Character-Based Interactive Storytelling

Marc Cavazza, Fred Charles, and Steven J. Mead, *University of Teesside, UK*

Interactive storytelling is a privileged application of intelligent virtual-actors technology. The authors introduce their character-based interactive storytelling prototype that uses Hierarchical Task Network planning techniques, which support story generation and anytime user intervention.

Interactive storytelling promises to be an important evolution of computer entertainment, introducing better narrative content into computer games and potentially supporting the convergence of traditional and interactive media. Previous work has described several paradigms for interactive storytelling,[1–3] each differing on various dimensions such as user involvement and relations between the character and plot. Our approach is character-based and essentially follows Michael Young's proposal[2] that autonomous actors, whose roles are implemented using real-time planning systems, should dynamically interact to generate the story.

Within the many possible implementations of interactive storytelling, we target a specific kind of application: letting users interfere, at any time, with a predefined storyline's progression. Furthermore, rather than give instructions, users can alter the environment by stealing an object or influence other characters by offering advice. The consequences of this intervention then affect the characters' behavior and alter the course of action, creating new dramatic situations and eventually leading to different story endings.

System overview

We developed our prototype using the Unreal Tournament game engine as a development environment (see www.unrealtournament.com). The interactive story appears as a real-time 3D interactive animation with subtitles corresponding to the characters' dialogue or important events. Users can physically interact with the characters and navigate through their environment using normal game controls, or they can verbally interact with them using a speech recognition system.

The test scenario we have been using is inspired by the popular US television sitcom *Friends* (www.nbc.com/Friends).[4] We chose a sitcom because, in this genre, the story ending and intermediate situations are equally relevant, which provides a more appropriate testbed for story generation. Furthermore, when developing the system, we defined various roles for each feature character and formalized these roles as plans; when the system executes a plan, it generates character behavior at runtime. Decomposing a plan into subgoals reflects an action's different stages, while the lower layers of the plan decomposition correspond to various ways to achieve these goals. For example, if the character Ross wants to ask out Rachel, then he must acquire information about her, gain her friendship, find a way to talk to her in private, and so forth. He faces several possibilities at each stage—for example, to gain information, he could steal her diary, talk to one of her friends, or phone her mother. These various possibilities correspond to subgoals in the description of Ross's plan, which can be further refined in the plan representation until they can be described in terms of terminal actions (that is, elementary actions carried out by the characters). The system then plays the actions in the virtual environment using standard Unreal animation sequences or additional animations that have been imported into the system.

One particularity of this character-based approach is how it uses the same basic mechanisms to support both story variability and interaction. Plan-based roles for the various characters are dynamically combined to generate multiple variants of an initial storyline.

1094-7167/02/$17.00 © 2002 IEEE

In the absence of any user intervention, this mechanism will produce a variety of plot instantiations. At the same time, user interaction can interfere with the characters' plans (for example, causing action failure) and trigger a replanning that varies the plot.

In our system prototype, we modeled the graphic environment using the game's level editor and modeled additional objects using 3d studio max and textures from several online resources. We imported the characters from online repositories (Brian Collins created the Ross character, "Austin" created Rachel, and Roger Bacon created Phoebe and Monica). We implemented the AI layer in C++ and integrated it in Unreal as a set of dynamic link libraries. UnrealScript defines all the functions that interface with Unreal's events—that is, those functions dealing with object interactions. We also fully integrated communication into Unreal using a speech recognition system (Babel Technologies' Automatic Speech Recognition (ASR) software development kit).

Planning techniques for character performance

A wide range of AI techniques has been proposed to support interactive storytelling systems, including planning techniques[1,2,4,5] and techniques for augmented truth-maintenance systems.[3] The technique used often depends on the interactive storytelling paradigm being implemented. However, there is no direct correlation between a given AI technique and a storytelling paradigm. For instance, Young has used planning to control the narrative rather than just the behavior of individual autonomous characters;[2] William Swartout and his colleagues have used planning for autonomous characters, but they also rely on causal narrative representations.[5]

We are mainly interested in the emergence of story variants from the interaction of autonomous actors, so our emphasis has been on the actors' behavior rather than on explicit plot representation or narrative control. Character-based systems provide a unified principle for story generation and interactivity. As such, they allow anytime interaction, whereas plot-based systems tend to restrict user intervention to selected key points in the plot representation. However, we still needed our planning formalism to accommodate the authoring aspects of the baseline narrative.

These knowledge-representation requirements led us to investigate planning techniques that we could use in knowledge-intensive domains, and we eventually opted for Hierarchical Task Networks planning.[6] We picked HTN planning because it is generally considered appropriate for knowledge-rich domains, which can provide domain-specific knowledge to assist the planning process.[7] It also appeared that we could naturally represent the characters' roles, which serve as a basis for our narrative descriptions, as HTNs in which the main characters' goals are decomposed into alternative actions.

Hierarchical Task Networks

A single HTN corresponds to several possible decompositions for the main task—in other words, we can view HTNs as an implicit representation for the set of possible solutions.[8]

Character-based systems provide a unified principle for story generation and interactivity. As such, they allow anytime interaction.

In the present context, each ordered decomposition constitutes the basis for a character's plan, and each HTN associated with an artificial actor contains the set of all possible roles for that character across story instantiations.

Although the set of all roles is sufficient, the set of story instantiations is at least an order of magnitude larger, because the story is composed of situations that are the cross-product of the actors' roles. This also provides a principled fashion for authoring these story variants, because that goal node in the network can subsume several ways of solving a narrative goal. For instance, if Ross needs to talk to Rachel in private, he can isolate Rachel from her friends by calling her aside, attracting her attention, asking her friends to leave, and so forth. This makes it easy to refine potential variants by adding extra options at authoring time. As representations, HTNs can capture essential properties of a character's role through the actions the agent takes toward its goals and the choices it faces.

There is a further need to categorize these actions according to narrative criteria. These categories should represent properties bearing relevance for intercharacter relationships, which we can match to the various actors' personalities. For instance, actions targeting other actors can be classified as "friendly," "rude," and so forth. If, when faced with the task of talking to Rachel in private, Ross interrupts her previous conversation and sends her friends away, we would tag the corresponding option in the HTN as "rude." In a similar fashion, we can categorize single actors' occupations according to their degree of sociability—for example, "lonely" or "sociable."

To some extent, these categories are part of an ontology of intercharacter relationships and can help determine how other characters will react to the actions taken. Intercharacter relationships, although obviously important in a *Friends* context, are a generic problem in interactive storytelling. The contents of the HTN are determined by considering each actors' role in the baseline story in isolation. These roles can be refined by providing additional options (this refine process is naturally supported by the HTN formalism). The search mechanisms associated with HTN planning also makes them a useful tool for debugging. Because HTNs are searched from the root node, which is also the main goal, it is easier to gain access to the corresponding state of the world. One additional reason for selecting HTNs as a formalism is that their graphic nature seems more supportive of the authoring phase than STRIPS-like planning formalisms. However, we have not yet been able to test this assumption with professional scriptwriters.

Figure 1 gives an overview of a typical HTN for a character. Pre- and postconditions for the various tasks (not explicitly represented in the figure) are associated with each task node. Preconditions for the lowest-level operators are constituted by the conjunction of executability conditions for their associated terminal actions (those actually acted in the 3D environment). For instance, if Ross wants to read information from Rachel's diary, the diary should be at its initial location, not in use by another agent or near any witnesses. Some of these conditions are obviously subject to change in a dynamic environment, so they become a main vehicle for interaction. The system directly implements postconditions through the effects of terminal actions, which are rolled back to the highest-level task node subsuming these actions.

Furthermore, we can compare HTNs to other forms of knowledge representation proposed in interactive storytelling. In particular, there is a formal equivalence between

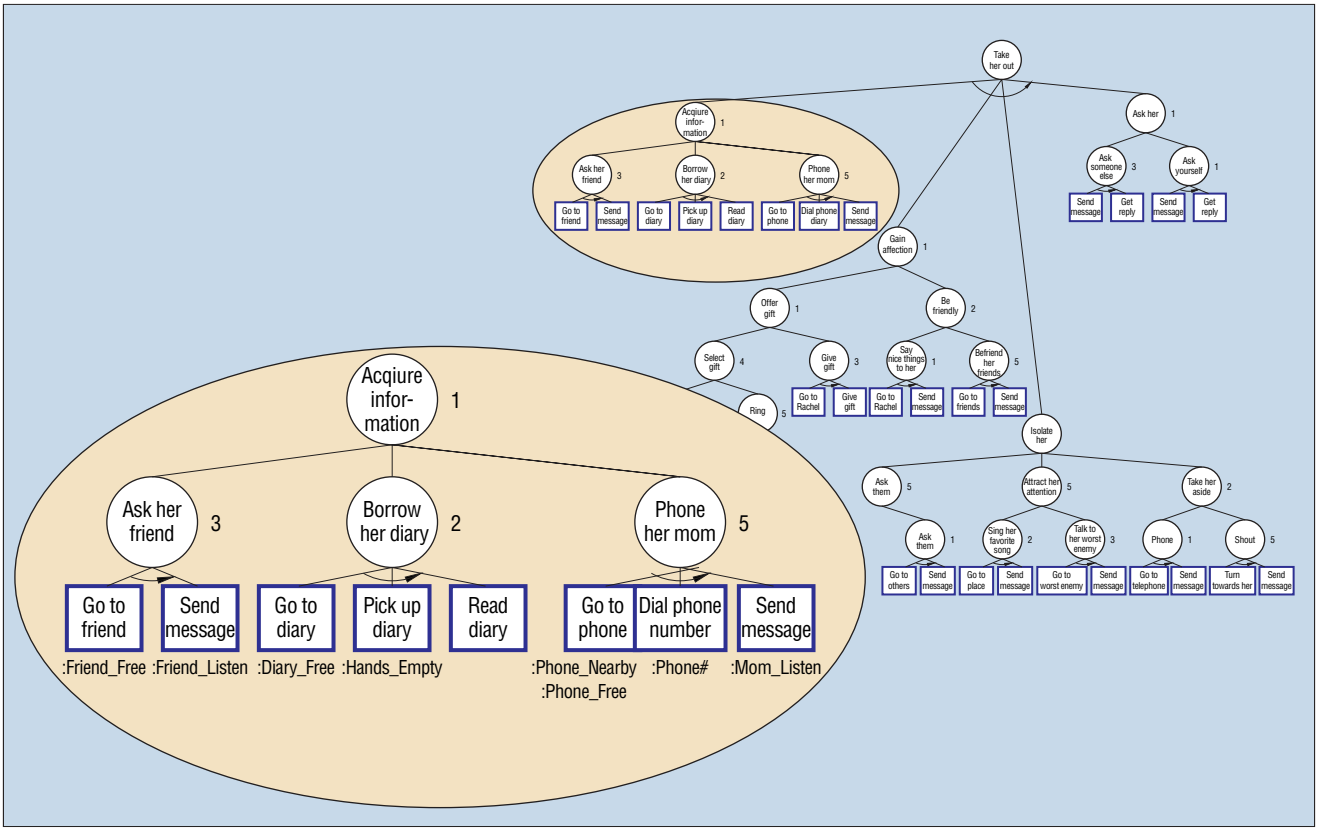

Figure 1. A Hierarchical Task Network for the main character, Ross.

subtasks of the HTN and narrative functions described in narratology that stand for key narrative actions seen from a given character's perspective. The difference lies in the fact that the *agentive* (or predicative) structure for the equivalent narrative functions lies outside the corresponding portion of the HTN, in the interaction with narrative objects and other characters filling up the roles for that narrative function. For instance, when seeking information about Rachel, Ross could talk to her friend Phoebe. If he talks to Phoebe, she will complement the agentive role of the corresponding narrative function. Also, whenever multiple characters interact, they potentially instantiate narrative functions "bottom-up" through the conjunction of activities from their respective HTNs.

HTN planning

Interactive storytelling requires interleaving planning and execution.[2] We have thus devised a search algorithm to produce a suitable plan from the HTN. Exploiting our total ordering assumption and subtask independence, the algorithm searches the HTN depth-first and left-to-right and executes any primitive action it encounters in the process. It allows backtracking when primitive actions fail (such as following competition for action resources by other agents, or user intervention). In addition, it attaches heuristic values to the various subtasks, so forward search can use these values to select a subtask decomposition (this is similar to the use of heuristics that Peter Weyhrauch described to "bias" a story instantiation[9]).

An essential aspect of HTN planning is that it is based on forward search while being goal-directed at the same time, because the top-level task is the main goal. (Other recent forward-search planning systems, such as the Heuristic Search Planner[10] or MinMin,[11] search forward from the initial state to the goal.) Consequently, because the system is planning forward from the initial state and expands the subtasks left-to-right, the current state of the world is always known (in this case, the current state reached by the plot).

When initially describing the roles, we chose to adopt total ordering of subtasks. Total-order HTN planning precludes the possibility of interleaving subtasks from different primitive tasks, thus eliminating task interaction to a large extent.[6] In the case of storytelling, the subtasks are largely independent because they represent the story's stages. Decomposability of the problem space derives from the inherent decomposition of the story into various stages or scenes—a classical representation for stories. Our use of HTN is currently associated with substantial simplifications of the associated planning problems, such as subgoal independence, empty delete lists, and total ordering of subtasks at AND nodes. However, this approach to planning seems consistent with the knowledge-intensive nature of interactive storytelling and some of its inherent properties, such as the temporal ordering of various scenes. Other planning techniques—ones more oriented toward a problem-solving approach, for example—could be used, such as one that manages resources and orders actions (see, for instance, D. Weld's "dinner date" example, which describes planning in a domain similar to our sitcom example[12]). However, it is still unclear under which conditions a more generic approach will benefit interactive storytelling.

In addition to their top-down plans, characters also react to specific events. For exam-

Figure 2. The emergence of situations: Ross meets Rachel by accident while still in the early phase of his plan.

ple, Rachel might become jealous whenever she sees Ross talking alone to another female character, or she might be upset if he is rude to one of her friends. These reactions dynamically update "mood" values that affect the other characters' plans. There is thus more to authoring than just describing the various subtasks for each actor's role in an HTN. It is also necessary to describe the character's reactions to various generic situations, mostly arising from the conjunction of actions from the characters' respective plans.

Interactive story generation

One main challenge in generating a story using a character-based approach is achieving story variability while preserving a well-defined story genre. In other words, in the course of various plot instantiations, different situations occur that generate different endings. However, these situations should generally fall in line with the sitcom genre. Having a consistent genre helps the user understand the course of events and decide whether to intervene and in what fashion.

Story generation results from dynamic interaction between the main characters' plans,[4] which correspond to a top-down approach, because characters' behavior is generated from their predefined HTNs. However, in the course of the action, situations might emerge that do not form part of the initial plans. The interaction between characters' plans results in random onstage encounters between agents that have the potential to create situations of narrative relevance. These interactions constitute a bottom-up approach (because plan-based behaviors don't account for these situations) and thus create a need for two specific mechanisms: *situated reasoning* and *action repair*.

Situated reasoning in plan-based actors' behaviors[13] originates from the discrepancy between an agent's expectations and action preconditions. One defining aspect of situated reasoning is that it is oriented toward obtaining a specific resulting state in a given situation.[13] Situated reasoning should include avoiding an undesirable result. One such example in interactive storytelling consists of reacting to situations that emerge from the spatial interactions of artificial actors. The system randomly positions the characters on the set before the story begins. Consequently, although characters will try to follow their independent plans, they might find themselves in situations that are not (and cannot be) explicitly represented as part of their plan—and the system can't ignore these situations.

One example is Ross meeting Rachel by accident while he is still at the early phase of his plan (see Figure 2). He can choose to talk to her or hide from her, but he can't, from a narrative perspective, walk past her without any interaction. One option that situated reasoning offers is to hide from her, and a user can implement this action by interrupting Ross' current action. Ross could also resume his initial plan: If his current action is to meet Phoebe, he can return to her after Rachel passes (not noticing him). In this specific case, hiding from Rachel does not impair subplan continuation.

Consider a similar case, where Ross wants to talk to Phoebe without Rachel knowing because he's afraid Rachel might get jealous (a feature actually implemented in the system). He might wait, but unlike the diary, Phoebe can in the meantime move to another location or engage in other activities, causing the initial intended action to fail. The interruption caused by situated reasoning can thus have an irreversible impact on the initial plan whenever time and duration or location constraints appear. However, even in this case, situated reasoning (hiding from Rachel) preserves the plot's relevance and coherence, because it is properly dramatized and constitutes a part of the story.

One of the main causes for action failure is not satisfying executability conditions. Consider the case where other agent behaviors affect the executability conditions.[11] One example is Ross needing to access Rachel's diary early in the story. This action can fail in several cases (corresponding to different contexts): the user hides the diary, Rachel is writing in it, Ross' sister Monica is in the same room so he cannot steal it, and so forth. The first case imposes replanning, because action repair cannot be applied to the user's nondeterministic behavior (for example, the user likely won't return the diary). The second situation can be a target for action repair, because Ross could simply wait until Rachel has finished her task. More interestingly, the latter case offers the widest range of options. Ross can choose another source of information about Rachel, wait for Monica to leave the room and resume his initial plan, or try to influence Monica so that he can still carry on his original action.

There is sometimes a fine line between action repair and situated reasoning. Strictly speaking, action repair should be dedicated to recovering from action failure. However, in our storytelling context, action failure is most often due to not satisfying executability conditions due to external factors. For instance, Ross cannot read Rachel's diary because it is missing, Rachel is using it, or Monica is in the same room. In other words, action repair is dedicated to restoring executability conditions or reaching the same final state as the original action, whereas situated reasoning essentially consists of inter-

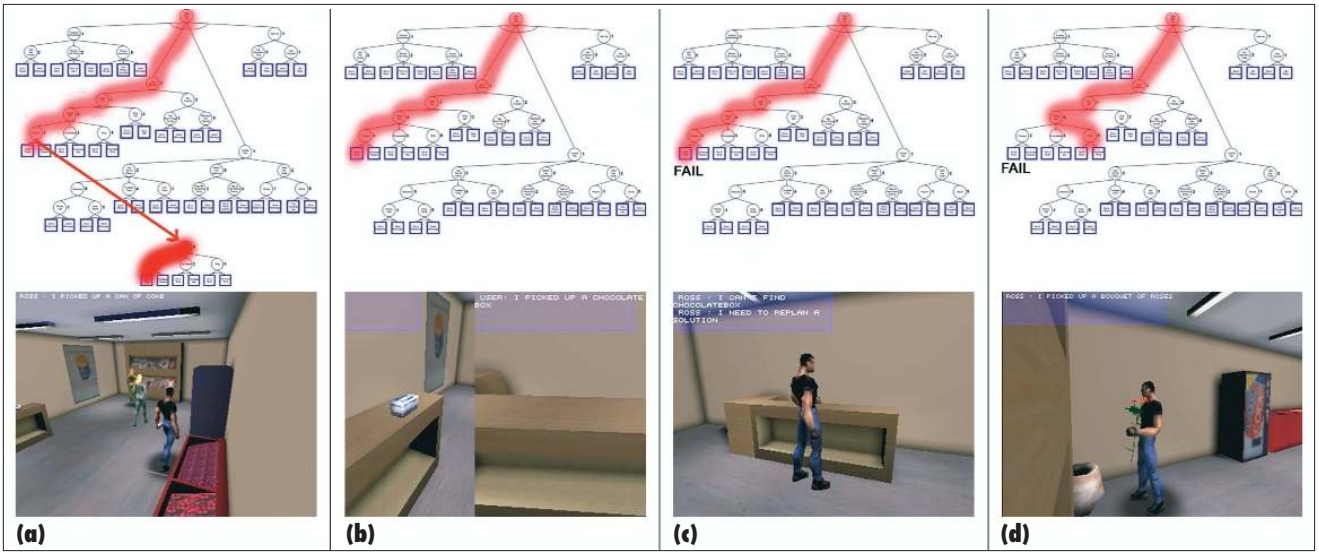

Figure 3. User intervention: (a) Ross goes to get a box of chocolates. (b) The user sees this and steals the chocolates. (c) Ross can't find them, so he (d) replans and gets roses instead.

rupting the current plan and dealing with a specific situation. It hence does so more from the dramatization perspective than from the planning perspective.

Although the basic elements of actors' behaviors are deterministic, several factors contribute to make the action nonpredictable from the user's perspective:

- The actors' initial positions on stage
- The interaction between actors' plans—the various characters essentially competing for resources for action (whether narrative objects or other characters)
- The random output of some terminal actions
- The characters' mood status
- User intervention

For instance, the initial positions on stage strongly influence the emerging situations. Depending on their positions and activities, Ross might not be able to acquire information from Phoebe before she leaves the apartment to go shopping. Consequently, similar conditions or user interventions might not always produce the same results.

User intervention and plot variation

The user watches the story as a spectator. He or she can follow the story from any character's perspective or navigate the virtual set while the action is in progress. Then, depending on the situation, the user can choose whether to interfere with the characters' goals. Characters' actions are dramatized through the timing of appropriate animations. Because the actors are playing a role rather than improvising, their actions are always narratively meaningful. Hence, if a character moves toward a given object, it likely bears significance on the story and can be a target for user intervention (for instance, if the user sees Ross moving toward Rachel's diary, he or she can steal or hide the diary).

Users can intervene any time—they don't need to wait for key situations or for the system to prompt them. However, it is important that they understand the story. Thus, users should be aware from the onset of the overall dramatic situation—namely, Ross' interest in Rachel. The system can best convey this using an opening full-motion video sequence, generated with the game engine.

A user can intervene by either acting on physical objects onstage that bear narrative relevance or by advising the characters using speech recognition. The possibility for physical intervention is based on the notion of narrative objects. These objects act as *dispatchers*—that is, they bear narrative significance because they are the compulsory objects of key narrative functions. Dispatchers naturally arise from the current course of action: when Ross seeks a gift for Rachel, objects such as flowers, chocolates, or jewelry become explicit potential targets for user interaction. These objects, now resources for actions, can force the character into replanning or action repair, thus creating a new course for the plot. The user simply uses the Unreal Tournament's ordinary "player" features to navigate in the virtual set to steal or hide narrative objects (the user, however, is not embodied through a character and thus maintains spectator status).

In Figure 3, a user steals the chocolate box, so Ross must offer Rachel roses (which happens to be a favorable gift). This situation can correspond to various sorts of user interventions, depending on the user's understanding of the plot. The user could have realized that Phoebe lied about Rachel's preferences and tried to help Ross. Or, the initial intention might have been to interfere with Ross' plan, in which case the user involuntarily helped him. Dispatchers crystallize choices both from the characters' perspective and from the user standpoint, the latter having to decide whether to interfere. We do not resort to the traditional notion of affordance nor to its implementation in current computer games, where potentially reactive objects are often signaled as such. Rather, we intend to use the same kind of narrative cues as traditional media, such as camera close-ups in films.

The other mode of interaction consists of influencing actors using speech recognition. Speech intervention is the most natural way of influencing the characters and is ideally suited to the interactive storytelling paradigm of user-as-spectator. Several interactive storytelling systems have reported the use of linguistic interaction,[1,5] essentially in the form of user–agent dialogue. The rationale being that, in these systems, the user is a member of the cast and acts by engaging in conver-

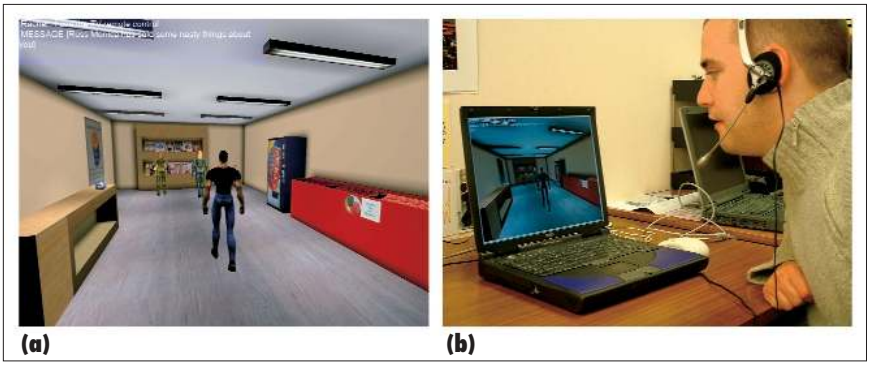

Figure 4. Giving spoken advice to characters: (a) Ross heads toward Rachel's room to read her diary; (b) the user warns him that Rachel is in the room.

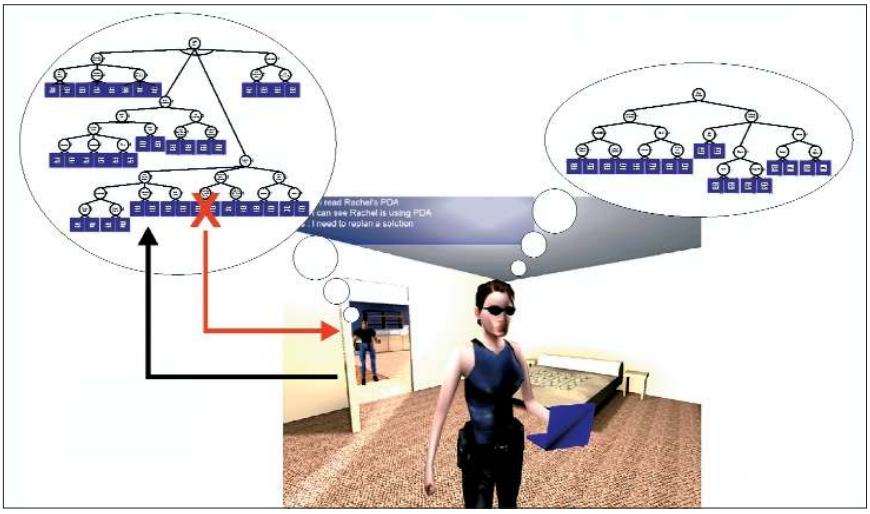

Figure 5. Character interaction and replanning.

sation with the virtual actors.

However, to be in line with our interaction paradigm, spoken input should not take the form of direct commands; otherwise, the user's role would shift from spectator to director. We designed our speech interface to analyze user advice in the form of isolated utterances, whose average length is between seven and 10 words. The grammar we defined for the ASR speech recognition system determines the linguistic coverage for user input. We designed this grammar using habitability principles—that is, syntactic and lexical variants that provide sufficient flexibility without requiring the user to memorize specific commands. We encoded the recognition grammar as flexible templates, which include optional sequences; we've encoded 90 such grammar rules into the system thus far. These provide sufficient coverage for an experimental system but would have to be greatly enhanced for a complete application. We performed a second level of template matching on the output from the speech recognition system, which associates semantic features with the recognized words. The resulting templates correspond to the semantic content of the user utterance, which influences the character's plan.

Giving advice rather than instructions assumes information of a more implicit nature (see Figure 4): for instance, when Ross heads toward Rachel's room to read her diary, the user might warn him that Rachel is in her room. To correctly process such advise, the system must recognize it as a speech act. We can prepare the system for this by using a semantic approach that maps the speech act's contents onto the tasks' preconditions (or the executability conditions of terminal actions for these tasks).[4] This approach also provides a unified principle for recognizing the speech act and computing its effects—in this case, for anticipating action failure and triggering the appropriate replanning. This approach seems well adapted to recognizing speech acts that affect specific tasks in an agent's plan, because it could identify them by mapping an utterance's semantic content to descriptors associated with a task (see Figure 5).

Other forms of advice exist, such as the "doctrine statements" Bonnie Webber and her colleagues have introduced.[13] These statements prescribe generic rules of behavior that only become active when relevant situations occur: for example, advising Ross to "be nice to Phoebe" will determine whether he interrupts a conversation between Phoebe and Monica when trying to gain information from Phoebe. This advice could keep Phoebe from lying to Ross about Rachel's preferences.

Finally, the user can directly provide information that will solve a subtask's goal. This is the case if the user tells Ross about Rachel's preferences (such as "Rachel really likes flowers"), solving the initial task of gaining information about Rachel and causing Ross' plan to proceed forward with this task solved. In this instance, the user can provide helpful information, or lie to him, and observe the consequences on the unfolding story. From an implementation perspective, subgoals in the HTN are labeled according to different categories, such as `information_goals`. When these goals are active, the system checks them against new information input from the natural language interface and marks them as solved if the corresponding information matches the subgoal content. In that sense, the system can recognize the speech act and compute its effect by mapping the semantic content of the natural language input to the semantic atoms occurring in some HTN's operators' pre- or postconditions.

Currently, our system can generate complete stories up to three minutes in duration. The dramatic action appears from Ross' perspective, although the user can switch viewpoints to another character or freely explore the stage while the plot unfolds. The action progresses until Ross asks Rachel out, and the story concludes with Rachel's answer. Figure 6 shows a sample story that the system produced.

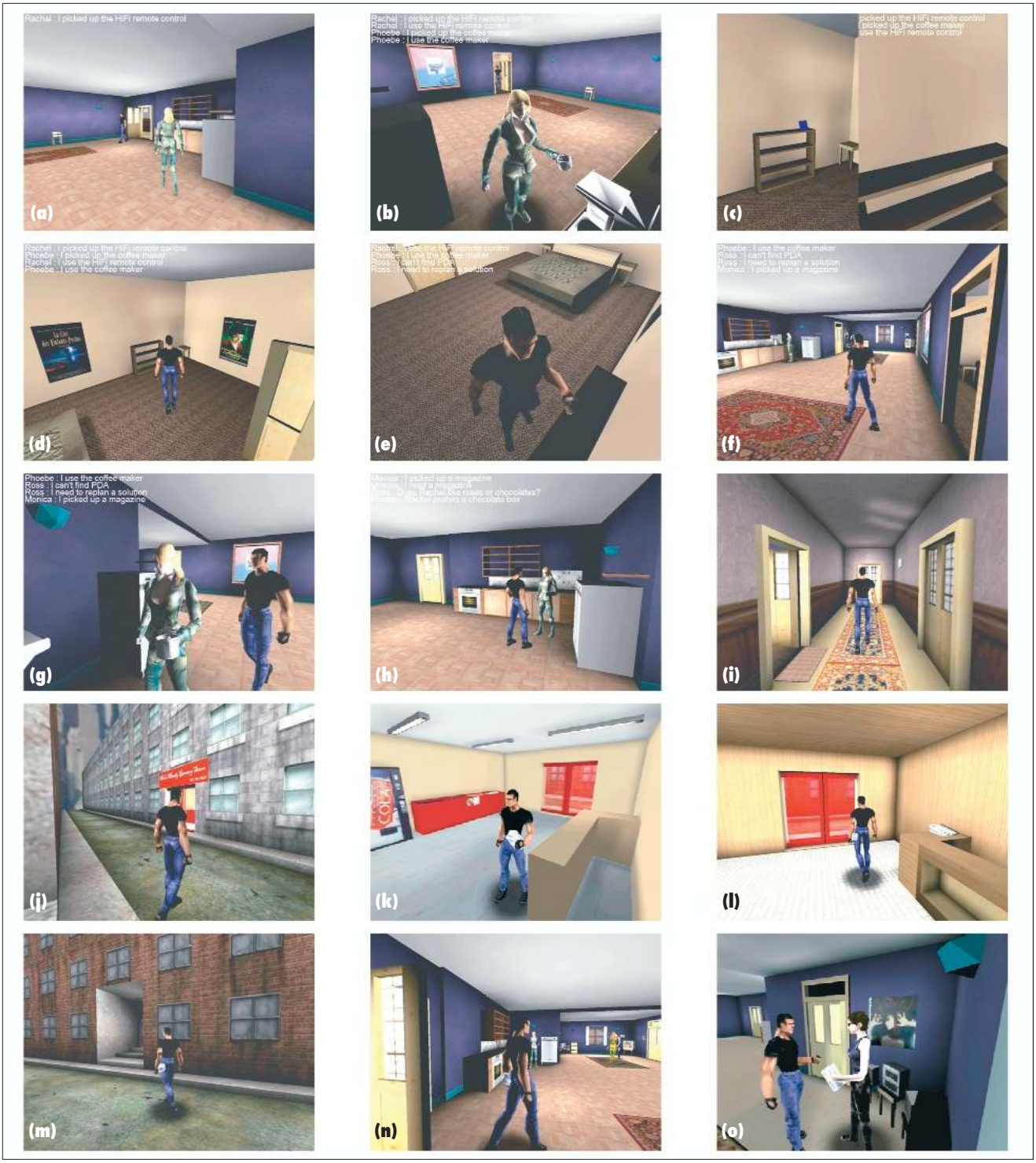

Figure 6. An example of story instantiation: (a) Ross goes to Rachel's bedroom to find her PDA. (b) Phoebe, who is preparing some coffee, doesn't see him. (c) The user discovers Ross' intentions and decides to steal Rachel's PDA (by removing it from the virtual environment). (d) Ross reaches the PDA's original location, unaware of user intervention and (e) can't find it. (f) Ross decides to ask Phoebe for information about Rachel. (g) He awkwardly interrupts Phoebe in her activities. (h) Upset by the intrusion, Phoebe lies about Rachel's preferences and tells Ross to give her a box of chocolates. (If Ross had been more careful when asking Phoebe, she would have responded sincerely to his request and told him to offer Rachel roses.) (i) After obtaining information from Phoebe, Ross leaves and (j) goes to a shop to buy chocolates. (k) He buys the box of chocolates and (l) leaves the store. (m) He returns to the apartment to offer the chocolates to Rachel. (n) He finds her alone and asks her out. (o) Unimpressed by his gift, she says no.

The Authors

Marc Cavazza is a professor of intelligent virtual environments at the University of Teesside, UK. His main research interest is artificial intelligence for virtual humans, with an emphasis on language technologies. He is currently working on interactive storytelling systems and on dialogue formalisms for conversational characters. He received his MD and PhD, both from the University of Paris 7. Contact him at the Univ. Teesside, School of Computing and Mathematics, Borough Rd., Middlesbrough, TS1 3BA, UK; m.o.cavazza@tees.ac.uk.

Fred Charles is senior lecturer in computer games programming at the University of Teesside, UK. He is currently undertaking a PhD in the area of virtual actors in interactive storytelling. He received his BSc in computer science and MSc in computer graphics from the University of Teesside. Contact him at the Univ. Teesside, School of Computing and Mathematics, Borough Rd., Middlesbrough, TS1 3BA, UK; f.charles@tees.ac.uk.

Steven J. Mead is a lecturer in the computer games programming department at the University of Teesside, UK. He is currently undertaking an MPhil in the area of intervention on autonomous agents' behaviors in interactive storytelling using natural language. He received his BSc in computer science and MSc in computer graphics from the University of Teesside. Contact him at the Univ. Teesside, School of Computing and Mathematics, Borough Rd., Middlesbrough, TS1 3BA, UK; steven.j.mead@tees.ac.uk.

We now need to develop evaluation methods that can measure a system's potential to generate stories and the narrative relevance of such stories.[2] At this time, we can only quantify our system's generative potential: A story instance consists of the conjunction of a set of terminal actions for each actor's plan. Assuming that the system synchronizes these actions in each scene, an order of magnitude for the number of stories is given by the sum across scenes of the product of individual characters' actions. For an average branching factor of three, this amounts to several hundred story variants, and for a branching factor of four, a few thousand. This order of magnitude does not evaluate the actual interest or dramatic value of the story variants: different actions carried by secondary characters, while formally contributing to a story variant, might have no real impact on the overall story.

However, character-based approaches have good potential for story generation. Despite the deterministic nature of their underlying techniques, many different factors contribute to the unfolding plot's unpredictability from the user's perspective. Future work will have to evaluate the approach's scalability: we plan to extend our prototype to develop more complex storylines and use multiple plans for each character to increase their interactions.

In the long term, our simplifying assumptions, such as decomposability and total ordering, will most certainly face limitations. All but the simplest stories involve intertwined plots and dependencies between actions taken. This would lead to investigating more generic, possibly domain-independent, planning techniques such as search-based planning.[10] However, in our current implementation, using knowledge-intensive planning techniques such as HTN planning simplified the narrative control problem, because narrative control was partly compiled in the representations. This might no longer be the case with generic planning techniques, which should be associated with narrative control mechanisms. As a first step in exploring these issues, we will study heuristic search planning in story improvisation, where only situational aspects are relevant—for instance, in cartoons. ■

References

1. M. Mateas, "An Oz-Centric Review of Interactive Drama and Believable Agents," *AI Today: Recent Trends and Developments*, M. Wooldridge and M. Veloso, eds., Lecture Notes in AI, vol. 1600, Springer Verlag, New York, 1999, pp. 297–328.
2. R.M. Young, "Notes on the Use of Plan Structures in the Creation of Interactive Plot," *Amer. Assoc. Artificial Intelligence Fall Symp. Narrative Intelligence*, AAAI Press, Menlo Park, Calif., 1999.
3. N.M. Sgouros, G. Papakonstantinou, and P. Tsanakas, "A Framework for Plot Control in Interactive Story Systems," *Proc. Am. Assoc. Artificial Intelligence* (AAAI '96), AAAI Press, Menlo Park, Calif., 1996, pp. 162–167.
4. M. Cavazza, F. Charles, and S.J. Mead, "AI-Based Animation for Interactive Storytelling," *Proc. IEEE Computer Animation*, IEEE Press, Pistcataway, N.J., 2001, pp. 113–121.
5. W. Swartout et al., "Toward the Holodeck: Integrating Graphics, Sound, Character and Story," *Proc. Autonomous Agents 2001 Conf.*, ACM Press, New York, 2001, pp. 409–416.
6. D.S. Nau, S.J.J. Smith, and K. Erol, "Control Strategies in HTN Planning: Theory versus Practice," *Proc. AAAI/IAAI-98*, AAAI Press, Menlo Park, Calif., 1998, pp. 1127–1133.
7. S. Kambhampati and J.A. Hendler, "A Validation Structure Based Theory of Plan Modification and Reuse," *Artificial Intelligence*, vol. 55, nos. 2–3, 1992, pp. 193–258.
8. K. Erol et al., "A Critical Look at Critics in HTN Planning," *Proc. 13th Int'l Joint Conf. Artificial Intelligence* (IJCAI-95), AAAI Press, Menlo Park, Calif., 1995, pp. 1592–1598.
9. P. Weyhrauch, *Guiding Interactive Drama*, doctoral dissertation, tech. report CMU-CS-97-109, School of Computer Science, Carnegie Mellon Univ., 1997.
10. B. Bonet and H. Geffner, "Planning as Heuristic Search: New Results," *Proc. European Conf. Planning* (ECP'99), Springer Verlag, New York, 1999, pp. 360–372.
11. J.C. Pemberton and R.E. Korf, "Incremental Search Algorithms for Real-Time Decision Making," *Proc. 2nd Int'l Conf. Artificial Intelligence Planning Systems* (AIPS 94), 1994, pp. 140–145.
12. D. Weld, "Recent Advances in AI Planning," *AI Magazine*, vol. 20, no. 2, 1999, pp. 93–123.
13. B. Webber et al., "Instructions, Intentions and Expectations," *Artificial Intelligence J.*, vol. 73, nos. 1–2, Feb. 1995, pp. 253–269.