Workshop on Real-Time Conversations with Virtual Agents,
Santa Cruz, CA, Sept. 2012

Operating in a Hierarchy of Time Scales for an Always-On Relational Agent

Charles Rich, Candace L. Sidner, Bahador Nooraei, and William Coon

Worcester Polytechnic Institute
Worcester, MA, USA
{rich,sidner}@wpi.edu

Abstract. Motivated by the challenge of designing a real-time conversational agent that is always on and builds long-term relationships, we have developed a software architecture that operates with control loops at three distinct time scales, from weeks down to milliseconds. We discuss the novel aspects of this architecture and how it is applied in a relational agent for isolated older adults.

1 Introduction

One of the long-held dreams of artificial intelligence is to create an agent (whether robotic, animated or only a disembodied voice) that functions as a permanent member of a human household. Because humans are deeply and fundamentally social beings, they cannot help but expect such a continuously present artificial agent, especially if humanoid, to become part of their network of personal relationships.

Previous work [2] has placed intelligent virtual agents in peoples homes for months at a time, but these agents were not "always on," i.e., it was up to the person to log in once a day to interact with the agent. There have also been long-term always-on agents in public settings, such as Roboceptionist [6], that did not, however, build a continuing relationship with any single person.

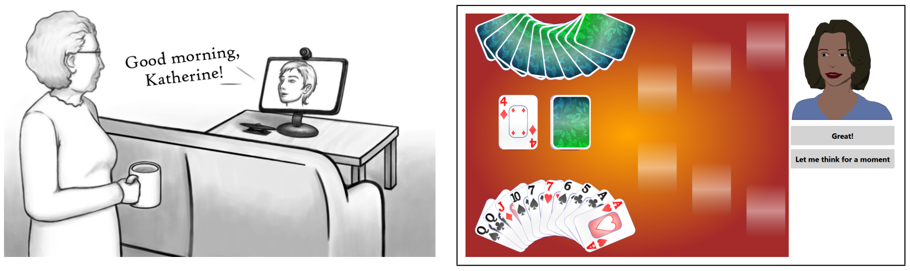

Fig. 1. Always-on agent concept and prototype.

The goal of this research is to develop and implement a computational model for building long-term relationships between humans and virtual agents, along with the interaction affordances to support this process. As theoretical starting points, we look to Grosz and Sidner's [7] collaborative discourse model and Bickmore's [1] relationship model. Our initial application focus is to provide companionship and social support and to promote wellness for older adults who are living alone.

The left side of Fig. 1 illustrates the basic concept of how the virtual agent will "live" in a person's home. Notice that because the agent is always on and has sensing abilities (using microphone, camera and passive IR motion detector), it can *initiate* interactions.

The right side of Fig. 1 is a screen shot from our current prototype in which the agent is playing a social game of cards with the person. The cartoon-like animated face is one of Bickmore's animation clients.[1] We are using a text-to-speech engine for the agent's utterances and touch menus for the user's contributions to the conversation (since speech recognition is problematic with our older population). The area of the screen where the card game appears will change for different activities; for example, a week-at-glance calendar will appear to support conversations about scheduling.

A typical interaction, lasting perhaps 20 minutes, might start when the agent notices the person walking by at a distance and calls out a greeting, such as "Good morning!". The person approaches the computer and a conversation about the weather ensues, which then develops into a social game of cards (Rummy). During the card game, the agent brings up the somewhat sensitive topic of the person diet. The conversation ends when the agent reminds the person that she needs to leave for a trip to the aquarium. Later in the day, they finish the card game together and the agent arranges a Skype call with the person's sister (hiding all the details of the actual Skype web interface).[2]

This paper focuses on one novel aspect of this project relevant to the Workshop on Real-Time Conversations with Virtual Agents, namely the fact that a scenario such as the one above requires the agent to act and reason in a hierarchy of times scales ranging from milliseconds to minutes to days and weeks. After describing some of the technical mechanisms we have developed to address this challenge, we will in Section 4 return to this scenario in more detail.

2 Time Scales

The typical interaction above illustrates the range of behaviors at different time scales that an always-on relational agent must support. We have found it useful to group these behaviors into the three-level hierarchy shown in Fig. 2:

1. *Relationship planning* takes place over hours, days and weeks and determines which activities are appropriate for the time of day and stage of the long-term relationship. For example, with a stranger, the agent only talks about

[1] See http://relationalagents.com

[2] Also see http://tinyurl.com/AlwaysOnVideo for a concept video.

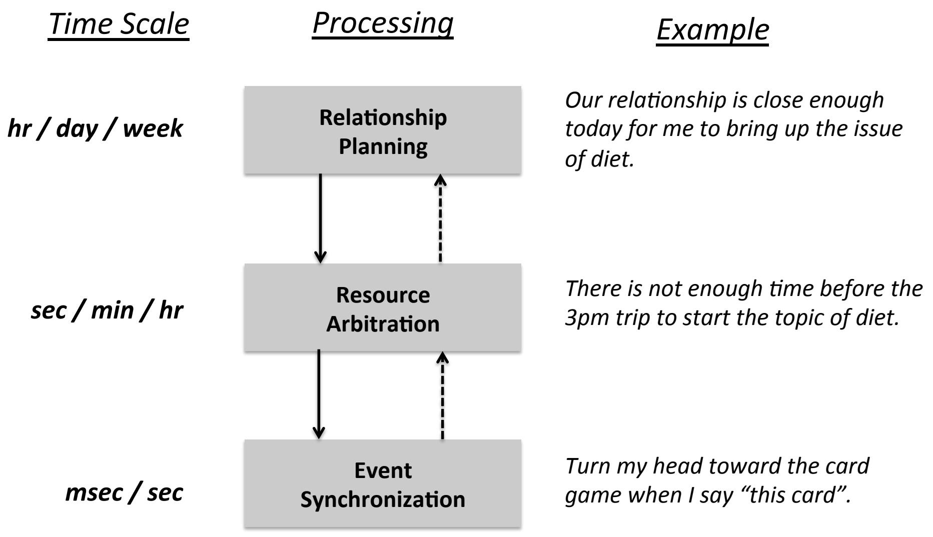
Time Scale
Processing
Example
hr / day / week
Relationship Planning
Our relationship is close enough today for me to bring up the issue of diet.
sec / min / hr
Resource Arbitration
There is not enough time before the 3pm trip to start the topic of diet.
msec / sec
Event Synchronization
Turn my head toward the card game when I say "this card".

Fig. 2. A hierarchy of time scales.

the weather and sports, but with a companion talking about diet and exercise is a possibility.

2. *Resource arbitration* happens in "soft" real time, i.e., on the scale of seconds to minutes to hours. The most visible resource that the agent needs to arbitrate is the current focus of attention in the conversation. The decision of when to switch the focus between activities depends on time and other factors, such as the relative importance of the activities. The temporal reasoning includes the expected duration of the activities (e.g., interrupting a card game for a brief reminder) and the urgency of the activities (e.g., switching to a Skype call when the other party is already connected is very urgent, whereas talking about diet can be be done anytime today).

3. *Event synchronization* is "hard" (sub-second) real time. For example, if the coordination of speech and gesture is even a fraction of a second off, it is perceived as unnatural.

3 Functional Architecture

Fig. 3 shows the functional architecture we have implemented to support operating at the three time scales described above and that we are using to develop our always-on relational agent application for isolated older adults. The *relationship manager* [5] operates at the relationship planning scale and the *real-time collaboration manager* [11] operates at the hard and soft real-time scales. It is

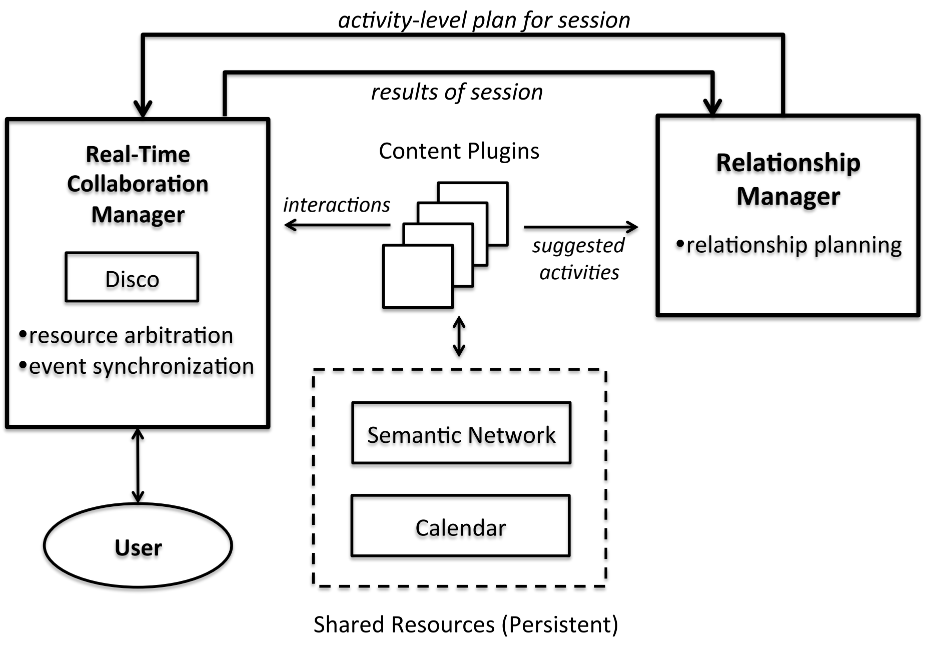

Fig. 3. Functional architecture of always-on agent.

useful to think of this architecture in terms of three nested "loops" or threads of control corresponding to the three time scales.

3.1 Relationship Planning

The outermost loop in the architecture is between the relationship manager and the collaboration manager and is represented by the two arrows at the top of Fig. 3. The communication between these two modules is in terms of high-level *activities*, such as discussing the weather, playing cards, having a Skype call, etc. Before each interaction session, the relationship manager produces a high-level plan for the session based on meta-information about each activity provided by the content plugins (see below). This plan consists of a partially ordered set of activities with conditionals (to allow the user to make choices) and other constraints. For example, the plan for the typical interaction discussed earlier would have had a weather chat, followed by either card-playing[3] or a baseball chat (the user chose cards), followed by a Skype call, followed by a story-recording activity (which was never reached). At the end of a session, the collaboration manager reports back to the relationship manager about which activities were started, which were successfully completed, how long was spent on each activity, etc. The relationship manager uses this information to plan the next session.

[3] Card playing is an example of a "container" activity, discussed further below.

The two innermost loops of control, for resource arbitration and event synchronization, operate within the collaboration manager. These two loops together manage all of the agent's real-time interaction with the user, including menus, speech and gestures.

3.2 Resource Arbitration

The resource arbitration loop is responsible for starting, stopping and switching between activities, based on the activity-level plan for the session, external conditions and a set of heuristic rules.[4] The arbitration loop uses Disco [13], the open-source successor to Collagen [12], to represent the discourse state, which includes the current activity-level plan tree and the focus of attention stack. The resource arbitration loop is implemented as a schema-based system, inspired by reactive architectures, such as [3].

A novel aspect of our heuristic rules for activity switching is the concept of a *container* activity. A container activity is an activity, such as playing cards, eating a meal or golf, that naturally co-extends in time with a conversation about something else. For example, when you play cards, it is very natural to intermingle your social comments about the card game with chatting about another topic, such as baseball or weather. However, it is not natural to similarly intermingle a conversation about baseball with a conversation about weather.

3.3 Event Synchronization

The innermost control loop of the collaboration manager concerns the moment to moment changes needed in the agent's presentation and behavior to coordinate between modalities (e.g., speech and gesture) and to respond to external events.

The coordination between modalities is similar in spirit to the Behavior Markup Language [15], except that we use Holroyd's event-driven approach [9], because we also intend to apply our architecture to robotic applications. For example, we have the ability to synchronize the turning of the agent's head toward the card game display to coincide with uttering the phrase "this card."

An example of a hard real-time response to an external event implemented in this control loop is barge-in. In our menu-driven system, barge-in is when the user touches a menu item while the agent is still speaking. When this happens, the agent's audio output needs to be stopped within a fraction of a second.

For more details of the internal architecture of the collaboration manager, see [11].

3.4 Content Plugins

Returning to Fig. 3, notice that the *content plugins* sit between the relationship manager and the collaboration manager and are accessed by both. The goal of

[4] This part of the architecture also arbitrates contention for resources other than the conversational focus, such as parts of the agent's body; but this is beyond the scope of this paper.

this aspect of the architecture is to provide authoring modularity. An application like our agent for isolated older adults will need to support a large number of different activities, and it should be easy for multiple developers to add new activities as the project evolves. Plugins currently under consideration or development include: agent's self-introduction (overview of its currently available activities), diet and exercise counseling, arranging and managing calls via Skype, daily update chat, weather chat, sports chat (baseball and basketball), calendar scheduling and reminding, social card playing, meditation, story recording, and humorous anecdotes.

For the purpose of planning a session, all the relationship manager needs to know is which high-level activities each plugin implements (one plugin can implement one or more related activities) and some meta-information about each activity including: which relationship stage it is appropriate for, how long it typically takes, and how much it increases the closeness between the agent and person when it is successfully completed.

Once an activity is started by the collaboration manager, the content of all the agent utterances and the user menu items is obtained from the plugins via an abstract API. All of the dialogue logic for a particular activity is thus encapsulated within the plugin. Plugins can be implemented using any technology. Some of our plugins are implemented using Disco for Games [8, 13]; others are coded in Java as simple state machines. Activities can also request the application screen area (i.e., where the card game appears in Fig. 1), which is a resource managed by the arbitration loop. Once an activity obtains control of this screen area, it can manage both touch input and graphics output in this area as needed for that activity.

Finally in Fig. 3, we have implemented a database consisting of a semantic network of people, places and events, and a calendar that are resources to all of the content plugins, as well as the relationship manager, for persistent storage of relevant information. Our implementation uses the OWL ontology system [10], because it is convenient for automatically making simple inferences using this information, such as the fact that the day and month of a person's birth is a recurring yearly birthday event.

4 Scenario Revisited

We now revisit the scenario introduced in Section 1 to point out in more detail examples of where each of the different time scales in the architecture are operating:

- The agent sees the person walking by and attempts to initiate an interaction with a greeting. [*arbitration* initiates new goal]
- The agent and person chit-chat a while about the local weather and also about some unusual weather in another city where the person's sister lives. [*synchronization* to support user barge-in during menu interaction]
- The agent suggests and the person agrees to playing a social game of cards (Rummy). [*arbitration* initiates new goal]

– During the card game, the agent looks toward the card display when it says "this card." [*synchronization*]
– After playing cards for ten minutes, the agent broaches the topic of improving the person's diet. [*planning* has added diet discussion to the list of allowable goals for this interaction; *arbitration* initiates this goal after an adequate amount of social "warm up" interaction has transpired]
– The agent notices and reminds the person that it is time for a previously scheduled Skype call with the person's brother. [*arbitration* initiates a new goal based on calendar and clock time]
– The agent pauses the card game and brings up the Skype video screen connecting the person to her brother (hiding all details of the actual Skype web interface).
– When the Skype call is done, the person abruptly leaves without saying anything to the agent.
– After a few minutes have passed, the agent concludes that the person intended to end the interaction. [*arbitration*]
– The agent updates its persistent model of the activities that occurred during the session and based on its rules, concludes that the long-term relationship has advanced from acquaintance to companion. [*planning*]

5 Conclusions and Future Directions

We have designed and implemented an architecture for real-time conversational agents that operates at three distinct time scales, from weeks down to millseconds, for relationship planning, resource arbitration and event synchronization. While many other conversational agent architectures, such as [4] and [14], operate at two time scales, corresponding roughly to our soft and a hard real-time loops, we do not know of any architecture that includes an explicit relationship planning cycle like ours. Furthermore, the temporal reasoning in our (resource arbitration) soft real-time loop is more sophisticated than these other systems.

We plan to evaluate the performance of our architecture in the field this coming year by placing always-on agents that will run autonomously for 4 to 6 weeks in the homes of a dozen or more isolated older adults.

Looking further into the future, we have designed our architecture to also be suitable for human-robot interaction. We plan to field test a robotic version of our always-on agent, using a yet-to-chosen desktop robot with an expressive face.

Acknowledgements. The always-on relational agent for isolated older adults is a joint project between WPI and Northeastern University. We therefore wish to acknowledge the contributions of Prof. Tim Bickmore, the principal investigator at Northeastern, and his students and staff, including Lazlo Ring, Laura Vardoulakis, Dr. Barbara Barry, John Connor Westfall, Kenneth Manning, Ramesh Manuvinakurike, Bridgette Collado and Ashley Kline. We also wish to thank JJ Liu and Mitch Wills for their contributions at WPI.

This material is based upon work supported by the National Science Foundation under Grant No. IIS-0811942 and IIS-1012083. 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 National Science Foundation.

References

1. Bickmore, T.: Relational Agents: Effecting Change through Human-Computer Relationships. PhD thesis, MIT Media Laboratory (2003)
2. Bickmore, T., Schulman, D., Yin, L.: Maintaining engagement in long-term interventions with relational agents. Int. J. of Applied Artificial Intelligence **24**(6) (2010) 648–666
3. Brooks, R.: A robust layered control system for a mobile robot. IEEE J. of Robotics and Automation **2** (1986) 14–23
4. Cavazza, M., de la Camara, R.S., Turunen, M.: How was your day?: A companion eca. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1. (2010) 1629–1630
5. Coon, W.: A computational model for building relationships between humans and virtual agents. Master's thesis, Worcester Polytechnic Institute (August 2012)
6. Gockley, R., Bruce, A., Forlizzi, J., Michalowski, M., Mundell, A., Rosental, S., Sellner, B., Simmons, R., Snipes, K., Schultz, A., Wang, J.: Designing robots for long-term social interaction. In: Proc. IEEE Int. Conf. on Intelligent Robots and Systems, Edmonton, Canada (2005)
7. Grosz, B.J., Sidner, C.L.: Attention, intentions, and the structure of discourse. Computational Linguistics **12**(3) (1986) 175–204
8. Hanson, P., Rich, C.: A non-modal approach to integrating dialogue and action. In: Proc. 6th AAAI Artificial Intelligence and Interactive Digital Entertainment Conf., Palo Alto, CA (October 2010)
9. Holroyd, A., Rich, C.: Using the behavior markup language for human-robot interaction. In: Proc. ACM Conf. on Human-Robot Interaction, Boston, MA (March 2012)
10. McGuinness, D., Van Harmelen, F., et al.: Owl web ontology language overview. Technical report, W3C Recommendation (2004)
11. Nooraei, B.: A real-time architecture for conversational agents. Master's thesis, Worcester Polytechnic Institute (August 2012)
12. Rich, C., Sidner, C., Lesh, N.: Collagen: Applying collaborative discourse theory to human-computer interaction. AI Magazine **22**(4) (2001) 15–25 Special Issue on Intelligent User Interfaces.
13. Rich, C., Sidner, C.L.: Using collaborative discourse theory to partially automate dialogue tree authoring. In: Proc. Int. Conf. on Intelligent Virtual Agents, Santa Cruz, CA (September 2012)
14. Swartout, W., Gratch, J., Hill Jr, R., Hovy, E., Marsella, S., Rickel, J., Traum, D., et al.: Toward virtual humans. AI Magazine **27**(2) (2006)
15. Vilhjalmsson, H., et al.: The behavior markup language: Recent developments and challenges. In: Intelligent Virtual Agents 2007, Lecture Notes in Artificial Intelligence. Volume 4722. Springer-Verlag, Berlin (2007) 99–111