Trigger-Action Programming for Personalising Humanoid Robot Behaviour

Nicola Leonardi, Marco Manca, Fabio Paternò, Carmen Santoro
HIIS Laboratory, CNR-ISTI
Pisa, Italy
{nicola.leonardi, marco.manca, fabio.paterno, carmen.santoro}@isti.cnr.it

ABSTRACT

In the coming years humanoid robots will be increasingly used in a variety of contexts, thereby presenting many opportunities to exploit their capabilities in terms of what they can sense and do. One main challenge is to design technologies that enable those who are not programming experts to personalize robot behaviour. We propose an end-user development solution based on trigger-action personalization rules. We describe how it supports editing such rules and its underlying software architecture, and report on a user test that involved end user developers. The test results show that users were able to perform the robot personalization tasks with limited effort, and found the trigger-action environment usable and suitable for the proposed tasks. Overall, we show the potential for using trigger-action programming to make robot behaviour personalization possible even to people who are not professional software developers.

CCS CONCEPTS

•Human-centered computing~Human computer interaction (HCI) • Computer systems organization~External interfaces for robotics • Software and its engineering~Software notations and tools

KEYWORDS

End User Development; Robot Personalization; Trigger-Action Programming

ACM Reference format:
Nicola Leonardi, Marco Manca, Fabio Paternò and Carmen Santoro. 2019. Trigger-Action Programming for Personalising Humanoid Robot Behaviour. In *2019 CHI Conference on Human Factors in Computing Systems Proceedings (CHI 2019), May 4–9, 2019, Glasgow, Scotland, UK.* ACM, New York, NY, USA. Paper 445, **13** pages. https://doi.org/10.1145/3290605.3300675

CHI 2019, May 4-9, 2019, Glasgow, Scotland, UK.

ACM ISBN 978-1-4503-5970-2/19/05...$15.00.
DOI: https://doi.org/10.1145/3290605.3300675

1 INTRODUCTION

Nowadays, general-purpose humanoid robots are becoming increasingly affordable and popular since they can help users in a variety of real-world tasks. According to Gartner, by 2025, three out of 10 jobs will be converted to software, robots or smart machines[1]. Thus, they are being progressively introduced into many domains spanning from industrial environments [32], entertainment, to education [22], work automation [26], and even therapy/assistive scenarios [10]. However, in spite of the potential opportunities offered by the highly technological and heterogeneous landscape in which robots can operate (in terms of devices, sensors, actuators and services involved), a key challenge remains customizing their behaviour so that they can satisfy the extremely contextualized, unique and often evolving needs of users. Indeed, programming context-dependent robotic behaviour in dynamic environments does not seem a viable solution in the long term, as their complete behaviour cannot be hardcoded at design time by developers, who cannot foresee all the possible situations that robots would encounter during use. Thus, a more viable solution –with a potentially higher impact– is to have the unskilled people to whom the robots are increasingly exposed, directly program the specific behaviour they need. This means enabling end users with minimal technical skills to personalize the behaviour of humanoid robots to quickly respond to and address their individual requirements.

Traditional end user software engineering [19] has been mainly limited to desktop and spreadsheets applications. More recently, a good deal of interest has arisen to investigate how end-user development [21] can be exploited in Internet of Things scenarios [24, 13, 25, 8, 7]. In this area an emerging paradigm is trigger-action programming [31] which allows people to easily connect many possible events with desired actions.

[1] https://www.gartner.com/binaries/content/assets/events/keywords/symposium/sym25/gartner-sym24-executive-report2.pdf

All the available robots can be programmed through some programming language [5], which is usually oriented to engineers and requires considerable effort to learn. Thus, the issue of making the development of robot applications easier has started to be considered. In general, the approaches proposed are either iconic data flow visual languages (such as Choregraphe [27]) or the use of some block-based programming languages [20, 33]. However, such solutions seem to work well when they consider scenarios in which the possible options to address are quite limited, but this is not true with modern humanoid robots which can flexibly react to many possible events and perform a wide variety of actions. Thus, we aim to investigate whether adopting a trigger-action paradigm, such as the one supported by tools such as IFTTT[2] and Zapier [3], can enable people without particular programming knowledge to personalize the behaviour of humanoid robots. In particular, we would like to determine whether its compact and intuitive structure connecting dynamic situations with expected reactions –without requiring the use of complex programming structures– can be effective in this perspective.

Trigger-action programming has been used for automating Web services or home control [11, 31] but, to the best of our knowledge, it has not been applied for end user personalization of humanoid robot behaviour. In addition, this approach opens the way to easily merge robots and Internet of Things (IoT) sensors into a continuum whereby no main difference is made between robots and the smart objects available in the user environment. In addition, new services can be created through the combination of the sensing/monitoring capabilities usually associated with IoT and the robots' capability of acting and interacting with the environment and especially humans. Thus, the contribution of this work is to show how a solution based on trigger-action rules can be used to make personalization of humanoid robot behaviour easier for people who are not professional software developers.

In the paper, after providing the main motivations of the research, we first analyse the state of the art in the field, then we describe the platform for personalization of robot behaviour, which has been integrated with the Pepper robot by SoftBank Robotics. Then, we report on a user study, which provides promising indications, and discuss its design implications. Lastly we draw some concluding remarks and directions for future work.

2 RELATED WORK

A necessary distinction to make is between industrial robots, developed to accomplish specific tasks in a well-defined environment, and social humanoid robots, usually exploited in natural environments coexisting with human beings with whom they must relate and exchange information. By social robots we mean humanoid robots who can help us at our jobs, in housework, in the care of children, elderly and disabled people, in hospitals, schools, hotels and so on. Such robots interact with us by voice, gestures and all the other modes typical of human communication.

According to [14] there are various possible levels of abstractions in programming a social robot. The lowest level consists of hardware primitives. These primitives concern the control and recovery of the state from the hardware devices, such as the control of the LEDs, of the motor actuators, the recovery of the data from the sonar, the video camera and the microphones. This level of abstraction is where for example the Pepper ROS (Robot Operating System)[4] specifications are positioned. The second level of abstraction consists in the algorithm primitives. These primitives are related to the robot ability to interact with the environment and a first functional approach to social interaction. Examples of this level of primitives are the recognition / speech synthesis, the tracking of faces (face tracking), localization of the source of noise. The third level consists of social primitives. These are atomic units of social interaction implemented through algorithmic primitives and in some cases through hardware primitives. For example, the ability to follow a person's gaze is a combination of face-tracking and head and torso motor control. They differ from hardware and algorithmic primitives because they are specific to the domain of social interaction. The fourth abstraction level is the emergent primitives, an abstraction level that combines two or more social primitives (in Pepper an example is the *ALDialog* [1] library). It can concern the ability to express emotions or the human-robot dialogue where the robot must combine different social primitives, such as face-tracking, speech recognition / synthesis, body movement, etc. The last abstraction level consists in the control primitives. It is the highest layer dedicated to the control of the various levels of the underlying primitives,

[2] https://ifttt.com/discover
[3] https://zapier.com/
[4] ROS.Org | Powering the World's Robots." http://www.ros.org/

managing them in a symbiotic and transparent way. For example, a finite-state machine used to manage a human-robot dialogue is a part of this level, combining dialogue and listening primitives, a rule-based system, or a programmed action structure. As described in [9] the best level of abstraction to address and on which to build an environment for programming humanoid robots is that concerning social primitives, because the emerging primitives are less reusable and customizable given their characteristic of combining heterogeneous contents and contexts. In the development of our solution, this has been taken into account, aiming to minimize the use and mix of different types of primitives and focusing on social primitives.

Various techniques have been explored for facilitating robot application development. One technique that has been considered is programming by demonstration [2], in which the robot is taught new behaviours by directly demonstrating the task to transfer, instead of programming it through machine commands. However, robot programming by demonstration is cumbersome for the end users, and usually works well only for specific cases. In general, there are two approaches [14, 9] to the development of interactive applications aimed at social robots: frameworks for experienced programmers, often requiring a GNU / Linux platform, and programming languages with a very steep learning curve, such as C++ and LUA. These approaches, on the one hand aim to optimize the performance, on the other hand are not suitable for those addressing these issues with limited technical background. The Bonsai framework [30] and the Robot Behavior Toolkit [18] are part of this group.

Another approach is the use of visual toolkits with more user-friendly interfaces that guarantee rapid interaction development without aiming to optimize the final resulting performance. In recent years, several environments have been released that go in this direction, such as Choreographe [27], Interaction Composer [16], Interaction Blocks [29], TiViPE [23], Target-Driven-Means [4]. However, they are not able to provide a comprehensive support, since Interaction Composer and Interaction Block lacks support for hardware primitives, while TiViPE lacks support for emergent primitives. Target-Driven Means aims to support specification of a collection of behaviours that run in parallel, but it requires technical knowledge in order to be applied.

In this area we can distinguish approaches that use data flow iconic representations of robot functionalities (such as Choregraphe), and those that use block-based languages introduced by Scratch [28] and which have already been applied in other domains [3, 12]. An example of a visual programming tool for user-friendly, rapid programming manipulators robots is in [17]. The system is designed to let non-roboticists and roboticists alike program end-to-end manipulation tasks. The authors present findings from an observational user study in which ten non-roboticist programmers were able to learn the system and program a robot to do manipulation tasks. Buchina et al. [6] propose a design of a Web-based programming interface that aims to support end-users with different backgrounds to program robots using natural language. Still in the attempt of lowering the barriers of programming robots, and then make it more accessible and intuitive for novice programmers, Weintrop et al. [33] present a block-based interface (CoBlox) for programming a one-armed industrial robot. The results show that participants using the block-based interface successfully implemented programs faster with no loss in accuracy while reporting higher scores for usability, learnability, and overall satisfaction. However, they considered a rather limited scenario, with one-armed industrial robots, and the participants were asked to program a “pick and place” routine, which is a relatively narrow set of functionality while in our study we consider humanoid robots, which can sense various pieces of information and perform a wide variety of actions.

Other examples of visual languages for robots are LEGO Mindstorms language (https://www.lego.com/it-it/mindstorms) and NAO Choreographe by Aldebaran. These languages handle the program as a set of blocks with defined inputs and outputs. These are connected to each other with “wires”, that might represent data flow or a sequence of execution. As previously highlighted [6], one of the main drawbacks of these languages is the poor applicability for describing complex behaviours, as they typically take a lot of space on the screen, and wires tend to tangle, eventually making the program hardly readable.

Overall, it seems clear that general solutions for supporting end user personalization of robot behaviour are still missing, and in addition it would be useful that such solutions be able to integrate the control of the robot behaviour with what happens in the surrounding environment, usually populated by various smart objects.

3 THE HUMANOID ROBOT TAILORING ENVIRONMENT

The proposed Tailoring Environment allows users who are not professional developers to specify the triggers and the

actions relevant for the customized behaviour they want to obtain. We assume that some general-purpose applications to control the robot behaviour have already been implemented by professional developers, and the users would like to add some specific behaviours that better meet their specific needs. They create the rules through a specific Tailoring Environment and such rules are sent to a Rule Manager, which is informed of the occurrence of relevant events and conditions by the Context Server to trigger such rules and indicate what actions consequently to perform to the robot (see Figure 1). The events are collected by a specific Context Server that can receive events also from sensors external to the robot in order to obtain an integrated management of what happens inside and outside the robot.

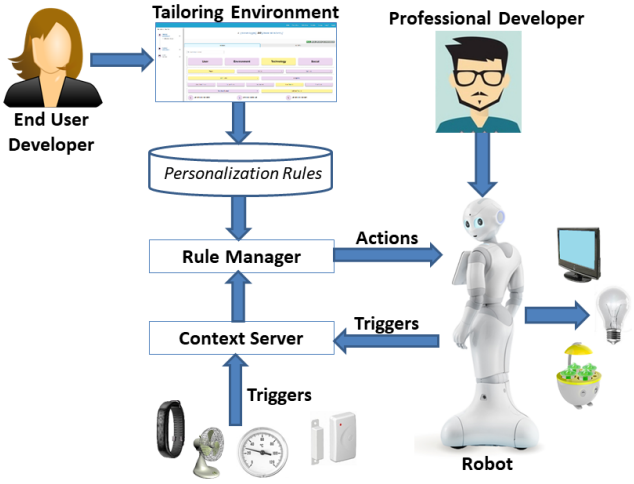

Figure 1: The Proposed Approach

3.1 Triggers

In general, the Tailoring Environment organizes the possible triggers according to whether they depend on the user, the environment, the technology or some social aspect. The Pepper-related triggers have been associated with the Technology dimension, interpreting Pepper as a further technological device. Thus, the Technology element is structured into "Pepper", "Devices" and "Appliances", which are the technologies that can generate relevant events. The selection of the triggers is performed by navigating through a hierarchical organization where the higher levels are used to group triggers that refer to similar aspects. Thus, the "Pepper" element has been refined into "Touch Events" and "Recognition": the first one deals with the management of the various types of touches and pressures of robot sensors, while the second one includes the various types of recognition supported by the robot. The "Touch Events" have been classified into: "Chest Button press", "Bumper events", "Head touch", "Hand Touch", "Tablet Events". Except for "Hand Touch", which is further subdivided into two elements (Right and Left Hand Touch), the other elements are refined into terminal elements (for instance "Bumper Events" has been refined into "Right Bumper Touch", "Back Bumper Touch", "Back Bumper Touch"). "Recognition" has been further refined into: Speech, Face, and Picture Recognition (see Figure 2), which are all terminal elements, hence the tool allows the user to specify some attributes of the corresponding trigger.

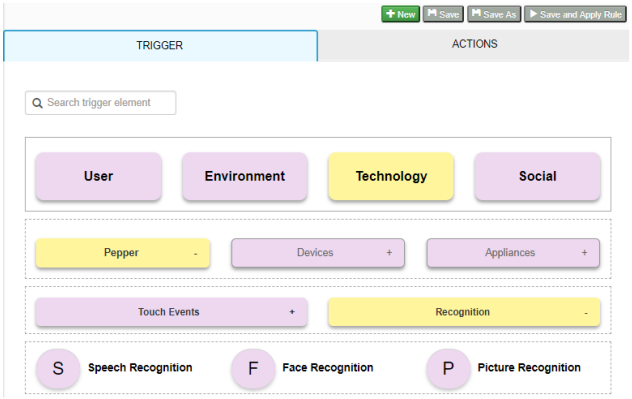

Figure 2: Robot-Related Triggers in the Tailoring Tool

3.2 Actions

Similarly, also the actions have been categorized in a hierarchical structure, according to the different types that devices and actuators can make available in the specific setting considered. They refer to the various actions that the robot can carry out, namely: Speak, Hand Movement, Animation, Position change, Video Camera activation, LEDs state change, Alarms, Reminders.

3.3 Trigger and Action Composition

While many existing trigger-action programming tools such as IFTTT mainly supports rules with one trigger and one action, this Tailoring Environment provides the possibility of applying AND and OR logical operators among the choices of the triggers, and obtaining more complex emergent primitive logics by combining multiple actions of heterogeneous nature in a parallel or sequential way (see Figure 3). During the rule composition, in the top part the tool provides feedback in natural language about what rule has been entered.

An aspect that has been specifically introduced to handle the actions carried out by the robot is the possibility to compose multiple actions in sequence or in parallel. However, while a sequential composition of

actions is always possible, the introduction of a parallel composition might lead to possible inconsistencies in the rules. This is the case when a user tries to combine in parallel two actions that involve the same object (e.g. open and close the hand simultaneously), or actions that overlap to some extent. The latter case can occur for instance when one action uses an animation involving some LEDs and another action simultaneously sets the colour of such LEDs. This situation is managed by the tool, which, when an inconsistent combination of actions is specified during the creation of a rule warns the user about the potential issue.

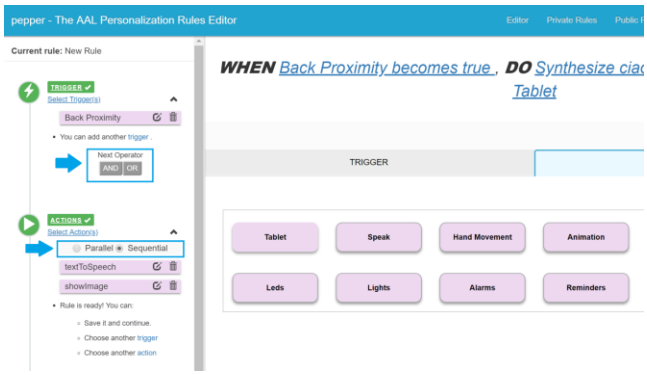

Figure 3: Action Composition Specification (See Left Hand Panel) in the Tailoring Environment

3.4 Integration between Robot and IoT Sensors

The Tailoring Environment also supports the definition of triggers generated by sensors external to the robot, and to send actions to external appliances (see Figure 4). Thus, it is possible to define events or conditions that depend on both types of heterogeneous aspects and have effects on multiple technologies as well. For example, it is possible to specify that when a person enters a room and the light is off, the robot can greet the user and the light will be turned on.

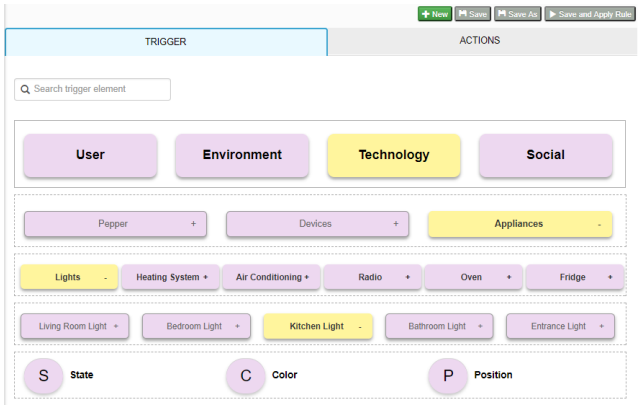

Figure 4: Selection of a Trigger Involving IoT Appliances

4 SOFTWARE ARCHITECTURE

For the development of our platform we were inspired by [15] but we had to address the specific aspects of humanoid robots, which were not considered at all in that work. The software architecture is composed of a number of software modules. Through the Tailoring Tool, users can define suitable rules following the trigger-action paradigm, connecting dynamic events and/or conditions (triggers) occurring in the current context with expected desired reactions (actions). The tool provides intuitive panels allowing even non-professional developers to select relevant contextual situations and consequent actions.

The hierarchy of trigger types relevant in the specific domain considered are modelled in a XSD file that the authoring tool loads when it starts. Thus, in order to properly support the specific, actual environment at hand (in our case including the Pepper robot with its specific possible events and actions), the Tailoring Tool needs to be 'populated' in terms of the actual triggers and actions available in the setting considered. To this aim, at initialization time, the tool asks the Context Manager the list of actual objects and sensors for populating the panel dedicated to triggers, and, similarly, it asks the application about the specific actions that it makes available, in order to be able to show them in the UI part dedicated to actions.

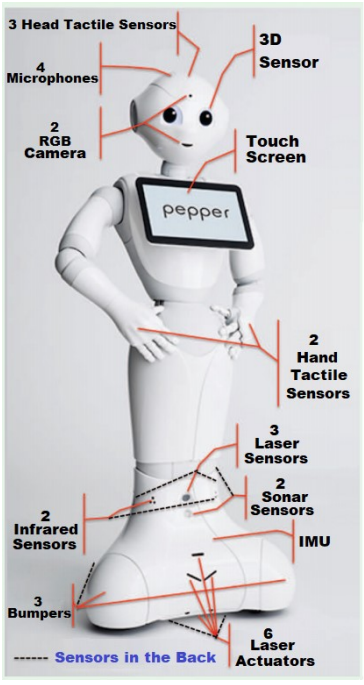

Figure 5: The Pepper Robot by SoftBank Robotics

The Robot we considered is Pepper, by SoftBank Robotics (see Figure 5), a 1.2-m-tall wheeled humanoid robot, with 20 degrees of freedom for motion in the body (17 joints for body language) and three omnidirectional navigation wheels to move around smoothly. It has a range of sensors to allow it to perceive objects and humans in its surroundings: i) six-axis inertial measurement unit sensor composed of a three-axis gyrometer and a three-axis accelerometer; ii) four microphones in the head to provide sound localization; iii) two cameras at the forehead and mouth positions; iv) one 3-D sensor located behind the eyes; v) three tactile sensors: one in the head and one on top of each of the hands; vi) three bumpers, one on each wheel position; vii) a tablet attached to its chest; viii) some laser sensing modules; ix) two loudspeakers, laterally placed on the left and the right sides of the head; two sonar sensors, one in front and one in back; and two infrared sensors at the base; x) Ethernet and Wi-Fi network connectivity.

4.1 The Main Modules

The flow of information exchanged between the various architectural modules to support personalization of robot behaviour is graphically represented in Figure 6. In particular, the robot has first to subscribe to the Rule Manager, to signal its willingness to receive relevant customizations from the platform. The customised behaviours are those specified by end users through a number of trigger-action rules that they create through the Tailoring Tool. After a user creates and saves a rule through the tool, the rule is sent in JSON format to the Rule Manager, the architectural module aimed at handling the rules at run time. In particular, the Rule Manager stores the rules as associated with a specific user and application. According to such rules, the Rule Manager subscribes to the Context Manager (the module that monitors the current context), to be informed when a change in the context verifies the trigger part of one (or more than one) rule.

Indeed, the Context Manager provides the functionalities to collect and store contextual data from external sources (e.g. robots, sensors, devices). For this purpose, it is composed of a centralized unit (the Context Server) and a set of software modules (Context Delegates), which are delegated to monitor relevant parameters of the context of use (e.g. temperature, noise, light, state of doors/windows) and are installed in relevant devices. The Context Delegates collect contextual data from the associated sensors and then continuously provide data which allows the Context Server to update its current view of the context state. For each subscribed event and condition that occurs, the Context Manager notifies the Rule Manager, which extracts the list of actions from the verified rules (i.e., the rules whose trigger part is verified) and sends them to the robot (and in turn to the relevant appliances, when necessary) for execution.

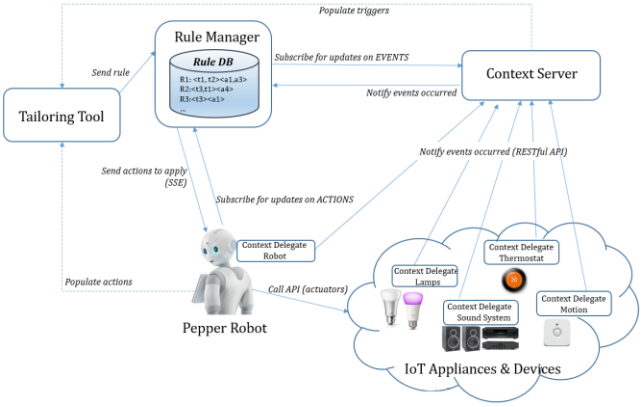

Figure 6: The Architecture of the Platform

In order to allow the Pepper robot to communicate with the other modules of the platform, a specific Context Delegate has been built: it is a Python application running on the robot and capable of capturing all the significant events generated by its sensors. In particular, it manages events including pressure (and gesture) on hand(s), head, bumper, tablet touch, motion detection, voice recognition, face detection/recognition, object recognition, proximity detection and battery charge level.

The Context Delegate associated with Pepper sends the events occurring in the robot to the Context Server by using RESTful APIs. In particular, it sends the events that have occurred to specific REST endpoints defined on the Context Server. Regarding the communication between Pepper and the Rule Manager, a Server Sent Events – based communication channel has been used. The rationale of using this one-way communication was due to the fact that, once the robot subscribes to the Rule Manager signalling its willingness to receive new personalization actions to execute (according to the rules that will be specified by end users), it only needs to receive the actions from it.

When an action message requires composite actions, it is possible to indicate sequential or parallel values in the *action Mode* field, thus identifying how to combine the multiple actions during execution. With this field it is possible to ask the robot to execute complex actions such as pronouncing a sentence and at the same time moving

the hands or the head, thus supporting the creation of emergent primitives in a customizable way.

4.2 The Robot Native Support Used in the Platform to Handle Triggers and Actions Main Modules

The NAOqi Framework [1] is the programming framework for programming Pepper. It is cross-platform (it is possible to develop on Windows, Linux or Mac), and cross-language (i.e. with identical APIs for both C++ and Python). The NAOqi executable process which runs on the robot is a broker. When it starts, it loads a preferences file that defines the libraries to load. Each library contains one or more modules that use the broker to advertise their methods.

Triggers. As for triggers, particularly interesting is the ALMemory module which i) provides information about the current state of Pepper's actuators and sensors; ii) provides methods to get and set named values; and iii) acts as a hub for the distribution of event notifications (i.e. provides methods to subscribe to events and also generate events). For the proxemics-related aspects, feedback from the back and front sonar managed by the ALSonar component is used. In particular, when certain thresholds (configurable) are exceeded, specific triggers are associated. The default value (1 meter) is the boundary between personal distance (between 45 cm and 1 meter), and social distance (between 1 meter and 4 meters).

Actions. We particularly focused on six main categories of "natural" interaction modalities: speech; gestures; facial expressions; haptic; tracking of the gaze; proxemics and kinesics. For instance, for face tracking, we exploit the default behaviour associated with the state of the interactive type, in which the robot "follows" (with its gaze and body movements) the person who has attracted its attention, making the robot exit the 'solitary' condition due to one of the following causes: motion detection; sound stimulus (generic noise, or a person who starts talking); draw attention through the movement of the arms or approach to the robot; touch (body or touch panel). Such "contact" modalities are managed by the ALAutonomousLife module. In addition, we support the possibility that the robot could be configured to show "natural human" behaviours (e.g. tiredness or euphoria) in relation to the battery level, or refuse to perform certain actions if "too tired" or perform actions at different speeds depending on its battery charge (which is managed by the BatteryChargeChanged event of the module ALMemory).

5 USER STUDY

We carried out a user test to understand to what extent our approach for programming humanoid robots via trigger-action rules can be effective for non-professional developers.

5.1 Participants

17 participants (13 females) with age ranging between 24 and 56 (mean=38.2, std. dev.=10) were involved in the user study. They were recruited by sending emails to various lists of the authors' research Institute explaining that we were looking for people who are not professional developers. In particular, people mainly working in administrative roles were recruited for the test. The main criterion for including people in the test was that they must not be (professional) developers. As for the education, 5 users held a High School degree, 3 a Bachelor, and 9 a Master Degree.

Their knowledge of programming languages was categorized in five different levels from no knowledge to very good knowledge: 1: No knowledge in programming; 2: Low Knowledge (which means: knowledge of HTML, CSS, and basic knowledge of JavaScript); 3: Medium Knowledge (knowledge of JavaScript, basic knowledge of either PHP or Java or C++); 4: Good Knowledge (good knowledge of either PHP or Java or C++); 5: Very Good Knowledge (knowledge of development languages at a professional level). All participants had a knowledge of programming languages between no knowledge and medium knowledge (Median = 2). In particular, 8 of them (47%) declared no programming experience, 3 of them (18%) had little experience, the remaining 6 (35%) declared medium programming experience. The majority of participants (14 out of 17) had never used any customization tool for smart environments: among the others, the only cited tool was IFTTT. In addition, users were asked whether, before the test, they already had some experience in programming a robot. Only two users replied positively (one user already used Lego Mindstorms, another used a mini ROV submarine), none of them dealt with a humanoid robot.

5.2 Test Organization

The test was done in a laboratory. One moderator observed the participants interacting with the tool (one at a time), annotating any issue, remark, and question they had. The test was organized in four phases: introduction and motivations, familiarization, test execution, questionnaire. We informed users about the organization of the test in four phases. We asked them to provide

honest feedback, saying that the test is an evaluation of the tool and not of their performance, and that any issue they could find is useful to improve the tool. We asked their permission to log their interactions, and informed them that the data would have been aggregated to avoid identification.

In the first phase participants received a brief introduction to the study, illustrating its main goals and motivations. Then, they were also provided with a description of the Tailoring tool and the trigger-action rule structure, as well as a brief explanation of the features of the tool. In addition, we provided users with a short video illustrating the tool capabilities and some example interactions for building trigger-action rules. Participants were strongly encouraged to analyse such introductory information remotely i.e. before the test, in order not to unnecessarily prolong the duration of the test. In addition, participants also had a brief introduction to the humanoid robot (i.e. available sensors and actuators).

In the second phase people familiarized themselves with the tool. Participants could freely interact with it for some time, to understand its use and features. In addition, in this phase users were first asked to provide in natural language three freely identified rules (R1, R2, R3) they would define for personalizing the behaviour of a robot.

After that, the moderator asked them to identify three rules of incremental structural complexity and specify them using the tool. The structure was provided by the moderator, e.g. "please identify a rule having 1 trigger and 2 actions". In particular, the three levels of complexity were the following: R4 (low complexity): a rule having one trigger and one action; R5 (medium complexity): a rule with 2 triggers and 1 action; R6 (high complexity): 2 triggers and 2 actions. The complexity of the R4-R6 rules increases progressively in terms of elements to specify, overall, having two elements to specify for R4 (1 trigger and 1 action), three elements to specify for R5 (2 triggers and 1 action) and four elements to specify for R6 (2 triggers and 2 actions). Thus, all the tasks associated with specifying R4-R6 were presented to users in the same order (according to an increasing complexity).

Then, users were presented with five specific behaviours written in natural language (they were all the same for all users) and participants were asked not only to model them using the tool, but also to actually apply and execute them on the robot, to check whether the Robot behaved as expected. The additional 5 rules that have been requested to specify with the tool in the test are described below.

Rule 7: As soon as the robot recognizes the sentence "could you please help me?", Pepper replies "I'd like to remember you an appointment that you have today" and then it shows on the tablet a textual message "You have a dentist's examination at 18". The purpose of the rule was to interact with the robot via a vocal command and then have a multimodal output from the robot, combining spoken and written text. This rule is structured into 1 trigger and 2 actions (combined through a sequential operator).

Rule 8: "In case the left hand of Pepper is pressed and the phrase "I'm feeling not very well" be pronounced, the robot replies with the phrase "I understand, I'll take care of it." The goal of this rule was to have a multimodal trigger (tactile + vocal), which activates a vocal response from the robot. This rule is structured into 2 triggers (combined with an AND operator) and 1 action.

Rule 9: "In case a movement is detected in the room Pepper says "Someone has entered the office" and simultaneously turns on the green eye LEDs for 5 seconds". The goal of this rule was to show the integration of the robot with the smart environment in which it is placed (a motion sensor was installed in order to enable the verification of the rule during the test). This rule is structured into 1 trigger and 2 actions (combined through a parallel operator).

Rule 10: "As soon the robot recognizes the user phrase "How are you?", it responds with "well, thanks!", and it does the animation named "Happy", and afterwards pronounces "and you?". The goal of this rule was to compose an answer showing some empathy to the interlocutor. This rule is structured into 1 trigger and 3 actions (combined through a sequential operator).

Rule 11: "As soon as a user approaches Pepper, the robot turns on red the light in the current room, records a 10s video and pronounces simultaneously the sentence "Did a person enter the room?". This last rule aims at defining a concrete role of the robot, i.e. with a surveillance purpose (recording a video) integrated with the appliances of the environment in which it is located (in our case a lamp was available in the room). This rule is structured into 2 triggers and 3 actions (combined through a parallel operator).

Such rules were identified to cover relevant aspects connected with human-robot interaction. Also such rules

(R7-R11) were presented to participants in the same order. The idea was again to progressively increase the 'complexity' of rules, which was mainly connected with two aspects: a higher number of triggers and actions to specify implied a more complex rule (i.e. more things to do); we also assumed that the use of a sequential operator was less difficult than using a parallel operator. Indeed, another key aspect that we wanted to analyse was the capability of users to correctly express rules in which the actions could be done either sequentially or in parallel. In particular, Rule7 and Rule10 exploited a sequential operator, Rule9 and Rule11 exploited a parallel operator (whereas Rule 8 just involved one action). After creating each rule in the R7-R11 set, users had to immediately execute them on the robot.

In the last phase users filled in an online questionnaire asking for some socio-demographic information (age, gender, education), and also indicating their expertise in programming, and whether they had previously used any system for building trigger-action rules. In addition, they filled in the SUS questionnaire, and they also rated some aspects of the proposed environment, and provided observations on positive/negative aspects they noticed on the assessed system and recommendations for its possible improvements. Most people compiled the final anonymous questionnaire remotely, so the possibility of social pressure was very low.

5.3 Results

Regarding rules R1-R3, the analysis highlighted that users would like the robot to show some social behaviour (e.g. greetings), or give them help in carrying out routine tasks and also provide assistance in case of potential risky situations. Regarding rules R4-R6, examples of rules created by users were (in natural language):

(Low complexity): On a specific date and time, the robot should provide the user with a vocal reminder "Remember to go out of office at 16.00"). In case the user says to Pepper "You're doing well", the robot verbally replies with "I'm very pleased about this")

(Medium complexity): In case the robot recognizes a specific person who says "What's the weather like?" Pepper will load on the tablet the web site of a weather forecast service. If a person is in front of Pepper and motion is detected in the surrounding, Pepper will send a vocal alarm by voice: "a person is in the office".

(High complexity): If Pepper is front-touched and also recognizes the vocal sentence "What do you do in life?", Pepper does a Hysterical animation and then says "I work a lot". When it is a specific time and date Pepper should turn on all Face Leds blue for 5 seconds and say in parallel: "Remember to take the medicine".

Please note that people, already in this test phase, used both parallel and sequential operator composition, although no specific requirement was put on any of such operator (e.g. to use one instead of another one). This can suggest that both operators are well received by users.

From the analysis of the interaction logs during the rule composition phase, useful information was extracted regarding the correctness, completeness and execution time of the tasks associated with rules R4-R11 (those which implied an actual use of the tool). All the participants completed the specification of the rules without making significant errors. There was one participant (P2) who, while specifying Rule7, instead of using the "Text to Speech" action for modelling the rule, used a different type of action (a "Reminder" having a voice type). However, this error was completely transparent from the point of view of the result of the task, since the robot correctly expressed the desired output, by verbally rendering the text specified by the user. As far as task completeness is concerned, all the tasks have been carried out by participants. In addition, the rules built were most of the time correct on the first try. The moderator provided some help only during rule execution, by giving advice e.g. when the speech was not recognized on the first attempt (sometimes users spoke far from Pepper or not loud enough).

As for the execution times, an overview is shown in Figure 7 representing the min, max and quartiles of times needed to create the rules in the test (the time needed for actually executing the rule on the robot was not considered in the diagram). Participants seemed to reach a quite steady state after completing the first rule that involved an active use of the tool (R4). As an explanation of the (significantly higher) time needed for modelling R4 compared to the other rules, it was likely because users needed first to have a more concrete idea of the possibilities offered by the tool in terms of triggers and actions, before actually specifying the rule. Then, users took the opportunity of the first task involving the use of the tool for modelling rules, to actually familiarise with it.

Regarding rules R7-R11, in terms of time dedicated to the specification of the rules we can note that their times were overall quite comparable, apart from R11 (2T, 3A,

PAR), which confirmed to be the most difficult task.

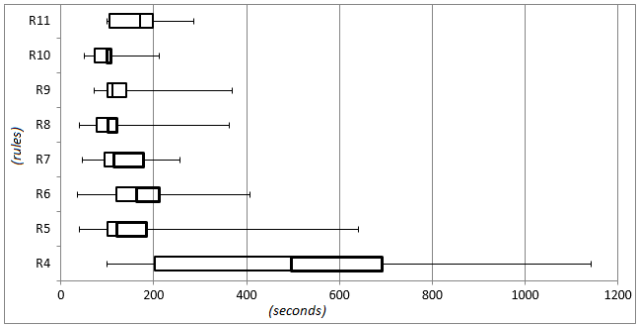

Figure 7: Box and Whisker Plot for the Execution Times

As for the SUS questionnaire, the average SUS score across all the users was 73.1. We also computed correlations between the programming experience and the time needed to specify the rules, finding a weak inverse correlation between the two (Pearson correlation index rho=-0.309). This suggests that the programming experience was not a factor that relevantly affected the time needed for specifying the rules, since the task times achieved by people across the three levels of programming experience are overall comparable. In addition, we asked further questions to participants, in particular:

Q1. After the application of a certain rule, Pepper behaved as expected? 13 users replied affirmatively. As for the remaining four ones, three of them said that there was an issue connected with Pepper having difficulties in recognizing user's speech. Nonetheless one participant said that this issue was solved by just repeating a bit louder the command directed to the robot. The remaining one said that Pepper did not correctly behave when a rule specifies to carry out an action at a specific time. However, this was due to a time format incorrectly specified in the rule. In addition, another user said that, when there was the rule in which the user had to touch the robot head, it changed its status to sleep instead of turning on the leds (as requested from the rule). This was due because of a built-in Pepper behaviour that puts the robot to sleep when its head is pressed for 3 seconds.

Q2. During the test was it clear to you at all times in what state was the robot? Twelve users replied affirmatively. As for the other five ones, two of them said that they had the feeling that Pepper did not quickly react to user-generated triggers ("sometimes Pepper does not react quickly to my requests"); another one said that it was unclear to him whether Pepper was actually 'listening' to user's commands. A factor that could have affected these users' perceptions could have been that during the test some delays occurred on the network, and this could have generated a less reactive behaviour of the robot. However, it is worth noting that, in order to better make users aware of Pepper's state, we programmed the robot in such a way that in case of recognition of a verbal expression the robot's eye LEDs were automatically turned on and set to green. This was appreciated by users, especially in case of network delays. Another user said that Pepper's animations were not that easy to be detected and distinguished especially at the beginning (when the user did not know what to expect from them), but this issue disappeared after a couple of repetitions. Another user said that at a certain point it seemed that the robot behaved according to additional, unexpected actions. This was because the robot has some predefined micro-behaviours added to those indicated by the users' specified rules.

Q3. Do you think this approach is useful for customizing the behaviour of the robot? (Likert scale, 7 levels, 7=best). The median value was 6 (min:4, max:7). Users overall found the approach very useful to customize the behaviour of a robot, and they especially appreciated the large set of triggers and actions available in the system. However, some of them encouraged to further improve the user interface of the authoring tool. For instance, one user noticed that sometimes there are many selectable elements in the UI at the same time (which can confound the user), and/or several selections are needed for reaching the lowest levels of the hierarchies of triggers and actions. Thus, the suggestion was to make the interaction quicker and more intuitive. A suggestion was to use more 'visual' paradigms. Furthermore, users were asked to identify the three aspects they appreciated most and the three ones they disliked most.

The three aspects users appreciated most. The vast majority of users mentioned the easiness of writing the personalization rules as one the aspects that they appreciated most, which corroborates overall the findings gathered by logs. One user mentioned the possibility for the users to create their own rules, the hierarchical structure in which triggers and actions are organized, and the possibility of being able to have actions performed sequentially or in parallel. Another user mentioned the good level of freedom in the choice of triggers and possible actions, the ease with which it is possible to set the robot functions without developing the code, and the ability to immediately see the output of the rules set.

The three aspects users did not like much. One user complained that the set of terms recognizable by the robot could have been made larger; the navigation for the

selection of actions was not always intuitive; the reactions of the robot did not seem to happen in real time in some cases, and therefore it was necessary sometimes to repeat the sentences. Some users said that the language used in the tool (i.e. for the hierarchies of triggers and actions) required some technical knowledge due to some technical terms used in it (e.g. "triggers"). One user said that there are some types of recognition that are currently missing (e.g. emotion recognition). Another user mentioned it was not completely clear how to combine multiple actions in sequence and in parallel in the same rule.

Further suggestions to improve the approach/tool. Finally, users were also asked whether they had further suggestions to improve the solution. One user said that the possible actions of the robot were a bit too many in some places, and he would have preferred limiting them to a lower number (five or six elements) at a time, so providing additional groupings when needed. Another user suggested using a more visual approach in the tool, as well as further expanding robot's recognition features by also including the recognition of emotions. One user suggested providing the user with more indications of the actual robot state in order to better make aware users of it.

One interesting aspect that emerged from the test was the fact that sometimes the interaction involving the robot was not perceived in the same way as interacting with a common piece of digital equipment but in a slightly different manner. Indeed, the human-like appearance of the robot suggested more human-like properties and behaviour, and therefore a less 'technological' connotation of the robot. This sometimes affected the users' mental model while they looked for specific triggers in the hierarchy, e.g. sometimes having difficulties in properly locating Pepper-related events under the "Technology" dimension.

6 DISCUSSION

Overall the test results were encouraging: users were enthusiastic about participating, and about their final results. The rules built were correct most of the time on the first try. As mentioned before, some users had hesitations in quickly locating Pepper-related events under the 'Technology' dimension, probably due to the human-like appearance of the robot, which suggested that the interaction with humanoid robots is not perceived by users as involving a technological device, but rather an entity similar to them. Although about half the participants did not have any programming experience, and almost none of them had any experience before the test with the technologies involved (i.e. programming a robot, using environments for customizing context-dependent behaviours), users were able to accomplish the proposed tasks, showing that the system is easy to learn and use for novice users. As a result of this study, a number of implications for design can be derived.

Offer the most appropriate abstraction level to end-user developers. The personalization of robot behaviour can be supported at various levels in robot programming, from basic hardware primitives to social behaviour. EUD tools should be able to hide the complexity of the many underlying technologies involved, and highlight the main conceptual aspects that need to be understood and manipulated through intuitive metaphors and programming styles. In the case of a humanoid robot it is important to: identify the events triggering some specific robot behaviour without having to learn the low-level sensing technology and communication protocols involved; set the robot reactions that are expected to be similar to those of a human, and thus should involve multiple modalities, which should be described in terms understandable by non-professional programmers.

Provide effective support to specify human-like robot behaviour. People tend to anthropomorphize the world around them. This is evident for interactions with humanoid robots able to respond and interact with humans in a natural way (e.g. showing emotions, performing gestures, speaking). In practice, this involves e.g. the ability to set animations, activate coloured LEDs, and control the robot's hands/arms while it is speaking. Thus, users need effective means for specifying human-like robot behaviour, which requires some level of structure, which can be accomplished for example by combining the possible actions sequentially and/or concurrently. This is different from what occurs with IoT appliances (e.g. lamps, radio), which usually involve performing simple sequential actions.

Integration of humanoid robots with IoT. Differently from industrial robots, humanoid robots can be used in various every day environments (equipped with several IoT devices/things/sensors). The possibility to detect what happens in the surrounding environment opens the way to exploit triggers that use the data detected by both the robot and IoT objects and to link the robot behaviour to what happens around it. Thus, EUD tools should provide users with suitable techniques for specifying such triggers to describe context-dependent robot behaviour.

Provide simulation and conflict analysis tools. Personalizing context-dependent robot behaviour may imply writing several rules. In such situations, users can have difficulties in forming an accurate model of the resulting behaviour. Thus, users would benefit from tools helping them to build an accurate mental model of the overall behaviour according to currently active rules. So, automatic support for simulating robot behaviour in specific contexts and for conflict analysis (to identify and resolve any interference between rules) should be developed to enable people with less programming expertise to maintain full control of their robot and environment.

Graceful degradation of robot behaviour. In order to support fluid and natural interactions, robots should show a good level of reactivity towards users. When this does not happen (due to unexpected failures), humans could feel a lack of control, which can undermine their user experience and confidence. Thus, we need suitable techniques to ensure that in unexpected situations the robot is able to 'gracefully degrade', by e.g. providing users with relevant explanations of the situation, trying to mitigate associated negative effects.

Support a high degree of behaviour reuse. Since the behaviours to specify tend to be multi-faceted, we should provide users with means to re-use already created rules or even some parts of such rules (e.g. blocks of actions) through familiar concepts to easily refer to them in further, more structured rules.

7 CONCLUSIONS

Robots are being increasingly used in many fields. However, their utility will remain limited unless end-users without technological expertise will be empowered to program them. In this paper we present a platform enabling users without significant programming experience to customise the behaviour of a robot immersed in an IoT smart environment through personalization rules expressed in trigger-action format. The results of a user study indicate that the EUD tool can be actually used for personalizing the robot behaviour and is well received by users. Overall, this work advances our understanding of how to make possible end-user development of personalization of interactive humanoid robot behaviour.

The study was an in-lab test, in which we gave explicit task assignments to limit the possibility of ambiguity and to better compare the collected results. In future evaluations we will consider challenging users through less explicit task instructions, and conduct longitudinal studies assessing the use of the tailoring tool for longer periods of time, to investigate whether further aspects emerge (e.g. if and how the way to personalise the robot would change over time).

REFERENCES

[1] Aldebaran 2018. NAOqi - Developer Guide — Aldebaran 2.4.3.28-R2 Documentation. Retrieved August 5, 2018 from http://doc.aldebaran.com/2-4/index_dev_guide.html.

[2] Sonya Alexandrova, Maya Cakmak, Kaijen Hsiao, and Leila Takayama. 2014. Robot Programming by Demonstration with Interactive Action Visualizations. In Proceedings of the 2014 Robotics: Science and Systems Conference. DOI: https://doi.org/10.15607/RSS.2014.X.048

[3] Emilia Ivanova Barakova, Jan Gillesen, Bibi Huskens, Tino Lourens, 2013. End-user programming architecture facilitates the uptake of robots in social therapies, Robotics and Autonomous Systems, Elsevier, 61(7), 704-713. DOI: https://doi.org/10.1016/j.robot.2012.08.001

[4] Vincent Berenz and Kenji Suzuki, 2014. Targets-Drives-Means: A declarative approach to dynamic behavior specification with higher usability, Robotics and Autonomous Systems, Elsevier, 62(4), 545-555. DOI: https://doi.org/10.1016/j.robot.2013.12.010

[5] Geoffrey Biggs and Bruce Macdonald. (2003). A survey of robot programming systems. In Proceedings of the Australasian Conference on Robotics and Automation, CSIRO, 1–10.

[6] Nina Buchina, Sherin Kamel and Emilia Ivanova Barakova. 2016. Design and evaluation of an end-user friendly tool for robot programming. In Proceedings of IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN '16). IEEE, 185-191. DOI: https://doi.org/10.1109/ROMAN.2016.7745109

[7] Nico Castelli, Corinna Ogonowski, Timo Jakobi, Martin Stein, Gunnar Stevens, and Volker Wulf. 2017. What Happened in my Home? An End-User Development Approach for Smart Home Data Visualization. In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17). ACM, New York, NY, USA, 853-865. DOI: https://doi.org/10.1145/3025453.3025485

[8] Fulvio Corno, Luigi de Russis, and Alberto Monge Roffarello. 2017. A High-Level Approach Towards End User Development in the IoT. In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17). ACM, New York, NY, USA, 1546-1552. DOI: https://doi.org/ 10.1145/3027063.3053157

[9] Enrique Coronado, Fulvio Mastrogiovanni, and Gentiane Venture, 2017. Node Primitives: An Open End-User Programming Platform for Social Robots, CoRR abs/1709.08363.

[10] Angelo Costa, Ester Martinez-Martin, Miguel Cazorla, and Vicente Julian, 2018. PHAROS - PHysical Assistant RObot System, Sensors, MDPI, 18, 2633. DOI: https://doi.org/10.3390/s18082633

[11] Joelle Coutaz and James Crowley, 2016. A first person experience with end-user development for smart home, IEEE Pervasive Computing, 15 (2), IEEE, 26:39. DOI: https://doi.org/10.1109/MPRV.2016.24

[12] José Danado and Fabio Paternò, 2014. Puzzle: A mobile application development environment using a jigsaw metaphor. Journal of Visual Languages & Computing, Elsevier, 25(4), 297-315. DOI: https://doi.org/10.1016/j.jvlc.2014.03.005

[13] Giuseppe Desolda, Carmelo Ardito and Maristella Matera, 2017. Empowering End Users to Customize their Smart Environments: Model, Composition Paradigms, and Domain-Specific Tools, ACM Transactions on Computer-Human Interaction, 24(2), ACM, Article 12, 52 pages. DOI: https://doi.org/10.1145/3057859

[14] James Diprose, Bruce MacDonald, John Hosking, and Beryl Plimmer. 2017. Designing an API at an appropriate abstraction level for programming social robot applications. Journal of Visual

Languages & Computing, Elsevier, 39, C, 22-40. DOI: https://doi.org/10.1016/j.jvlc.2016.07.005

[15] Giuseppe Ghiani, Marco Manca, Fabio Paternò, and Carmen Santoro, 2017. Personalization of Context-dependent Applications through Trigger-Action Rules. ACM Transactions on Computer-Human Interaction, 24(2), ACM, Article 14, 33 pages. DOI: https://doi.org/10.1145/3057861

[16] Dylan Glas, Satoru Satake, Takayuki Kanda, Norihiro Hagita. 2012. An interaction design framework for social robots, In Proceedings of the 2012 Robotics: Science and Systems Conference. DOI: https://doi.org/10.15607/RSS.2011.VII.014

[17] Justin Huang and Maya Cakmak. 2017. Code3: A system for end to-end programming of mobile manipulator robots for novices and experts. In Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction (HRI '17). ACM, New York, NY, USA, 453-462. DOI: https://doi.org/10.1145/2909824.3020215

[18] Chien-Ming Huang and Bilge Mutlu. 2012. Robot behavior toolkit: generating effective social behaviors for robots. In Proceedings of the seventh annual ACM/IEEE International Conference on Human-Robot Interaction (HRI '12). ACM, New York, NY, USA, 25-32. DOI: https://doi.org/10.1145/2157689.2157694

[19] Andrew Ko, Robin Abraham, Laura Beckwith, Alan Blackwell, Margaret Burnett, Martin Erwig, Chris Scaffidi, Joseph Lawrance, Henry Lieberman, Brad Myers, Mary Beth Rosson, Gregg Rothermel, Mary Shaw, and Susan Wiedenbeck. 2011. The state of the art in end-user software engineering. ACM Comput. Surv. 43, 3, Article 21 (April 2011), 44 pages. DOI: https://doi.org/10.1145/1922649.1922658

[20] Jannik Laval, 2018. End User Live Programming Environment for Robotics. Robotics & Automation Engineering Journal, 3(2), June 2018.

[21] Henry Lieberman, Fabio Paternò, Markus Klann, Volker Wulf, 2006. End-User Development: An Emerging Paradigm. In: Lieberman, Henry, Paternò, Fabio and Wulf, Volker (eds.), Springer, End-user development (Human-Computer Interaction Series), 1-8. DOI: https://doi.org/10.1007/1-4020-5386-X_1

[22] João Paulo Cardoso de Lima, Lucas Mellos Carlos, José Pedro Schardosim Simão, Josiel Pereira, Paulo Manoel Mafra, Juarez Bento da Silva, 2016. Design and implementation of a remote lab for teaching programming and robotics, IFAC-PapersOnLine, 49(30), 86-91. DOI: https://doi.org/10.1016/j.ifacol.2016.11.133

[23] Tino Lourens and Emilia Ivanova Barakova. 2011. User-friendly robot environment for creation of social scenarios. In Proceedings of the 4th International Work-Conference on the Interplay Between Natural and Artificial Computation, IWINAC'11, Springer, Lecture Notes in Computer Science, 6686, 212–221. DOI: https://doi.org/10.1007/978-3-642-21344-1_23

[24] Panos Markopoulos, Jeffrey Nichols, Fabio Paternò and Volkmar Pipek, 2017. Editorial: End-User Development for the Internet of Things. ACM Transactions on Computer-Human Interaction 24, 2, Article 9 (April 2017), 3 pages. DOI: https://doi.org/10.1145/3054765

[25] Georgios Metaxas and Panos Markopoulos. 2017. Natural Contextual Reasoning for End Users. ACM Transactions on Computer-Human Interaction 24, 2, Article 13 (April 2017), 36 pages. DOI: https://doi.org/10.1145/3057860

[26] Yoha Oishi, Takayuki Kanda, Masayuki Kanbara, Satoru Satake, and Norihiro Hagita. 2017. Toward End-User Programming for Robots in Stores. In Proceedings of the Companion of the 2017 ACM/IEEE International Conference on Human-Robot Interaction (HRI '17). ACM, New York, NY, USA, 233-234. DOI: https://doi.org/10.1145/3029798.3038340

[27] Emmanuel Pot, Jérôme Monceaux, Rodolphe Gelin, Bruno Maisonnier. 2009. Choregraphe: a graphical tool for humanoid robot programming. In Proceedings of the 18th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN'09). IEEE, 46-51. DOI: https://doi.org/10.1109/ROMAN.2009.5326209

[28] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, and Yasmin Kafai. 2009. Scratch: programming for all. Commun. ACM 52, 11 (November 2009), 60-67. DOI: https://doi.org/10.1145/1592761.1592779

[29] Allison Sauppé and Bilge Mutlu. 2014. Design patterns for exploring and prototyping human-robot interactions. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14). ACM, New York, NY, USA, 1439-1448. DOI: https://doi.org/10.1145/2556288.2557057

[30] Frederic Siepmann and Sven Wachsmuth, 2011. A Modeling Framework for Reusable Social Behavior. In R. De Silva & D. Reidsma (Eds.), International Conference on Social Robotics, Work-in-Progress Workshop Proceedings, Springer, 93-96.

[31] Blase Ur, Elyse McManus, Melwyn Pak Yong Ho, and Michael L. Littman. 2014. Practical trigger-action programming in the smart home. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14). ACM, New York, NY, USA, 803-812. DOI: https://doi.org/10.1145/2556288.2557420

[32] Jan Van den Bergh, Fredy Cuenca Lucero, Kris Luyten, and Karin Coninx. 2016. Toward specifying Human-Robot Collaboration with composite events. In Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN'16), New York, NY, USA, IEEE Press, 896-901. DOI: https://doi.org/10.1109/ROMAN.2016.7745225

[33] David Weintrop, Afsoon Afzal, Jean Salac, Patrick Francis, Boyang Li, David C. Shepherd, and Diana Franklin. 2018. Evaluating CoBlox: A Comparative Study of Robotics Programming Environments for Adult Novices. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18). ACM, New York, NY, USA, Paper 366, 12 pages. DOI: https://doi.org/10.1145/3173574.3173940