A social robot empowered with a new programming language and its performance in a laboratory

Biel Piero E. Alvarado Vasquez
Center of Automatics and Robotics
CSIC-UPM
Madrid, Spain
biel.alvarado@upm.es

Fernando Matia
Center of Automatics and Robotics
CSIC-UPM
Madrid, Spain
fernando.matia@upm.es

***Abstract*—For the last years, social robots have been used to serve as tour-guide robots in museums, laboratories, trade fairs, etc. So they are intended to perform tasks which will be affected by the interaction with people. In order to perform these tasks, a task planner must be developed in order to integrate the several modules that the robot contains in its software architecture. In this paper a new programming language was developed in order to command the actions of the robot over the place where the social robot is going to perform its actions. Lexical, syntactical and semantical analyzers were developed based on a simple language syntax with the possibility to create events. These events can come from different sensors installed like the RFID, cameras, etc. The testbed is Doris, an interactive robot developed in the Centre of Automation and Robotics. The results show that the new language can merge all the modules from Doris, executing the actions (trajectories, speeches and actions based on events) specified i n t he p rogram p rovided b y t he d eveloper without making any changes in the lower layers of the framework.**

***Index Terms*—Social Robots, interactive robots, system integration, task planning, programming language**

I. Introduction

Social robots are very useful for interacting with people in different environments. Work in this area has been performed for many years such as the case of MINERVA, a social robot for interaction by [1], RHINO [2] [3] which is another robot dedicated to work in museums. Another robot is NAO from Aldebaran Robotics [4]. [5] presents another work for human-robot interaction, in particular, robot emotion, speech and facial expressions to determine if people feel comfortable in front of a machine that exhibits basic rational and intelligent behaviour. Another research in interactive robots is [6] presenting RHINO XR3 and MARK 3. And another architecture can be found at [7].

Concerning task planning, several research has been done to implement the sequential execution of tasks. Different types of programming languages can be found in [8], showing the applications (general or robot-specific) o f e veryone o f them. ROBOL/0 [9] is a programming language oriented to the navigation behaviour. Other programming language mentioned is [10] intended to provide actions to a robotic arm. Another work performed by [11] where the researchers propose a new logic programming language called GOLOG which is based on PROLOG. A related work can also be found in [12] which is an extension of GOLOG including concurrency. A programming language with a syntax similar to JavaScript, Lua and Python, is created to program swarm robots [13]. [14] introduces a method to use written natural language instructions to program assembly tasks for industrial robots. Another contribution can be found at [15] by designing a visual programming language for robots end users called RURU. With the extended use of XML for different purposes, [16] created L-IRL, a procedural language for industrial robot programming based on XML code.

The aim of this research is to provide Doris with a language that can integrate the different modules of the architecture in order to perform tours and interact with people at the environment. Doris is going to move around the Centre of Automation and Robotics (CAR). The subsystems to be integrated into the new language are the localization system, the path planning system, the mapping system, the emotions systems, the facial system and the lip syncing system.

The structure of the paper is as follows: In section II the Doris architecture will be brefly explained. The new programming language is briefly explained in section III, detailing the development of an interpreter, showing the grammar and the functionality of every instruction of the new language. Finally, a script and the results shown in section IV and its execution will be shown in a video.

II. Doris, the social robot girl

Doris is an interactive robot, conceived to work at museums, theaters, trade fairs and different indoor environments. It builds on the upgrading of Blacky and Urbano, the two previous robots that the group at CAR has been developing over recent years at the Universidad Politécnica de Madrid [17] [18]

A. Hardware architecture

The hardware consists of three parts:

- a mobile platform developed by Adept, as shown in figure 1a.
- a skeleton, which replicates the human body physical appearance, as shown in figure 1b.

- a head, attached to the skeleton, used for interaction, as shown in figure 1c

B. Software architecture

The software is a Client-Server application, enabling direct communication between Doris (server) and the (client) that controls it. Figure 2 shows how each module is organized in Doris. Each block on the figure details a task that is asynchronously under execution on a different thread, which means that in the higher level of the architecture, there is a task handler.

III. The programming language for Doris

Why is it necessary to introduce a new programming language in a social robot if there are thousands of them already? If the robot is intended to work as a tour-guide, why its own actions are not programmed directly in its program core? These are the questions made when a new programming language is designed for a robot. The most appropriate answer is that each robot user could design a program in order for the robot to act as the user wants in a simple programming language and without the necessity of recompiling the core program which manipulates the robot, always remembering that Doris is intended to work in a closed environment which is very variant and in which different events can take place.

Understanding that a tour is a set of actions to be performed by Doris, the idea of creating a programming language, which is as any other simple programming language but that also includes direct commands of Doris movements, configuration of what, when and how to say sentences and direct actions over different sensors data, appears.

A. Requirements of the language

Grammar, which is going to be used to program the sequence of actions for Doris, must contain common features of all programming languages like input/output instructions, conditionals clauses, loops, arithmetic operations and boolean operations. Besides these features, the new programming language for Doris allows variables definition which can be either numeric or literals or defined as an array which can contain both numeric and literals in the same vector.

B. General grammar structure

The grammar of the language is expressed as next:

⟨*compound_stmt*⟩ ::= ⟨*if_stmt*⟩
| ⟨*while_stmt*⟩
| ⟨*for_stmt*⟩
| ⟨*funcdef*⟩
| ⟨*vardec*⟩
| ⟨*sizeof_stmt*⟩
| ⟨*say_stmt*⟩
| ⟨*turn_stmt*⟩
| ⟨*goto_stmt*⟩
| ⟨*move_stmt*⟩

⟨*call-function*⟩ ::= ⟨*identifier*⟩ ’(’ [⟨*parameter-list*⟩] ’)’

⟨*literal-string*⟩ ::= ’ ‘‘ ’ [a-zA-Z0-9]* ’ ” ’

⟨*statement-list*⟩ ::= ⟨*statement*⟩ ⟨*statement-list*⟩ | ⟨*statement*⟩

⟨*statement*⟩ ::= ⟨*ident*⟩ ’=’ ⟨*expression*⟩ | ⟨*compound_stmt*⟩

⟨*parameter-list*⟩ ::= ⟨*ident*⟩ [’,’ ⟨*parameter-list*⟩]

⟨*expression*⟩ ::= ⟨*def*⟩ [(’==’ | ’!=’ | ’⟨=’ — ’⟩=’ | ’⟨’ — ’⟩’) ⟨*def*⟩]

⟨*def*⟩ ::= ⟨*sum*⟩ [(’+’ | ’-’ | ’and’ | ’or’ | ’not’) ⟨*sum*⟩]

⟨*sum*⟩ ::= ⟨*factor*⟩ [(’*’ | ’/’ | ’%’) ⟨*factor*⟩]

⟨*factor*⟩ ::= ⟨*ident*⟩ | ⟨*number*⟩ | ⟨*expression*⟩

⟨*number*⟩ ::;= ’#’ ⟨*digit-part*⟩ [’.’ ⟨*digit-part*⟩]

⟨*digit-part*⟩ ::= (⟨*digit*⟩)*

⟨*digit*⟩ ::= ’0’ ... ’9’

where different types of statements and basic grammar are detailed as how to call a function or how literals and numbers are defined. A basic program is shown as next:

```
function main()
    say[face=happy, attention=front](”Hello Everybody”)
     return 0
endfunction
```

Fig. 3: Basic Program in the language developed for Doris

C. The if statement

The **if** statement is used for conditional execution.

⟨*if_stmt*⟩ ::= ’if’ ’(’ ⟨*expression*⟩ ’)’ ⟨*statement-list*⟩ [’else’ ⟨*statement-list*⟩] ’endif’

The language developed for Doris should be able to allow the execution of a group of instructions when certain expression is true. In cases where that condition is false, a suite of the **else** clause is executed if present

```
if(boolean_condition)
    say[face=happy, attention=front](”Hello Everybody”)
else
    say[face=happy, attention=front](”Bye Everyone”)
endif
```

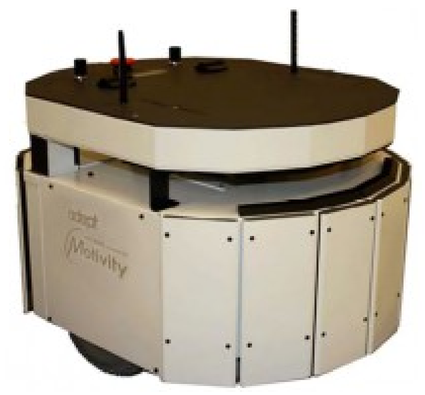
Fig. 4: **if** statement sample

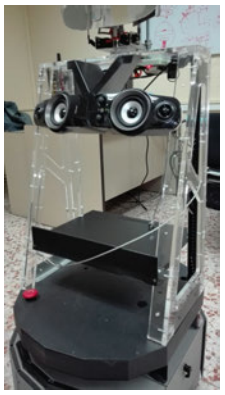
D. The while statement

The **while** sentence is used for repeating the execution of a block of statements while certain condition is true.

⟨*while_stmt*⟩ ::= ’while’ ’(’ ⟨*expression*⟩ ’)’ ⟨*statement-list*⟩ ’endw’

This instruction is needed in cases where Doris is intended to repeat certain tasks, like asking the same question or moving forward until some point is reached.


(a) Platform


(b) Skeleton

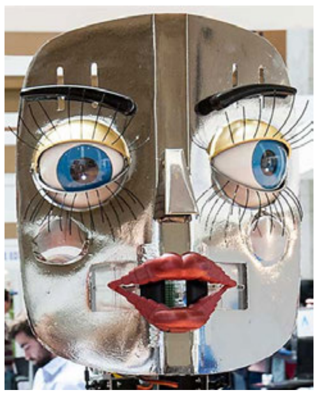

(c) Head

Fig. 1: Doris hardware overview

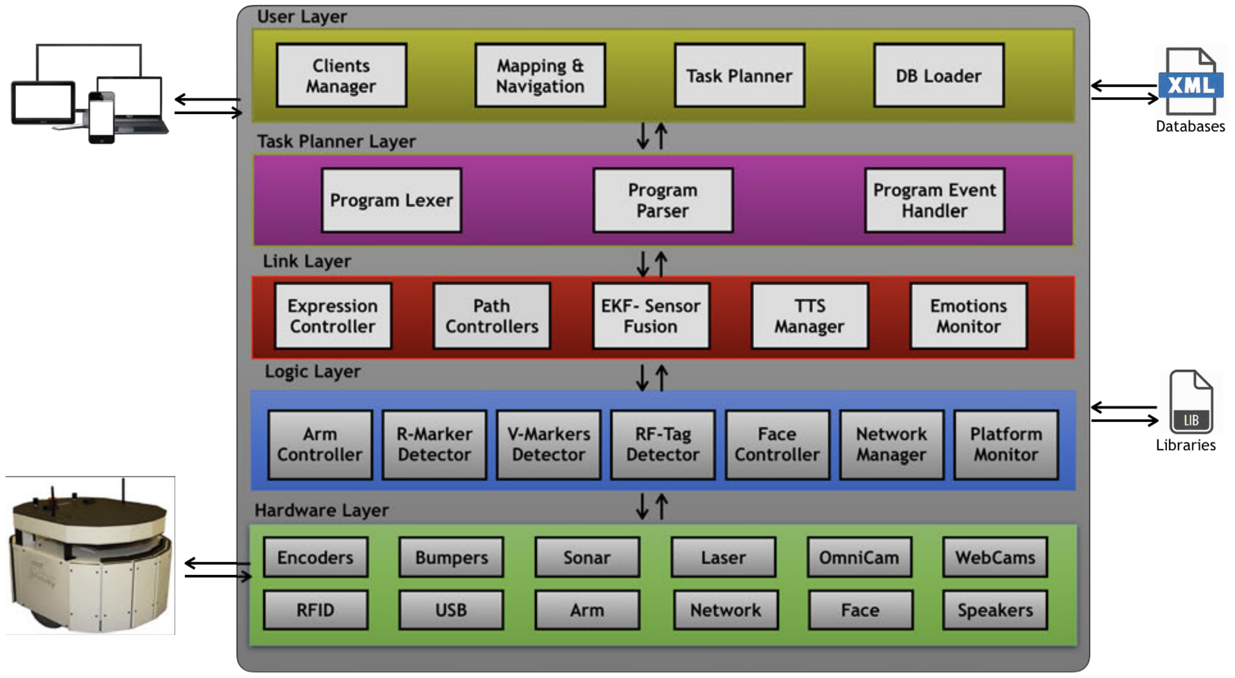


Fig. 2: Software architecture of Doris

```
while(not location == "door")
    move(next)
endw
```

Fig. 5: **while** statement sample

E. The for statement

The **for** is used to iterate over a sequence of elements like a list or an array, for instance, a list of tags read from RFID sensor.

$\langle for\_stmt \rangle$ ::= 'for' '(' $\langle expression \rangle$ ':' $\langle expression \rangle$ ':' $\langle expression \rangle$ ')' $\langle statement\text{-}list \rangle$ 'endfor'

```
for(var i = #0 : i < sizeof(tags) : i = i + #1)
    if(tags[i] == "2804")
        say("this painting is from Salvador Dali")
    else
        if(tags[i] == "2804")
            say("This is a Cardenal Cisneros sculpture")
        endif
    endif
endfor
```

Fig. 6: **for** statement sample

As seen in algorithm 6, the statement is very similar to C language **for** statement, which firstly is a declaration of a variable, then, secondly, the condition and finally the upgrade of the variable.

F. Function definitions

Functional programming has always been a very good idea in all programming languages. This was included in order to tell Doris the principal function to be executed and also create action routines which can be called or not from the main function or other functions.

The grammar is expressed as next:

$\langle$*funcdef*$\rangle$ ::= 'function' $\langle$*ident*$\rangle$ '([$\langle$*parameter-list*$\rangle$]')' $\langle$*statement-list*$\rangle$ ['return' $\langle$*expression*$\rangle$] 'endfunction'

```
function hallway(name)
    say[face=happy, attention=front]("Hello Everybody")
    say("name")
     return 0
endfunction
```

Fig. 7: Function sample

A new set of statements can be declared into the function, and it can contain parameters passed by values and also return a value if the **return** clause is stated.

G. Variable declaration

As in every language, variables are used for storing information. For Doris they are very helpful to handle information coming from sensors, specially the information provided from the sensors. The grammar is:

$\langle$*vardec*$\rangle$ ::= 'var' $\langle$*ident*$\rangle$ ['=' ($\langle$*expression*$\rangle$ | $\langle$*call-function*$\rangle$)]

$\langle$*vardec*$\rangle$ ::= 'var' $\langle$*ident*$\rangle$ ['=' '[' $\langle$*expression*$\rangle$ ',' $\langle$*expression*$\rangle$ ',' ... ',' $\langle$*expression*$\rangle$ ']']

```
var portraitCode = "8056"
var portraitCodes = ["8056", #8326, "8126", "8123"]
```

Fig. 8: Variable declaration

H. The say statement

This command indicates that Doris needs to say a specific phrase.

$\langle$*say_stmt*$\rangle$ ::= 'say' ['[' 'attention=' $\langle$*attention-item*$\rangle$ ';' 'face=' $\langle$*face-item*$\rangle$ ']'] '(' $\langle$*literal-string*$\rangle$ ')'

As it can be observed in the grammar, the instruction can be executed with extra options. These options are concerning to the state of the face and the attention direction.

```
say("My name is Doris")
say[face=smiley; attention=left]("Let's move to the left side
of the room")
```

Fig. 9: **say** instruction

I. The turn statement

The **turn** command makes the robot turn the amount of degrees specified as a parameter. The degrees can be specified as a variable, as a result of a function or writing the value in degrees.

$\langle$*turn_stmt*$\rangle$ ::= 'turn' '(' ($\langle$*number*$\rangle$ | $\langle$*ident*$\rangle$ | $\langle$*expression*$\rangle$) ')'

```
turn(#90)
```

Fig. 10: **turn** instruction

J. The move statement

The **move** command tells the robot to move to the next or the previous waypoint specified in the tour database.

$\langle$*move_stmt*$\rangle$ ::= 'move' '(' ('next' | 'prev') ')'

```
move(next)
```

Fig. 11: **move** instruction

K. The goto statement

The **goto** command indicates the robot where to go. The grammar is specified as next:

$\langle$*goto_stmt*$\rangle$ ::= 'goto' '(' ($\langle$*number*$\rangle$ | $\langle$*ident*$\rangle$ | $\langle$*expression*$\rangle$) ',' ($\langle$*number*$\rangle$ | $\langle$*ident*$\rangle$ | $\langle$*expression*$\rangle$) ',' ($\langle$*number*$\rangle$ | $\langle$*ident*$\rangle$ | $\langle$*expression*$\rangle$) ')'

The function contains three values: the first is the sector identifier in the same map, if #-1 is especified, then the Robot will remain in the same sector where it is located, and the other two parameters are position X and Y in the plane and inside the sector boundaries. An example is shown next:

```
goto(#-1, #4, #3)
goto(#2, #4, #3)
```

Fig. 12: **goto** instruction

L. The sizeof statement

The **sizeof** command returns the actual size of an array. If the variable specified is not an array, it will return 1;

$\langle$*sizeof_stmt*$\rangle$ ::= 'sizeof' '(' $\langle$*ident*$\rangle$ ')'

```
var portraitCodes = ["8056", #8326, "8126", "8123"]
var s = sizeof(portraitCodes)
```

Fig. 13: **sizeof** instruction

IV. Results

In order to join all the submodules into a single execution, a program for touring in the Centre of Automation and Robotics at Universidad Politécnica de Madrid has been developed. The program is listed in listing 1. The program consists in going to the entrance of the laboratory and then the robot starts to speak to the visitors. Then the program makes the robot move to the first location which is going to be explained. As stated, the explanations is based on points of interest which are previously defined and when the robot arrives to the point it will execute the following command. In this specific case the commands to provide an explanation is the **say** instruction.

As it can be observed, events are included in the program. These events are:

1) **onRfidTagsDetected**: which subscribes a function to the event and allows to the interpreter to know what to do with the tags.
2) **onSectorChange**: which notifies to the interpreter what to do when a new sector is loaded in the robot.

Listing 1: Program that performs a tour in the Centre of Automation and Robotics

```
var finishedTour = #0

onRfidTagsDetected(rfidEvent)
onSectorChanged(newSectorEvent)

function rfidEvent(tagsList)
    var i = #0
    while(i < sizeof(tagsList))
        if(tagsList[i] == "e280-1160-6000-0207-28b5-2a04")
            say("That big box that is placed right there at the bottom, it is where I am stored to go out the university")
            say[face=afraid]("That thing scares me a lot")
        endif
        i = i + #1
    endw
endfunction

function newSectorEvent(sectorId)
    say("New sector achieved")
    if(sectorId == #0)
        say[face=happy]("Now we are at the place where I have been built. This is the Mobile Robots Room")
        explainMoileRobots()
        finishedTour = #1
    endif
endfunction

function explainMoileRobots()
    goto(#-1, #1.6, #2.27)
    turn(#0)
    say[face=happy]("If you look at the top of that light brown closet, you can find VAD 1, which was the first mobile robot at the laboratory")
    goto(#-1, #2.4, #2.27)
    turn(#90)
    say[face=sad]("Then we have here our little fish Nemo. Some of you may have seen the movie Finding Nemo. When he got lost he arrived here and became a this little robot to be used for exploration")
    goto(#-1, #4.8, #2.27)
    turn(#90)
    say[face=surprise]("Right there is another mobile robot called Blacky. But as you can observe, it does not have a gorgeous face like mine and people did not know how to interact with it")
    goto(#-1, #6.6, #2.27)
    turn(#90)
    say[face=surprise]("And finally, here is my predecesor Urbano. He was touring around museums performing almost the same work like me, but he has turned old the poor guy.")
    turn(#180)
    say[face=surprise]("There is a poster near Blacky and can be observed that he and Urbano have arrived to the Valencia Science Museum")
    say("as I am a girl of her house, I stay right here.")
    say[face=happy]("Well people, There is nothing left to add besides saying that behind me is the desk of my maker and where he has given me life, like Frankenstein to his creation.")
    say[face=happy]("Thank you for everything and have a nice day.")
endfunction

function main()
    goto(#-1, #0.9, #11.50)
endfunction
```

A full video, named **Full System Test: Touring in the Laboratory**, where the robot is performing the program mentioned above can be found at the group's webpage gallery Intelligent Control Group UPM-CSIC - Gallery: http://blogs.upm.es/controlinteligente/en/gallery/

V. Conclusions

A complete interactive robot for tours in museums and laboratories has been developed by developing a language to be able to command all the submodules inside the Doris architecture.

The programming language proposed for task planning has a very simple grammar so that any basic programmer can be able to create a new tour easily. This programming language integrates different components of the robot like the sensors, face and voice interaction, emotions, navigation, etc. The language is a mix between C, Pascal and JavaScript languages which can be written in a more simplistic way than other languages developed at the moment for robot's task planning.

The language also integrates the face and voice modules which also needed to be synchronized with the lips movements and the navigation of the robot when arriving to the specific place.

Acknowledgement

The results in this paper report the research supported by the DPI2017-86915-C3-3-R: Técnicas de Inteligencia Artificial y Ayuda a la Navegación Autónoma that is under by the Spanish Ministry of Science, Innovation and Universities.

This research has also received support from the RoboCity2030-IV-CM project (Robótica aplicada a la mejora de la calidad de vida de los ciudadanos. fase IV; S2018/NMT-4331), funded by Programas de Actividades I+D en la Comunidad de Madrid and cofunded by Structural Funds of the EU.

References

[1] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz, “Probabilistic algorithms and the interactive museum tour-guide minerva,” *Journal of Robotics Research*, vol. 19, pp. 972–999, November 2000.

[2] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, “Experiences with an interactive museum tour-guide robot,” *Elsevier Artificial Intelligence*, vol. 114, pp. 3–55, October 1999.

[3] W. Burgard, A. B. Cremers, D. Fox, D. Hanel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, “The interactive museum tour-guide robot,” in *AAAI '98/IAAI '98 Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, pp. 11–18, 1998.

[4] R. G. Boboc, M. Horaţiu, and D. Talabă, “An educational humanoid laboratory tour guide robot,” *Procedia - Social and Behavioral Sciences*, vol. 141, pp. 424–430, 2014.

[5] D. Vogiatzis, C. D. Spyropoulos, S. Konstantopoulos, V. Karkaletsis, Z. Kasap, C. Matheson, and O. Deroo, “An affective robot guide to museums,” in *4th International Workshop on Human-Computer Conversation*, 2008.

[6] M. Tekerek, “A human robot interaction application for robotic education,” *Procedia - Social and Behavioral Sciences*, vol. 1, no. 1, pp. 2164–2169, 2009.

[7] R. Stricker, S. Muller, E. Einhor, C. Schroter, M. Volkhardt, K. Debes, and H.-M. Gross, “Konrad and suse, two robots guiding visitors in a university building,” in *Autonomous Mobile Systems (AMS 2012)*, pp. 49–58, 2012.

[8] S. Yang, X. Mao, B. Ge, and S. Yang, “The roadmap and challenges of robot programming languages,” in *IEEE International Conference on Systems, Man, and Cybernetics*, pp. 328–333, 2015.

[9] S. Suzuki and S. Yuta, “Analysis and description of sensor based behavior program of autonomous robot using action mode representation and analysis and description of sensor based behavior program of autonomous robot using action mode representation and robol/0 language,” in *IEEE/RSJ International Workshop on Intelligent*, vol. 3, pp. 1497–1502, 1991.

[10] J. Lapham, “Robotscript™: the introduction of a universal robot programming language,” *Industrial Robot: An International Journal*, vol. 26, no. 1, pp. 17–25, 1999.

[11] H. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. Scherl, “Golog: A logic programming language for dynamic domains,” *Journal of Logic Programming*, vol. 31, no. 1-3, pp. 59–83, 1997.

[12] G. D. Giacomo, Y. Lespérance, and H. J. Levesque, “Congolog, a concurrent programming language based on the situation calculus,” *Elsevier Artificial Intelligence*, vol. 121, no. 1-2, pp. 109–169, 2000.

[13] C. Pinciroli and G. Beltrame, “A programming language for robot swarms,” *IEEE Software*, vol. 33, no. 4, pp. 97–100, 2016.

[14] M. Stenmark and P. Nugues, “Natural language programming of industrial robots,” in *IEEE ISR*, pp. 1–5, 2013.

[15] J. P. Diprose, “End user robot programming via visual languages,” in *IEEE Symposium on Visual Languages and Human-Centric Computing: Graduate Consortium*, pp. 229–230, 2011.

[16] M. Lutovac, G. Ferenc, JelenaVidaković, Z. Dimić, and V. Kvrgić, “Usage of xml and p code for robot motion control,” in *Mediterranean Conference on Embedded Computing*, pp. 162–165, 2012.

[17] B. P. E. Alvarado, F. Matia, and R. Galan, “Improving indoor robots localisation by fusing different sensors,” in *IEEE International Conference on Intelligent Robots and Systems (IROS'18)*, pp. 2616–2623, 2018.

[18] B. P. E. Alvarado, R. Gonzalez, F. Matia, and P. de la Puente, “Sensor fusion for tour-guide robot localization,” *IEEE Access*, vol. 6, pp. 78947 – 78964, 2018.