Retrofitting Synthetic Dialog Agents to Game Characters For Lifestyle Risk Training

Susann Luperfoy

Stottler Henke Associates, Inc.
48 Grove Street Suite 205
Somerville, MA 02144-2500
luperfoy@stottlerhenke.com

Abstract

This paper describes a game-based Intelligent Tutoring System (ITS) that engages learners in interactive role-play as a form of indirect health communication about lifestyle risks. Successful instruction must not only inform patients of correlations and risks but also equip them to overcome well-known cognitive obstacles to wise lifestyle choices. Through the game, HEDONIST, we mean to address these sub-rational decision processes that are inaccessible to conventional didactic instruction methods. This is an experiment in capturing the pedagogical value of experiential learning, especially cognitive dissonance, learning by teaching others, and the social support value of ad hoc social networks that form spontaneously inside the game and persist outside. For instance, to excel in the game HEDONIST the player must use dialog to persuade Non-Player Characters (NPCs) to modify their simple decision functions by 'teaching' them about the consequences of substance abuse, unsafe sex, eating disorders, smoking, etc. The ITS integrates the commercial game engine NeverWinterNights™ with intelligent dialog and instructional components that convert 3D graphical game engine characters into intelligent dialog agents. The focus of this paper is the reusable dialog agent architecture and its interface to the backend game-engine system.

Motivation

The health communication task that motivates this project concerns high-risk lifestyle choices—substance abuse, gambling, tobacco consumption, dangerous sexual practices, binge drinking, DUI, obesity, overextended credit, and unnecessary sports injuries. Unwise choices are prevalent in spite of nearly universal awareness of the cause-effect relationships between, say, smoking and lung cancer, obesity and diabetes, unsafe sex and HIV. A partial explanation is the difficulty humans have evaluating probability ratios; we are easily thrown off by framing effects (Tversky, et al., 1974) and subconscious "cognitive illusions"(Piattelli-Palmarini, 1994) that undermine rational decision-making about gambles. Added blame goes to seductive messages carefully designed to exploit these cognitive deficiencies. The problem is still further exacerbated in high-stress environments with stretches of idle time that tend to inspire risk-seeking behavior especially in young people. So to be successful, health communication must not only inform patients of correlations, likelihoods, and consequences but also equip them to overcome cognitive obstacles to wise lifestyle choices.

Simulation-based training systems have been successfully applied to a range of instructional goals. Mannequin simulations of trauma patients train casualty care physicians in perceptual-motor skills, rapid situation assessment under stress, and unambiguous human communication often over low-bandwidth, high-noise telemedicine channels. Other simulation-based Intelligent Tutoring Systems (ITS) convey procedural knowledge or 'soft' cognitive skills required for diagnosis and treatment decisions. The lifestyle decision-making skills that are the focus of this project call for a third genre of instructional system designed to convert *inert knowledge* about cause-effect relations into actively usable awareness.

Approach

The training value of any simulation-based ITS depends on the user's ability to feel immersed in the instructional scenario that it presents. That means being able to operate the training application without focusing conscious attention on it as a software system. User-centered engineering (UCE) techniques address this need by guiding the design of training systems that are natural for the learner to use and better optimized to the cognitive tasks that will result in retention and transfer of learning. For some applications, mixed-initiative spoken dialog interaction between human and synthetic dialog agents (DA's) will be a key component of the user-centered design. The dialog mode of interaction supports the indirect learning objective by allowing users to engage the synthetic agents in a way that feels natural and intuitive. Without claiming any advances in the state of understanding of affective computing directly, we hope to

capitalize on the benefits attested in (Picard, 1997). Further, the speech modality lets the user control the simulation while keeping their eyes, hands, and focus of attention on the training exercise and its representation in the simulation.

For this project we designed and prototyped an Intelligent Tutoring System (ITS) that engages learners in a role-play game, HEDONIST (HIV, Economics, Drugs, Obesity, Nicotine, Injury, Safe-sex Tutor). The game integrates the commercial game engine NeverWinterNights™ with AI-based dialog and instructional components that convert game engine characters into intelligent dialog agents. We adapted a tested architecture for knowledge based spoken dialog systems (Luperfoy, et al., 1998) to construct simulation-based training systems with mixed-initiative dialog.

Overview

The first health communication scenario selected is a military situation in which the human player assumes the role of a Commanding Officer (CO) and controls the corresponding Player Character (PC) that engages in spoken dialog with Non-Player Characters (NPC) representing the Executive Officer (XO) and subordinate staff. When the PC is in sufficient proximity with a pair of NPCs engaged in conversation, their voices become audible so that the human player overhears them. Salient aspects of their communicative and non-communicative behavior get recorded as part of the dynamic situational context of the training scenario.

The remainder of this paper describes the generic dialog architecture and the reusable components that used to accomplish this dialog behavior and how they are instantiated for each new dialog application. We then report on the process and result of applying that architecture to the HEDONIST training system that integrates dialog-enabled ITS technology with a popular on-line video game. We will not elaborate further on the instructional content for lifestyle risks, the pedagogical theory chosen for this training application, nor the advantages of the resulting ITS for training effectiveness; our topic for this symposium is restricted to the dialog architecture and the procedure for constructing a system to instantiate any pedagogy requiring dialog processing. We sketch the basic structure of the ITS interaction only to situate our discussion of the architecture that supports it.

The following sections define six categories of computational dialog processing. We derive functional requirements dictated by those categories and present a software architecture that supports those functions. Then we illustrate the process and result of applying the architecture to the conversion of a COTS multiplayer online game into a platform for designing innovative lessons using a simulation-based ITS with dialog-enabled animated 3D agents.

Human-Machine Discourse

The range of discourse phenomena that can occur in a multi-user simulation or game environment used for instruction can be classified into six categories of computational dialog functionality. (1) **Human-system** dialog lets the user command and control the application using a dialog-based user interface, e.g., “Create another enemy tank battalion”, or “Zoom in on that bridge”, or “Bring up the topological map overlay.” (See Walker, et al. 2002 for a comparative evaluation of several human-system dialog interface designs.) (2) **Computer-mediated human-human** dialog lets players, instructors, operator/controllers, and observers communicate with each other in the context of the simulated scenario (Miller, et al. 1996). (3) **Automated analysis** of those human-human dialogs either in real time or retrospectively (Jurafsky, et al., 1997; Glass, et al., 2002) can assist in evaluation of student and/or training system performance. These analyses in (3) can also be used to model spontaneous human-human dialog for a fourth category of computational dialog, namely, (4) **computer-generated** synthetic agent dialog that is overheard by the user as part of the situational context of the unfolding scenario, but involves no human speaker. Examples include synthetic dialog between dueling chat bots available on in EMACS or on the worldwide web. (5) **Human-DA (Dialog Agent) dialog** lets players engage the dialog-enabled synthetic agents of the simulation world and hear contextually-appropriate DA responses, e.g., “Fifth platoon, decrease speed by two zero miles per hour”, “Yes, Sir. Fifth platoon decreasing speed”, “First platoon, what is your position?” (Webber et al., 1995; Goldschen, et al. 1998; Nielsen, et al., 2002). (6) **Human-tutor dialog** is a related form of human-machine dialog that provides a personification of the ITS as a disembodied coach (Luperfoy, 1996) or, depending on the application constraints, as a Non-Player Character (NPC) with an overt screen presentation (Graesser, et al., 2002). The dialog-enabled tutor can offer ‘over-the-shoulder’ verbal coaching during the exercise, or it can collect and save observations for After-Action Review when it can engage the user in meta-level dialog about the lesson, the scenario, or the user’s performance.

While the implementation project referenced in this paper involves all six forms of dialog processing, we restrict this discussion to categories (5) and (6) to illustrate the human-DA dialog between a HEDONIST player and the dialog-enabled NPC that occupies the role of the player’s XO.

Dialog Agent Capabilities

In this section we define the three component capabilities of an intelligent DA. These three modules, Context Tracking, Pragmatic Adaptation, and Dialog Management constitute the central contribution of this paper to ITS engineering.

Context Tracking

Humans as Context Trackers: Spontaneous human language contains context-dependent referring expressions, including pronouns, indexical references ("tomorrow", "us", "that room"), elliptical phrases ("No, it doesn't."), definite noun phrases, and other forms that receive their semantics in full or in part from the context of their occurrence. (See Appendix A for a table of examples.) In order to interpret these **dependent** forms when they occur, humans mentally track the salient elements of the communicative context and the perceptually shared situational context that can **sponsor** the occurrence of subsequent dependent forms. An oversimplified description of the tracking process is that when we hear a new utterance we consult our context representation to find sponsors for any dependent forms in the new utterance, and we add new sponsors to the context representation to prepare for subsequent dependent forms. In human dialog, the speaker's context includes their conceptual model of the state of the listener. That model of the listener guides the speaker's composition of each new dialog utterance. In return, the human listener will often assist the speaker by offering verbal or nonverbal indicators of how well their participation in the dialog is going, e.g., nodding, puzzled facial expression, or vocal back-channeling ("I see," "uh-huh," "Go on.").

Thus, humans come to the human-machine situation well equipped as listeners and speakers, able to hold up their end of a cooperative mixed-initiative dialog. They also bring valuable expectations of how the synthetic DA will behave. We use the human dialog behavior as a development time model to guide our design of a cooperative DA, and at runtime we rely on the mental and communicative behavior of the user to reinforce the DA's algorithms for updating context information.

Dialog Agents as Context Trackers: The DA in Types (5) and (6) dialog can be viewed as a mediator between the human user and the backend software application. Its job is to make sense of each new human input (statement, query, or command) relative to its own internal representation of the context. It must then translate the input into a well-formed command in the language of a backend application. It updates its internal context representation with the input communicative event and issues the command through the backend API. The DA then intercepts the backend response to that command, translates the response into a context-appropriate natural language output utterance (statement, question, suggestion) for the user, and finally, updates its internal context representation with that output communicative event to prepare for the next user input.

For simulation-based ITS dialog, the context tracker must record salient situational as well as communicative information. The DA needs awareness of the backend simulation/game, its possible states, error conditions, its syntax for well-formed input commands, its rules of engagement, as well as events and state changes that happen to occur at runtime.

Consider the following hypothetical input command to a game-based ITS, "Don't do that. Just tell her to go get it and send it to him on the other bridge." To interpret this input utterance as the user's desire that you command Olivia to locate a particular vehicle and report it to Samuel who is on a bridge other than the bridge in focus, the DA will consult its context representation to find at minimum, entities for Olivia, Samuel, the particular vehicle in question, and the spatial layout and location of entities, including at least two bridges, the speaker (PC) and the embodied DA (NPC) itself. To serve as a personified tutor, the DA must also have access to the evolving student model, the training objectives, the curriculum model, remediation options, and other knowledge sources required to deliver the desired pedagogical approach.

The combined situational and communicative context is initialized at the start of the session. Then all salient communicative and situational events/state changes that occur are recorded by the context tracker during the exercise. A DA that maintains even limited versions of the above forms of contextual information can be construed as having 'beliefs' about the external world, the user, the backend application, and about the dialog itself.

Pragmatic Adaptation

Pragmatic adaptation is one way humans and synthetic DA's demonstrate intelligent behavior in dialog: humans and DA's use pragmatic knowledge and the current context to supplement a literal interpretation of each communicative act in order to arrive at an actionable understanding of the speaker's underlying intent. Earlier we said that the DA must translate the user input into a well-formed command in the language of a backend application. This translation process involves pragmatic adaptation of meaning into action. Each human or synthetic DA uses its unique perspective on the world of reference and its internal goals and plans, to decide how to respond to that speaker's intent through an appropriate action, an appropriate verbal response, or both.

Thus, pragmatic adaptation sits at the boundary between communication and action. Understanding your utterance

"Do you know how to open the window?" as a request for specific action is a complex feat requiring resolution of context-dependent forms, indexical references, indirect speech acts, and more. But that is only part of the task. Having understood you, I must still decide on an appropriate action by reasoning about consequences of various actions (or inaction) relative to your intent. For example, I could answer your yes/no question "Why yes, I do," open the window in silence, open the window with self narration "Yes sir, I'm opening the window now," report an execution problem, e.g., tell you that the windows in this building don't open, request a clarification of your intent, "Do you mean this window here or that one there," propose an alternative action "How about if I turn on the air conditioning instead," or simply ignore the request altogether. The Pragmatic Adaptation component of our architecture lets us model this human ability to convert an indirect speech act into the appropriate response based on situational and communicative context.

Dialog Management and Repair

The dialog management skill requires knowing such things as when to interrupt, when to relinquish the floor to another speaker, how to backchannel (e.g., nodding versus vocalization "Uh-huh"), and how to repair disfluencies. Even in human dialog between two people who are well acquainted, dialog disfluencies occur frequently during normal communicative exchange. Thus, competent speakers of all languages have developed skills for preventing, detecting, diagnosing, and repairing the inevitable disfluencies that arise. Indeed, the dialog repair mode so defined is not an aberration but is as much a part of successful interaction as the primary topic dialog.

For humans or DA's, dialog repair requires a repertoire of strategies to deal with various forms of dialog disfluency and to service the needs of a given user relative to a given backend. For example, if the DA is stuck on an ambiguity, it can guess (randomly select one of the interpretations), procrastinate the decision as long as possible, or request a clarification from the user. If the input is interpretable, but it translates to a command that is impossible to execute or nonsensical in the current context, the DA can report the error and suggest an alternative action, it can try to diagnose the problem and present the user with options for action to remedy the situation, or it can make a unilateral repair and watch for objections from the user.

McRoy (1996) presents a thorough treatment of dialog repair including prevention of dialog disfluencies. While prevention is essential for any serious work on repair dialog per se, we will not address it further in this paper. Our description of the repair process (in humans or synthetic DA's) comprises the following subtasks.

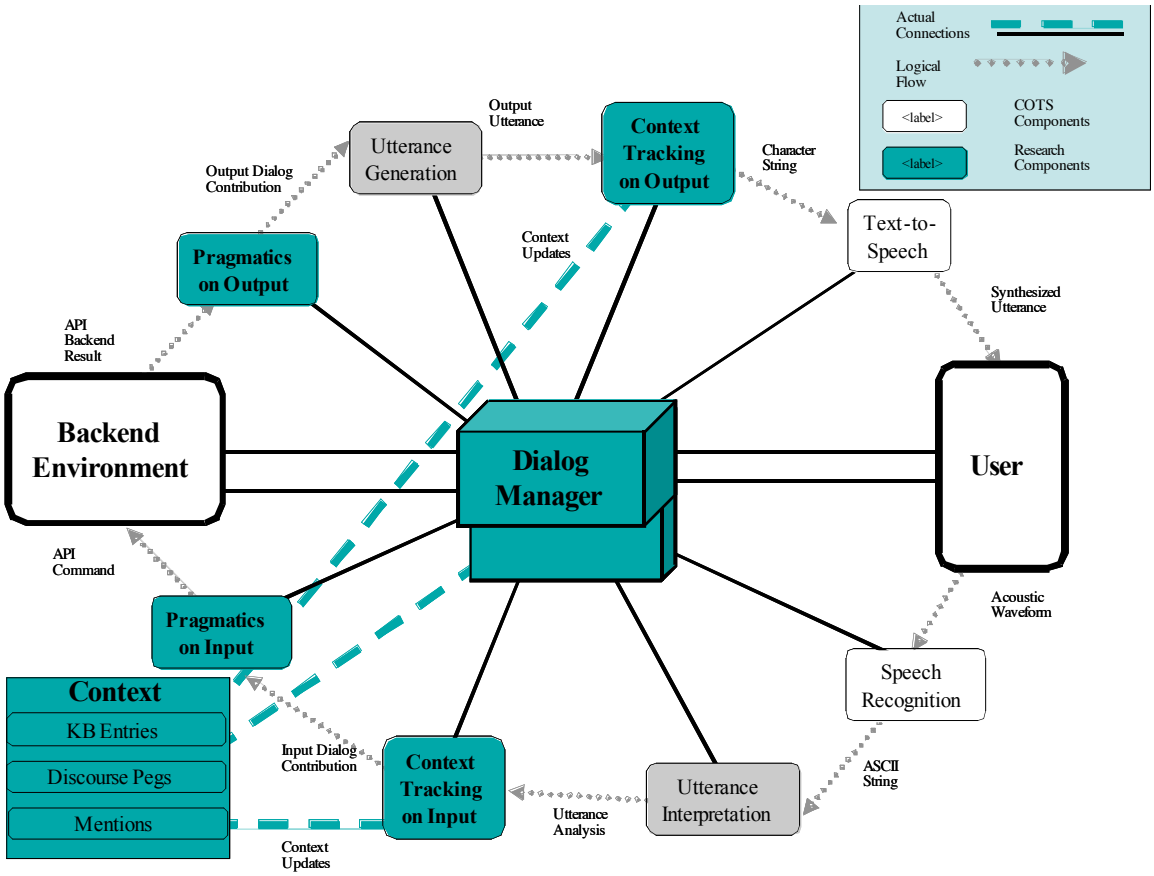
1. **Detect**: One of the parties in the dialog must recognize that there is a problem, otherwise the dialog continues, hampered by propagated effects of the miscommunication.

2. **Diagnose**: A determination must be made as to the source of the dialog trouble. This diagnosis can be made unilaterally, or collaboratively by the two interlocutors. For automated systems, as with human dialog interaction, disruptions to understanding can take place at a number of levels. We have adopted the model of collaborative communication defined by Clark et al. (1997), borrowing their eight levels of presentation acceptance to use as points of potential interpretation failure. In this way we distinguish categories of dialog disfluency based on which component of the dialog agent system indicates difficulty in carrying out its step in the analysis: speech recognition, utterance interpretation, context tracking, or internal elements of pragmatic adaptation (e.g., domain model incompatibility, user model conflict, or ill-formed backend command).

3. **Devise Recovery Plan**: Even given a successful detection and diagnosis, we must query user interface design parameters to determine the preferred method for recovery. For example, upon determining that a user misconception is the cause of an illegally stated question or command, the system has the options of (a) correcting the command without bringing it to the user's attention, (b) correcting the command and reporting back to the user the proper formulation of the command, or (c) reporting the problem without correcting it and suggesting that the user reissue the command using a legal formulation.

4. **Execute Recovery**: For spoken dialog systems this recovery plan must be executed in collaboration with the other dialog agent(s). Since human dialog agents are unpredictable in repair dialogs as they are in primary dialogs, the DA may have to respond to user input that fails to match behavior prompted for during the repair. For example, the repair prompt "Do you mean this bridge?" calls for a yes/no answer but the user can surprise the DA with "That's not a bridge," or "The assembly area," or "Please repeat."

5. **Close and return to the primary dialog**: Once the dialog trouble has been resolved, both system and user must be brought jointly to the understanding that the next utterance is a return to the primary dialog. Options for achieving this return step include an overt closure statement, or appropriate embellishment of or wrapper around the next utterance to unambiguously associate it with the primary dialog.



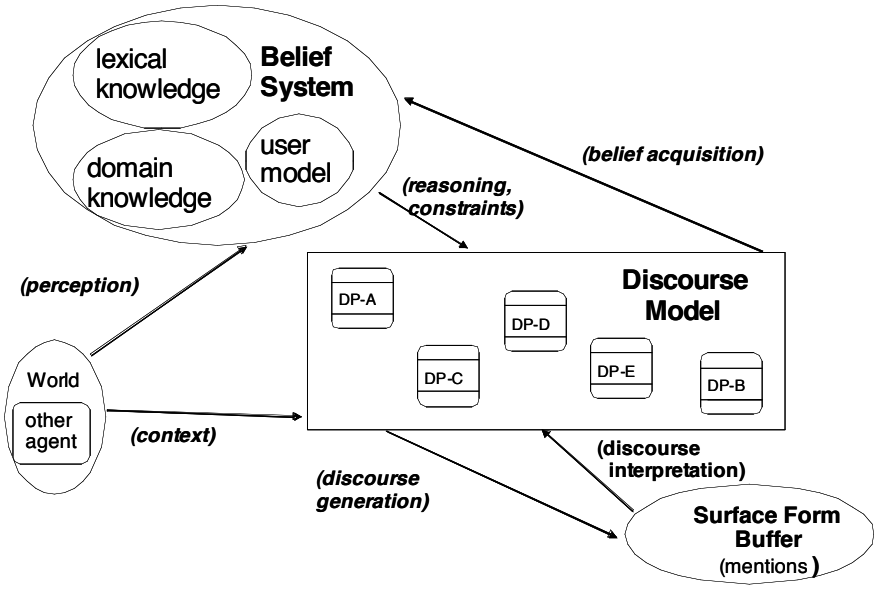
Dialog Agent Architecture

The capabilities of an intelligent DA as defined above are implemented in HEDONIST using the software architecture shown in Figure 1. Each DA has an animated screen character, an NPC in the multiplayer game NeverWinterNights™ that occupies the position of the Backend Environment in the architecture.

Context State Representation

The three-level representation of context shown in Figure 2 is based on a computational theory that partitions information about surface communicative forms (Mentions), discourse-level abstractions of entities that have been mentioned or can be mentioned in the near future (Discourse-Pegs), and knowledge of the world of reference (KB objects). These three types of information available to the DA, have distinct procedures for access, updating, and decay over time. The **Belief System**, implemented as a Knowledge Base (KB), represents the DA's beliefs about the world under discussion, which may include a model of the DA's human dialog partner (stored in the user model), ontologies, rules of inference, analogical reasoning engine, and more. In the current implementation we are using our in-house General Representation Inference and Storage Tool (GRIST) populated with an ontology representing NeverWinterNights™ entities, states, events, and rules of inference.

The **Discourse Model** encodes the DA's current understanding of the information content that is currently in shared focus. Content of the discourse model involves systematic uncertainty and is understood by the DA's meta-cognitive awareness to be potentially incomplete or flawed. Information in the Discourse Model is organized around abstract objects called Discourse-Pegs (DPs), that represent the DA's current focus of attention. DPs decay from prominence only when they are ignored by both speakers. When neither DA nor human user mentions a DP for a time it loses its ability to license, or *sponsor* new dependent forms and is eventually replaced by new DPs for new constructs in focus. There is one **Surface Form Buffer** for each modality channel (keyboard, speech, joystick, mouse, output graphics, eye tracker, etc.) and its content is supplied by a processor that captures input and output communicative events and interprets them to a level equivalent to first-order predicate logic. Unlike DPs, the objects at the surface level, called **Mentions**, decay rapidly as a function of time so that linguistic forms, sounds, keystrokes, etc., are soon lost to the context representation while new Mentions replace them. A new Mention can



refresh an existing DP or cause a new DP to be introduced into the Discourse Model.

This context representation and updating framework was designed (Luperfoy, 1991) to model cognitive processes exhibited in human dialog interaction: the ability to understand an explanation without believing it, the ability to use knowledge about the world and inferential reasoning to construct an internally consistent model of a counterfactual world, the ability to say things that one does not believe to be true, and the inability to interpret context-dependent references to concepts that have fallen out of discourse focus due to simple passage of time or due to overwriting by new communicative events that intervene. The model enables these behaviors by distinguishing discourse interpretation from assimilation of beliefs, distinguishing private perception and internal reasoning, from joint observations of situational context shared between speaker and hearer as the common ground (Clark, et al. 1989).

One Discourse-Peg can relate to multiple surface Mentions. The Mentions may be directly related to each other syntactically, or indirectly related semantically through the DP that they share. For example, in "*They drove an **M1A1** to the assembly area. They were forced to abandon the **tank** to recover wounded so the **vehicle** is still there.*" the Mentions for "tank," "M1A1," and "vehicle" do not show linguistic dependence, but are related semantically through their reference to a common DP. The Context Tracker accesses and updates its own representation of context and lets remaining components of the DA access it to reason about appropriate interpretations and their own next actions.

Dialog Management

The Dialog Manager (DM) is the facility of the DA that controls the interaction between the human user and all system components that contribute to the user's experience of the dialog. This includes Context Tracking and Pragmatic Adaptation. The DM orchestrates the firing of modules to process input speech and generate output responses, update context, and translate input requests and queries into well-formed commands in the language of the backend API, then translate backend output into context-appropriate natural language. The DM property settings control the 'personality' of the DA that the user experiences. Three implementation features that help create DA personality are mixed-initiative dialog, dialog troubleshooting, and back-channeling.

Mixed-initiative interaction has long been recognized for its advantages in user-interface design. Instead of long complicated utterances, commands can be spoken in shorter, more natural segments that are easier to both interpret and confirm. If the user's original input is not sufficient for the Utterance-Interpretation subtask, the DA can elicit the missing information, and even suggest an intelligent choice of default values for remaining gaps based on information from the context manager.

But mixed-initiative interaction requires dialog situation awareness. When the DA is responding by voice or prompting the user, the user must be able to interrupt it or "barge in." Likewise, if the DA is in the process of speaking to the user and any other higher priority event takes place, the DA will interrupt its own output and either discard it or record it for later processing. The DA can be designed to seize the initiative when it needs to get the user's attention for any reason—e.g., when it has completed an off-line task that the user had requested, when there is an incoming call, when a new player logs on to the game, etc.

Dialog-Enabled ITS

The goal of this implementation project is to create engaging instructional game-based ITS that helps learners make informed lifestyle choices, in part by overcoming influences of propaganda and peer pressure. We selected the multiplayer game engine NeverWinterNights™ by Bioware© because of its well designed 3D characters and graphical backdrop, its well-defined API, popularity in the target user community, and the game modules that have been designed by that community. Using the game authoring tools provided by Bioware© we then created our own game modules to serve the idiosyncratic needs of our training system. Then we integrated the game engine into the Backend-System slot of the DA architecture and defined the rules of the game HEDONIST.

The HEDONIST scenario entails a player logging on in the role of Commanding Officer (CO) and engaging a dialog-enabled NPC in the role of the CO's Executive Officer (XO). Other NPCs that populate the CO's staff are termed At-Risk Agents (ARAs). NPCs (ARAs and XO) converse with each other in purely synthetic dialogs that become audible to the player and get recorded as part of the shared context, only when the Player's Character (PC), i.e., the avatar for the CO that represents the player, moves into proximity of the NPC avatars who are talking.

An ARA's lifestyle choices are driven by its internal configuration of decision functions. The player's is to persuade each ARA to modify its internal function so that as many ARA's as possible make wise lifestyle choices. The player can only modify ARAs indirectly, through argumentation and discussion. As a player in the role of CO progresses in mastery, the challenge is increased through assignment of larger numbers of ARAs who are more difficult to persuade and/or who lack self-discipline to make choices consistent with their internal values.

Status and Future Plans

The HEDONIST implementation is a limited prototype with minimal versions of each component in the DA architecture. In this prototype, the player, XO, and ARAs are restricted to dialogs about smoking. After the DA interprets a spoken input from the user, "How many people in my staff are smokers?" "How long has Andrew been smoking?" or "Tell Andrew to stop smoking," it translates the interpreted result into a well-formed command to the backend, e.g., an SQL query, or a call to the natural language output system to generate an audible utterance such as "Andrew, stop smoking" that gets conveyed from the XO to the ARA named Andrew. In the prototype, ARAs are blindly obedient and robotically self-disciplined so that the XO always conveys the message to the ARAs who always change their value function in response to CO commands and always choose behavior consistent with their value functions.

The prototype leverages our own prior results for knowledge-based DA construction—including the architectural framework, the Context Trackers (for input and output), our GRIST Knowledge Base system that constitutes the third tier of the Context Representation, the Pragmatics Adaptation modules (for input and output) that form the boundary between communication and action, and the Dialog Manager that controls the overall interaction.

We constructed a temporary grammar and lexicon and defined finite state machines for sentence parsing and generation of output utterances. These are introduced strictly as placeholders for future components to be obtained from sources of mature technology available in the computational linguistics community. In the prototype grammar (enumerated below) personal pronouns are recognized as legal fillers of the <person> slot, and are resolved relative to the Mentions and DPs in the current Discourse Model. When the prototype ITS game is initialized, Tim and Sally are the only known ARAs and the player can introduce new ARAs to their staff through assertions to the XO, e.g., "George reports to me." The current placeholder lexicon allows the <person> slot to be filled by Tim, Sally, George, or Andrew and the prototype grammar contains these formulas:

<smokeQues> = does <personOrPronoun> smoke;

<smokeJust> = why does <personOrPronoun> smoke;

<follow> = follow me | come here;

<stopFollow> = stay here | stop following me | stop there;

<age> = how old is <personOrPronoun>;

<smokeStart> = when did <personOrPronoun> start smoking;

<subjectPronoun> = he | she | they;

<objectPronoun> = him | her | them;

<smokeCommand> = tell (<person> | <objectPronoun>) to stop smoking;

<numberSmokers> = how many smokers are there;

<smokingTell> = <personOrPronoun> smokes <quant> (pack | packs) a day;

<personCreate> = <person> is in my unit;

<quant> = one | two | three;

<person> = Tim | George | Andrew | Sally;

<personOrPronoun> = <person> | <subjectPronoun>;

We used COTS or open source products to populate remaining components of the architecture, including STT (speech to text), TTS (text to speech), and the commercial NeverWinterNights™ game engine. The prototype system runs in distributed client-server mode over a local network of personal computers. Appendix B enumerates the utterances and corresponding responses from the prototype DA.

Summary

Bad decisions involving health, safety, finances, and legal risks continue to harm individuals, degrade job performance, and increase health care costs. Self-destructive behavior does not always result from a lack of knowledge about cause-effect relations involving lung cancer, AIDS, and the dangers of DUI. Rather, lifestyle decisions involve non-rational reasoning and emotion so our game-based ITS is designed to approach the learner on a cognitive and affective level that is inaccessible to rational argumentation. We hope to address three causes: (1) human inability to evaluate probability ratios; (2) seductive advertisement and peer pressure that promote high-risk behavior, and (3) idle time that encourages risk-seeking behavior. The prototype domain of instruction was the risks associated with tobacco, and the research aim was helping users to overcome psychological limitations stemming from framing effects.

We constructed the game HEDONIST by integrating our existing architecture for knowledge-based spoken dialog interaction with a COTS videogame. While the dialog architecture has been used in prior implementations to personify the disembodied controller of a simulation system, and to allow synthetic agents to respond to spoken commands, this is its first application to transforming a reactive but non-communicative Non-Player Character (NPC) into an intelligent Dialog Agent. The resulting proof-of-concept demonstration represents the starting point for ITS applications to address a range of training objectives. The hypothesis to be tested is that such a system can help decision makers to be less vulnerable to destructive propaganda and fallacious but intuitively appealing arguments that downplay negative consequences of risky behavior.

Acknowledgements

Jim Ong, David Duff contributed technical suggestions on the design and construcion of the demonstration system. Jeremy Ludwig helped design and implement the NeverWinterNights™ API link and remaining software development was performed by Eric Domeshek, Eli Holman, and David Struck of Stottler Henke Associates.

References

Clark, H. and E. Schaeffer (1989) "Contributing to Discourse" Cognitive Science 13:259-294.

Duff, D. and S. LuperFoy (1996) "A Centralized Troubleshooting Mechanism for a Spoken Dialog Interface to a Simulation Application" International Symposium on Spoken Dialog, Philadelphia.

Glass, M. and B. DiEugenio (2002) *MUP The UIC Standoff Markup Tool*. In proceedings 3rd SIGDial workshop on Discourse and Dialog.

Goldschen, A., Harper, L.D., Anthony, E.R. (1998) The Role of Speech in a Distributed Simulation: The STOW-97 CommandTalk System.

Graesser, A., et al. (2002) "Why-2 Auto Tutor" oral presentation to ONR workshop on Tutorial Discourse.

Jurafsky, D., Bates, R., Coccaro, N., Martin, R., Meteer, M., Ries, K., Shriberg, E., Stolcke, A., Taylor, P., and Van Ess-Dykema, C. (1997) "Johns Hopkins LVCSR Workshop-97 SWBD Discourse Language Modeling Project Final Project Report.

Luperfoy, S. (1991) "Discourse Pegs: A Computational Analysis of Context-Dependent Referring Expressions." Ph.D. dissertation, University of Texas at Austin.

LuperFoy, S. (1996) "Tutoring Versus Training: A Spoken Language Dialog Manager for Instructional Systems" TWLT-11 Twente Workshop on Language Technology Number 11, Dialog Management in Natural Language Systems. University of Twente.

Luperfoy, S., D. Loehr, D. Duff, K. Miller, F. Reeder, and L. Harper (1998) "An Architecture for Dialog Management, Context Tracking, and Pragmatic Adaptation in Spoken Dialog Systems". In proceedings 36th Annual Meeting of the Association for Computational Linguistics.

McRoy, S. and G. Hirst (1995) "The Repair of Speech Act Misunderstandings by Abductive Inference" Journal of Computational Linguistics, vol. 21 no. 4.

Miller, K., S. LuperFoy, E. Kim, D. Duff (1995) "Some Effects of Electronic Mediation on Spoken Bilingual Dialog: An Observational Study of Dialog Management for the Interpreting Telephone" Electronic Journal of Communication.

Nielsen, P., Koss, F., Taylor, G., Jones, R. M. (2002) Communication with Intelligent Agents, in proceedings I/ITSEC 2002.

Piattelli-Palmarini, M. (1994) Inevitable Illusions: how mistakes of reason rule our minds. Wiley & Sons.

Picard, Rosalind (1997) Affective Computing. MIT Press.

Walker, M., A. Rudnicky, R. Prasad, J. Aberdeen , E. Bratt, J. Garofolo, H. Hastie, A. Le , B. Pellom, A. Potamianos, R. Passonneau, S. Roukos, G. Sanders, S. Seneff, D. Stallard. (2002) "DARPA Communicator Cross-System Results for the 2001 Evaluation" in proceedings, ICSLP (International Conference on Speech and Language Processing).

Webber, B. (1995) Instructing Animated Agents: Viewing Language in Behavioral Terms. *Proc. International Conference on Cooperative Multi-modal Communication*, Eindhoven, Netherlands, May.

APPENDIX A: Examples of Context Dependence in Human Dialog

Phenomenon	Example	Comments
Intra-sentential anaphora	*DA: And so, what about nicotine, is* **it** *also addictive?*	The pronoun finds its **sponsor** (nicotine) in the current sentence.
	USER: Yes, only 26 people lost **their** lives?	"26 people" sponsors "their"
Inter-sentential anaphora	USER: Yes, **it** is a addictive substance that can hook you quickly.	Here, the pronoun finds its *"sponsor"* (nicotine) in an earlier sentence.
Ellipsis	*DA: Given your definition of illegal, is binge drinking also illegal?* USER: Yes **it is**.	Reconstruction of the elliptical expression yields, "Yes, it is also illegal." The sponsor of "it" is DDT so pronoun resolution yields, "Yes, DDT is also toxic."
	DA: **By whom**?	Main verb is elided
	DA: And **what about** nicotine?	The operator "what about(x) " gets the interpretation APPLY-PROPOSITION-TO(x)
Discourse Deixis	USER: **That** is controversial.	The deictic adverbs, "this," "that," "these," "those," etc. are *sponsored* by something in the prior discourse.
	USER: No it was earlier than **that**.	
	USER: **This** is undecided.	
Totally Dependent Definite Noun Phrase	*DA: Okay. I understand* ***the term*** *now.*	This "term" is a second mention of (and sponsored by) a term that was mentioned earlier.
Partially Dependent Definite Noun Phrase	DA: Was it **the purpose**	*The "purpose" is new to the discourse but dependent on a purposeful event mentioned earlier.*
One-Anaphor	DA: Okay, you are telling me about **one** in which nicotine killed someone.	*This new event partially depends on a concept mentioned earlier, the class of deadly events.*
Quantifier as One-Anaphor	USER: I am not aware of **any**?	Quantifiers can behave as one-anaphoric expressions introducing new entities by depending on entities mentioned earlier
	Does serin gas have ***other*** *uses, unrelated to warfare?*	
Indexical	It has been active since 1982 and **now** has members numbering upwards of---	Functional relationship between indexical expressions, "now," "me," "you," "here," "yesterday," etc. and the situation of the utterance.
	DA: Okay, **you** are telling **me** about one in which serin gas was released into a population.	
	USER: **I** am not aware of any?	

APPENDIX B: Input Utterances and Corresponding Responses

Utterance Template	Dialog Agent (XO) Response
Does <person> smoke? How old is <person>? Why does <person> smoke? When did <person> start smoking?	Queries the GRIST knowledge base looking for a certain type of assertions. If one is found, the content is used to generate an appropriate response. If not, a standard response is used. If the <person> named is not a known symbol, the system replies as such.
Stay here. Stop there. Stop following me.	Sends a message to the NeverWinterNights™ server telling it to turn off the XO following behavior.
Come here. Follow me.	Sends a message to the NWN server telling it to turn on the XO following behavior.
<person> is in my unit.	If a person is referenced who is not currently in the knowledge base, then a new Symbol with that name is created. Otherwise, the system replies that it already knew that.
<person> smokes (one | two | three) (pack | packs) a day.	Adds or replaces an assertion into the GRIST knowledge base that matches the content of the input utterance. If <person> is not resolvable that is indicated.
How many smokers are there?	Queries the knowledge base to determine how many smokers there are. This is done by looking for agents that have a smokingQuantity slot filled in. This is the slot that is queried by the Does <person> smoke query, and is updated by the input utterance just above.
Tell <person> to stop smoking.	If <person> is resolvable, and a smoker, then the XO says "<person>, stop smoking", and <person>'s smokingQuantity slot is set to null, effectively decrementing the number of smokers. If <person> is not a smoker, the XO says that the task cannot be done.
Who are the smokers?	Like 'how many smokers are there', this query checks the knowledge base to see who is a smoker, but instead of just counting, it puts them into a list.