This document is published in:

Robotics and Autonomous Systems (2011), 59(12), p. 1102–1114.

DOI: 10.1016/j.robot.2011.07.009

This work has been supported by the CAM Project S2009/DPI-1559/ROBOCITY2030 II, developed by the research team RoboticsLab at the University Carlos III of Madrid.

End-User Programming of a Social Robot by Dialog

Javi F. Gorostiza, Miguel A. Salichs

RoboticsLab
Charles III University of Madrid
Systems Engineering and Automatics Department
Av. Universidad 30
28911 Leganes (Madrid)

Abstract

One of the main challenges faced by social robots is how to provide intuitive, natural and enjoyable usability for the end-user. In our ordinary environment, social robots could be important tools for education and entertainment (edutainment) in a variety of ways. This paper presents a Natural Programming System (NPS) that is geared to non-expert users. The main goal of such a system is to provide an enjoyable interactive platform for the users to build different programs within their social robot platform. The end-user can build a complex net of actions and conditions (a sequence) in a social robot via mixed-initiative dialogs and multimodal interaction. The system has been implemented and tested in Maggie, a real social robot with multiple skills, conceived as a general HRI researching platform. The robot's internal features (skills) have been implemented to be verbally accessible to the end-user, who can combine them into others more complex following a bottom-up model. The built sequence is internally implemented as a Sequence Function Chart (SFC), which allows parallel execution, modularity and re-use. A multimodal Dialog Manager System (DMS) takes charge of keeping the coherence of the interaction. This work is thought for bringing social robots closer to non-expert users, who can play the game of "teaching how to do things" with the robot.

Keywords: Sequence Function Charts, Petri Nets, Instruction-Based Learning, Natural Programming, Human-Robot Dialogs, Dialog Manager System, Semantic Grammars, Social Robotics

1. Introduction

This paper discusses a Natural Programming System (NPS): its design, implementation and first experimental results. Programming a social robot can be seen as an interesting application within the HRI, in which not only can the robot increase its capacity domain, but also and more importantly, the end-users enjoy and have fun while learning how to program it. In other words, building programs in a robot could be interesting for the user if the programming interface offers a natural and easy interaction platform.

On the other side, it is neither practical nor reasonable trying to put all the robot's capabilities in an initial fixed implementation, since the adaptation to the human environment clearly involves the enrichment of this initial set of skills. One possible way of performing this adaptation is by combining the initial skills into new ones. This new and dynamic set could include more complex capabilities, favoring the adaptation of the robot to the user's necessities.

In [1] a distinction is made between manual and automatic programming systems. The former implies a direct access to the system code and is typically performed by expert programmers. The latter is more appropriate for non-expert or nave users, since they can use an interface that makes the programming easier. The automatic programming systems are divide into three types: Learning Systems, Programming by Demonstration (PbD), and Instructive Systems. In social robotics, there are multiple researching works on the two former, but less work has focused on the latter.

A collaborative robot learning architecture has been discussed in many works such as [2] that shows studies in HRI with Leonardo robot; or in [3], where the game "hide and seek" is proposed as a scenario for robot

Email addresses: jgorosti@ing.uc3m.com (Javi F. Gorostiza), salichs@ing.uc3m.com (Miguel A. Salichs)
URL: http://roboticslab.uc3m.es/ (Javi F. Gorostiza), http://roboticslab.uc3m.es/ (Miguel A. Salichs)

learning. In [4] and [5] a Programming by Demonstration (PbD) system is shown. In [6] a cognitive model provides the robot not only with autonomous learning but also with autonomous reasoning about essential aspects in natural interaction. In [7] this system is implemented for a collaborative human-robot interaction. Besides, in [8] and [9] a dialog system allows the robot to increase its knowledge about the environment for navigation, interacting with the user in a natural way.

In this work, we present a novel Natural Programming System (NPS) as an example of Instructive System. We conceive naturalness in interaction as it is said in [10]: "[human] communication has become to be thought of as a relationship, an act of sharing, rather than something someone does to someone else". Therefore, in our interactive system users can naturally adapt themselves in similar ways they do with other humans.

The presented NPS works in combination with a frame-based Dialog Manager System (DMS) that is able to change the dialog context at run-time. The main goal of the dialog is to complete a finite set of information slots from the user's utterance. This set of slots, which defines the dialog context, can be changed online. An extension for multimodal dialogs has also been included.

The new skill built by the user is represented as a Sequence Function Chart (SFC) that makes it possible to build multiple programming structures, such as single sequences, sequence selection (selection mode), simultaneous sequences (parallel mode) and loops, and not just a serial sequence of actions, as in another similar works like [11], [12] or [13].

In general, learning from instruction should coexist with learning from other sources such as observation, imitation, demonstration, interaction with the environment, analogy, etc. There are several developments that focus on such methods, but only a few do on NPS. This paper focuses on the language used for the representation of the created sequence, on how to make the initial robot's capabilities verbally accessible to the end-user, and on how to give the verbal tools to create new skills in the robot by an efficient and natural enough dialog.

In [11] both autonomous and guided learning methods are used at once. A virtual SOAR agent is able to learn multiple kinds of knowledge through a combination of analytical and inductive techniques, combining top-down Instruction Based Learning system with a full suite of contextually-guided responses to incomplete explanations. By a situated explanation, the agent is able to achieve a general learning from specific cases. Our system makes up the new robot's capabilities in a bottom-up way, that is, from simple primitives to complex skills. In this way, the user could combine the initial capabilities into new ones. The system has been implemented and tested in a real social robot, and an interesting discussion about the problems and challenges found is given at the end of the paper.

In [14] the proposed system is based on Commonsense Reasoning (in particular, ASP) for Human-Robot Collaboration, mixing a Dialog System with a task planner, so the robot is able to extract information from the dialog with the user and transfer it into inner representation, what is very similar to the method used in the present work. They use 4 atomic actions that are combined in complex tasks. In our work we use more actions and also a set of conditions as nodes for the created sequence. We include interaction in the creation of such sequence and a deep relationship system between natural language and action language.

Programming a robot is not a new research issue. We can find one of the first programming systems of the type proposed here in [15], where the movements of the human hand are translated into a sequence of robot primitives. Similar works have been presented in [16], where a conversational-based programming system for the HRP-2 robot is shown. The robot is programmed for assembling a specific piece of furniture in a set, involving spoken interaction and visual perception. In [17] a robot is also taught for a predetermined task. The system can use a task model that is examined for determining the missing information in the model, and asks for it to the user, which is very similar to the frame-based dialog system proposed here.

In [12] a vacuum robot is programmed for a room cleaning task. Verbal commands and gestures are translated into a sequence of cleaning actions. The system allows the user to indicate with hand movements the path that the robot has to follow during the cleaning task. But the created sequence is very simple and no conditions are allowed.

In all these works, the user programs the robot for a specific task. Our approach tries to involve more general scenarios; therefore, the task is not fixed *a priori*, but chosen by the end-user. The challenge we want to face up to is to make all the robot's skills accessible to the end-users so that they can have fun using and combining them into new skills, while at the same time, they increase the robot's capabilities in an easier way by natural interaction with it.

The robot can guide the user in this frame by showing the possibilities in each interaction context and by precise queries about the necessary information, maintaining the coherence of the interaction process.

The organization of the paper is as follows. Section 2

relates to the robotic platform where the NPS has been implemented and tested. In section 3 we describe, the Sequence Verbal Edition (SVE) system as a whole, and mentioning its different parts: the HRI Skills, the Dialog Manager System (DMS) , the Sequence Generation System (SGS) and the Sequencer. These parts are explained in next sections. In section 4 we describe the HRI skills involved in the edition of the sequence. In section 5 we describe the Dialog Manager System (DMS). In section 6 we describe how the sequence is created by HRI. In section 7 we give a first results of the NPS, main part of the paper. We discuss about more general results we have found in section 8. Finally we have the conclusions in section 9 and future works in 10.

2. The Platform: Maggie and Sequences

This section describes the platform used for the implementation of the NPS. Maggie's hybrid control architecture works be means of skills that can be sequenced. Next sections describe the procedure that allows this control mode.

2.1. The Control Architecture

The control architecture where the developed NPS is framed in is called Automatic-Deliberative Architecture [18], [19]. This architecture is based on modular executing units called *skills*. Actions and conditions, which are the nodes that make up the sequences, are implemented by these skills.

In the AD architecture there are two communication methods between skills: the Event System (ES) and the Short Term Memory (STM) system. The former works according to the *publisher-subscriber* paradigm. Therefore, at any instant one skill can send any punctual event to its subscribers. The STM is like a shared *blackboard* where any skill writes or reads any type of data.

2.2. The Sequence

The sequences have been usually used in robotics as representation of plans or any task that could be divided into sub-sequences of smaller elements, such as sub-tasks, commands, actions, etc. For its relative simplicity and its versatility, our approach uses a sequence representation language based on a Discrete Event System standard: the Sequence Function Chart (SFC) [20]. The sequence is alterable, that is, it could be modified at run-time, since it is implemented as an `XML` structure that also includes `Python` scripts.

There are two types of nodes that are connected in alternate order in a sequence: *steps* and *transitions*. The former is associated with the actions that the robot can do. Each step can be activated or deactivated. The transitions are associated with a condition made from variables of the environment or the own robot. These transitions control the activation and deactivation of the actions to which they are connected.

2.3. Maggie: the Social Robot

Maggie is the social robot where all the systems described here have been implemented and tested. In our project, it is important that the end-user has fun "playing" to building a program with and for Maggie. By her physical looking, Maggie is friendly, attractive and fun. She is an anthropomorphic shape robot of 1,40 meters tall that takes elements of cartoons characters and cute animals (Fig. 1).

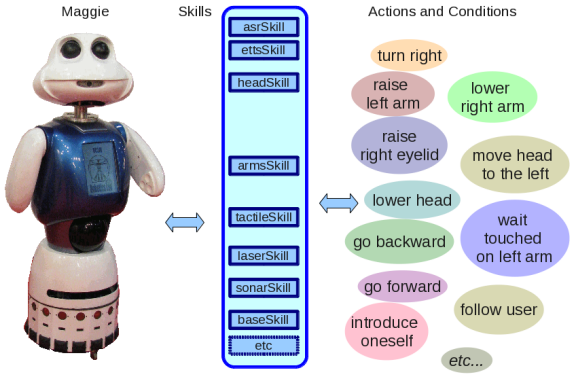


Figure 1: Maggie's capabilities are implemented in skills. Two sets of actions and conditions are used to edit a sequence. One skill could perform more than one action/condition, and *vice versa*

In the hardware level, Maggie has sensors for the different interaction modes: tactile, telemeter, microphone, camera, RFID sensor, etc; and multiple actuators for verbal and non-verbal expression: Maggie can move the head, the eyelids, the arms, move or rotate, speak, etc.

In the control level, Maggie has several skills for interaction, such as automatic speech recognition, emotional text-to-speech synthesis, person following, face detection, teleoperation by voice, user identification, etc. She also has skills that gather internet information and allow her to give the news or the weather report, email handling, etc. Other skills include the control of infrared devices, such us the television or the radio.

The skills can be as simple as for example a "control-arm-skill", that just controls the arm's movement, or as complex as a "follow-person-skill", that makes use of

simpler skills: the one that controls the laser telemeter, the one that controls the base's movements ("control-base-skill"), etc. The set of all these robot's skills is in constant growing. More details about Maggie's features and some of her capabilities can be found in [19], [21].

2.4. Skills and Sequence Elements

One action/condition can involve one or more skills, and one skill can implement one or more actions/conditions. For instance, the action "greeting" involves the skills "control-arm-skill", "control-head-skill" and "etts-skill" (emotional text-to-speech synthesis). On the other side, the skill that controls the arm (control-arm-skill), for instance, implements actions such as moving the arm up or down and conditions related to its position, knowing if the arm is completely up or down. There are also skills that implement just one action, for example the "follow-person-skill" commented above.

In the programming level, the actions and conditions are implemented using the methods that the Application Program Interface (API) of the robot offers. This API covers from low-level functions directly related to the hardware (simple skills) to high-level functions related to complex skills. The skills are also implemented using this API.

One important feature of the presented system is that the created sequence can also be seen as a new skill, and therefore could be reused to subsequent new sequences (new skills).

2.4.1. Actions

The actions are implemented in the steps of a sequence. Each step can have two different activation states: *activated* or *deactivated*. When the step is activated, it executes its action. In our system, this execution is implemented as a `Python` function, that is built at execution-time. This function combines the `Python` methods implemented in the API. This construction is made following the instructions of the end-user that the robot perceives from the results of the human-robot conversation. Actions are active until they finish or another action deactivates it. For instance, "moving the head up" finishes when the head is in a particular position that is interpreted as "being up", but "following a person" finishes with another action that is "finish to follow a person".

Some examples of actions can be: "move up/down the left/right arm", "move up/down or to the right/left the head", "go forward", "go backward", "turn left-/right", "activate person-following", etc.

2.4.2. Conditions

A condition is a logic proposition that makes an evaluation as a function of the variables of the sequence itself, the robot's features and/or the environment. In our system, each condition is implemented as a `Python` function that performs the evaluation of the condition. This function is executed if the transition is enabled, that is, if all the preceding steps are active. If the condition result is true, then the trigger is performed; otherwise, no evolution movement is made.

Most evaluation functions are associated with the ending of the preceding action. For instance, the evaluation function of the action "raise-left-arm" is that the arm is already in the up pest position.

Some of the used conditions are: "if head/shoulder/arm is touched", "when the arm is in the top", etc.

2.5. Innate primitives

The sets of actions and conditions are directly related to the sets of skills of the robot (see Fig. 1). As described above, Maggie's skills are multiple, but for the present work and as a starting point, a reduced initial set of actions and conditions has been chosen.

In other natural programming systems, the action primitives are designed one by one from a set of user's utterance [13]; therefore, the primitives are well adapted to the user's speech. These methods have the advantage of easily covering a great range of the user utterance if the application domain is well constrained. Nevertheless, these design methods for the "innate set" do not assure that the primitives are going to cover all the action possibilities of the robot. In our approach, it was very important to cover all such possibilities, and Maggie has many of them. Therefore, the innate primitive set is closer to the low-level robot action-perception space. In the speech or symbolic level, the semantic-CFG rules perform the necessary transformation from the user's utterance, which is natural, into the action-perception robot space, which follows a formal model. The DMS is in charge of guiding the users and translating their natural language into an internal representation in terms of these primitives.

In summary, as an initial set of actions, we have chosen the basic movements of the robot's DOF (head, eyelids, arms and body), and some skills as "following a person" and "make an introduction"; and as an initial set of explicit conditions we have used the touch-skill. Therefore we have tested the NPS with 21 different actions and 7 different explicit conditions. Nevertheless, these sets can be easily increased *ab libitum*.

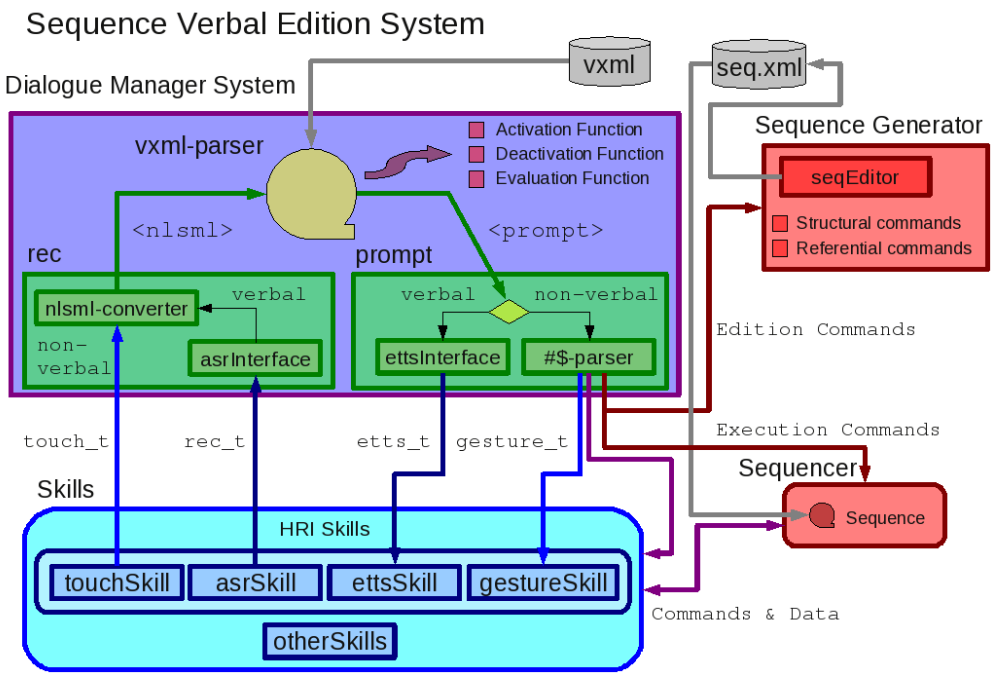


Figure 2: Sequence Verbal Edition System including the Dialogue Manager System. The sequence is built on-line by human-robot dialogs.

3. Sequence Verbal Edition System

The Sequence Verbal Edition (SVE) system provides the connection between the external user's utterance and the internal robot's skills, making these skills accessible by voice for the user. This system allows the user to build new skills (new sequences) from primitive skills, and reuse the created skills for subsequent sequences.

Fig. 2 shows a general diagram of the whole implemented system, that is divided into the following main parts:

- **HRI Skills**, that allows the robot to perceive and express verbal and non-verbal information used for HRI.
- **Dialogue Manager System** (DMS), that communicates the HRI expressive and perceptive skills, and performs the main control of the whole system.
- **Sequence Generator System** (SGS), that creates the sequence in an XML structure, from the appropriate commands from the DMS.
- **Sequencer**, that parses the sequence and executes, pauses or stops it at run-time. It is also controlled by the DMS.

The DMS communicates with the SGS, that takes charge of building the new sequence by two types of commands: referential and structural ones, which will be discussed in section 6.1.

The Sequencer reads and parses the new created sequence, and takes charge of executing, pausing and stopping it. The execution of the sequence involves sending and receiving commands to/from the skills of the architecture, the ES and the STM system.

The dialog is implemented following the standard `voiceXML`[1] with some particular extensions. Dialog is implemented in a set of structures that are parsed at run-time by the DMS. These structures define the *context* of interaction, that is: *what* the robot is going to say, *when* and *how* is going to say it, and *what* the robot is going to be able to detect from user's communicative acts. In the implementation of the dialog, it is also defined how to interpret this obtained information.

In our NPS, such interpretation defines how to create the sequence. Nevertheless, the DMS is an open system that is also used for other purposes. For instance, the DMS have been included as a part of some robot's skills, such as the one that controls the television via an infrared transmitter. By this skill, user and robot dialog

[1] A description of the W3C standard for voiceXML-2.0 could be consulted here: http://www.w3.org/TR/voicexml20/

in a specific interaction context where the user indicates by voice what command he/she wants to order to control the TV: turn it on/off, turn volume up/down, change the program, etc. In this way, each robot's skill that need to interact with the user can implement an specific vxml structure, so the DMS can change from one context to another dynamically.

4. HRI Skills

The execution of the dialog involves the gathering of information from the perception skills (verbal and non-verbal) and the articulation of communicative acts via speech and gestures. In this section we briefly discuss about the skills that allows the robot and the user to interact each other. It is out of the scope of this work to get deep on HRI issues, and these skills here are used as a means of the NPS, the real purpose of the paper.

Fig. 2 includes two speech skills: asrSkill and ettsSkill, and two non-verbal skills: touchSkill and gestureSkill. Maggie has more HRI skills, but here we show the ones that have been used for the SVE system.

The asrSkill is the result of a long research on Automatic Speech Recognition that has been published on [22]. This skill performs the **automatic speech recognition** and sends the robot a structure called `rec_t` that contains the recognition result, that is, the recognized words and the relevant verbal information. This procedure will be explained in section 4.1.

The **Text-to-Speech synthesis** is made by the ettsSkill, that receives the structure `etts_t`. This structure contains some control parameters for synchronization between other skills that want to use the speech synthesis. It also contains the text to be synthesized and some expression parameters. The synthesis process controls some prosodic features such as the main pitch, volume and velocity, and some sound features such as equalization or effects (reverb, chorus, etc). All this control allows the system to generate a speech that expresses very simple emotions, such as nervousness, happiness, sadness or calm.

In natural language, it is usual to find non-verbal sounds and fillers such as "emm", "amm", "aha", whistles, yearns, yawns, laugh, etc. The ettsSkill is also able to synthesize these sounds.

The capacitive sensors in Maggie are managed by the **touchSkill**, that sends an event with the states of all the sensors when one of these sensors changes its state. This event is perceived by the DMS, as will be explained in section 4.2.

Maggie is able to express herself both by verbal and **non-verbal communication acts**. The DMS can emit both verbal and non-verbal prompts. From the architecture point of view, there is no significant difference between sending a command to the ettsSkill and sending it to the gestureSkill. For instance, in a "greeting prompt" the robot is able to say "Hello" and make the appropriate gestures at the same time: gaze the user, move the arm, wink one eye, etc.

How the communicative act is performed in its different modes is defined in the dialog implementation like the system presented in [23], that is based on the Multimodal Presentation Markup Language (MPML) presented in [24], where the developer can easily add non-verbal tags inside the text that is going to be synthesized. For instance, when Maggie expresses an affirmative message she sends at the same time one utterance like *"Aha"* to the ettsSkill and the event "HEAD_AFFIRM" to the gestureSkill, that performs the assert gesture with the head. Therefore each expressive mode is decoupled in a different skill and it is the developer who perform the synchronization between modes in the vxml implementation.

4.1. Semantic Grammars and Speech Recognition

For the speech recognition and synthesis a commercial engine by *Loquendo*[2] is used, for both Spanish and English.

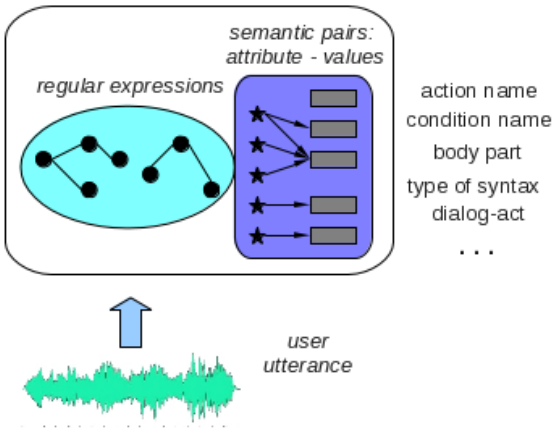


Figure 3: asrSkill works using CFG grammars and semantic assignation.

Fig. 3 shows how the speech recognition, `asrSkill`, works. It uses a Semantic Context-Free Grammar (CFG), that specifies two types of information:

[2] http://www.loquendo.com/es/

- **Regular expressions** that define the words and the syntactic rules that combine these words into a set of sentences called Regular Language. This language specifies the utterance that the robot is able to recognize. This part of the grammar is called *literal*.

- **Semantic attributes.** This is the *semantic* part of the grammar. We define a discrete set of attributes whose values are assigned depending on the recognized utterance from the user.

The literal part is a CFG that specifies the exact formal language that can be recognized. In our experiments, we have found that this language can be wide enough for a specific context, without significant misunderstandings or other errors. For instance, we have used about 100 CFG rules with about 300 different words and we have obtained more than a 95% of recognition accuracy when the user utters something inside the regular expression of the grammar.

The utterance that contain some elements out of the terms and/or syntactic rules described in the CFG have more probabilities of causing some type of misunderstanding. Although the dialog itself helps to coherence recovering (see section 5), the literal part of the grammar can also include one special variable (`$GARBAGE`) that allows the recognition of terms that are out of the defined context-free language.

The semantic part is in charge of taking the information that is relevant for the robot from the user's utterance. In our context, the semantic attributes are directly related to the sequence editing commands that will be explained later. Some of these semantic attributes have been shown in Fig. 3: `action-name` (move, turn, activate,...), `condition-name` (if-touched), `body-part` (head, eyelid, left-arm, base,...), `type-of-syntax` (verb, verb + direct object, verb + direct object + adverbial of place), `dialog-act` (co - conversational opening, ky - response acknowledgment or "yes" answer, sd - statement, etc), and so 8 different semantic attributes.

As an example of how the semantic grammars work, let's analyze this very simple grammar made by two rules and two semantic attributes:

```
$opening = "hello | how are you [doing]"
           {<@da "co">};

$closing = "([good] bye | see you)"
           {<@da "cc">};

$grammar = $opening | ($closing $GARBAGE);
           {<@type "greet">}
```

This grammar makes the recognition of greetings such as *"hello", "good bye", "bye", "bye Maggie"* possible. Two semantic attributes have been defined: `da` (dialog act) and `type`. The former can take two values: "*co*" (conventional opening) or "*cc*" (conventional closing). The attribute "`type`" takes the value "*greet*".

Notice that the use of the variable `$GARBAGE` after the "conversational closing" allows the recognition (but not identification) of any undefined term placed after the "*closing*" rule. For instance, utterance such as "bye Maggie", "good bye forever", "see you later" would lead to a positive recognition result.

Semantic attributes act as an information filter and allow us to get the exact information useful for the application (in the example, simply identify that there has been a greeting). Naturally, there is a lot of information that is not directly perceived by the semantic attributes. For example, prosodic and timing features of the user utterance, that would have to be detected by different ways, what is not an important limitation, as this information is not used by the NPS.

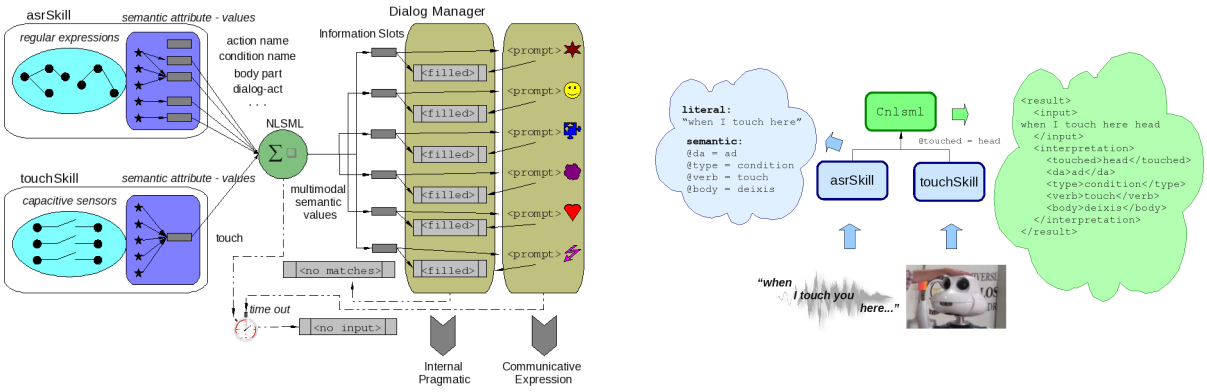
The grammar is loaded and parsed at run-time; therefore, it can be modified and changed without the necessity of stopping the speech recognition process. In other words, it is possible to modify and add more Regular Expressions, that is, more recognizable words and syntactic rules, and to modify and add more Semantic Attributes with their values. This will allow us to label new built skills dynamically, which could be verbally referenced in the future.

4.2. Multimodal Fusion and Semantic Grammars

The asr interface allows for robust speech recognition, even in very noisy environments. The etts interface is able to synthesize speech and paralinguistic features. But, as natural interaction is multimodal, we wanted to incorporate in the DMS the possibility of handling information from other modes, both in expression and perception. Multimodal interaction belongs to a large enough issue and it is not the central part of this paper, but we have developed a particular method of fusing together speech and tactile information that is briefly discussed in this section.

In the perception side, the information perceived from the sensory skills is transformed into a structure expressed in the Natural Language Semantic Markup Language (NLSML[3]). The multimodal fusion is made at variable periods where all the communicative information that has been perceived is represented by NLSML data and sent to the DMS.

[3] http://www.w3.org/TR/nl-spec/



(a) Multimodal Inputs in the Dialog System

(b) Multimodal fusion: example of a deixis resolution.

Figure 4: The DMS is able to handle information from different modes. For example, we show how touch and speech can be fused.

Fig. 4(a) shows how the information from the asrSkill (speech recognition) and the touchSkill (touch events perception) is fused into a single NLSML structure. In short, the semantic attributes could come from any perceptive skill, and not just from the asrSkill; therefore, the semantic attributes can take their values from both verbal and non-verbal modes. In the figure, some semantic attributes for sequence edition have been shown: "action-name", "condition-name", "body-part", etc. All of these attributes, and their corresponding assigned values, are merged and sent to the DMS to fill the information slots.

Fig. 4(b) shows an example of multimodal fusion of information from these two perceptive skills: asrSkill and touchSkill. The speech recognition returns the recognized literal text and a set of semantic attribute values, what is fused with the semantic attribute value related to the touch event. Later, the DMS can disambiguate the attribute value $@body = deixis$ with the attribute $@touched = head$, since "deixis" is the cue value that indicates the mandatory disambiguation. In Fig 2 and Fig 4(a) just the touchSkill has been included, but any other multimodal perception skill could be added in an analogous way.

The main advantage in representing all the multimodal information in one single standard language is that we make opaque enough the origin of such information to the DMS, that takes charge of controlling the interaction. Therefore, if the DMS is waiting for an information-slot regarding with a part of the robot body, for the DMS point of view it would be the same to mention this part by speech, "the right arm", or directly to touch the right arm.

5. Frame-Based Dialog Manager System and References to Sequence Nodes

In [25] a wide range of different dialog systems is given according to the task complexity, from the simplest systems, such as Finite-State scripts, to the most complex ones, such as Agent-based models. Following the nomenclature presented, the technique used for our DMS is frame-based. However, the DMS also allows us to easily change the context or shift between different dialog topics, where each topic defines a different finite set of information slots, that is, a different frame.

The dialogs used here are mixed-initiative, which allows the robot to perform a communicative act when necessary. For example, to manage misunderstandings, coherence lacks or if a particular information is missing, the robot can take the initiative for re-prompting, asking for confirmations, giving further information, etc. The robot is also able to manage some temporal aspects of the dialog such as interruptions (barge-in property), turn changes, prosodic control of the speech, user silences, etc.

There are three main types of elements in the vxml structures that define the dialog:

- **Information slots.** Specific internal variables that are filled from the user's communicative acts.
- **Communicative acts for prompts**, which are used by the robot for verbal and non-verbal expression.
- **Temporal parameters** that handle some important dynamic aspects of the dialog: turn organization, silence time-out and barge-in property.

The first two relate to the content of the dialog messages and will be described in section 5.2. The third

type refers to the temporal aspects and dynamic the interactive process involving every natural dialog.

5.1. Control of the Temporal Aspects of the Dialog

Social robots have to be able to manage not only structural, semantic or static aspects of the user's communicative acts, but also temporal aspects of interaction such as synchronization, pace, exchange of turns, handling of silence moments, etc. In our implementation, some of these aspects have been taken into account. For example, when it is the robot's turn and the user starts talking, which is called a barge-in, the robot transfers the turn to the user and stops the ongoing utterance. When the turn is given to the user, e.g., after the robot performs a query, there is a critical time out, in which the robot waits for the user to speak. After this period, if the user remains in silence, the DMS throws a `<no input>` event (see the time out parameter in Fig. 4(a)) so that the robot can re-take the turn and react to such a silence. Thus, the robot is always active, giving a greater impression of empathy and avoiding those moments when the user just expects some reaction or movement from the robot.

5.2. Control of the Spatial Aspects of the Dialog

By "Spatial Aspects of Dialog" we mean those aspects related to the content of the shared messages that are not directly related to the dynamics of the dialog as an interactive process. That is, we refer to the static information shared between participants.

In the perception side, the content of the information that the robot can perceive is defined in the ASR grammars, which are designed so that the semantic attributes directly coincide with the slots of information in the `voiceXML` implementation of the dialog.

In the expression side, the robot's queries, statements and confirmations are implemented as canned text and using templates.

For the dialog implementation all the necessary semantic attributes for the edition of the sequence are included: the ones associated with the details of an action or a condition and the ones associated with any edition command, structural or referential ones. Each edition command is associated with a specific discrete set of information slots. For instance, for adding an action, the system has to know that the edition command is of type "structural", that this command is "adding an action", the name of such an action, its parameters, and so on. The main goal of the dialog is to complete this set of information slots, making the appropriate queries by means of the HRI.

Other information slots and their associated semantic attributes have also been designed and implemented to handle other usual interactive scenarios, as will be discussed in section 6.3.

5.3. Operation of the Dialog System

The vxml structures that define the dialogue implementation are divided into *forms*. Each form, that represents a "concrete dialog", is also divided into a set of information slots, called *fields*. In the voiceXML-2.1 specification[4] the parsing from the asr recognized text is done by Context-Free-Grammars that are defined inside the vxml structure. Since our approach includes multimodal semantic attributes, such parsing is made outside the vxml structure, by the asrSkill when the inputs are verbal and by the NLSML-converter (see Fig. 4(b)), that is in charge of gathering all semantic attributes for the dialogSkill.

Therefore, the fields of the vxml structure are directly related to the multimodal semantic attributes: each field in the vxml structure has to correspond with one and only one semantic attribute in the nlsml structure. Thus, when the Automatic Speech Recognition Skill (asrSkill) or another perception multimodal skill performs an assignation of the perceived values to one or more semantic attributes, the corresponding fields, in the dialog structure, are filled with such values.

As an example, let's consider the utterance "Raise the right arm". The verb "raise" is "translated" by the asr semantic grammar into two semantic values: `verb = move` and `limit = Top`[5]. Each semantic value is also assigned to the corresponding field (`verb` and `limit`) in the dialog implementation. The system builds the activation function of the action by combining the semantic values in terms of the robot's API. For this example, the Python object used is `rightArm`, as follows:

Activation Function:= `rightArm.move(ArmTop)`

And so for each of the methods of the classes in the robot's API.

The dialog flux is controlled by the Form Interpretation Algorithm (FIA), that is in charge of visiting each field and checking if it is filled (see Fig. 5). Each field has two main parts. One is executed when the visited field is not filled yet. The text for querying about the

[4] See http://www.w3.org/TR/voicexml21/

[5] In this example "Top" value refers to the position the arm is moved to and it is fixed a priori for simplicity. However, it could also be considered as another possible parameter of the action. This could be easily implemented adding a new rule in the semantic grammar, in the asrSkill.

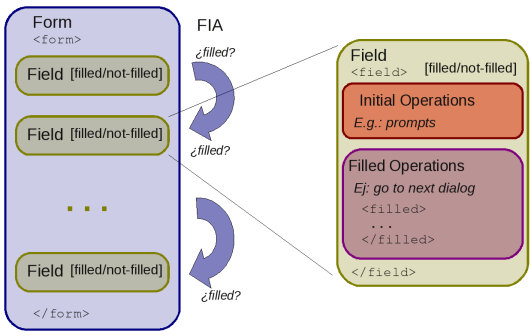


Figure 5: voiceXML structure and Form Interpretation Algorithm

concrete field should be placed in this part. In Fig. 4(a) this part has been represented as a set of "prompts" (communicative expression). The other main part of the field is executed when the field is filled. In the filled part, is where any assignation, dialog movement or action related to the field value is performed (this has been represented as "internal pragmatic" actions in Fig. 4(a)).

The dialog flux can also change the context, that is, the frame of the dialog. For instance, the whole dialog can go from a greeting-exchanges scene to a sequence edition context, to a goodbye-scene, etc. Since each context or "local dialog" has its own information slots (fields), it can also access to global slots (variables) completed from other dialogs.

5.4. Associating Actions and Conditions with Communication Acts

In this section it is described how our system relates the relevant information obtained by user's communication acts and the elements of the domain of actions and conditions of the robot.

How to automatically associate multimodal inputs with robot actions has been treated in several works. In [26] it is proposed a multimodal fusion system based on mirror neurons theory (MIRA). The system associates actions with the appropriate body part and then associates the word form of user utterance with an action.

Here we use a particular method where user utterance verbal inputs are partitioned by the semantic rules of the CFG in a way that they match a action-verb, and the necessary parameters.

Maggie's actions and conditions are represented as Python scripts in ASCii text. Each of this scripts or functions is made by an object, a method and some parameters. The object represents the part of Maggie that is involved in the action, for instance Maggie's left arm, her head, the speech capability, etc. The method represents the command that the object is going to execute, and it usually matches a verb. For instance, commands like "move" (for body parts) or "say" (for speech capacity). Finally, the parameters specify `where` and `how` to perform the action, for instance, moving the left arm to the top, moving two meters forward, saying "hello", etc.

This actions attributes are directly corresponded with semantic attributes. Each sensor skill (asr, touch, vision, etc) is in charge of giving semantic values to these attributes.

6. Interactive Sequence Edition

6.1. Semantics and Verbal Commands for the Edition of a Sequence

A sequence or program is essentially a graphical structure, therefore from the developer point of view a GUI will be the easiest tool for creating a graphical structure. An open question is if for a non-expert user it is easier to use a GUI environment or a verbal interaction system for creating a robot program. For instance, in [27] it is presented a graphical environment where the user can develop complex behaviors on a virtual Nao robot. However GUI usually have several limitations. For instance Choregrapher does not allow to program a real Nao at runtime, therefore there is not HRI at all. In this work we use verbal and non-verbal HRI skills for creating the robot behavior.

The main goal of this work implies to make the sequence verbally accessible. Therefore, we have studied how to design a system that translates verbal utterance into graphical information.

As shown in [28], humans spontaneously divide the instructions into small and basic units of actions during explanations. This is because the human short-term memory has a capacity of about seven (plus or minus two) elements, called "chunks" of information. We wanted to better know the steps humans use to describe the edition of a sequence by speech. Some users (around ten) were asked to verbally describe a SFC. As a result, we can conclude that humans utter two types of editing commands:

- **Referential commands**, that establish in what part of the sequence the structural commands are performed, e.g., establishing the focus in one specific node.
- **Structural commands**, that build the sequence itself, e.g., adding a node, opening a sequence selection, etc.

This important distinction could be compared with the graphical edition of any structure using a pointer in a

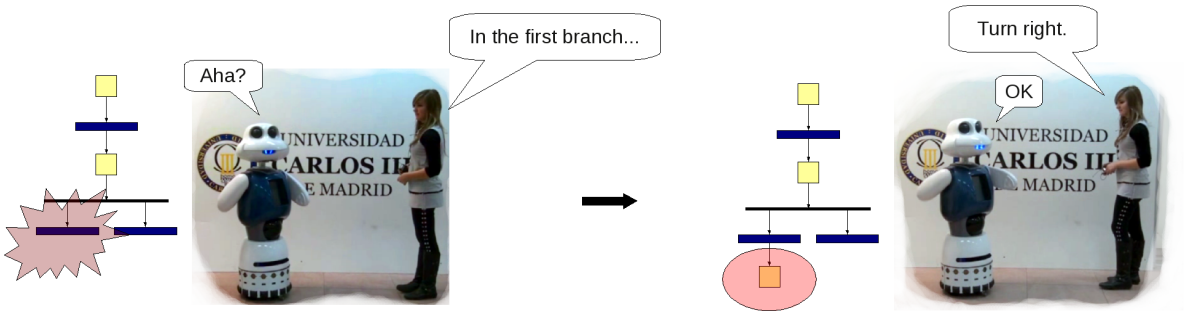


Figure 6: Before a structural command, the user specifies the focus in a specific part of the sequence (referential command).

screen. To make any modification in the graphic, first of all, users will point and select the part of the graphical structure they want to modify. Then, they will perform the specific modification.

Verbally, the users act in the same way. First of all, they establish the focus of attention in the nodes of the sequence that they want to edit and then, they perform the sequence modification. Sometimes the focus of attention is implicitly established by default or by assignation. These cases have also been taken into account in the present work. For instance, in Fig 6, it is shown two scenes where the user establishes the focus of edition before ordering an addition of an action.

6.1.1. Referential Commands

To establish the focus of attention for the sequence edition, the user can perform the following referential commands:

- **Reference to a node** that is in some part of the already created sequence, e.g., *"After raising the left arm..."*.
- **Reference to a branch,** which allows the system to distinguish between different branches inside the sequence, e.g., *"In the first branch..."*.
- **Reference to a structure of multiples branches.** This type of reference is used to conclude a part of the sequence made by some branches in parallel, such as a conditional selection or a parallel execution, e.g., *"Finish all the branches, and then, wait until I touch your left shoulder"*.

If the user does not specify a focus of attention explicitly, the SGS takes one by default. For instance, if the user adds an action, it is supposed that the next structural command is going to refer to that action; thus, the SGS establishes the focus of attention in that new added action by default.

6.1.2. Structural Commands

In the implemented system, the end-user can perform the following structural commands:

- **Addition of a node.** The node could be a step (action) or a transition (condition). For instance, the utterance *"Raise the left arm"* will add an action; the utterance *"then..."* or *"Wait until I touch your head"* will add a condition.
- **Removing a node.** All the nodes (steps or transitions) in the edition focus will be removed, e.g., *"Remove the last action"*.
- **Opening or closing a structure of multiple branches**, such as a sequence selection or simultaneous sequences, as explained above. For instance, *"Consider four possibilities"* will open a selection between four branches; *"Do three things, at once"* will open three simultaneous branches.
- **Loops**, that is, the union of one part of the sequence with a previous one, e.g., *"go back to the begining"*.

Notice that the addition of a node involves the definition of the content of such a node.

The study of the commented corpus shows that, although the formal language of the utterance involved in each of the structural commands can have multiple variations, these ones can be covered by the appropriate CFG definition. We also have to take into account that the dialog can disambiguate any unknown utterance, misunderstanding or lack of coherence.

6.2. Syntax Integrity of the Sequence

No any possible combination of structural commands is allowed. For instance, the user cannot add two nodes of the same type (two consecutive steps or two consecutive transitions), and it is not possible to begin a selection branch with a step instead of a transition. To

solve these integrity problems, the SGS takes into account the context of the edition process: what type of node is in the focus of attention, if the edition is part of some sequence in parallel, etc; and automatically adds the necessary nodes before the performance of the specific structural command.

Fortunately, the number of cases that can compromise the structure integrity of the SFC is limited. These are the cases that we have taken into account:

1. Beginning with a transition.
2. Adding one transition after another transition.
3. Adding one step after another step.
4. Beginning a selection branch with a step.
5. Beginning a parallel branch with a transition.
6. Appropriate convergence of parallel and selection branches.

For the first and the second cases, the system automatically adds a "trivial step" between transitions, which is an action that does nothing. For the third case, the system adds the appropriate transition, that is, the transition associated with the ending of the preceding step. The case number four has no sense, because one of the most essential things in a selection branch is its beginning condition. In this case, the robot will directly query what transition the user wants to add for the beginning of the specific conditional branch. For the case number five, another trivial step is added; and for the last case, depending on the context, trivial steps or trivial transitions are added in the appropriate branches so the convergence could follow the standard definition of the SFC.

The semantic integrity of the sequence is more difficult to achieve. In simple sequences (with no parallel branches), one possible solution is to associate some pre-conditions for all the actions. For instance, the precondition to move the arm down is that the arm is not down yet. Another example: it is not possible to move the same arm up and down at the same time. Furthermore, as the number of possible actions grows from the new sequences the end-user creates, it is not possible to consider all the individual semantic paradoxes of a sequence one by one. Therefore, it is necessary to develop a general formal model of the semantic of the robot's Action Dynamic Space, which is one of our works in progress currently.

6.3. Interaction Scenarios in the Edition of the Sequence

The sequence verbal edition is divided into several interaction scenarios or frames of context. The dialog moves between these scenarios depending on the user's communication acts and the dialog context.

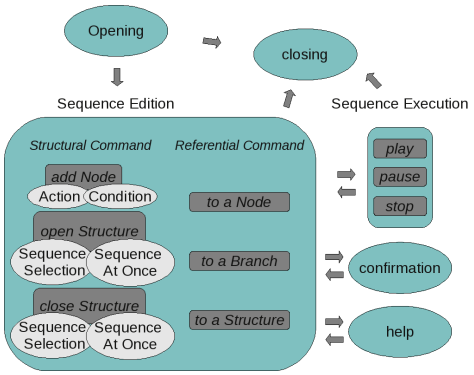


Figure 7: The dialog system changes the context in different frames.

Fig. 7 shows a simple diagram with the main of these scenarios:

1. Structural edition, as explained in 6.1.2.
2. Referential edition, as explained in 6.1.1.
3. Confirmations, e.g., if the recognition accuracy is low.
4. Information and clarification for contextual help.
5. Related to the communication process: conventional opening, conventional closing, misunderstandings, etc.

Each scenario has a certain independence from the rest, what makes it very easy to add as many scenarios as needed without changing the system significantly. The scenarios are implemented as voiceXML forms, and the information gathered in one scenario can be used in another one. For instance, if the robot perceives the user's name, later, in a confirmation scenario, the robot could call the user by his/her name.

Therefore, each scenario is associated with a set of information slots and each information slot is associated with a strategy of getting it from the user. The easiest of such a strategy is querying directly or indirectly.

Fig. 8 gives an example of dialog. It shows the relation between the user's utterance, the filled semantic attributes (and also slots of information) and the missing slots that are necessary to complete the edition command. In the example, the user wants to add the action *"Put your left arm down"* but does not specify with clarity what arm is involved in the action. The system automatically detects this lack, takes the initiative and asks for the specific slot of information.

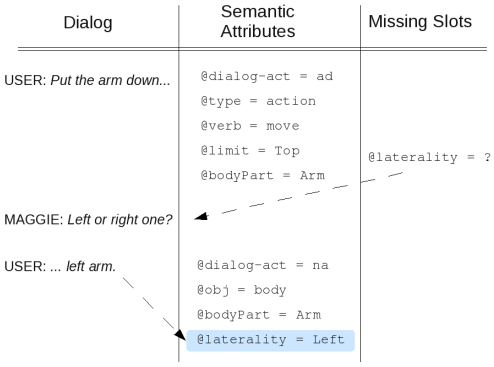


Figure 8: Example of dialog to add an action. Two dialog acts are involved: `"ad"`, action directive, and `"na"`, narrative/descriptive answer.

7. Experimental Examples

7.1. Dialog and Sequence Edition

This section shows a simple example of dialog between the robot and the user for the edition of a sequence. In the example, the user builds the sequence drawn in Fig. 9, which includes a sequence selection between two branches and a loop. The language used in the dialogs is Spanish, but here we have translated them in English.

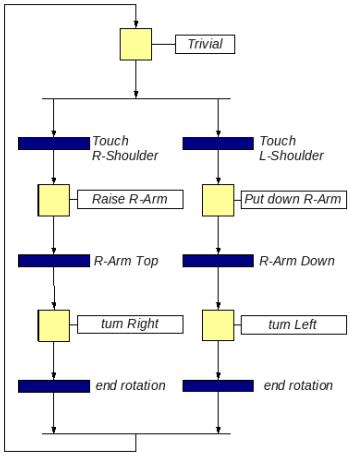


Figure 9: Example of a SFC that includes a selection of two branches.

First of all, the user and the robot maintain a dialog to know each other, with greetings, exchange of personal data, etc.

USER: Hello Maggie!
MAGGIE: Hello. How are you doing?
USER: My name is Javi.
MAGGIE: Nice to meet you, Javi.
MAGGIE: Now that we have met... let's have fun.
MAGGIE: For example, do you want to teach me a program?
USER: Yeah!
MAGGIE: OK!

The robot is able to handle different types of greetings, such as "hello" or "how are you?", in different ways. It also has the name of the user as an information slot; thus, it takes the initiative and performs the query for the user's name. Finally, the robot brings the conversation to the main topic of the system: the verbal edition of a sequence.

In the next part of the dialog, the user performs the beginning of a sequence selection between two branches.

MAGGIE: Tell me what you want me to do...
[*Time out for silence has been exceeded, so the robot re-prompts*]
USER: You consider two possibilities...
MAGGIE: I can't hear you. Tell me what you want me to do...
[*At this moment the asrSkill has a speech recognition result: the dialog continues...*]
MAGGIE: I do a selection with two options...
MAGGIE: What's next?
USER: In the first branch...
MAGGIE: In branch number one... what?
USER: ...if I touch your right shoulder...
MAGGIE: Adding a condition... done! What's next?
USER: Raise your right arm...
MAGGIE: Coming up... done! What's next?

Once the sequence selection is built, the user establishes the focus of edition in one of the branches by the specific referential command. The focus of edition can be changed to suit the user; therefore, it is not necessary to finish the edition of one branch to begin to edit another one, and the user can change the focus from one branch to another, alternatively. On the other side, notice how the focus of edition changes automatically after a structural command. In the example, when the user adds the condition *"if I touch your right shoulder"*, the focus is set in that node. Therefore, the next added action is placed just after the previous condition.

In the following part, there is a recognition with a low level of confidence; therefore, the robot tries to confirm what it has understood:

USER: ...and turn right.

MAGGIE: Coming up...

[*The recognition confidence is below the threshold; thus, the dialog goes to a confirmation sub-dialog.*]

MAGGIE: I have understood "turn right".

MAGGIE: Are you sure you want to add this action?

USER: Yes.

MAGGIE: Perfect!... done!

MAGGIE: Then what? What's next?

The dialog continues with the edition of the second branch. In this part, as there is lack of information in the user's utterance, the robot takes the initiative.

USER: In the second branch...

MAGGIE: In branch number two... what?

USER: If I touch your left shoulder...

MAGGIE: Adding a condition... done! Next one?

USER: Put your arm down.

[*Notice that the user has not specified which arm he is referring to; thus, the robot queries for that information slot*]

MAGGIE: Left or right one?

USER: Right one.

MAGGIE: Mhm... done! Next one?

USER: Turn left.

MAGGIE: Coming up... done! What's next?

Finally, the user completes the edition of the sequence.

USER: To conclude...

MAGGIE: Concluding the sequence selection... then?

USER: ...go back to the beginning.

MAGGIE: ...back to the beginning. And then?

USER: Execute the sequence.

MAGGIE: I execute the sequence...

[*The robot executes the created sequence. Notice that, while the sequence is in execution, the user could continue editing and modifying it.*]

In some cases, the robot adds the necessary transitions to maintain the consistency of the sequence (see Fig. 9). Therefore, a "trivial transition" (that is always true) is added as the ending of each branch. Notice that, after adding these transitions, they have been set as the focus of edition, which is used for adding the loop. The loop just connects the last transitions in the focus with the first step of the sequence. The dialog example finishes with the execution of the already created sequence. However, the user can continue editing the sequence while the sequence is being executed, adding new structural elements.

7.2. New Created Skills as Sequences

The main goal of the present work is related to the creation of new skills[6] from primitive ones, as an edutainment interactive process for the end-users. This section shows two new skills that have been created from the interaction between the robot and the user by the NPS presented in this work.

Fig. 10 shows a sequence that allows the user to operate the robot by the capacitive sensors on its head, shoulders and hands. It is a sequence selection with a loop.

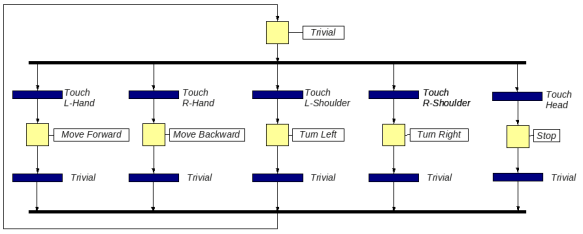


Figure 10: Example of a sequence edited by the user by dialogs. This sequence represents a skill that controls the movements of the robot using the touch sensors.

For example, if the user touches the robot's left hand, the robot begins to move forward and keeps moving until the user touches its head, and so on.

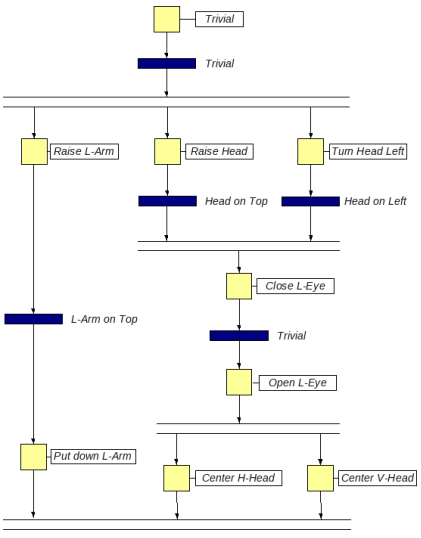


Figure 11: Example of a sequence edited by the user by dialogs. This sequence represents a skill that performs a simple greeting.

Fig. 11 shows a simple greeting skill. The sequence starts with three branches at once: raising the left arm,

[6]Each new created skill is in fact a new sequence.

raising the head and moving it to the left. Once the movements of the head are done, the robot winks the left eye. The wink action is made by closing and immediately opening the eye. After the robot winks the left eye, the head goes again to the center position in both horizontal and vertical axes. At the same time, the robot raises the left arm, and then put it down. Notice that the action "putting the left arm down" waits for the condition that the arm is at the top.

Several new similar skills can be made by the proposed NPS. The system represents an useful and promising starting point with some differences with respect to similar systems and several limitations that we will discuss next.

8. Discussion

The NPS presented here makes it possible to easily implement multiple functionalities in a social robot by verbal interaction, as we can infer from the first tests. However to give a good, complete and rigorous evaluation results of one NPS, we should have the appropriate qualitative and quantitative methods, metrics and evaluation goals, e.g. the population to involve in the tests, the parameters to measure, the design of tests and other measure methods, etc. All of this is inside an interesting researching issue that is big enough to be presented in a separate work.

Nevertheless, some first tests have been done with less than ten different volunteer subjects: non-expert users between 16 and 34 years old, and some people of the lab. In the experiments, users held a conversation with Maggie building a program. In all cases users received some instructions on how to use the NPS before performing the sequence construction. The main purpose of these tests was to get a first impression on how good the system is and what limitations should be considered for future developments. What follows is a qualitative discussion about the first results of these tests.

8.1. Degree of Entertainment and Fun of the Robot

The degree of the robot fun has been measured by a qualitatively observation of the interaction scenes between user and robot, and taking into account the amount of time that user keeps the interaction with the robot. Deeper non verbal analysis of gaze, proximity, body orientation changes, prosodic study, etc could improve and formalize the measurement of this parameter.

In general, Maggie looks quite attractive and fun for end-users that approach to the robot with expectation and interest. They enjoy the way the robot express herself, verbally (with suprasegmental prosodic features, fillers and verbal sounds as her infectious laugh), and non-verbally with simple gestures that make Maggie looks "alive". Physically, Maggie has been designed to be friendly and attractive, nevertheless we have found that Maggie looks uncanny for children under 6 years old. This is probably because of the big size of the robot (1.40 meters tall), big head and big eyes that blink[7]. In the other side, Maggie looks attractive but not very fun to young users over 18 years old, that start getting bored before younger users.

First experimental tests show that end-users enjoy a lot playing with Maggie the game of "teaching how to do things". It looks like end-users see Maggie as a delicate "baby-agent" that is still in development, which produces both a kind of sense of superiority and tenderness. Also a sensation of mate is perceived be kids (about 9 years old). Some kids assert to prefer Maggie to other electronic devices as the Wii, just because with Maggie "you play with all the body" (in words of one kid).

The NPS easily impress end-users in two ways: in the interaction side, and in the execution side, since the robot performs what the user asks to do, that is, because they see that the robot responds to what they are ordering and programming.

8.2. Difficulty of Use

The robot is not still able to completely adapt itself to the complexity of the user's communication space (verbal: vocabulary and possible utterance, and non-verbal: gestures, rhythms, accents, etc), as humans do like is described in [29]. For instance, natural human-human interaction implies that the sender is also a receiver and *vice versa*.

At the moment, it is the user who has to learn what type of utterance are the most efficient for the NPS. Nevertheless, we have found that end-users perceive this adaptation quickly and intuitively from the explanations and contextual help the robot gives, or if an expert user is also present and gives some clues and more details about robot capabilities, and the vocabulary the robot is able to detect.

Part of this usability lies in the robot's ability in recognizing user communication acts asynchronously in time: speech recognition is quite robust and efficient to dynamic variation in user utterance: repetitions, incomplete utterance, terms out of the robot's vocabulary, etc.

[7] Children under 6 usually react with fear at the blinking of the robot.

The use of CFG is a limitation both in vocabulary, as in the set of sentences that the robot can perceive. This constrain implies that the user has to learn this vocabulary and sentences. However, the dialog system helps to maintain the consistency of the interaction when the user utters something outside the formal language the robot is able to understand. In these cases, the robot reacts proposing to the user example utterance that are inside the definition of the CFG, and that explain what the robot is able to do with utterance like "tell me what to do, for example, say go forward, raise the eyelids, etc."

9. Conclusions

In this paper, we present a Natural Programming System : the design method we have used, the implementation in a social robot, and some of first (very limited) experimental results with expert and non-expert users. The NPS presented is one of the first extensive applications of voiceXML in robotics and it describes one of the few complete learning systems around.

The implemented NPS allows end-users not only to improve the capabilities of the social robots on-line, but also to do so in an interactive and entertaining way.

We have discussed the use of the SFC as a versatile and dynamic description language for robot skills as sequences. From manufacture, the social robots will come with a finite set of primitive capabilities as low-level actions ("turning right a number of degrees", "moving the head up", "say *hello world*", etc) or high-level sensorimotor skills ("following a person", "going through the corridor", etc). A SFC allows the combination of these primitive skills and the generation of new complex ones allowing complex programming structures such as parallel concurrency, selection modes and loops.

In this work, a set of primitive skills has been made completely accessible to the user's voice, implementing a system that makes it possible not only to access them, but also to combine them into new sequences by human-robot dialogs at run-time. The overall system has been successfully implemented and tested in Maggie, a real social robot that interacts with the user in a natural way, that is, including verbal and non-verbal interaction, mixed-initiative dialogs, turn taking adaptation and user's silence management.

10. Future Works

As a future research direction we plan to include more interaction in the sequence construction. For instance, instead having the primitive action raising-left-arm, that the user says "begin to raise the arm...", and then, "...stop there. This is a new action called raising-left-arm." Therefore new skills could be designed while the actions/conditions are in execution and before the whole skill is defined.

One close application is to adapt the sequence execution so the robot could perform a dancing choreography. This will be made including a Beats-Per-Minute (BPM) detector as another perception skill in the architecture. The sequence will make its movements according the detected BPM.

Some exhaustive experiments with children at different ages are also in progress. Some NPS features will be studied in depth: the degree of fun of the system, difficulty and usability for non-expert users and degree of edutainment, that is, what the user learn using the NPS. In this sense, the robot acts in fact as a teacher, performing a tutorial for the child. Therefore, the system could also be used for children to acquire some programming skills in similar ways as the "Betty's Brain" system discussed in [30]: the user will learn how to build programs by programming the robot by interaction, what can be seen as an interesting application in edutainment.

For the evaluation tests we take into account works from different disciplines. For instance, as cited in [31], engagement is a key metric for HRI evaluation. For a quantitative measure of engagement we plan using a Game Engagement Questionnaire similar as the proposed in [32], and a verbal and non-verbal video analysis, measuring acquisition time for capturing attention, duration of maintaining interest, etc.

Obviously, the system could be improved in different ways. Adding new features in HRI skills will improve the NPS considerably. For instance, adding skills that percieve user feature as position, gesture, etc and linking these skills with conditions it would be possible to create more complex sequences. In this way, "follow-person-skill" could be implmented in this way, instead of being an atomic action.

11. Acknowledgments

The research leading to these results has received funding from the RoboCity2030-II-CM project (S2009/DPI-1559), funded by Programas de Actividades I+D en la Comunidad de Madrid and cofunded by Structural Funds of the EU.

The authors also gratefully acknowledge the funds provided by the Spanish Ministry of Science and Innovation (MICINN) through the project named "A New Approach to Social Robots" (AROS) DPI2008-01109.

References

[1] G. Biggs, B. Macdonald, A survey of robot programming systems, in: in Proceedings of the Australasian Conference on Robotics and Automation, CSIRO, Vol. 1, 2003.

[2] C. Breazeal, A. Brooks, D. Chilongo, J. Gray, G. Hoffman, C. Kidd, H. Lee, J. Lieberman, A. Lockerd, Working collaboratively with humanoid robots, Vol. 1, 2004, pp. 253–272.

[3] J. G. Trafton, A. C. Schultz, D. Perznowski, M. D. Bugajska, W. Adams, N. L. Cassimatis, D. P. Brock, Children and robots learning to play hide and seek, in: HRI '06: Proceeding of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction, ACM Press, New York, NY, USA, 2006, pp. 242–249. doi:http://doi.acm.org/10.1145/1121241.1121283.

[4] S. Calinon, F. Guenter, A. Billard, On Learning, Representing and Generalizing a Task in a Humanoid Robot, IEEE transactions on systems, man and cybernetics, Part B. Special issue on robot learning by observation, demonstration and imitation 37 (2) (2007) 286–298.

[5] M. Hersch, F. Guenter, S. Calinon, , A. Billard, Dynamical system modulation for robot learning via kinesthetic demonstrations, IEEE Transactions on Robotics 24 (6) (2008) 1463–1467.

[6] M. Skubic, D. Perzanowski, S. Blisard, A. Schultz, W. Adams, M. Bugajska, D. Brock, Spatial language for human-robot dialogs, Systems, Man and Cybernetics, Part C, IEEE Transactions on 34 (2) (2004) 154–167.

[7] N. Cassimatis, J. Trafton, M. Bugajska, A. Schultz, Integrating cognition, perception and action through mental simulation in robots, Tech. rep., Naval Research Laboratory (2004).

[8] G.-J. M. Kruijff, H. Zender, P. Jensfelt, H. I. Christensen, Clarification dialogues in human-augmented mapping, in: HRI '06: Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction, ACM, New York, USA, 2006, pp. 282–289.

[9] G. J. M. Kruijff, H. Zender, P. Jensfelt, H. I. Christensen, Situated dialogue and understanding spatial organization: Knowing what is where and what you can do there, in: IEEE International Workshop on Robot and Human Interactive Communication, RO-MAN, Hatfield, UK, 2006, pp. 328–333.

[10] W. Schramm, How communication works, Process and Effects of Communication (1954) 3–26.

[11] S. B. Huffman, J. E. Laird, Flexibly instructable agents, Journal of Artificial Intelligence Research 3 (1995) 271–324.

[12] S. Iba, C. J. Paredis, P. K. Khosla, Interactive multimodal robot programming, International Journal of Robotics Research 24 (1) (2005) 83–104.

[13] S. Lauria, G. Bugmann, T. Kyriacou, E. Klein, Mobile robot programming using natural language, Robotics and Autonomous Systems 38 (2002) 171–181.

[14] X. Chen, J. Ji, J. Jiang, G. Jin, F. Wang, J. Xie, Developing high-level cognitive functions for service robots, in: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1 - Volume 1, AAMAS '10, 2010, pp. 989–996.

[15] T. Harima, H. West, Natural robot programming system, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE/RSJ, Raleigh, USA, 1992.

[16] P. F. Dominey, A. Mallet, E. Yoshida, Progress in programming the hrp-2 humanoid using spoken language, in: International Conference on Robotics and Automation (ICRA07), 2007.

[17] K. Iwase, J. Miura, Y. Shirai, Teaching a mobile robot to take elevators, in: International Conference on Machine Automation, Osaka, Japan, 2004, pp. 453 – 456.

[18] R. Barber, M. Salichs, A new human based architecture for intelligent autonomous robots, in: The Fourth IFAC Symposium on Intelligent Autonomous Vehicles, 2001, pp. 85–89.

[19] M. A. Salichs, R. Barber, A. Khamis, M. Malfaz, J. Gorostiza, R. Pacheco, R. Rivas, A. Corrales, E. Delgado, D. García, Maggie: A robotic platform for human-robot social interaction, in: Submitted to IEEE International Conference on Robotics, Automation and Mechatronics (RAM 2006), IEEE, Bankgok, Thailand, 2006.

[20] I.E.C., Preparation of Function Charts for Control Systems, IEC 848, 1988.

[21] J. F. Gorostiza, M. A. Salichs, R. Barber, A. Khamis, M. Malfaz, R. Pacheco, R. Rivas, A. Corrales, E. Delgado, D. García, Multimodal human-robot interaction framework for a personal robot, in: RO-MAN 06: The 15th IEEE International Symposium on Robot and Human Interactive Communication, IEEE, Hatfield, U.K., 2006.

[22] F. Alonso-Martín, M. A. Salichs, Integration of a voice recognition system in a social robot, Cybernetics and Systems: An International Journal 42 (4) (2011) 215 – 245.

[23] C. Shi, T. Kanda, M. Shimada, F. Yamaoka, H. Ishiguro, N. Hagita, Easy development of communicative behaviors in social robots, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2010), Taipei, Taiwan, 2010, pp. 5302 – 5309.

[24] Y. Nishimura, S. Minotsu, H. Dohi, M. Ishizuka, M. Nakano, K. Funakoshi, J. Takeuchi, Y. Hasegawa, H. Tsujino, A markup language for describing interactive humanoid robot presentations, in: Proceedings of the 2007 International Conference on Intelligent User Interfaces, Honolulu, Hawaii, USA.

[25] J. F. Allen, D. K. Byron, M. Dzikovska, G. Ferguson, L. Galescu, A. Stent, Towards conversational human-computer interaction, AI Magazine 22 (4) (2001) 27–38.

[26] S. Wermter, C. Weber, M. Elshaw, C. Panchev, H. Erwin, F. Pulvermller, Towards multimodal neural robot learning, Robotics and Autonomous Systems 47 (2-3) (2004) 171 – 175.

[27] E.Pot, J. Monceaus, R. Gelin, B. Maisonnier, Choregraphe: a graphical tool for humanoid robot programming, in: The 18th IEEE International Symposium on Robot and Human Interactive Communication (ROMAN-2009), Toyama, Japan, 2009, pp. 46 – 51.

[28] M. G., The magical number seven, plus or minus two: Some limits on out capacity processing information., The Psychological Review (1956) 81–97.

[29] R. L. Birdwhistell, Kinesics and Context. Essays on Body Motion Communication, University of Pennsylvania Press, Philadelphia, USA, 1970.

[30] G. Biswas, D. Schwartz, K. Leelawong, N. Vye, TAG-V4, Learning by teaching: A new agent paradigm for educational software, Journal Applied Artificial Intelligence 19 (2005) 363–392.

[31] A. Steinfeld, T. Fong, D. Kaber, M. Lewis, J. Scholtz, A. Schultz, M. Goodrich, Common metrics for human-robot interaction, in: 2006 Human-Robot Interaction Conference, ACM, 2006.

[32] J. H. Brockmyer, C. M. Fox, K. A. Curtiss, E. McBroom, K. M. Burkhart, J. N. Pidruzny, The development of the game engagement questionnaire: A measure of engagement in video game-playing, Journal of Experimental Social Psychology 45 (4) (2009) 624–634.