Available online at www.sciencedirect.com
ScienceDirect
Knowledge-Based Systems 20 (2007) 592–606
Knowledge-Based SYSTEMS
www.elsevier.com/locate/knosys

A goal-oriented interface to consumer electronics using planning and commonsense reasoning

Henry Lieberman [a,*], José Espinosa [b]

[a] MIT Media Laboratory, 20 Ames Street, E15-384A, Cambridge, MA 02139, USA
[b] MIT Media Laboratory, 20 Ames Street, E15-383, Cambridge, MA 02139, USA

Received 4 February 2007; accepted 17 April 2007
Available online 4 May 2007

Abstract

We are reaching a crisis with design of user interfaces for consumer electronics. Flashing 12:00 time indicators, push-and-hold buttons, and interminable modes and menus are all symptoms of trying to maintain a one-to-one correspondence between functions and physical controls, which becomes hopeless as the number of capabilities of devices grows. The root of this problem is that the devices do not have knowledge of (a) their own capabilities (b) the scenarios where they can be of use. By using commonsense knowledge and an AI partial-order planner we propose instead to orient interfaces around the goals that users have for the devices.

We present Roadie, a user interface agent that provides intelligent context-sensitive help and assistance for a network of consumer devices. Roadie uses a commonsense knowledge base to map between user goals and functions of the devices, and an AI partial-order planner to provide mixed-initiative assistance with executing multi-step procedures and debugging help when things go wrong.

Keywords: Commonsense reasoning; Planning; Consumer electronics; Goal-oriented interfaces

1. The crisis in consumer electronics interfaces

Current consumer electronics are getting more and more complicated, threatening to outstrip the competence that can be reasonably expected from their intended users. For example, a typical consumer camera, the Canon S500, has 15 buttons, two dials, 4×2 mode switches, three menus of five choices in each mode, each with two or three values, 7 on-screen mode icons, and so on.

We attribute the growing complexity of consumer electronics interface design to the desire to maintain the one-to-one correspondence between functions and controls that worked well for simpler devices. But as the number of functions of a device grows, controls get overloaded, leading to heavily-moded interfaces, push-and-hold buttons, long and deep menus, and other confusing and error-prone interface elements. For this reason, the devices have interfaces with bad affordances [30] where there is no easy way to help the user perform the basic operations on the device.

The next generation of consumer electronics devices will incorporate processing and networking, making things potentially more complex if we stick to manual operation, but also opening up new possibilities for automating cooperation between multiple devices.

We propose to re-orient the interface around the goals of the user, rather than the functions of the device. Something, then, has to map between the user's goals and the concrete functions of the device. We propose to fill this gap with Roadie, an interface that makes use of commonsense knowledge and a partial-order planner to give the user proactive advice, automate complex tasks, and provide debugging help when things go wrong.

2. Users need help with many scenarios of use

It is not only the "normal operation" of the device that users need help with. There are other scenarios associated with consumer devices that users need help with. The

* Corresponding author. Tel.: +1 617 253 0315.
E-mail addresses: lieber@media.mit.edu (H. Lieberman), jhe@media.mit.edu (J. Espinosa).

0950-7051/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.knosys.2007.04.008

advent of powerful computing and communication in devices gives us the potential of providing help with these scenarios, as well as merely invoking functions of the device.

What can I do "out of the box"? When the user first acquires the device, how do they know what it can do? How do they know what its capabilities and limitations are? Devices should be self-aware, self-explaining, and self-revealing. Onboard memory, processing and networking can access and display information like introductory tutorials, user group messages, examples of use, etc. just when they are needed. The system should describe its capabilities and limitations in terms that the user can understand.

Oops, it doesn't work! Devices should also be self-debugging. Devices should know what the possibilities for error are, and give users sensible options for investigating the problem, mapping the behavior of the device to their expectations, and plausible routes to a solution or to seeking more assistance. The interface should help generate hypotheses concerning what might have gone wrong. It should test those hypotheses automatically, when possible. If the system cannot test a hypothesis it should give to the user an explanation of what might be wrong, how he or she can test it, and the steps he or she should follow to correct the problem.

Don't do that to me again! Devices should accept feedback on their behavior and modify their behavior accordingly. They should have the capability of customizing device operation for particular situations and automating common patterns of usage.

I should be able to... Devices should enable unanticipated, but plausible, patterns of use. Especially when several devices are networked together, users should be able to compose the results of using one device with the input of another without learning arcane procedures; converting file formats, figuring out how to patch cables, etc.

I want to do ... The information presented to the user, and the information exchanged between the user and the system, should always be in the context of the user's goals. For this kind of dialogue, the system needs to have a good idea of the relations between the user's actions and the goals he or she is trying to accomplish.

To sum up, our goal is to create an interface that is goal-oriented, self-describing, self-revealing and self-debugging.

2.1. Goal-oriented interfaces

For example, a reasonable goal for the user after opening the freezer is to want to defrost something. Thus if a system can sense the door opening, the microwave should suggest the defrosting function.

2.2. Self-describing

The system should describe its capabilities and limitations in terms that the user can comprehend. If the devices cannot perform a desired action, they should give the user an explanation of why the goal failed and how to fix it. If the state of the device interferes with another action, it should inform the user of the conflict and the actions necessary to resolve it.

2.3. Self-revealing

In the normal process of manipulating the devices, the user may be required to make choices in situations where he or she might not have a good understanding of the consequences. In this case, the system should inform the user about the trade-off of each choice. Informing the user of the consequences of trade-offs has the following advantages: (a) it prevents the user from experiencing an undesirable and potentially irreversible consequence of a system action; and (b) it helps the user back trace to the right point if he or she wants to change the behavior of the system.

2.4. Self-debugging

Fixing devices when things go wrong is one of the most frustrating things about dealing with consumer electronics. A common reason for problems is when the users use of the device is outside the designer's anticipated scenario of use, or there is a piece that it is not working as expected. Fixing this problem forces the user to introspect about the system's internal state – which might be hidden by the device designer – and to figure out what is wrong and devise a strategy to fix it.

3. Introducing Roadie

In this paper, we present Roadie, a prototype consumer electronics interface oriented towards the needs of non-expert users. The project name comes from the person who is in charge of setting up the audio and video devices during music concert tours.

Roadie's main goal is help naïve users in interacting with home electronics devices. This goal is reached by giving the system, first, knowledge about the functioning of the devices, and, second, the goals and desires of the user. The former knowledge is achieved by a partial-order planer and the latter by using the open mind commonsense knowledge database (see Section 4).

Roadie's interface is currently deployed as a window on a computer screen, but it can be ported onto a PDA, a cell phone or a universal remote control. In the long-term future, it might indeed be advantageous to totally redesign the hardware control interface to each device to be more goal-oriented. But for the moment, it is not our intention to completely replace the conventional button-and-knob interface to each device.

The Roadie interface (see Fig. 1) shows dynamic dialog boxes that

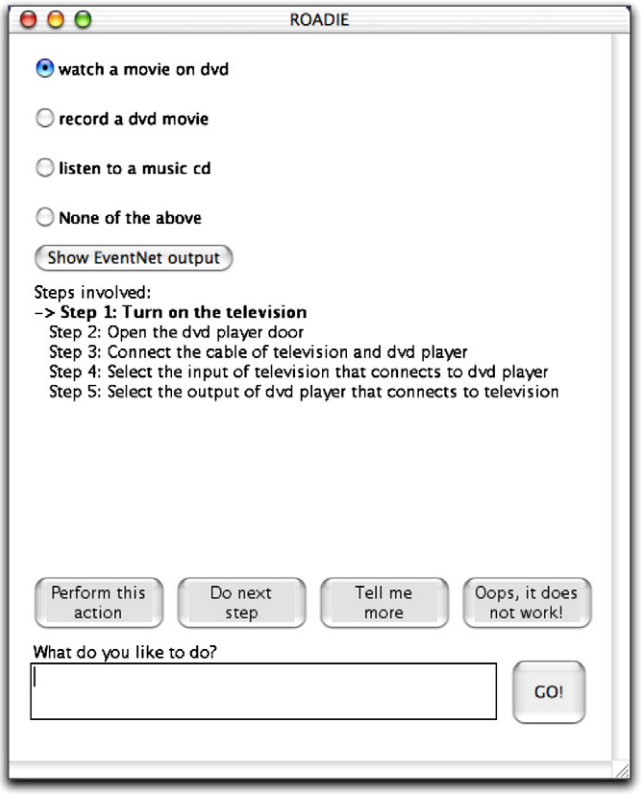

Fig. 1. Screen shoot of Roadie user interface.

Show steps of a procedure.
Provide controls for executing the procedure.
Provide explanation.
Display alternatives for what to do next.
Provide facilities for help and giving feedback.
Provide a box for unrestricted natural language input for the user to state goals or ask questions.

At the top of the interface are the suggested goals. When the user picks one of the options, the planner calculates a plan to reach the goal. The answer is mapped to English by the device interface, and rendered by the user interface, highlighting the action that is going to be executed next.

The user can control the execution of the steps by using the "Perform this action" (do all the steps at once), and the single-step "Do next step" button. The "Tell me more" button provides more detailed explanation of why the steps help accomplish the goal, and the "Oops, it does not work!" button launches a debugging dialog.

In addition, the interface has a "What do you want to do?" text box where the user can use natural language to communicate with the system. While we are using some natural language understanding, as explained below, we do not rely on being able to completely understand arbitrary English. We also are anticipating the possibility that we could use speech input for the natural language component.

Roadie's interface is intended to be *fail-soft* – provide intelligent assistance when it is helpful, but not replace conventional push-the-button interaction if that turns out to be more convenient in a given case, or if Roadie does not have the knowledge or language understanding capability to correctly understand and implement the user's intention.

3.1. Roadie device requirements

Roadie is designed to operate with devices that (1) provide means to control their functions via external software, and (2) that can query their state by external software. The first requirement allows Roadie to control the devices on the user's behalf; the second allows Roadie not only to watch the state changes of the devices and interpret them as the user's actions, but it also monitors the devices by looking for direct user interaction.

Unfortunately, the devices available to us at this time do not meet these two requirements. The first requirement can be easily mimicked by using an infrared controller to override the device's remote controls. But there is no easy approach to extending API's of current devices to get their states. Controlling the devices without getting their states leads to clumsy design, since the controller's internal representation of the devices becomes unsynchronized every time the user interacts with the devices using the traditional front panel interface controls.

The devices' manufacturers are aware of this problem and created the UPnP [38] standard. Unfortunately, the manufacturers have not started to build devices that fully comply with this standard. Furthermore, they sometimes do not fully expose to the applications programmer all the necessary controls and states to accomplish a given task. Some manufacturers are interested in implementing sets of branded devices that coordinate using a proprietary protocol that prevents systems like Roadie from fully implementing general interaction with the device.

To overcome this problem, we created a set of simulated devices to test Roadie. We hope that in the near future UPnP or a similar standard will become widely accepted allowing deploying Roadie in real devices.

The simulated representation of each device has two parts. The first contains the device's states (if the device is on or off; what device is connected to what input/output; if the DVD player has a disk inserted, etc.) and the device's simulated functionality (an MP3 file plays when you hit the "Play" button on the CD player to show that a simulated CD would be playing; a quicktime video represents the operation of the television). The second contains the visual interaction elements like:

Front panel controls: e.g. the power on/off button, the radio tuning knob.
Front panel display indicators: e.g. the power on/off light, the radio frequency display.
Back panel input/output jacks: When connected, these display what they are connected to.
Image of the device hardware.

Fig. 2 shows an example of the simulated devices used by Roadie.

Roadie assists the user in manual operations by a dialog box instructing the user how to perform that operation (see Fig. 3). In the current version, this help is a static picture but can be changed to movies, "how-to's or tutorials.

4. Roadie's internal architecture

Fig. 4 shows a diagram of Roadie's system architecture.

4.1. Commonsense knowledge and EventNet

Roadie needs to have knowledge about what the motivations, desires and goals of the users are and the connections between goals and actions. For this, we created a semantic network called EventNet [11], used by Roadie to create a list of possibilities for user goals and plausible candidates for action sequences to accomplish those goals. The plan recognizer uses knowledge mined from the OpenMind Commonsense Knowledge Base [36], a knowledge base of 800,000 English sentences describing everyday life, contributed by volunteers on the Web. We also use ConceptNet, a semantic network derived from parsing the sentences in OpenMind and applying a spreading activation algorithm.

EventNet uses temporal knowledge from ConceptNet, and spreading activation to infer a possible set of antecedent or subsequent actions.

By traversing the network from antecedents to subsequent nodes the system is making forward projections about what is plausible to happen in the future. If the network is traversed in the opposite direction, EventNet is making inferences about possible past actions.

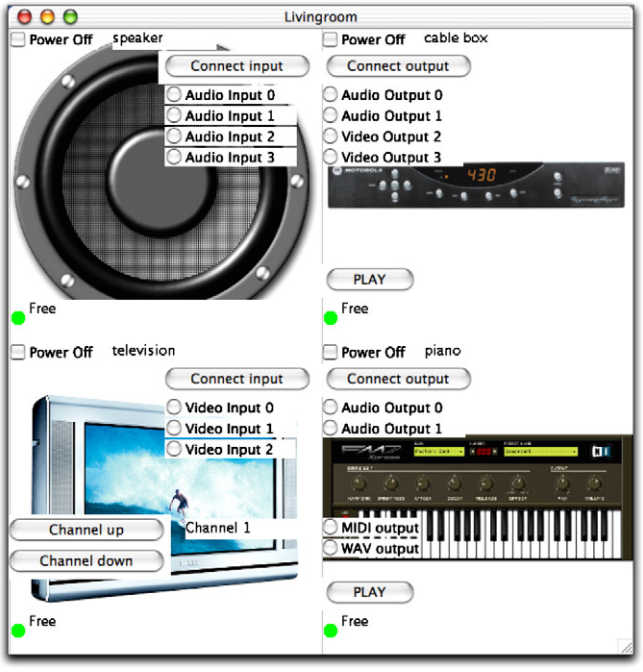

Fig. 2. Roadie device simulation.

Fig. 3. Picture explaining how to connect a television.

EventNet links (Fig. 5) represent before and after relations between events, expressed in a first-person perspective (''I rain'' in the above example, should be interpreted as "I observe that it is raining'').

EventNet does not distinguish between event sequences that express causal relations and those that do not. EventNet links can either represent cause-effect relations; express a goal, a sequence of actions, and a result; or simply a contingent observation of temporal occurrence. For example, "I drink coffee" and "I feel alert" are causally related, but "I drink coffee" and "I go to work" are not necessarily causally related, even though "I drink coffee" often occurs before "I go to work" (there may, of course, be a reason why coffee drinking often occurs before work). Their function is to generate candidates for possible goals consistent with a goal the user expresses to Roadie, or a possible sequence of actions and/or consequences for accomplishing a user-supplied or Roadie-generated goal.

Note that some of these nodes cannot be executed at the same time, some are necessary and others depends on personal preferences, but all of them represent some stereotypical situation. The following equation shows how inference is performed using spreading activation

$$n_i = \sum_{j=\text{links}(n_i)} \text{energy}(n_j)^* \text{weight}(n_i, n_j) \quad (1)$$

Note that EventNet may contain links that are patently garbage. In the example, "I walk in the rain" and "I get wet" are plausible continuations of "I (observe that it is) raining", but "I paint someone's house" is not. How did that get there? First, it may have come directly from OpenMind – a malicious or sarcastic user may have typed it in, and we have not verified each and every statement in the knowledge base. Second, it may have come from a misinterpretation by our natural language understanding system, whose capabilities are admittedly limited.

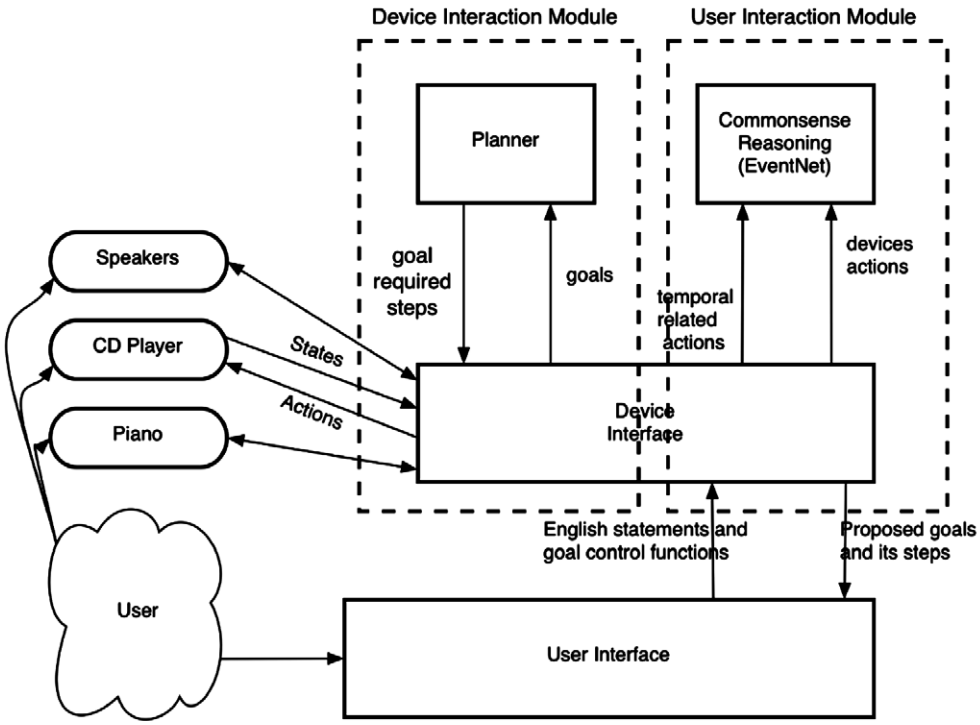

Fig. 4. Roadie system architecture.

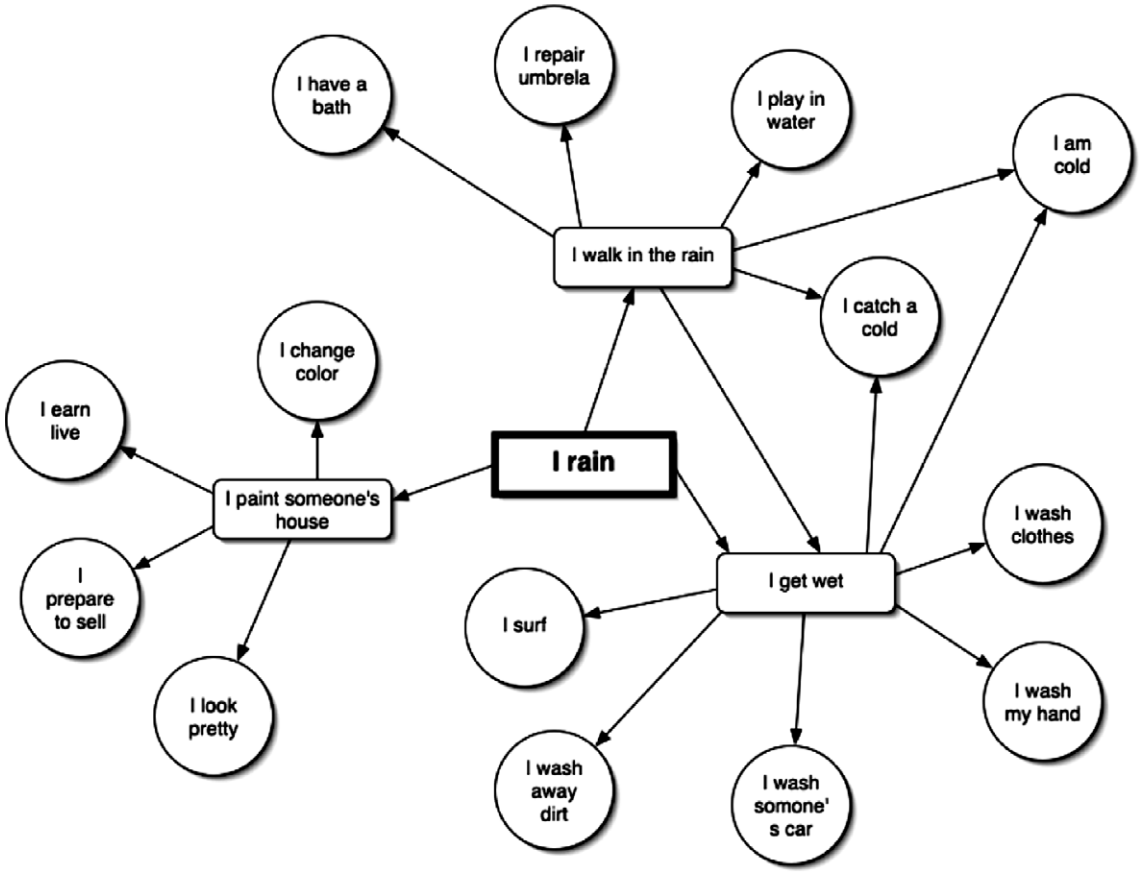

Fig. 5. The EventNet temporal semantic network.

In Roadie, as in several other of our commonsense knowledge applications that use EventNet and ConceptNet, the impact of such garbage statements is kept under control by *intersecting* the results of retrievals with domain knowledge that comes from plausible operations in the application. That generally causes the garbage to be filtered out.

If Roadie detects the event, "I plug in my guitar", EventNet continuations may produce "I play music", "I record music", and "I start dancing". However, intersecting the device functions, capabilities and goals with the EventNet results approves playing and recording music, but no function of any device that Roadie knows about has anything directly to do with dancing. If, on the other hand, the user types a goal, "I would like something to dance to", it is likely to infer that playing music is appropriate.

Further details on the use of EventNet are presented in the scenario.

4.2. User interaction module, commonsense plan recognizer

The user interaction module maps the actions, goals and desires of the user to a format that the device controller can understand. This component works as a complement to the normal device's interface, sensing the user's interactions with the devices. It uses EventNet to find the implications of the user's actions. We also used the Web to automatically collect pairs of device actions linked by temporal relations [Mihalcea, personal communication].

For example, if the user plugs in his or her guitar, the system infers that it is likely that the user wants to play music. It is also responsible for providing an explanation about the behavior and functionality. The input nodes are calculated using templates with English descriptions of the device's changes of state. Then the output nodes are matched against a text description of the available goals.

The planner is used to infer the set of actions needed to configure the devices to satisfy the user's goal. The planner decomposes the desired states to single actions that the devices can execute and creates alternative actions when something unexpected occurs. Also, the planner keeps track of the recently performed actions and whether they succeed or not. If it is impossible to accomplish the goal, the system uses this information to provide the user advice to debug and potentially correct the problem.

We acknowledge that planners generally assume the completeness and accuracy of their descriptions, and these assumptions might not be fully met when the knowledge comes from the common sense knowledge base. However, in practice our experience in the consumer electronics domain is that plans generated turn out to be reasonably plausible, within the limitations of the implementation. Our task models, too, are incomplete, but generally fit people's common sense models of the operation of these devices. Efforts in the industry to standardize device and task models, or to publish device, operation, and task descriptions in Semantic Web form would give us a source of information more reliable than piecing these descriptions together from disparate Web sources. But we do not think it necessary to wait for these efforts to come to fruition before being able to provide useful assistance. We use relatively a simple planner, but could perhaps benefit from more sophisticated hierarchical task models and planners.

Roadie uses the standard Graphplan [2,3] implementation, from Avrim Blum and Merrick Furst at CMU. Graphplan is a partial-order planner that has the capability of incrementally constructing a plan graph, and propagating constraints and truth-values through the graph as it is being built. This incrementality is useful to Roadie in enabling it to interactively respond to user's physical actions in the device setup without recalculating the entire graph at each step. Graphplan is reasonably efficient, and less sensitive to goal order than some other approaches. It guarantees it will find the shortest plan among those in which independent actions may take place at the same time.

In addition, the planner's goals serve as a model of the capabilities of the available devices. This knowledge helps to constrain the broad options provided by EventNet. The planner is also responsible for finding the states of the devices that are conflicting with the desired goal.

Note that the system uses two separate inference engines. EventNet is in charge of inferring what the user's goals are, and the planner is in charge of executing those goals. Both systems work in synergy: the planner helps EventNet to narrow its output to just the relevant information, and in exchange EventNet does not need to display all the possible device actions, just the relevant ones.

4.3. Device interface

The device interface is the module responsible for making the devices communicate with the rest of the system. It is responsible for controlling and monitoring the devices, querying EventNet and sending the goals to the planner. This module has a text string for each change in state of the device, like "turn on the device," "I insert a music CD." In addition, it has all the possible goals that might be reached with the current devices: both natural language, and as a planner goal with the slots and its acceptable types. For example, it has ⟨"*play the music CD", (play-music-cd [cd-player-device] [speaker-device])*⟩ for playing a music CD. So, to set up the action *play-music-cd* it looks for CD players and speakers and sets those particular devices into the planner. Also, English templates for each possible planner step are used to create explanations in natural language. The matching between two phrases are made using EventNet's semantic link algorithm (Fig. 6).

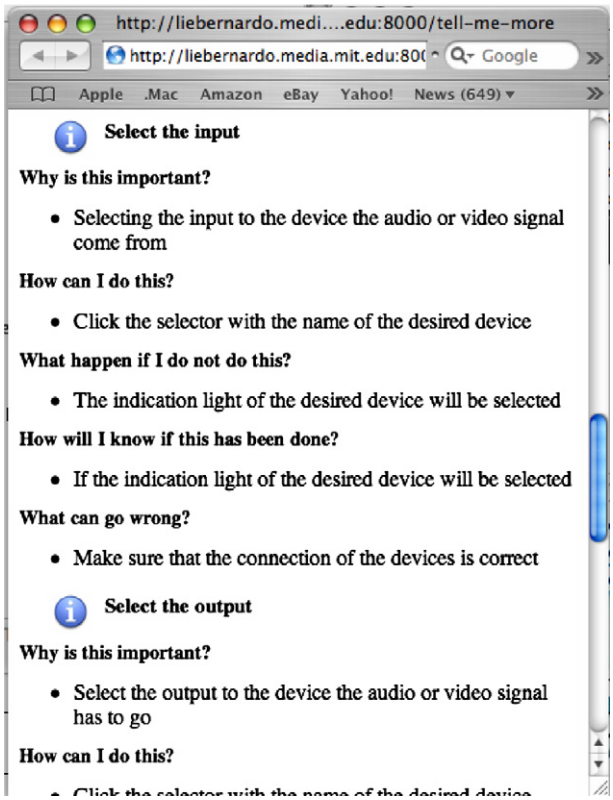

Fig. 6. Roadie's documentation for each procedure step.

4.4. Debugging information

We do not assume that action sequences will never fail. Problems inherent to devices – malfunctions or misunderstandings between the user and Roadie – might emerge. Debugging consists of looking for the causes of unexpected results.

Roadie's "Oops, it does not work" button is intended to be the portal to Roadie's debugging assistance capabilities. The first line of defense in Roadie's approach to debugging is to produce customized documentation for the specific debugging situation by instantiating general information sources with knowledge about the present device configurations, goals, and states. This is organized as a set of responses to anticipated questions, in the tradition of Burke and Hammond's FindMe systems [9].

For each step, we can show why the step is important, how the user can perform the step, what the consequences are of *not* doing this step, what the results are of performing it, and the things that might go wrong while trying to perform the step. If the user does not find this information sufficient to solve the problem, the system can automatically send queries to online search engines, user manuals, user group forums, etc.

Future work in debugging will include an approach based on matching recorded sequences of user actions to Commonsense plan representations, similar to Wagner and Lieberman's Woodstein [24]. Woodstein provides a browser that allows users to browse a hierarchical display of goals and recorded actions, coupled to the states that result from each action. They can then compare, step-by-step, their expectations for each goal or action step, with the states that result, to uncover discrepancies. Woodstein provides the means for interactively tracing both control and data flow.

5. User scenario

5.1. Recording to a CD

The user turns on the DVD player, using its front panel switch. Roadie queries EventNet for the set of *temporally related events* for the action *"turn the DVD player on."* For this action, EventNet returns, among others, these answers:

"watch hours of the world's best nature programs,"
"hit play,"
"record a music cd,"
"listen to a music cd,"
"insert a disk,"
"insert a dvd,"
"leave the room,"
"push the television," and
"turn on the home theater projector."

Some of the actions, like "*leave the room*," are ambiguous. Others are just true in a very narrow context, such as "*watch hours of the world's best nature programs*". Again, the idea to generate a broad range of possibilities, and let further constraints from the context, other actions, and interaction with the user narrow down the search space.

Keep in mind that we have not programmed in advance all the possible goals that the user might have, and all the implications of these goals, EventNet is useful in generating at least *some* plausible possibilities for subsequent events, no matter what the user's goal and situation is, as long it could reasonably be considered part of common sense knowledge.

Then, Roadie tries to match the EventNet answers with a device description. Using text matching as a method to find the likely goals allows flexibility to add new goals to the set of devices, while filtering the nodes that are out of context since they do not match any goal.

The set of suggested actions are:

"watch a movie on dvd,"
"listen to a music cd",
"record a music cd."
"record a movie on dvd"

The user wants to record a music CD on his home system, so he picks the third choice. This goal needs two parameters: a device capable of playing the music CD, and a device capable of recording the music CD. The

system keeps track of the recency of usage of the devices and asks the planner for a set of actions to accomplish the goal. Note that the system prefers the DVD player instead of the CD player to play a CD even though more than one device has that capability, since it is the device that the user was most recently using.

The planner calculates a plan. The output of the planner is a partially ordered set of actions. One of the advantages of using a planner is that the system is able to find the configuration to accomplish the goal even if it is necessary to change some settings deeply buried on a device interface, or to set the state of a remote device into a particular mode. Roadie uses the planner output and a set of English templates – one for each possible planner step – to communicate to the user the steps involved in performing this task. The planner's explanation is shown in the Roadie interface:

1. Connect the cable of the recorder and the DVD player
2. Open the DVD player door
3. Turn on the recorder
4. Select the input of the recorder that connects to the DVD player
5. Select the output of the DVD player that connects to the recorder

Note that some of these actions can be performed directly by the system, while others (like inserting the DVD) cannot. Roadie shows four control buttons:

- **"Perform this action"** This button will perform all the steps listed to accomplish the goal at once. If one of the actions needs the user's manual intervention, the system will instruct the user about what he or she needs to do. Roadie knows if a step fails, in which case the planner is called again to find an alternative. If there is no alternative plan, the system will tell the user which step of the process went wrong, along with suggestions for how to solve the problem.
- **"Do next step"** This button behaves like the button "Perform this action", but instead of executing all the steps at once, it executes them one step at a time. This permits the user to observe physical effects of each action.
- **"Tell me more"** This button tells the user why each step is important, how he can perform the step, what can happen if the step is not finished, and how he can determine if the step has been performed correctly.
- **"Oops, it does not work!"** This button queries an online search engine for information about the step. This button can be specialized to use the device's user forums or vendor-provided information. Knowledge about user goals, device states, and other context items can be fed to the search engines directly by the device, rather than asking the user to end their interactions with the device and log into a conventional computer. Again, we intend to use this button as a portal to more sophisticated debugging capabilities, such as found in Woodstein [24].

These facilities not only provide the user with concrete information and things to do, but also facilitate the user's learning more about the devices' principles of operation.

The user picks the button "Perform this action," and Roadie starts to execute the steps until it reaches the action *"Connect the cable of the recorder and the DVD player"*. The system cannot perform the action by itself, so it asks the user to perform this action. Roadie shows the user a picture of the correct input and connector. The "Tell me more" button explains to the user what a connection is, the different jack types, and the differences between input and output devices and other relevant information about this step. A similar dialog is displayed when the system needs the user to insert a disk.

After listening a couple of songs, the user types, *"I want to watch a movie"* in the "What would you like to do?" dialog box. Roadie recognizes the pattern *"I want to"* as a user goal, then passes it to EventNet to figure out the desired goal. Roadie queries EventNet and matches the user's goal to the functions "watch a DVD movie," and "watch television". The user selects the option "watch a DVD movie".

Roadie realizes that it is not possible to use the DVD player since it is being currently used to record music, but there is also a CD player. At this point the user has three possible options,

- Perform both actions,
- Play only the movie, or
- Record the CD.

Roadie displays the dialog shown in Fig. 7. In this dialog the system explains the options for changing the devices' configuration. If the user ignores the suggestion, the current configuration is kept.

Roadie displays options that provide alternative means of accomplishing the user's goals in blue, marked by a triangle containing an exclamation point. Roadie displays options that require a user to abandon one of their goals to resolve a conflict in red, marked by a stop sign containing a hand.

Roadie first displays a message that says *"The action "watch a movie on dvd" cannot be performed since the device "dvd player" is being used to "record a music cd from dvd player to recorder."* The first button says *"Move "record a music cd" from "dvd player" to "cd player," then perform "watch a movie on dvd"* and a note warning the user that the current task, "Recording a music cd from dvd player to recorder" will be abandoned. The second button says *"Stop "record a music cd from dvd player to recorder" and do "watch a movie on dvd"*.

The new desired goal is sent to the planner and the control buttons are displayed. While this scenario is simple, it illustrates Roadie's capability of dealing with the problem of conflicting goals. Conflicting goals are a common source of difficulty and problems in operating devices. People experienced in operating audio and video equipment often

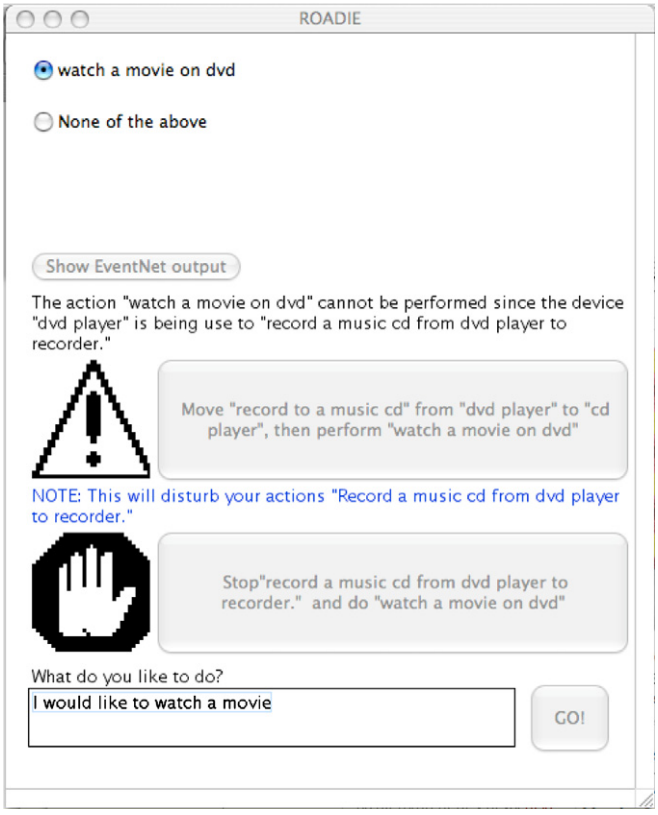

Fig. 7. Roadie showing two possible ways of resolving conflicting goals.

have sophisticated and successful techniques for resolving goal conflicts.

5.2. Watch the news

The above scenario presented a case of inferring goals from actions. Now, we'll show one where the user explicitly states a goal. The user types into the "What do you want to do?" dialog box the phrase "I want to watch the news." This goal is sent to EventNet and then matched with the available goals; the proposed actions are "watch television," and "listening to the radio." The user selects "watch television" and sets the devices by clicking the "Perform this action" button. Note that "listening to the radio" is proposed even though the user said "watch" the news, and you cannot watch a radio. This is because the commonsense intent of "watch the news" is to become informed about the day's news, not to watch something.

To perform this action, the user needs to connect the cable box to the television. But the user does not make the connection correctly. The user realizes that something is wrong, then type in the "What do you want to do?" box, "Why can't I see an image?" Roadie identifies the pattern "Why ..." as if something is wrong, then it tries to find the problem and correct it. Roadie uses the devices sensors to look for the cause of the problem. Also it knows that it cannot sense the states of the cables, and that cable connections are a frequent source of mistakes. It then shows a picture of the correct way to connect the cables, and recommends that the user check the connections. Furthermore, Roadie provides a "Tell me more" option, where the system gives more explanation about how to know if the connection is correct, information about the different types of inputs, outputs, jacks, etc.

A second user turns on the DVD player making the options *"watch movie on dvd," "record dvd movie,"* and "*listen to music cd"* appear, and he selects the first option. Roadie realizes that the television is busy watching the news, and remembers that also "listening the radio" might satisfy the goal. This will free the television to watch the DVD while satisfying the goal of listening to the news. To warn the user about this conflict and a possible solution Roadie displays a similar dialog to the one in the previous scenario. This scenario also shows how Roadie can track device states and user actions, and find concrete actions compatible with multiple high-level goals.

5.3. Kitchensense

In this section we will describe Kitchensense, an augmented reality Kitchen that uses the same techniques that Roadie uses to provide the people cooking with context aware information [17]. Whereas the first scenario dealt with audio and video equipment, similar user interface problems exist for kitchen appliances such as microwaves, dishwashers and food processors.

The environment uses its built-in sensors to tag the user's activities, for example a drawer sensor is attached to the sentence "I open the kitchen drawer." Then, this sentence is sent to EventNet, and the output is processed by Kitchensense to trigger relevant functions.

Kitchensense uses the information from its sensors and the EventNet plan recognizer to show device functions that might be relevant to the user activity. For example, when the user opens the refrigerator and gets close to the microwave, Kitchensense sends the sentences *"I open the refrigerator,"* and *"I walk to the microwave"* to EventNet, the top answers are: *"I cook food," "I eat lunch," "I reheat food," "I take ice cream out," "I read newspaper," "I set cup on table," "I breathe fresh air,"* and *"I took food out of the fridge."* Then Kitchensense matches these sentences to the functions in the electronic appliances, suggesting the functions "Cook" and "reheat" of the microwave.

It is important to point out that both Kitchensense and Roadie use the same version of EventNet. This let us test the effectiveness of having a large database of commonsense knowledge as an enabler of novel knowledge into interactive applications [25].

6. Preliminary evaluation

We performed experiments to evaluate the contribution of Roadie to making consumer electronics interfaces more user-friendly and effective. The scenarios we chose to test are ones in which consumers are likely to face problems,

such as (a) familiarizing themselves with new devices, (b) performing complex multi-step processes involving multiple devices and requiring data transfer among devices, and (c) debugging problems when things go wrong.

We would have liked to test Roadie with physical devices controlled by software, to present a more realistic scenario to the user. As explained above, we were unable to implement Roadie with physical devices, and so were forced to perform tests on our simulation. However, there were some advantages to using a simulation. Because we pushed participants out of their "comfort zone" and familiar devices, they had to pay more attention. When it happened that they did make mistakes that they might not have made with a physical device, this provided an opportunity to test our debugging capabilities.

Because of these considerations, we consider this preliminary evaluation to yield qualitative results only. When we are able to implement a hardware version of Roadie that at least roughly provides the user with an experience physically similar to operating today's consumer electronics devices, we would like to reconduct experiments with a larger number of participants. Nevertheless, the results of our preliminary evaluation were encouraging.

The experiment consists of four scenarios. The first trains the users on how to use the devices and the agent. During this scenario the users were able to ask the experimenter questions. The next three scenarios were the test experiments. The first scenario is relatively straightforward; the rest of the scenarios introduce complications that exercise some of Roadie's particular capabilities, through the potential for resource conflicts, or by requiring additional technical information to be able to perform them correctly, and the potential for error. These scenarios are typical of everyday operation of consumer electronics devices.

The first scenario consists of playing the piano and watching television. The first task was performed by the experimenter and the second by the participant. During these tasks, the users are allowed to ask questions.

For the second scenario, the user was presented with a television, a set of speakers, a CD player and a DVD player, and was asked to perform the scenario described in Section 5.1. The third scenario involves recording from an electronic piano. In this scenario the piano can be connected to just one device at a time, and in order to record and hear the music, an amplifier is needed. The users were not given any instructions as to how the devices work, but this knowledge is necessary in order to successfully perform this task. The fourth and last scenario consists of playing a movie DVD, similar to scenario number two.

The participants were twelve individuals from the university community, mostly women, roughly half students. The first experimental condition had Roadie turned off for the first three scenarios, and turned on for the last one. The second condition had Roadie turn on for the first three scenarios, and turned off for the fourth scenario.

We performed two analyses of the data. The first one is a between-subjects analysis comparing the four scenarios comparing the performance when Roadie is on and off. The second one is a within-subjects analysis comparing how users perform on the scenarios "Play a CD on the DVD player" and "Play a DVD on the DVD Player"with Roadie on and off; this comparison is valid since both tasks involved the same steps – turn on the CD/DVD, connect the devices together, put the CD/DVD on, and play the disk – therefore can be considered similar (Tasks two and three are not similar, and thus excluded from the within-subjects analysis).

During this preliminary analysis, indications are that users finished the task in less time and using fewer steps with Roadie turned on than off for both the between-subjects and within-subjects analysis. The results of the between-subjects case are shown in Table 1, and the results of the within-subjects case are shown in Table 2.

One surprising finding during the experiment was that a few users misconstrued the explanation of the steps, not as a presentation of the system's plan, but as a list of steps that the system expected the user to follow. We hypothesize that this was due to our lack of clearly labeling the intent of the step list, and to the users' inexperience with systems that autonomously execute planned procedures. We should better label the step list, and clearly explain in advance the division of labor between the system and the user. Due to the small sample size and some skewing of the results because of this unexpected misunderstanding on the part of certain subjects, we do not report confidence levels on our results.

Other users complained that Roadie's interface is too verbose to be used during normal operation but agree that it would be useful when the devices are not working as expected or in novel situations. To fix this we envision a dual-mode interface. The first mode will be a standard remote control aimed at normal operation, and the second

Table 1
Average time and number of clicks before the user ended the given task (between-subjects analysis)

		Roadie on	Roadie off
Play a CD on the DVD player	Avg. time	88.33	111.50
	Std. time	44.89	78.34
	Avg. Clicks	10.33	13.67
	Std. Clicks	4.72	7.23
Play a DVD and a CD at the same time	Avg. time	171.33	179.83
	Std. time	99.11	114.47
	Std. time		
	Avg. clicks	19.00	32.83
	Std. clicks	8.39	9.52
Record the piano	Avg. time	202.17	444.00
	Std. time	194.03	239.96
	Avg. clicks	23.00	59.33
	Std. clicks	21.81	15.37
Play a DVD on the DVD player	Avg. time	54.67	77.17
	Std. time	26.4	57.92
	Avg. clicks	7.33	15.83
	Std. clicks	3.83	8.86

Table 2
Average time and number of clicks before the user ended the given task (within-subjects analysis)

		Roadie on	Roadie off
Play a CD on the DVD player or play a DVD on the DVD player	Avg. time	71.50	94.33
	Std. time	35.56	74.51
	Avg. clicks	8.83	14.75
	Std. clicks	4.39	7.79

will activated Roadie when the user finds a problem, or does a new task.

The tables below indicate a trend showing that participants were able to perform tasks faster and with fewer clicks in general with Roadie than without. "Std. time" and "Std. clicks" refer to the standard deviation on the time and number of clicks to complete the task, respectively.

6.1. Related work

Our discussion of related work will fall into four categories. First, we look at the few projects that have directly tried to tackle the problem of simplifying consumer electronics interfaces and making them effective for the problems we are considering, such as planning complex actions, making device behavior context-sensitive, and debugging. Next, we consider related work regarding some of the particular AI interface techniques used by Roadie, namely mixed-initiative interfaces, goal-oriented and Commonsense interfaces, and self-explanatory interfaces.

6.2. Interfaces to consumer electronics devices

To overcome the current usability crisis, researchers have proposed constructing Ambient Intelligent environments "that are sensitive and responsive to the presence of people" [5]. This approach classifies user scenarios in terms of *rituals* and *routines*. A ritual is something that the user values as a meaningful experience while routines are mundane tasks in everyday life. This distinction varies among individuals [8]. One of the goals of the Ambient Intelligence [7] approach is to magnify the pleasure in the rituals and minimize the interaction in the routines.

De Ruyter created a context-aware remote control where the state can be changed in response to the input of home sensors. In order to make the end user feel in control of the context-aware remote control, the user can modify its look-and-feel and contextual rules using special tools. His work acknowledges that it is hard for the end user to significantly decrease the amount of programming in the devices, and recognizes that a new programming metaphor needs to be developed [8].

However, de Ruyter does not propose any fundamentally new approaches either to controlling individual devices, or to programming behaviors for sets of devices. There is no provision for expressing high-level goals that are not directly reflected in concrete device operations, nor any provision for planning or debugging, both strengths of Roadie. Roadie also has rules, whose sources is EventNet, but whose representation is natural language. It is easy to imagine expanding the current Roadie functionality to allow correction of the rules while the system is in use.

The subfield of model-based interface design seeks to develop declarative descriptions of interfaces that can be automatically adapted or reconfigured for different platforms. MOBI-D Is a general framework to map abstract to concrete actions. This system is able to map the user's specifications for data needs to user interfaces [32].

Kühme exposes a technique called an *adaptive action prompter* interface, where the tools that are shown change based on the user's previous actions. This system uses modes and recency of use to show the user the most relevant tasks in a contextual menu [18]. While much work in this area concerns desktop computers, some work on adaptive interfaces has taken phones and consumer electronics devices as a target domain.

One popular approach is to build universal remote controls, able to control every single device [40]. This is done by downloading the specification from the appliance to the personal universal control (PUC), and it will render a suitable interface. The main goal of this research is to generate a general framework where the current interfaces are mapped to portable computers or cell phones [28]. Having these dynamic interfaces allows hiding or graying-out functions that are not available in a given context.

An approach to simplifying the internal operation of device networks is to model each device as an autonomous agent and use multi-agent communications and negotiation techniques to build an autonomous home [19]. These models try to predict the user's actions and intentions [5]. The main problem with this approach is that they cannot help the users when they are trying to perform a novel action, or when some agent of the system is not functioning as designed. They are also better suited to device-to-device communication rather than direct interaction with the user.

The MIT house_n project built a home equipped with multiple sensors. Volunteers live in this house for periods of time while the house collects real usage data [16]. This research aims to build houses that help their habitants to live long and healthy lives, reduce energy consumption, and integrate learning into their everyday activities in the home [15]. Similar experimental house-of-the-future projects also exist at Georgia Tech, Microsoft, Phillips, and elsewhere. These houses contain appliances such as washing machines and microwave ovens that could be targets for our approach. These projects emphasize sensor technology rather than explicit user input. We have already explored a kitchen application in Section 5.3.

Another approach to simplifying the user interface interaction is to use voice commands instead of controls. Precise is able to reliably translate *semantically tractable* sentences to SQL queries [29]. This approach has been applied to

generate PDDL statements to control consumer electronic devices [39]. This allows users to correctly control devices with direct verbal orders like "Increase the temperature 5°." This requires the user and the system to share the same vocabulary for talking about the concrete device operations. Roadie will be able to expand this approach to have more open-ended sentences like "It is too cold here" and infer that the user wants to increase the temperature. Both approaches can be merged by reformulating EventNet nodes to a format that complies with Precise's semantic tractability requirements.

6.3. Mixed-initiative interfaces

Collaborative or *mixed-initiative* interfaces are software agents that cooperate with the user to satisfy certain goals. The outstanding system within this paradigm is Collagen. This system uses SharedPlan [14] as a conversational model between the user and the interface, which enables it to effectively keep track of its interactions with the user. In addition, the agent interacts with the user interface in the same way a real user does, giving the user feedback concerning the operations performed by the agent [34].

The Collagen architecture has been applied to consumer electronics [35], in a system called DiamondHelp. This approach reduces the configuration overhead when the user knows what function of the device to use. DiamondHelp is a help system that explains procedures that accomplish high-level tasks to the user in terms of their concrete device operations.

Collagen works by having two avatars, one representing an agent and the other the user. Both agents can communicate by directly manipulating a shared user interface. This leads to an interactive paradigm where both are able to observe the actions and comment on them. In addition to direct manipulation, they can use their avatars to communicate to each other, using scripted dialogs using the SharedPlan theory. The user can communicate with the agent by clicking one of the proposed alternatives. In order to work, this approach needs both the agent and the user to have mutual beliefs; if there is a substantial belief discrepancy the SharedPlan scripted dialogs might not make sense. To solve this problem DeKoven [6] states that for the successful implementation of a collaborative interface, it needs to express what it believes the users' goals are and what the steps are to accomplishing the goal.

Roadie follows DeKoven's suggestions by presenting the user with an explanation of what it wants to do and the set of relevant EventNet nodes. These pieces of information shown the current system goals and beliefs.

One of the current problems with Collagen is its inability to deal with natural language, although exploratory work has been done [37]. Roadie is able to enable the user to implement a collaborative approach where the user can state his goals in plain English, and then the system proposes a set of goals.

Roadie guesses the user's goal and then configures the devices. And then, if a problem occurs the system provides information to find and solve the problem. Collagen takes a different approach by modeling the interaction between the user and the devices as a dialog. During the conversation the user and the agent agree on the function to use and try to reach the goal and in a cooperative way even when problems appear.

6.4. Goal-oriented and commonsense interfaces

An early application that used commonsense computing to infer the user's goals is Hugo Liu's Goose, a goal-oriented search engine. This search engine goes one step beyond current keyword matching of current search engines by reformulating user's queries to satisfy their goal. It is able to reformulate the query "*My cat is sick*" to "*Find a veterinarian in Cambridge, MA*" [26].

Furthermore, commonsense computing has been used to create a proactive calendar that suggests To-Do tasks based on the user's events. When the user adds a new appointment of event into the calendar, the application asks EventNet for the temporal related nodes. The answered nodes are filtered by the application, picking the ones that have patterns used in a To-Do list, the selected phrases are shown to the user. This application was designed to help the user remember common steps and prerequisites needed to perform an action. For example, if the user adds the event "Go to the beach" the system will suggest bringing a beach ball and sunscreen [11].

Other examples of commonsense interfaces can be found in [25]. Many of these interfaces share Roadie's approach of using commonsense to infer goals from concrete actions. Roadie is a goal-oriented interface in the same way than Goose is; it maps the user's goals to actions in a format and operations that the device can understand.

A variety of applications have used AI planners interactively. A planner has been used for controlling the configuration and operation of a disk management on the HPUX system [2], scheduling in a manufacturing line [20], in air traffic control [21], and in scheduling for a call center [12]. Writer Aid is a collaborative writing assistant that helps to identify and insert relevant papers from online resources. This application uses a planner to identify both a relevant bibliography and the action to perform [1]. Puerta's MOBI-D, discussed earlier, also uses a planner to plan interface configurations. All these systems, along with ROADIE, use planners to gain flexibility in the ways they can control a certain machine. The planners allow them to successfully accomplish unanticipated scenarios if they are possible.

6.5. Self-explanatory interfaces

The ability of systems to introspect their state and change it is called *reflection*. This ability means that a system is able to change its own behavior in order to satisfy its goals [10]. This ability is important in computer systems

since it allows the system to reason about its state when something does not go as expected. Furthermore, in case the system is not able to solve the problem, it can provide a good insight of what went wrong, possible causes, and what can be done to solve the problem.

Although reflection has just been implemented in a handful of *experimental programming* languages (see references in [27]) some popular languages do allow some degree of introspection. They can know the capabilities of the objects and variables at run-time [31]. Having programming languages with built-in support for introspection and reflection will help to build programs and applications using this programming paradigm.

The most significant system using this approach is Expect, a knowledge acquisition and reasoning tool [4]. This system has the ability to infer which pieces of knowledge are required, which are necessary to perform certain reasoning, and provide an explanation of why. For example: if the system is trying to ship merchandise to Los Angeles it will ask for the airports and seaports in this city and explain that this information is needed as input to the desired inference operation [13].

Woodstein is a debugging interface for web processes like on-line purchases. It provides reflection by allowing the user to go back to the webpage where an action occurred and introspect if the data shows it is correct or not. To introspect, the user can ask why and how something happened, and it can also tag the data as successful or unsuccessful [24].

Reflection has also been applied to programming environments. ZStep 95 [22] is a programming environment that keeps a memory of all the states of the program's execution. This history allows, while debugging, not only a full history of the execution of the code that can be stepped forward and backward, but it also answers questions like "When did the variable have that value?" or "Where in the code was that expression evaluated?"

Roadie provides introspection since it is able to change the configuration of its devices to satisfy the user's goals. In addition, it adds introspection to its internal beliefs by providing an explanation of why a certain action is suggested.

6.6. Expanding roadie's capabilities

6.6.1. Learning from user's habits

Learning from user's habits can be done in two ways. First, we can raise the weight of the links when goals are chosen and lower them when they are not chosen. Also, since Roadie can show the output of EventNet's temporal traces, the user should be able to mark the output links that are incorrect. A learning facility would also allow us to streamline the interface in the case that the user wants to perform simple tasks that they already know how to do and the system behaves as expected.

6.6.2. Allowing the user to set custom goals

Roadie's current use is to map the whole space of a user's possible activities to a small set of previously known goals. This approach has a fundamental flaw, as the system designer either has programmed in advance the available goals, or the goals that have to be added from somewhere else, such as the general-purpose OpenMind knowledge base.

This limited scenario can be expanded by adding programming by example [23] techniques. We've left a hook for that eventuality in providing a "Do something else" alternative when the list of Roadie's suggested goals is presented. Our intent is that the user should be able to train the system to perform a new task by demonstrating a sequence of actions. The action sequence can be generalized into a procedure for accomplishing that task, using machine learning techniques similar to explanation-based learning. Subsequently, that task can be automatically suggested using the techniques presented in this paper.

6.6.3. Improving Roadie user interface

One of the main problems with Roadie is that it heavily depends on natural language to communicate with the user. This characteristic makes the system very intrusive when the users wants to perform simple tasks that they already know how to do and the system behaves as expected. On the other hand, when problems arise it is good to have a rich interactive communication channel.

This can be improved by changing the Roadie interaction schema. First, the system should treat the tasks the user should perform and the ones that the system can perform differently. Then, if the suggested goal does not need user interaction, it should be presented in a different way than if it needs the user's attention. Furthermore, the system can assign icons to the more frequent operations, heavily reducing the amount of text displayed by the system. This brings the problem of how to easily and intuitively assign icons to similar tasks like playing a CD on the CD player or on the DVD player.

Then, when problems arise – an operation fails, the user complains about certain behavior of the system, etc – Roadie will show its current interface.

7. Discussion

7.1. Implications of goal-oriented interfaces

Raskin [33] argues we should rethink the computer interfaces as a small set of always accessible core operations and then build more complex operations around this small set. His approach, while appealing, has two fundamental limitations. First, there is no one-size-fits-all method for selecting those small core operations. So the user ends up having a huge set of core capabilities, and then wondering which capability he wants and how to access it. Second, it is easy to map a simple goal to simple actions but the task of mapping from the goal "emphasize these ideas" to "make the text bold" is still left to the user. With a goal-oriented interface, like Roadie, this distinction gets blurred. Since the system can map high-level goals to

low level actions, there is no need to keep the core functions small.

7.2. A plan recognizer with commonsense vs. other plan recognizing approaches

Statistical or logical plan recognizers can be easily used to mimic the basic EventNet operations. Statistical plan recognizers can be trained on a corpus of associations between sequences of actions and statements of goals, and build up correlations incrementally. Logical plan recognizers deduce correspondences between actions and goals from first-principles axiomatizations of specific knowledge domains.

The first advantage of using EventNet is that the systems built on top of it are able to provide the user with the reason for the suggestion, unlike statistical approaches, and allow the user to correct the system's knowledge base. Statistical learners require huge numbers of examples, and inessential quirks in the corpus can result in skewing the statistically-derived generalizations. Second, EventNet can work with more open-ended scenarios than is typically the case with logical recognizers. The Commonsense knowledge used by EventNet is typically more plentiful, intuitive and robust against minor variations than handcrafted logical representations.

7.3. Scruffy vs. neat knowledge approaches

In Artificial Intelligence there is an ongoing debate between the *scruffy* and *neat* approach, as two separated and incommensurable approaches. The *neat approach* argues that AI theory should be rooted in mathematical rigor and the *scruffy approach* prioritizes the experimentation of multiple ideas by prototyping.

Roadie is able to use an hybrid approach. It uses Graphplan, a first order logic planner to manipulate the devices, and EventNet to interact with the user. Graphplan is a classical neat algorithm while EventNet is a scruffy one; each one does its part and complements the other one.

Graphplan controls the devices since consumer electronics have a well defined and predictable logic of operations that can be managed by the planner.

EventNet tries to infer the operations that the user wants to accomplish by having an unstructured model of typical human behavior. Using rigorous mathematics to directly model goals and desires of people is an extremely complicated, if not impossible task. Conversely, you cannot ask EventNet for the "right" answer to a given query, but only for what is plausible. For this reason, EventNet or a similar algorithm is unsuitable for performing the task at which Graphplan excels.

The decision of picking a *neat* or a *scruffy* algorithm depends on the nature of the problem that is being solved. If there is a mathematical representation of the problem, a *neat* algorithm might be the best choice. But, if the characteristics of the problem make it difficult to have a computable mathematical representation a *scruffy* algorithm might be better suited.

8. Conclusion

In this paper we show an interaction schema for consumer electronics. This schema uses a plan recognizer based on the OpenMind Commonsense corpus to find out the users' intentions and propose relevant device action sequences. In addition, it uses a planner to control and manipulate the devices, saving the users the trouble of dealing with low level configuration and helping them to debug the devices when something does not goes as planned.

Acknowledgements

Our thanks to all the media lab sponsors for their support of this project, and to Rada Mihalcea of the University of North Texas for help us to mine the data from the Web.

References

[1] Tamara Babaian, J. Barbara Grosz, Stuart M. Shieber, A writer's collaborative assistant, in: Proceedings of the Seventh International conference on Intelligent User Interfaces, ACM Press, 2002, pp. 7–14.
[2] Rosy Barruffi, E. Lemma, Michela Milano, P. Mello. Application of planning techniques for system configuration tasks, in: Proceedings of the Fourth Workshop of the OpenView University Association, OVUA'97, Madrid, Spain, 1997, April 2–4.
[3] Avrim Blum, Merrick Furst, Fast planning through planning graph analysis, in: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI 1997), Montreal, Canada, 1997, pp. 1636–1642.
[4] Jim Blythe, Jihie Kim, Surya Ramachandra, Yolanda Gil. An integrated environment for knowledge acquisition, in: Proceedings of the Sixth International Conference on Intelligent User Interfaces, ACM Press, 2001, pp. 13–20.
[5] Diane J. Cook, Michael Youngblood, Edwin O. Heierman, Karthik Gopalratnam, Sira Rao, Aandrey Litvin, Farhan Khawaja, MavHome: an agent-based smart home, in: Proceedings of the IEEE International Conference on Pervasive Computing and Communications, IEEE Press, 2003, pp. 521–524.
[6] A. Elyon, M. DeKoven, Help Me Help You: Designing Support for Person-Product Collaboration, PhD Thesis, Delft University of Technology, 2004.
[7] Boris de Ruyter, Emile Aarts, Ambient Intelligence: visualizing the future, in: Proceedings of the Working Conference on Advance visual interfaces, ACM Press, 2004, pp. 203–208.
[8] Boris de Ruyter, Richard van de Sluis, Challenges for end-user development in intelligent environments, in: Henry Lieberman, Fabio Paterno, Volker Wulf (Eds.), End-User Development, Springer, 2006.
[9] R. Burke, K. Hammond, B. Young, The FindMe approach to assisted browsing, IEEE Expert (1997) 32–39.
[10] Paul. Dourish, Developing a reflective model of collaborative systemsACM Transactions on Computer–Human Interaction, ACM Press, 1996 (pp. 40–46).
[11] Jose Espinosa, Henry Lieberman, EventNet: inferring temporal relations between commonsense events, in: Proceedings Fourth Mexican International Conference on Artificial Intelligence, Springer Publisher, Monterrey, Nuevo Leon, Mexico, 2005, pp. 14–18.

[12] Alex Fukunaga, Ed Hamilton, Jason Fama, David Andre, Ofer Matan, Illan Nourbakhsh, Staff scheduling for inbound call center and customer contact centers, AI Magazine 23 (4) (2002).
[13] Yolanda Gil, Knowledge refinement in a reflective architecture, in: Proceedings of the Twelfth National Conference of Artificial Intelligence (AAAI-94), vol. 1, AAAI, 1994, pp. 520–526.
[14] B.J. Grosz, C.L. Sidner, Plans for discourse, in: P.R. Cohen, J.L. Morgan, M.E. Pollack (Eds.), Intentions and Communication, MIT Press, Cambridge, MA, 1990, pp. 417–444.
[15] Stephen S. Intille, Designing a home of the future, IEEE Pervasive Computing (2002) 80–86.
[16] S.Stephen, I. K. Larson, J.S. Beaudin, J. Nawyn, E. Munguia Tapia, P. Kaushik, A living laboratory for the design and evaluation of ubiquitous computing technologies, in: Extended Abstracts of the 2005 Conference on Human Factors in Computing Systems, New York, NY: ACM Press, 2004.
[17] Chia-Hsun Jackie Lee, Leonardo Bonanni, Jose Espinosa, Henry Lieberman, Ted Selker, KitchenSense: augmenting kitchen appliances with shared context using knowledge about daily events, in: International Conference on Intelligent user interfaces, Sydney, Australia, January 2006.
[18] Thomas Kühme, Uwe Malinowski, James D. Foley, Facilitating interactive tool selection by adaptive prompting, in: INTERACT '93 and CHI 93 Conference Companion on Human Factors in Computing Systems, ACM Press, 1993, pp. 149–150.
[19] Victor Lesser, Michael Atighetchi, Brett Benyo, Brett Horling, Anita Raja, Regis Vincent, Thomas Wagner, Xuan Ping, Shelley XQ Zhang, The intelligent home testbed, in: Proceedings of the Autonomy control Software Workshop (Autonomous Agent Workshop) January 1999.
[20] Terrence Harvey, Keith Decker. Planning alternatives for an intelligent scheduler, in: IJCAI 2001 AI in Manufacturing Workshop, 2001.
[21] Wolfgang Hatzack, Bernhard Nebel, The operational traffic control problem: computational complexity and solutions, in: A. Cesta (Ed.), Proceedings of the Sixth European Conference on Planning (ECP'01), 2001.
[22] Henry Lieberman, Christopher Fry, Step 95: a reversible, animated source code stepper, in: J. Stasko, J. Domingue, M. Brown, B. Price (Eds.), Software Visualization: Programming as a Multimedia Experience, MIT Press, 1997.
[23] Henry Lieberman (Ed.), Your wish is my command: programming by example, The Morgan Kaufmann Series in Interactive Technologies, 2001.
[24] Henry Lieberman, Earl wagner: end-user debugging for electronic commerce, in: Proceedings of the eighth international Conference on Intelligent user interfaces, ACM Press, 2003.
[25] Henry Lieberman, Hugo Liu, Push Singh, Barbara Barry, Beating common sense into interactive applications, AI Magazine 25 (4) (2005) 63–76.
[26] Hugo. Liu, Henry. Lieberman, Ted. Selker, Goose: a goal-oriented search engine with commonsense, in: Proceedings of the second international conference on Adaptive Hypermedia and Adaptive Web-Based Systems, ACM Press, 2002, pp. 253–263.
[27] Pattie Maes, Concepts and experiments in computational reflectionConference proceedings on Object-Oriented Programming Systems, Languages and Applications, ACM Press, 1987, pp. 147–155.
[28] Jeffrey Nichols, Brad Myers, Thomas Harris, Roni Rosenfeld, Stefanie Shriver, Michael Higgins, Joseph Hughes, Requirements for automatically generating multi-modal interfaces for complex appliancesFourth IEEE International Conference on Multimodal Interfaces, IEEE Press, 2002, p. 377.
[29] Ana-Maria Popescu, Oren Etzioni, Henry Kautz, Towards a theory of natural language interfaces to databases, in: Proceedings of the Eighth International Conference on Intelligent User Interfaces, ACM Press, 2003, pp. 149–157.
[30] Donald. Norman, The Psychology of Everyday Things, Perseus, New York, 2002.
[31] Patrick O'Brien. Guide to Python Introspection. http://www-128.ibm.com/developerworks/library/l-pyint.html.
[32] Angle Puerta, Jacod Eisenstein, Towards a general computational framework for model-based interface development system, in: Proceedings of the Fourth International Conference on Intelligent User Interfaces, ACM Press, 1998, pp. 171–178.
[33] Jef. Raskin, The Humane Interface: New Directions for Designing Interactive Systems, First Ed., Addison-Wesley Professional, 2000.
[34] Charles Rich, Candy Sidner, Neal Lesh, Collagen: applying collaborative discourse theory to human-computer interaction. Special Issue on Intelligent User Interfaces, AI Magazine 22 (4) (2001) 15–25.
[35] Charles Rich, Candy Sidner, Neal Lesh, Andrew Garland, Shane Booth, Markus Chimani, DiamonHelp: a graphical user interface framework for human–computer collaboration, in:IEEE International Conference on Distributed Computing Systems Workshops, June 2005, pp. 514–519.
[36] Push Singh. The public acquisition of Commonsense knowledge, in: Proceedings of AAAI Spring Symposium on Acquiring (and Using) Linguistic (and World) Knowledge for Information Access, 2002.
[37] Candace L. Sinder, Clifton Forlines, Subset Languages for Conversing with Collaborative Interface Agents. in:International Conference on Spoken Language Processing (ICSLP), September 2002.
[38] Universal Plug and Play Device Architecture, 2000. <http://www.upnp.org/download/UPnPDA10_20000613.htm>.
[39] Alexander Yates, Oren Etzioni, Daniel Weld, A reliable natural language interface to household appliances, in: Proceedings of the Eighth International Conference on Intelligent User Interfaces, ACM Press, 2003, pp. 189–196.
[40] Gottfried Zimmermann, Gregg Vanderheiden, Al Gilman, Prototype implementations for a universal remote console specification, in: ACM Special Interest Group on Computer–Human Interaction, ACM Press, 2002, pp. 510–511.