Tools for Programming Human Robot Interaction

James P. Diprose
Dept. of Computer Science
University of Auckland
New Zealand
jdip004@aucklanduni.ac.nz.

Abstract— Whilst robots are increasingly being used in scenarios that involve human-robot interaction, it is still difficult to program them to interact with humans. This is because current programming tools either require programmers to work at low abstraction levels or they lack features needed to implement particular aspects of human-robot interaction. Our goal is to create an API that is both capable of programming a wide range of human-robot interaction scenarios and is easy to use by the various users of human-robot interaction programming tools. We have taken a first step toward this API by developing an exemplar, high level API for programming social interaction and evaluating it with the Cognitive Dimensions framework. We plan to explore other aspects of human-robot interaction, including navigation and manipulation and exploring how they should be integrated with our existing primitives for programming social interaction.

Keywords— application programming interfaces, api, usability, design, cognitive dimensions, human robot interaction, social robot interaction, navigation, manipulation, humanoid robot.

I. INTRODUCTION

The vision of robots and humans interacting with each other in daily life is progressively becoming a reality. There are already many applications where robots interact with humans, including: companions for the elderly [1], interactive theatre [2] and robotic butlers [3]. Whilst much research is devoted to human-robot interaction, it is still difficult to create applications such as those mentioned above. There are three reasons for this. First, the tools used to create human-robot interaction scenarios lack support for many human-robot interaction requirements. Second, tools that do support human-robot interaction requirements often express them at too low an abstraction level. Lastly, there is little recognition of the diverse range of users and methods that share the same goal of creating human-robot interaction.

A lack of support for human-robot interaction requirements means that programmers are not able to fully express the nuances of human-robot interaction. For example, understanding human speech involves understanding what the speaker said [4], how they said it [4], who was speaking [4] and who they were speaking to. If the programming tool doesn’t support these requirements, then the programmer can’t include them in applications they create.

On the other hand, supporting human-robot interaction requirements, but representing them at low abstraction levels is a problem because the programmer has to play programming games to attain their goal. For example, a programming tool could support most of the requirements of understanding human speech by providing sound source localisation, tracking, separation and speech recognition (see [5] for details). However, it would be difficult and time consuming to directly use these algorithms to create human-robot interaction because the programmer has to use abstractions below the application level.

Lastly, there is little recognition of the diverse range of users and methods that share the same goal of creating human-robot interaction. End user programming [6]–[8], programmers [9] and artificial intelligence systems [10], [11] all have the shared goal of creating human-robot interaction. It is likely that different interfaces will need to be created for each target audience because they each have different needs and represent human-robot interaction in different ways.

Our intention is to create an API that is both capable of programming a wide range of human-robot interaction scenarios and is easy to use by the various users of human-robot interaction programming tools. The most significant trade-off when designing such an API resides between two design decisions: the expressability vs the usability of the API. Enough features need to be included in the API so that it is useful (expressability), but they need to be represented so that they are easy to learn and use (usability).

Our approach to producing an API that balances these two design decisions is based on a combination of the Three Examples pattern from the Evolving Frameworks Pattern Language [12] and the Cognitive Dimensions framework [13]. To discover the features required in the API, we choose three exemplar human-robot interaction applications and create APIs for each of them [12]. The generic API emerges by finding abstractions common to all three APIs [12]. This is similar to the methodology used by Li [14] to create a generic framework for event integration specification. To maximise usability, we use the Cognitive Dimensions framework [13] as a design tool to weigh up the trade-offs that occur when designing each API. Then when each API is finished, a usability study is performed to understand how real users perceive it.
II. SOCIAL INTERACTION

We have already created an API that is designed to meet the requirements of social interaction. It was used to program a scenario where a Nao humanoid robot acts as the host of a game show where two human players compete against each other. The Cognitive Dimensions Framework was used as a design tool when the API was being designed. The API contains a number of specially designed classes and functions, for example, the API provides a function called say_to which controls both robot speech and gesture (Table I). The programmer specifies the person or people the robot speaks and gazes at, as well as how and when the robot should gesture. A Cognitive Dimensions evaluation was performed and showed that programmers with no robotics knowledge were positively impressed by the notation and that its organization, domain specific interfaces and object oriented nature positively affected several Cognitive Dimensions.

<table>
<thead>
<tr>
<th>say_to</th>
<th>Parameters</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>text (String), audience (Object, Query)</td>
<td>robot.say_to('Hi', people)</td>
<td>robot.say_to('Hi', people)</td>
</tr>
<tr>
<td></td>
<td>robot.say_to('&lt;wave&gt; Hi &lt;/wave&gt;', people)</td>
<td>robot.say_to('&lt;point target={(0)} who is that? &lt;/point&gt;', people, person1)</td>
</tr>
</tbody>
</table>

III. FUTURE WORK

Our goal is to create a more general framework for programming human robot interaction, with better support for the requirements of human robot interaction, a much higher abstraction level and interfaces for different audiences. We have made a first step toward this goal by exploring the multiplayer quiz scenario, creating an API to program it and evaluating the API with the Cognitive Dimensions framework.

The next step (which is currently underway) involves exploring scenarios that involve navigation and manipulation as well as social interaction, so that appropriate abstractions for the unexplored aspects of human-robot interaction can be discovered. A possible scenario is where a robot interacts with a group of people socially, asking them for coffee orders and then fetching & delivering the coffee to them. This involves discovering the navigation and manipulation primitives needed to fetch & deliver the coffee as well as how they should be combined with primitives for specifying social interaction. For example, how would a programmer specify arm movements for carrying coffee as well as gestures for interacting with people.

The last step involves tailoring the interfaces for social interaction, navigation and manipulation for different target audiences. For example making APIs for both programmers who want to create human-robot interaction themselves and programmers that want to integrate human-robot interaction with an artificial intelligence system. The following research questions summarise our future work:

- How can we abstract robot navigation and manipulation so that they are easy to program, yet can still able to express their respective requirements?
- How can we create a harmonious level of abstraction for navigation, manipulation and social interaction so that they can all be used together?
- How should the interfaces provided for social interaction, navigation and manipulation differ, depending on the target audience?

REFERENCES