.

Bohus, D., and Rudnicky, A. 2006. *A K-hypotheses + Other Belief Updating Model*, in Proc. of the AAAI Workshop on Statistical and Empirical Methods in Spoken Dialogue Systems, 2006.

Bohus, D., and Rudnicky, A., 2005. *Sorry I didn't Catch That: An Investigation of Non-understanding Errors and Recovery Strategies*, in Proc. of SIGdial-2005, Lisbon, Portugal.

Bohus, D., and Rudnicky, A., 2003. *RavenClaw: Dialog Management Using Hierarchical Task Decomposition and an Expectation Agenda,* in Proc. of Eurospeech 2003, Geneva, Switzerland.

Bohus, D., and Rudnicky, A., 2002a. *LARRI: A Language-based Maintenance and Repair Assistant,* in Proc. of IDS-2002, Kloster Irsee, Germany.

Bohus, D., and Rudnicky, A., 2002b. *Integrating Multiple Knowledge Sources in the CMU Communicator Dialog System,* Technical Report CMU-CS-02-190.

Bos, J., Klein, E., Lemon, O., and Oka, T., 2003. *DIPPER: Description and Formalisation of an Information-State Update Dialogue System Architecture*, in Proc. of SIGdial-2003, Sapporo, Japan

Cepstral, LLC, 2005. Swift™: Small Footprint Text-to-Speech Synthesizer, http://www.cepstral.com.

Cole, R., 1999. *Tools for Research and Education in Speech Science*, in Proc. of the International Conference of Phonetic Sciences, San Francisco, USA.

Glass, J., Weinstein, E., Cyphers, S., Polifroni, J., 2004. *A Framework for Developing Conversational Interfaces*, in Proc. of CADUI, Funchal, Portugal.

Harris, T. K., Banerjee, S., Rudnicky, A., Sison, J. Bodine, K., and Black, A. 2004. *A Research Platform for Multi-Agent Dialogue Dynamics*, in Proc. of The IEEE International Workshop on Robotics and Human Interactive Communications, Kurashiki, Japan.

Harris, T. K., Banerjee, S., Rudnicky, A. 2005. *Heterogenous Multi-Robot Dialogues for Search Tasks,* in AAAI Spring Symposium: Dialogical Robots, Palo Alto, California.

Huang, X., Alleva, F., Hon, H.-W., Hwang, M.-Y., Lee, K.-F. and Rosenfeld, R., 1992. *The SPHINX-II Speech Recognition System: an overview*, in Computer Speech and Language, 7(2), pp 137-148, 1992.

Kawamoto, S., Shimodaira, H., Nitta, T., Nishimoto, T., Nakamura, S., Itou, K., Morishima, S., Yotsukura, T., Kai, A., Lee, A., Yamashita, Y., Kobayashi, T., Tokuda, K., Hirose, K., Minematsu, N., Yamada, A., Den, Y., Utsuro, T., and Sagayama, S., 2002. *Open-source software for developing anthropomorphic spoken dialog agent*, in Proc. of PRICAI-02, International Workshop on Lifelike Animated Agents.

Raux, A., Langner, B., Bohus, D., Black, A., and Eskenazi, M. 2005, *Let's Go Public! Taking a Spoken Dialog System to the Real World,* in Proc. of Interspeech 2005, Lisbon, Portugal.

Raux, A., Bohus, D., Langner, B., Black, A., and Eskenazi, M. 2006 *Doing Research on a Deployed Spoken Dialogue System: One Year of Let's Go! Experience,* in Proc. of Interspeech 2006, Pittsburgh, USA.

RavenClaw-Olympus web page, as of January 2007: http://www.ravenclaw-olympus.org/.

Rudnicky, A., Thayer, E., Constantinides, P., Tchou, C., Shern, R., Lenzo, K., Xu W., and Oh, A., 1999. *Creating natural dialogs in the Carnegie Mellon Communicator system,* in Proc. of Eurospeech 1999.

Seneff, S., Hurley, E., Lau, R., Pao, C., Schmid, P., and Zue V. 1998 *Galaxy-II: A reference architecture for conversational system development,* in Proc. of ICSLP98, Sydney, Australia.

Stenchikova, S., Mucha, B., Hoffman, S., Stent, A., 2007. *RavenCalendar: A Multimodal Dialog System for Managing A Personal Calendar*, in Proc. of HLT 2007, Rochester, USA.

Ward, W., and Issar, S., 1994. *Recent improvements in the CMU spoken language understanding system,* in Proc. of the ARPA Human Language Technology Workshop, pages 213–216, Plainsboro, NJ.