Robotics: Science and Systems 2011
Los Angeles, CA, USA, June 27-30, 2011

An Interaction Design Framework for Social Robots

Dylan F. Glas, Satoru Satake, Takayuki Kanda, Norihiro Hagita

Intelligent Robotics and Communication Laboratories, ATR, Kyoto, Japan
{dylan, satoru, kanda, hagita}@atr.jp

***Abstract*—We present a novel design framework enabling the development of social robotics applications by cross-disciplinary teams of programmers and interaction designers. By combining a modular back-end software architecture with an easy-to-use graphical interface for developing interaction sequences, this system enables programmers and designers to work in parallel to develop robot applications and tune the subtle details of social behaviors. In this paper, we describe the structure of our design framework, and we present an experimental evaluation of our system showing that it increases the effectiveness of programmer-designer teams developing social robot applications.**

Keywords-human-robot interaction; software development; social robotics; programming interfaces;

I. INTRODUCTION

The field of social robotics is still young, and although much research has focused on details of creating humanlike interactions for social robots, little attention so far has been paid to the development process itself, which is usually performed by programmers. However, this is really a cross-disciplinary process integrating technical knowledge of hardware and software, psychological knowledge of interaction dynamics, and domain-specific knowledge of the target application.

The development of social robot applications faces not only the conventional challenges of robotics, such as robot localization and motion planning, but also new challenges unique to social robots, including new kinds of sensory-information processing, dialog management, and the application of empirical design knowledge in interaction. Examples of this design knowledge include maintaining acceptable interpersonal distance [1], approaching people from a non-frontal direction [2], and controlling the duration and frequency of eye contact [3], all of which have been shown to be important for social robots.

Applications developed in a research context are usually small-scale and engineered by small groups of highly-capable individuals. However, scaling this process to the level of real-world commercial deployment requires a collaborative design process involving people with different areas of expertise.

For example, algorithms and software modules are often developed for information-processing tasks like human tracking, social group detection, gesture recognition, prediction of human behavior, or dynamic path planning. Development of such modules fundamentally requires programming expertise.

Other tasks do not, by their nature, require programming ability. These include scripting the robot's utterances, choosing gestures, and structuring the sequence of the robot's actions. Sometimes the specialists most qualified to design the interaction flows or contents of robot behaviors are non-programming researchers or domain experts. However, these specialists are often required to rely upon programmers for development and modification of interaction flows and behavior contents.

Such a design process is inherently inefficient. To improve efficiency in the design of social robotics applications, a structured framework is necessary to enable these fundamentally distinct aspects of social robot application development to be conducted in parallel.

In this paper we propose a framework which uses clearly-defined layers of abstraction to allow this kind of parallel development. In our framework, programming specialists are free to focus on low-level programming tasks like hardware interfacing or data processing. These low-level components are then encapsulated and presented to interaction designers via an easy-to-use graphical interface for developing interaction flows and fine-tuning details of the robot's utterances and gestures.

II. RELATED WORK

Related research has explored robotics development frameworks, dialogue management, and the handling of gestures and nonverbal communication.

A. Development frameworks

Many powerful development tools exist for programming robot systems [4], and some frameworks such as ROS and Player/Stage have been adopted widely by robotics specialists. Development environments such as Choreographe [5] enable smooth motion and behavior planning for complex operations such as dancing. Some development environments are targeted towards novice users or even young children [6]. However, all of these systems generally focus on conventional robotics problems such as navigation, mapping, and motion planning.

Some development environments are targeted more specifically towards development of robots for social interaction, including the capacity for developing dialogue management in addition to conventional robotic capability [7, 8]. However, these systems are still based on programming or scripting, and are not intuitive for nontechnical users.

For dialogue development, the CSLU RAD toolkit [9] provides a flowchart-based interface for building dialogue flows, but its inputs and outputs are limited to speech only. A framework for social robots will need to handle many kinds of sensor inputs and actuate both speech and robot motion.

This work is supported by the NEDO (New Energy and Industrial Technology Development Organization, Japan) project, 'Intelligent RT Software Project.'

B. Dialogue management

1) Traditional dialogue management

There are three main approaches that have been used to create dialogue management systems: state-based, frame-based, and plan-based [10].

State-based systems generate utterances and recognize users' responses according to a state-transition model, like a flowchart. This approach is simple and intuitive, and thus easy to implement and often used in working systems.

Frame-based systems fit users' responses into pre-defined slots in "frames" to estimate user goals. These are often used for telephone-based dialogue systems, *e.g.* for providing weather or transportation information. Frame-based systems can handle more complex information, but involve more effort for preparation of such frames of knowledge.

Plan-based, or "agent-based," systems use a set of rules to change the internal states of an agent to navigate through conversation, *e.g.* [11]. These can handle the most complex interactions, but require very advanced natural-language processing and well defined sets of rules. This approach is often used in research but rarely used in working systems [10].

We chose a state-based approach, as it is the simplest of these three, and it is sufficient to represent the flow of a simple conversational interaction. As our system is aimed at non-programmers, simplicity and clarity are important for usability.

Although handling user-initiated interactions is one weak point of state-based approaches, it is possible to build rich interactions by designing them to be robot-initiated. This may appear to be a disadvantage of state-based modeling, but it is worth noting that robot-initiated interactions are often necessary in order to set expectations for a robot's capabilities.

2) Dialogue management in robotics

In robotics, dialogue management has sometimes been studied while taking real-world difficulties into consideration. For example, Matsui et al. integrated multimodal input for a mobile office robot [12, 13]. Roy et al. used POMDP's to take account of speech recognition errors in state-based transitions of dialogue [14, 15]. For social robots, many architectures for cognitive processing have been developed [16, 17]. The BIRON system has used state-transition models [18] and common-ground theory [19] to direct dialogue. However, the majority of such systems have been task-oriented, that is, aimed primarily at communicating commands or teaching information to a robot [8, 20, 21]. This is different from social dialogue, where the goal of the robot may be to entertain, interest, or persuade a customer.

Some research has focused on using generic dialogue patterns to generate dialogue for human-robot interaction, e.g. [22, 23]. Such research has been directed towards modeling exchanges such as factual confirmations, but not stylistic aspects such as politeness or subtlety of wording. Although such patterns enable automation of certain simple exchanges, it is not yet possible to create humanlike social interaction based on dialogue patterns alone. For applications focusing on social interaction, human knowledge is still needed at the level of implementing dialogue flow.

C. Nonverbal communication

For embodied robots, interaction includes not only dialog management, but nonverbal communication as well. Many aspects of nonverbal behavior have been explored, such as the use of gestures and positioning [4, 7, 11, 22, 24, 25], gaze control [3, 26, 27, 28], and nodding [27, 29]. Nakano *et al.* also developed a mechanism to generate nonverbal behavior based on speech context in an embodied conversation agent [30].

Our proposed architecture allows such nonverbal behaviors to be implemented in the robot. Both implicit behaviors, such as gaze-following, and explicit behaviors, such as gestures synchronized with the dialog, are supported.

III. Interaction Design Framework

A. Division of Roles

The concept of division of roles drives the design of our proposed approach. Roughly speaking, we can categorize the main developers of a robot application into "programmers" and "designers." Developers in these two roles contribute in different ways to the implementation of a social robotics application. These different contributions must be reflected in the design framework and user interface.

1) Programmer

There are several tasks which by their nature require programming expertise.

Hardware interfacing: Adding new sensors or actuators to the system will require work at the robot driver level to enable the new components to operate with the robot's control system.

Data processing: New recognition techniques or machine learning algorithms will be necessary to help the robot understand the situation in its environment.

Behavior development: Basic interactive robot behaviors need to be developed, *e.g.* a behavior for approaching a moving person in a socially-appropriate way, based on tracking information from an external sensor network.

2) Interaction Designer

The tasks of an interaction designer center around the creation of content for the robot's interactions, and the creation of logical sequences of robot behaviors to be executed. Specifically, design tasks include the following:

Dialogue generation: An interaction designer will need to specify the robot's utterances and gestures. To tune the robot's performance, a designer could adjust the speed of the robot's actions or speech, or insert appropriate pauses.

Interaction flow design: By linking the robot's behaviors into sequences, a designer can create interaction flows. The designer needs to consider the order in which the robot should present information, when it should ask questions, and how it should respond to a person's actions. Non-dialogue elements could be used in these flows, such as driving to a new location or approaching a customer. An understanding of HRI design principles would be useful for an interaction flow designer.

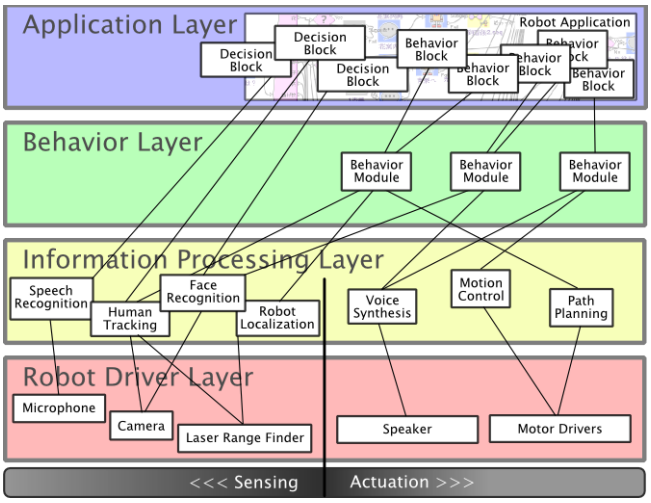

Figure 1. Four-layer robot control architecture.

Content entry: It may also be necessary to enter large amounts of domain-specific content, such as items in a restaurant menu, details about products in a store, directions to locations in a shopping mall, or information about seasonal events. This task might require a designer with specific domain knowledge relevant to the target application.

B. Robot control architecture

Our system uses a four-layer architecture, shown in Fig. 1. While similar to other modular architectures, the emphasis here is on the encapsulation of low-level control and processing into simple components such as behaviors and explicit gestures, which can easily be used by a non-programming designer to create social interaction flows.

1) Robot Driver Layer

The lowest layer is the robot driver layer, which contains hardware-specific driver modules. These modules support abstract interfaces that hide minor differences between similar robots, such as different motor or joint configurations, or size differences (*e.g.* slightly longer/shorter arms, human-size or baby-size, etc.). This enables the same applications and behaviors to be used with different robots, as long as they are functionally similar, *e.g.* wheeled humanoid robots. Our architecture currently supports four robot platforms.

The concept of modular drivers is not new, and in theory it should be possible to implement a system like ours on top of popular modular middleware frameworks such as Microsoft's Robotics Developer Studio or Willow Garage's ROS.

2) Information Processing Layer

The information processing layer contains sensing and actuation modules. Sensing modules are components related to recognition of environments and activities in the real world. Examples include localization, human tracking, face detection, speech recognition, and sound source localization.

Actuation modules perform processing for tasks like path planning or gaze following. Some knowledge about social behavior is implemented here. Following the approach in [31], we classified non-verbal behaviors as *implicit*, which do not need to be specified by designers, and *explicit*, which need to be synchronized with utterances. Based on the state of conversation (*e.g.* talking, listening, or idling), components in this layer generate implicit behaviors such as gaze control.

Some frameworks (*e.g.* Microsoft RDS), are primarily targeted towards development at this level for robotics research or education, but our framework considers this layer mainly as infrastructure to enable the creation of higher-level behaviors.

3) Behavior Layer

The concept of a robot "behavior" as a combination of sensor processing and actuation is used both in behavioral robotics, *e.g.* [32], and in social robotics [25]. Examples for social robots include guide behaviors incorporating speech, gesture, and timing, or approach behaviors which react to a person's trajectory [33].

In our architecture, behaviors are implemented as software modules in the behavior layer which execute actions and react to sensor inputs. They can incorporate social knowledge, for example, by specifying gestures like tilting the robot's head to one side while asking a question [31]. It is also possible to design behavior modules to be configured by designers from the application layer. This is a powerful concept, as it enables the development of flexible, reusable behavior modules.

4) Application Layer

The highest layer is the application layer, where designers can develop social robot applications. Using "Interaction Composer," the graphical interaction development environment shown in Fig. 2, non-programmers can access behavior and sensor modules in the underlying layers. This software enables interaction flows to be built by assembling behavior and decision blocks into sequences resembling flowcharts. [1]

C. Interaction Composer

It is important to note that Interaction Composer (IC) is not simply a graphical programming language. Its graphical representations map directly to the underlying software modules, making it a tool that bridges the gap between designers and programmers.

1) Behavior Blocks

Behavior blocks (the blue blocks in the flow example shown in Fig. 2) allow the designer to use the behavior modules defined by programmers. These can represent behaviors like asking a question or giving directions. By configuring the properties of a behavior, the designer creates a "behavior instance." A behavior flow may contain many instances of general behaviors like "Talk" and "Ask". When a programmer creates a new behavior module, a corresponding block becomes available for designers to use in IC.

A developer can also allow a designer to provide arguments for behaviors. By configuring behaviors through IC, a designer can easily use behaviors in different ways without knowing the details of the program embedded in the behavior.

---

[1] Interaction Composer is a part of the Intelligent RT Software project supported by NEDO. The software is available by request until the end of 2011 at http://www.irc.atr.jp/ptRTM/. Its availability after that period is currently under discussion.

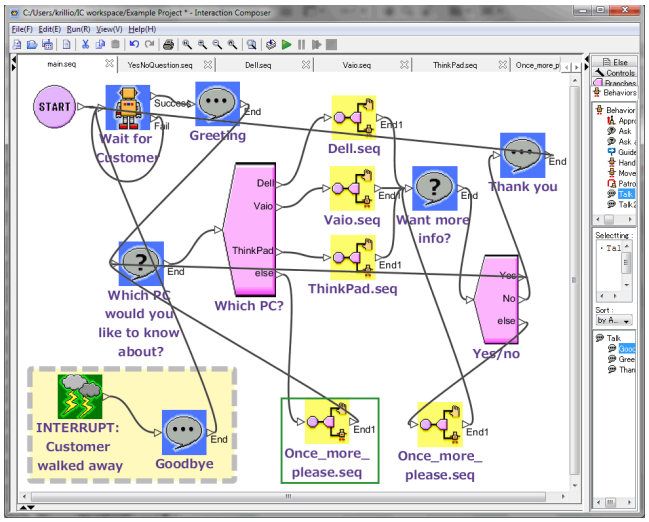

Figure 2. Screenshot of Interaction Composer.

Fig. 3 shows two possible configurations of the "Approach" behavior. The behavior itself involves complex information processing such as dynamic path-planning for approaching from the frontal direction of the person. However, the concept of "approach" is easily understood, and thus a designer can use the behavior without knowing the details of internal mechanism. There are three arguments prepared for the "Approach" behavior: the robot's speed, the distance at which to speak to the target person, and the contents of utterance. The designer could configure the behavior for "slow speed" and "social distance (1.5 m)" for approaching a waiting person; or with "fast speed" and "public distance (3 m)" in case of catching up with a person who forgot an item.

IC also supports easy addition of robot gestures. Fig. 4 and 5 show screens for editing gestures to be associated with utterances. In Fig. 4, a designer inputs an utterance, (”How about this laptop?”), and then chooses a part (”this laptop”) to add a gesture. By clicking the "reference" button in Fig. 4, a new screen for reference (pointing) gesture appears (Fig. 5), in which the designer can choose a pre-defined label ("LaptopA") for a pointing gesture. The robot will do the pointing gesture when it utters the "this laptop" phrase toward the object labeled "LaptopA". Other gestures, such as “emphasis,” “big,” “small,” etc. can also be selected.

The robot also used implicit gestures, causing it to move its arms and head slowly while idling, more actively while talking, and tilting its head to the side while asking questions. When explaining the different products, explicit behaviors were also included in the utterances, such as emphasis gestures and pointing to the products being explained.

2) Decision Blocks

The flow of the interaction can be controlled by using decision blocks (the pink blocks in Fig. 2) to direct the execution flow based on data from sensor inputs or internal state variables. These blocks enable the designer to work directly with human-readable data from the information processing layer (see Sec. III-B-2).

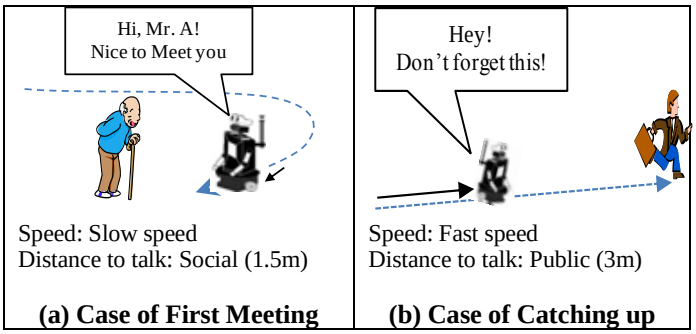

Figure 3. Example configurations of an "Approach" behavior

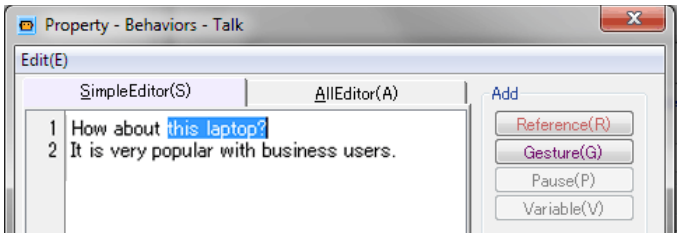

Figure 4. Selection of the utterance where a robot points to an object

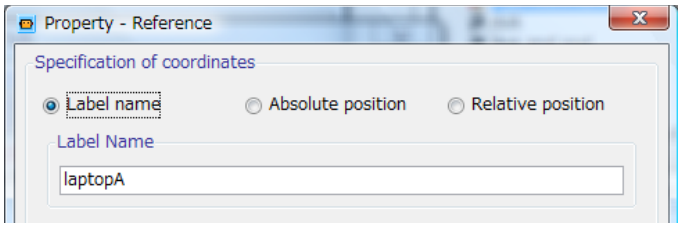

Figure 5. Selection of a pointing target

Examples of sensor inputs are the “ListenWord” (speech recognition results) and “DistanceHumanToLabel” (human position tracking) variables. For example, if a designer creates a scenario for shop assistant robot, the designer could specify that the robot should ask if the customer is looking for a desktop when “DistanceHumanToLabel(DesktopPC) <=1000”, meaning the customer is standing within one meter of the desktop PC. This information comes from the information processing layer, so the designer needs to know nothing about the implementation of the tracking algorithm.

“Sequential” and “random” decision blocks are also provided, which can be used to select a different output node each time they are executed. These blocks can be useful for adding lifelike variation to behaviors which are often repeated.

3) Sequences

Using only behavior and decision blocks, a program can quickly grow to be unmanageable in size and complexity. To manage this complexity, our system enables encapsulation of execution flows into subroutines, which we call “sequences.”

Sequences (the yellow blocks in Fig. 2) can be edited as separate execution flows, and then used as blocks within other sequences. They are a powerful tool, increasing readability and enabling structured development and debugging of interactions.

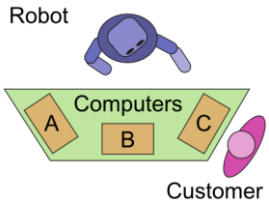

Figure 6. Layout of robot and computers in experiment space.

Some uses for sequences include encapsulating common tasks such as confirmation questions or delineating sections of a dialogue flow.

4) Interrupts

The flowchart-based representation of a dialog flow is helpful for structured, robot-driven interactions, but it does not allow us to easily react to unexpected situations. For example, if a customer walks away during an interaction, it might be best for the robot to stop speaking and begin searching for a new customer. Yet, a flow which performs such a check after every utterance would be tedious to build and hard to read.

For situations like this, we provide an "interrupt" mechanism. When the conditions of an interrupt are satisfied, the robot's execution flow will jump to a specified sequence.

For example, an interrupt could monitor the robot's on-board laser range finder, interrupting the flow if no human is detected in front of the robot. If this interrupt is triggered during a conversation, it means the customer has walked away, so instead of finishing a one-sided conversation, the robot could search for a new customer.

IV. Experimental Evaluation

We conducted an experiment to evaluate the effectiveness of our parallel design approach using Interaction Composer. In our experiment, teams of one programmer and one designer collaborated to develop a small application for a shopkeeper robot at a computer store. An example of one application developed in this experiment can be seen in the video in [34].

A. Conditions

We used a between-participants experimental design with two experimental conditions: *with-IC* and *no-IC*. In the *with-IC* condition, the designer used Interaction Composer to build the behavior flow, while the programmer worked in C to solve the programming problems. For the *no-IC* condition, Interaction Composer was not provided. Instead, the designer created interaction flows on paper, and the programmer implemented them in C, while also working on the programming tasks.

B. Experimental Setup

For this experiment we used a Robovie R-2 humanoid robot. Speech recognition was performed using the ATRASR speech recognition engine [35], and face detection was performed using a custom application written using OpenCV.

The experimental environment was laid out as shown in Fig. 6, with a stationary robot placed behind a table. Three laptop PC's were placed on the table, and a customer stood across the table from the robot, looking at the PC's. As the customer moved around to examine different PC's, the robot could determine the customer's position by turning its head and using face detection, and it could conduct simple conversations with the customer using speech recognition.

C. Task Specifics

To choose an appropriate balance of tasks, we considered what a typical preparations for a deployment of social robots might entail. Let us assume robots are to be deployed in a retail shop as sales associates. This would require sophisticated social interactions such as providing product information, making recommendations based on customer needs, explaining special offers, and gently encouraging customers to buy more expensive products or accessories. The robots would need to display professionalism as their actions reflect on the shop and influence customers' purchasing decisions. Assume further that the core robot system itself is a stable system for commercial use, but it has recently been upgraded with new sensors.

The design tasks to prepare for such a deployment might include developing hundreds of explanations, creating many different patterns of interaction sequences, fine-tuning the timing and gestures, and testing the smoothness of flow transitions. Programming tasks might include developing and testing recognition software for use with the new sensors. In such a situation, it would clearly be advantageous to have the interaction content developed by domain experts familiar with the products and sales techniques in the shop, enabling the programmers to concentrate on the programming tasks.

Although a real development cycle would require many people and several months, we designed this experiment to be completed within a single day. To demonstrate our proposed approach, representative design and programming tasks were chosen which could be achievable within a few hours of work. The task specifications were the same for both conditions.

1) Design Task

The design task for this experiment was to develop content and an interaction flow enabling the robot to explain at least two features (price, CPU speed, etc.) of each of three computers in a socially smooth conversation with a customer.

Participants were given a set of behaviors and functions, presented in Table I along with their C API equivalents for the *no-IC* condition. A list of available gestures was also provided. In both conditions, gestures were added by placing markup tags in the text to be spoken, as in the following example.

```
This PC has <gesture type="emphasis"> six
hours </gesture> of battery life.
```

1) Programming Task

The programming task focused on the processing of sensor data, which is a common task for robotics programmers. We provided participants with a face detection application for identifying whether a customer was present and where they were standing. As real-world data is noisy, the programmer's first task was to create a simple filter to remove false face detections based on features such as height and width.

The second task was to compute the customer's position in space, based on the face detection data. This would enable the robot to turn its head towards the customer while interacting.

TABLE I. BEHAVIORS AND VARIABLES

Behavior	C Function	Description
Talk	void talk(string text);	Speak and/or perform gestures.
Ask	void ask(string text, int time, string expectedResponses);	Speak, then listen for a spoken response within a given time limit.
LookForFace	int lookForFace(); Return 0 for left, 1 for center, 2 for right, 3 for none.	Look for a face to the left, center, and right of the robot.

Variable	C Function	Description
faceDetected	int isFaceDetected(); Return 1 if face detected, 0 if none.	True if a face is currently visible
listenWord	int isSpeechResult(string result); Return 1 if the speech result was equal to *result*, or 0 otherwise.	Most recent speech recognition result

D. Fairness of conditions

For this experiment it was essential to provide exactly the same capabilities in both the C interface and the Interaction Composer (IC) interface. To make the interfaces as equivalent as possible, we created a single C function corresponding to each behavior template available in IC, as shown in Table I.

For example, Fig. 7 shows a simple flow in IC. The same flow could be built using our C interface as follows:

```
while (lookForFace()==3) {}
talk("Welcome to my computer shop!");
```

C equivalents of the conditional, sequential, and random decision blocks were not provided, as this functionality is easily available in C, using `if` statements, `for` loops, and the `rand()` function.

The equivalent of sequences is also trivial to implement in C, simply by defining a function, and interrupts were not used for either condition in this experiment. Thus, all functionality available in IC was also easily usable in the C interface.

E. Participants

32 pairs of participants (49 male, 15 female, average age 24.8 years) took part in this experiment. Designers were required to have no computer programming experience, and programmers were required to have basic proficiency in the C language. Programmers were also given an entry-level C programming test before the experiment, and their scores were used to choose the condition for their trial. This enabled us to balance the skill levels of the programmers between conditions.

F. Procedure

After 2 hours of instruction, 3.5 hours were given for developing the robot application. Each hour, we evaluated the progress of the application using a checklist of 17 requirements, and we gave the participants feedback about missing features or serious problems.

The requirements checklist was independent of the experimental condition, and was strictly an evaluation of the robot's outward behavior, not the underlying implementation. Examples include the following:

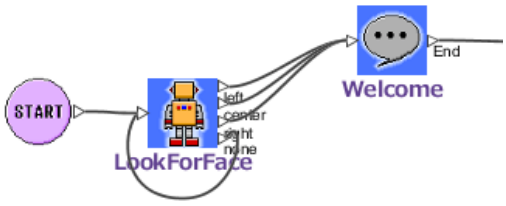

Figure 7. Example flow using IC.

- Greet the customer when they arrive.
- Introduce at least two features of each product.
- Explain only features requested by the customer.
- Show variety in utterances when they are repeated.
- Say goodbye only when the customer has left.

G. Evaluation

We evaluated the overall quality of the completed applications and performed secondary evaluations of the individual subtasks.

1) Primary Evaluations

We first measured the overall quality of the completed applications using the requirements checklist, for a score from 0 to 17, where 9 points represents completion of all basic tasks.

We also conducted an interactive evaluation, in which two evaluators, blind to the experimental conditions, spent 10 minutes interacting with the robot for each application and gave subjective quality ratings on a 100-point scale. These evaluators considered things like the appropriateness of utterances, naturalness of gestures, and how the robot made them feel as a customer.

2) Secondary Evaluations

We measured performance on the programming tasks by testing the accuracy of the face detection filter and noting whether the second programming task had been completed, as many teams skipped this optional task due to time pressure.

To quantitatively measure the complexity of the interaction design, we counted the number of unique utterances used in each interaction flow, expecting that interactions of higher quality will display a greater variety of utterances.

V. RESULTS

Results of primary evaluations regarding the overall performance of the robot application are shown in Fig. 8, and results from secondary evaluations are shown in Fig. 9.

A. Overall achievement

For the interactive evaluation, we averaged the ratings between the two evaluators. A one-way ANOVA conducted for the interactive evaluation results revealed a significant main effect ($F(1,30)=20.659$, $p<.001$, partial $\eta^2=.408$). A one-way ANOVA conducted for the checklist scores also revealed a significant main effect ($F(1,30)=9.905$, $p=.004$, partial $\eta^2=.248$). Applications in the *with-IC* condition significantly outperformed those in the *no-IC* condition in both evaluations.

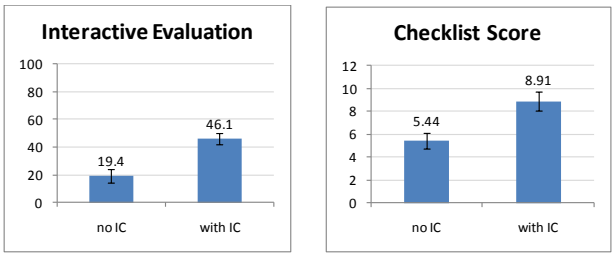

Figure 8. Results for primary evaluations.

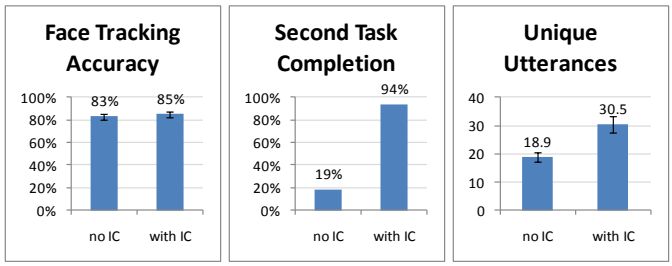

Figure 9. Results for secondary evaluations.

B. *Secondary evaluations*

Face tracking accuracy was similar between the two conditions. A one-way ANOVA conducted for face-tracking accuracy revealed no significant effect ($F(1,30)=.299$, $p=.589$, partial $\eta^2=.010$). As we intentionally balanced the ability levels of programmers between conditions, is unsurprising that we did not see a significant difference in this task. In the *no-IC* condition, programmers typically gave this task priority and completed it before working on the interaction flow. For Task 2 completion, however, a chi-square test revealed a significant difference between conditions ($\chi^2(1) = 18.286$, $p<.001$, $\varphi=.758$).

Only 19% of programmers in the *no-IC* condition completed the second task, compared with 94% in the *with-IC* condition. Programmers in the *no-IC* condition were usually too busy building the interaction flow to work on this lower-priority programming task.

Finally, a one-way ANOVA was conducted for the number of unique utterances, revealing a significant main effect ($F(1,30)=12.760$, $p=.001$, partial $\eta^2=.298$). These results showed significantly more utterances in the *with-IC* condition, indicating that the increased involvement of the designer enabled the creation of more complex interaction flows.

VI. Discussion and Conclusions

A. *Observations*

During our experiment, different teams used our design framework with varying degrees of success. One thing we observed is that a clear understanding of the interface between the programming side and the interaction design is critical for productive collaboration. Some designers using IC misunderstood the functionality of the robot's behaviors, for example, confusing the *LookForFace* behavior and the *faceDetected* variable. These designers built flows with redundant or incorrect use of behaviors.

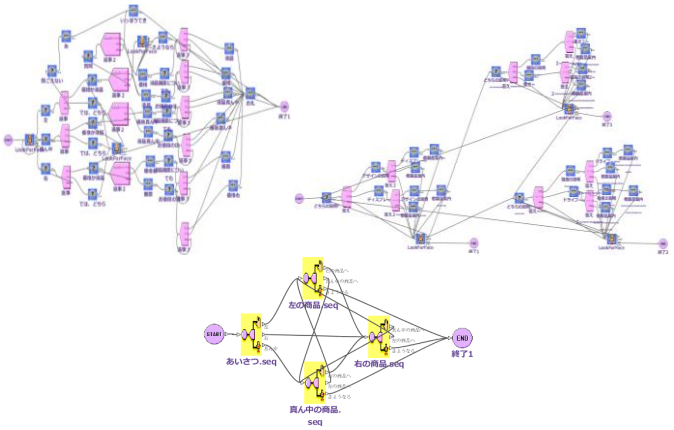

Figure 10. Examples of flow organization.

All teams in the *with-IC* condition used sequences, but some were more effective than others. Many designers built nearly the entire flow within a single sequence, *e.g.* Fig. 10 (upper left). Others took advantage of the graphical freedom in IC to arrange the elements into visual groups, *e.g.* Fig. 10 (upper right). More than half of the teams organized their flows by using top-level sequences, *e.g.* Fig. 10 (bottom). The most successful teams used sequences extensively (the best used 11, while most used 5 or fewer) to encapsulate tasks like asking confirmation questions or confirming the customer's presence.

In some of the more successful *no-IC* cases, programmers took strong initiative in the interaction design, using the designer's flow as a rough guideline since the designer did not have a clear understanding of what was difficult or easy to implement. The best programmers were able to generate logical flows equivalent to average-performance flows in the *with-IC* condition, but their flows lacked advanced features like checks to see if the customer had moved, variation in utterances, and small talk. These features were present in many *with-IC* flows.

Many unsuccessful *no-IC* cases failed because of mistakes in software design. As text is a linear medium, it can be hard to see the structure of an interaction flow just by looking at source code. Without a visualization of the structure, many programmers forgot to handle important contingencies, such as the case when no speech recognition result is received.

B. *Scalability*

Scalability is a concern for any programming environment, and while our results show that the state-based approach we use for dialog management is useful for simple flows, its scalability and flexibility remain important questions.

We have found this approach useful for long, mostly-linear flows, and if the robot guides the interaction by asking questions, people's responses are usually predictable. We have also found it to be effective for interactions where the robot needs to make simple responses to a large number of keywords, *e.g.* providing directions to one of 90 places in a shopping mall.

The state-based approach is not as effective when the robot needs to remember interaction history, *e.g.* offering to explain only information it has not already presented. In these situations the complexity of the flow rises exponentially with the amount of state that needs to be remembered. Such

situations are fairly common in social interactions, so for some applications we have developed custom behavior modules to handle interaction history.

So far, the current balance between functionality and ease-of-use has been sufficient for our applications. No doubt this balance will change in the future as applications become more complex. However, the principle of enabling designers and programmers to collaborate through parallel development will only become more important as complexity increases.

C. Conclusions

In this paper we have presented a novel interaction design framework which enables non-programmers and programmers to work in parallel to develop interactive applications for social robots. In our experiment we have validated that this new design approach increases efficiency and application quality.

Structuring the development process to reflect the unique roles of designers and programmers should help to increase efficiency and enable both programmers and designers to produce higher quality work, making this a first step towards a scalable development process that will eventually be applicable to commercial social robotics applications.

REFERENCES

[1] M. L. Walters, *et al.*. "Human approach distances to a mechanical-looking robot with different robot voice styles," *Proc. 17th IEEE International Workshop on Robot and Human Interactive Communication (ROMAN)*, pp. 707-712, 2008.

[2] K. Dautenhahn, *et al.*, How may I serve you?: a robot companion approaching a seated person in a helping context, *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, 2006.

[3] B. Mutlu, T. Shiwa, T. Kanda, H. Ishiguro, and N. Hagita, "Footing in human-robot conversations: how robots might shape participant roles using gaze cues," *4th ACM/IEEE International Conference on Human-Robot Interaction*, pp.61-68, 2009.

[4] J. Kramer and M. Scheutz, "Development environments for autonomous mobile robots: a survey, autonomous robots," Autonomous Robots, vol. 22(2), pp. 101-132, 2007.

[5] E. Pot, J. Monceaux, R. Gelin, and B. Maisonnier, "Choreographe: a graphical tool for humanoid robot programming," in *Proc. 18th IEEE Intl. Symposium on Robot and Human Interactive Communication*, pp. 46-51, Toyama, Japan, Sept. 27-Oct. 2, 2009

[6] B. Erwin, M. Cyr, and C. Rogers, "LEGO engineer and ROBOLAB: Teaching engineering with LabVIEW from kindergarten to graduate school." *Intl. Journal of Engineering Education*, 16(3), pp. 1-12, 2000.

[7] A. van Breemen, "Scripting technology and dynamic script generation for personal robot platform," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3561-3566, 2005.

[8] M. Nakano, *et al.*, "A two-layer model for behavior and dialogue planning in conversational service robots," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1542-1548, 2005.

[9] M. F. McTear, "Modelling spoken dialogues with state transition diagrams: experiences with the CSLU toolkit," in *Proc. 5th International Conference on Spoken Language Processing (ICSLP'98)*, pp. 1223-1226, Sydney, Australia, 1998.

[10] M. F. McTear, "Spoken dialogue technology: enabling the conversational user interface," *ACM Computing Surveys*, vol. 34 (1), pp.90-169, March 2002.

[11] M. Dragone, T. Holz, B.R. Duffy, G.M.P. O'Hare, "Social Situated Agents in Virtual, Real and Mixed Reality Environments," *Int. Conf. on Intelligent Virtual Agents*, 2005.

[12] H. Asoh, *et al.* "A spoken dialogue system for a mobile office robot," *6th European Conference on Speech Communication and Technology*, pp. 1139-1142, 1999.

[13] T. Matsui, *et al.*, "Integrated natural spoken dialogue system of Jijo-2 mobile robot for office services," *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pp.621-627, 1999.

[14] F. Doshi, N. Roy, "Efficient model learning for dialog management," *ACM/IEEE Intl. Conf. on Human-Robot Interaction*, pp.65-72, 2007.

[15] N. Roy, J. Pineau, and S. Thrun, "Spoken dialogue management using probabilistic reasoning," *38th Annual Meeting of the Association for Computation Linguistics*, 2000.

[16] J. G. Trafton, M. D. Bugajska, B. R. Fransen, and R. M. Ratwani, "Integrating vision and audition within a cognitive architecture to track conversations," *3rd ACM/IEEE Annual Conference on Human-Robot Interaction*, pp.201-208, 2008.

[17] M. Scheutz, P. Schermerhorn, and J. Kramer, "The utility of affect expression in natural language interactions in joint human-robot tasks," *1st ACM/IEEE Annual Conference on Human-Robot Interaction*, pp. 226-233, 2006.

[18] M. Kleinehagenbrock, J. Fritsch, and G. Sagerer, "Supporting Advanced Interaction Capabilities on a Mobile Robot with a Flexible Control System," *IEEE/RSJ Int'l Conference on Intelligent Robots and Systems*, pp.3649-3655, 2004.

[19] S. Li, B. Wrede, and G. Sagerer, "A dialog system for comparative user studies on robot verbal behavior," *IEEE Int. Symposium on Robot and Human Interactive Communication (RoMan 2006)*, pp.129-134, 2006.

[20] A. Clodic, R. Alami, V. Montreuil, S. Li, B. Wrede, and A. Swadzba, "A study of interaction between dialogue and decision for human-robot collaborative task achievement," *16th IEEE International Symposium on Robot and Human interactive Communication*, pp. 913-918, 2007.

[21] O. Lemon, A. Bracy, A. Gruenstein, and S. Peters, "The WITAS multi-modal dialogue system I," *European Conf. on Speech Communication and Technology*, pp. 1559–1562, Aalborg, Denmark, 2001.

[22] J. Peltason, B. Wrede, "Modeling human-robot interaction based on generic interaction patterns," *AAAI Fall Symposium: Dialog with Robots*. Arlington, VA, USA: AAAI Press; 2010.

[23] M. Denecke, "Rapid prototyping for spoken dialogue systems," *Proc. of the 19th international conference on Computational linguistics*, p.1-7, August 24-September 01, 2002, Taipei, Taiwan.

[24] B. Scassellati, "Investigating models of social development using a humanoid robot," Biorobotics, MIT Press, 2000.

[25] T. Kanda, H. Ishiguro, M. Imai, and T. Ono, "Development and evaluation of interactive humanoid robots," Proceedings of the IEEE, vol.92 (11), pp. 1839-1850, 2004.

[26] B. Mutlu, J.K. Hodgins, and J. Forlizzi, "A storytelling robot: modeling and evaluation of human-like gaze behavior," *IEEE-RAS International Conference on Humanoid Robots*, pp. 518-523, 2006.

[27] C. Breazeal, *et al.*, "Effects of nonverbal communication on efficiency and robustness in human-robot teamwork," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 708-713, 2005.

[28] C. L. Sidner, C.D. Kidd, C. Lee, and N. Lesh, "Where to look: a study of human-robot engagement," *Proc. of the 9th International Conference on Intelligent User Interfaces*, pp. 78-84, 2004.

[29] C.L. Sidner, *et al.*, "The effect of head-nod recognition in human-robot conversation," *1st ACM/IEEE Annual Conference on Human-Robot Interaction*, pp. 290-296, 2006.

[30] Y. I. Nakano, G. Reinstein, T. Stocky, and J. Cassell, "Towards a model of face-to-face grounding," *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pp. 553-561, 2003.

[31] C. Shi *et al.*, "Easy use of communicative behaviors in social robots," *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, 2010.

[32] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation*, vol. 2(1), pp. 14-23, 1986.

[33] S. Satake, *et al.*, "How to approach humans? Strategies for social robots to initiate interaction," *4th ACM/IEEE International Conference on Human-Robot Interaction*, pp.109-116, 2009.

[34] http://www.youtube.com/watch?v=4Zi4NzzEfbQ

[35] T. Shimizu *et al.*, "Spontaneous dialogue speech recognition using cross-word context constrained word graph," *Proc. ICASSP*, pp.145-148, 1996.