Building Conversational Agents with Basilica

Rohit Kumar **Carolyn P. Rosé**
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
rohitk@cs.cmu.edu cprose@cs.cmu.edu

Abstract

Basilica is an event-driven software architecture for creating conversational agents as a collection of reusable components. Software engineers and computer scientists can use this general architecture to create increasingly sophisticated conversational agents. We have developed agents based on Basilica that have been used in various application scenarios and foresee that agents build on Basilica can cater to a wider variety of interactive situations as we continue to add functionality to our architecture.

1 Introduction

Conversational Interfaces apply the metaphor of agent to an interface which allows the user to conversationally interact with the machine using natural language through speech or text. The current state of the art in the area of conversational interfaces is largely dominated by spoken dialog systems (SDS). These SDS are most often used for the purpose of accessing information from a database over the telephone. Other common applications of conversational agents include computer aided instruction (CAI) and human-robot interaction (HRI).

Conversational Agents in most of today's SDS, CAI and HRI are designed to work within the scope of specific task domains which allows the scientists and engineers working on such systems to ensure satisfactory and relevant interaction with the user most of the time. Within the task domain, such agents can display intelligent interactive behavior like helping the user use the interface, asking remedial questions (Bohus and Rudnicky, 2005), shaping the user behavior (Tomko and Rosenfeld, 2004) by using alternative phrasing of utterances, responding to user affect (D'Mello et al., 2008) through text, voice and gesture, engaging the user through the display of presence via backchannels (Ward, 1996) and embodiment (Cassell et al., 1999).

As more and more of these intelligent interactive agents get built for many task domains (Raux et al., 2005; Bohus et al., 2007; Gockley et al., 2005; Amtrak Julie; …) that surround our everyday life, we observe a gradual transition in the use of the conversational agent technology to be a form of situated interaction. One of the characteristic requirements of this transition towards ubiquity of such interactive agents is the capability to sense and trigger behavior in a context sensitive way.

In most conversational interfaces today, the only trigger used by the agents is that of initiation of conversation usually by sensing user presence through a telephone call, proximity detection or user login into a virtual environment. The initiation event is followed by a scripted task-oriented conversation with the agent. These scripts could be fairly complex depending on the representational formalism underlying the script. Most of the common software architectures/platforms used to create conversational agents like TellMe Studio, Voxeo Prophecy, Olympus (Bohus et al., 2007), DIPPER (Bos and Oka, 2003), etc. use one or more of these presence sensing techniques and one of the many existing scripting languages including VoiceXML, SALT, TuTalk (Jordan et al., 2007) and Ravenclaw (Bohus and Rudnicky, 2003) task specification language among others.

However, in our recent work on building conversational agents situated in collaborative learning environments, we have discovered the need for a software architecture for creating agents that persist in an interactive environment in which human users interact with these agents as well as with each other. In this situation, the agents need to be able to sense many kinds of triggers at many points of time and choose to respond to some of those triggers through a variety of modalities including conversation. This observation was the motivation for creating Basilica which is our architecture for building conversational agents. In section 2, we talk more about the intricacies of Basilica and agents built on this architecture. Section 3 describes some of application scenarios in which we are using Conversational Agents based on Basilica.

2 Basilica Architecture

In order to meet the need for an architecture that enables development of Conversational Agents as a collection of behavioral components that can sense triggers and respond to those appropriately, we created the Basilica architecture.

In this architecture, we model sensing and responding as two types of components that make up conversational agents. The sensing components referred to as *Filters* observe stimuli from various kinds of input sources and other components. They can also generate stimuli for other components. On the other hand, *Actor* components generate responsive behavior that may be observed the user(s) and other components. Basilica provides the software elements required to tie Filters and Actors together through *Connections* that carry *Events* over them. We think that many of the state of the art intelligent behaviors listed in section 1 can be implemented as dyads of filter and actor components.

The minimal set of behavioral component classes listed above can easily be extended. For example, certain agent designs may need memory components and coordination components which bridge across multiple actors or filters that do not necessarily share events with each others. Timer components may be used to generate regulated stimuli. Besides belonging to one of these classes of components, certain components may act as wrappers to external systems. For example, we use wrapper components to integrate TuTalk dialog management system (Jordan et al., 2007) for some of the instructive behavior exhibited by our agents. Also, certain components act as wrappers to the environment in which the agent is present. These wrappers help in easily integrating the same agent with multiple environments without having to change any underlying components except the wrappers to the environment.

We believe that fairly intelligent conversational agents can be built for situated interaction applications by incrementally building a large number of behavioral components. Each of these components represent a decomposition of the agent's perceptive and cognitive capabilities. Among the agents we have built using Basilica, we observe that some of these capabilities are common across agents. Hence the corresponding behavioral components get re-used in many cases. Some instances of component re-use are mentioned in Section 3.

Note that recently there has been other work on modeling conversational agents as a decomposition of components. Jaspis (Turunen and Hakulinen, 2003) models the agent as a collection of *managers*, *agents* and *evaluators* which synchronize with each other through *transactions*. RIME (Nakano et al., 2008) distributes cognitive capabilities across a collection of *experts* of two types. However, *evaluators* and *agents* are configured as a pile of components whereas our filters and actors are configured as a network. Hence, designing conversational agents with Basilica gives the flexibility to change the network topology. Also, while Jaspis agents are stateless, actors in our architecture need not be stateless. In other work on event-based multi-layered architectures (Raux and Eskenazi, 2007), events are used for communication between layers as a mean to provide higher reactive compared to pipeline architectures. While we share this motivation, definition of events is extended here as events are used for all kinds of communication, coordination and control in Basilica.

3 Current Application Scenarios

In 2008, we built three conversational agents to support learners in collaborative learning environments. Also, we are currently using Basilica to develop a cross-lingual assistive agent to support non-Spanish speaking 911 dispatchers in the southern states of the US. In this section, we will discuss these four conversational agents briefly.

CycleTalk is an intelligent tutoring system that helps college sophomores studying Thermodynamics learn about principles of designing Steam cycles. In our recent experiments, we have studied the effectiveness of conversational agents in this intelligent tutoring system (Kumar et al., 2007; Chaudhuri et al., 2008). Student use the system both individually and in pairs. The conversational agent monitors student interaction in a chat room as the students work on solving a design problem. The tutor provides the students with hints to help touch upon all the underlying concepts while the students work on the design exercise. Also the agent brings up reflective dialogs when it detects a relevant topic in the students conversation. One of the problems we observed over the years with the use of instructional dialogs in collaborative environments is that the students tend to ignore the tutoring agent if it interrupts the students when they are talking to each other. Basilica helped us in resolving this problem by implementing a component that tells that student that help is available on the topic they are talking about and they can ask for the dialog support when they are ready. Basilica gives the flexibility to change the intervention strategy used by the agent when it is speaking with more than one student.

In another version of this system, the tutoring agent prompted the students with some motivational prompts occasionally as we observed that many of the students found the design exercise very demanding to complete in the time permitted for this lab exercise. We found that the use of motivational prompts improved the student's attitude towards the automated agent.

We developed another agent to help college level mathematics students working on problem solving. This agent operates in a collaborative environment which includes a whiteboard. As in the case with the CycleTalk agent, the agent used here also helps the students with hints and dialogs. The component required for those behaviors were re-used as-is with modifications only their configuration files. Besides these behaviors, the agent coordinates the problem solving sessions for the team by presenting the team with problems as images placed on the whiteboard and helping the students stay on track by answering questions about the amount of time left in the problem solving session.

Recently, we modified the environment wrapper components of our CycleTalk agent and integrated them with a SecondLife application (Weusijana et al., 2008). This integration helps developers of conversational agents create interactive agents in the SecondLife virtual environment.

Finally, in a currently ongoing project, we are building an agent that would interpret Spanish utterances from a distressed 9-1-1 caller and work with a human dispatcher who does not know Spanish to attend to the call. We model the agent in this scenario after a human translator who does not just translate the caller's input to English and vice versa. Instead the translator partners with the dispatcher to provide service to the caller. Partnering conversational agents with a human user to help another human user in a different role is a novel application of interactive agents.

4 Building Agents using Basilica

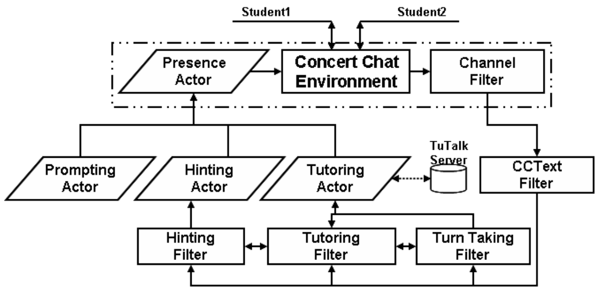

Figure 1. Components of the CycleTalk Agent

Building conversational agents using Basilica involves the process of representing the desired agent as a decomposition of components. Figure 1 above shows the components that make up the CycleTalk conversational agent we mentioned in Section 3. The rectangles represent Filters and the parallelograms represent Actors. Connections are shown as solid lines. In a detailed design, these lines are annotated with the events they carry.

Once an agent is designed, the agents and filters required for the implementation of the agent can be either re-used from the pre-existing components of Basilica or implemented as Java objects that extend the corresponding component class. Often the programming task is limited to implementing handlers and generators for the events received and sent out by the component. Theoretically, the validity of a component can be verified if it can handle and generate all the events as specified in the design diagram.

As we continue to develop more conversational agents on this architecture, we intend to create development tools which would easily translate a design like Figure 1 to the implementation and facilitate validation and debugging of the agent.

5 Demonstration Outline

The demonstration of our architecture will give the audience an opportunity to interact with the agents we have described in section 3 and discuss how we can design such agents using Basilica. We will have a poster to aid the discussion along with ability to probe into the code underlying the design of these agents. Attendees will be able to understand the process involved in building agents with Basilica and assess the effort required. Additionally, if we have any specialized development tools to automatically map agent design as described in Section 4 to Java code, we will demonstrate those tools. Up to date information about Basilica can be found at http://basilica.rohitkumar.net/wiki/

Acknowledgements

This work is supported by NSF REESE/REC grant number 0723580.

References

Dan Bohus and Alex Rudnicky, 2005. *Error Handling in the RavenClaw dialog management architecture*, HLT-EMNLP-2005, Vancouver

Stefanie Tomko and Roni Rosenfeld, 2004. *Shaping Spoken Input in User-Initiative Systems*. Interspeech 2004, Jeju, Korea

Antoine Raux, Brian Langner, Dan Bohus, Alan Black, and Maxine Eskenazi, 2005. *Let's Go Public! Taking a Spoken Dialog System to the Real World*, Interspeech 2005, Lisbon, Portugal

Dan Bohus, Sergio Grau, David Huggins-Daines, Venkatesh Keri, Gopala Krishna A., Rohit Kumar, Antoine Raux, and Stefanie Tomko, 2007. *Conquest - an Open-Source Dialog System for Conferences*, HLT-NAACL 2007, Rochester, NY

Amtrack Julie, http://www.networkworld.com/news/2003/0619julie.html

Justin Cassell, Timothy Bickmore, Billinghurst, M., Campbell, L., Chang, K., Vilhjálmsson, H. and Yan, H., 1999. *Embodiment in Conversational Interfaces: Rea*, CHI'99, Pittsburgh, PA

Nigel Ward, 1996. *Using Prosodic Clues to decide when to produce Back-channel Utterances*, ICSLP 96

Sidney D' Mello, Tanner Jackson, Scotty Craig, Brent Morgan, Patrick Chipman, Holly White, Natalie Person, Barry Kort, Rana el Kaliouby, Rosalid W. Picard and Arthur Graesser, 2008, *AutoTutor Detects and Responds to Learners Affective and Cognitive States*, Workshop on Emotional and Cognitive Issues, ITS 2008, Montreal

Rachel Gockley, Allison Bruce, Jodi Forlizzi, Marek Michalowski, Anne Mundell, Stephanie Rosenthal, Brennan Sellner, Reid Simmons, Kevin Snipes, Alan C. Schultz and Jue Wang, 2005. *Designing Robots for Long-Term Social Interaction*, IROS 2005

Dan Bohus, Antoine Raux, Thomas Harris, Maxine Eskenazi and Alex Rudnicky, 2007. *Olympus: an open-source framework for conversational spoken language interface research* HLT-NAACL 2007 Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technology, Rochester, NY

Johan Bos and Tetsushi Oka, 2003. *Building Spoken Dialogue Systems for Believable Characters*, 7th workshop on the semantics & pragmatics of dialogue

TellMe, https://studio.tellme.com/

Voxeo Prophecy, http://www.voxeo.com/products/

Pamela Jordan, Brian Hall, Michael Ringenberg, Yue Cui, Carolyn P. Rosé, 2007. *Tools for Authoring a Dialogue Agent that Participates in Learning Studies*, AIED 2007

Dan Bohus and Alex Rudnicky, 2003. *RavenClaw: Dialog Management Using Hierarchical Task Decomposition and an Expectation Agenda*, Eurospeech 2003, Geneva, Switzerland

Markku Turunen, Jaakko Hakulinen, 2003. *Jaspis - An Architecture for Supporting Distributed Spoken Dialogues*, Eurospeech' 2003, Geneva, Switzerland

Mikio Nakano, Kotaro Funakoshi, Yuji Hasegawa, Hiroshi Tsujino, 2008. *A Framework for Building Conversational Agents Based on a Multi-Expert Model*, 9th SigDial Workshop on Discourse and Dialog, Columbus, Ohio

Antoine Raux and Maxine Eskenazi, 2007. *A Multi-Layer Architecture for Semi-Synchronous Event-Driven Dialogue Management*, ASRU 2007, Kyoto

Rohit Kumar, Carolyn Rose, Mahesh Joshi, Yi-Chia Wang, Yue Cui, Allen Robinson, *Tutorial Dialogue as Adaptive Collaborative Learning Support*, 13th AIED 2007, Los Angeles, California

Sourish Chaudhuri, Rohit Kumar, Carolyn P. Rose, 2008. *It's not easy being green - Supporting Collaborative Green Design Learning*, ITS 2008, Montreal

Baba Kofi A. Weusijana, Rohit Kumar, Carolyn P. Rose, 2008. *MultiTalker: Building Conversational Agents in Second Life using Basilica*, Second Life Education Community Convention, Purple Strand: Educational Tools and Products, 2008, Tampa, FL