Learning to develop Interactive Applications with the EVA Robot

Renan Martins CS de Lima
Marcelo Marques da Rocha
Débora Christina Muchaluat Saade

Road map

ÿ **Development of Interactive Applications for the EVA Robot**

EvaML – XML-based language

ÿ EvaSIM - Simulator

EvaML

(An XML-Based Language for Session Specification)
Interactive with the EVA Robot)

EvaML (Objectives)

- Greater control over entering and editing commands and their parameters.
- Abstraction of programming language elements (Macros)
- Enable the creation of scripts independent of the robot's control interface.
- A language created to facilitate the development of interactive applications using the EVA robot.

EvaML

- Commands that control the robot's multimodal interaction elements are present:

  Light, userEmotion, qrRead, and **userID** components.

- Elements **for** *creating* and *manipulating* variables
- **Random** number generation
- Conditional controls using **switch** and **case** elements
- Others…

EvaML (XML Basics)

- *XML* - eXtensible Markup Language

XML documents are structured as "trees" of elements .

- An *XML* tree starts at a root element and branches out from the root. for the child elements

tutorial link https://www.w3schools.com/

EvaML (XML Basics)

- All *XML* elements must have a closing tag.

XML tags are case-sensitive .

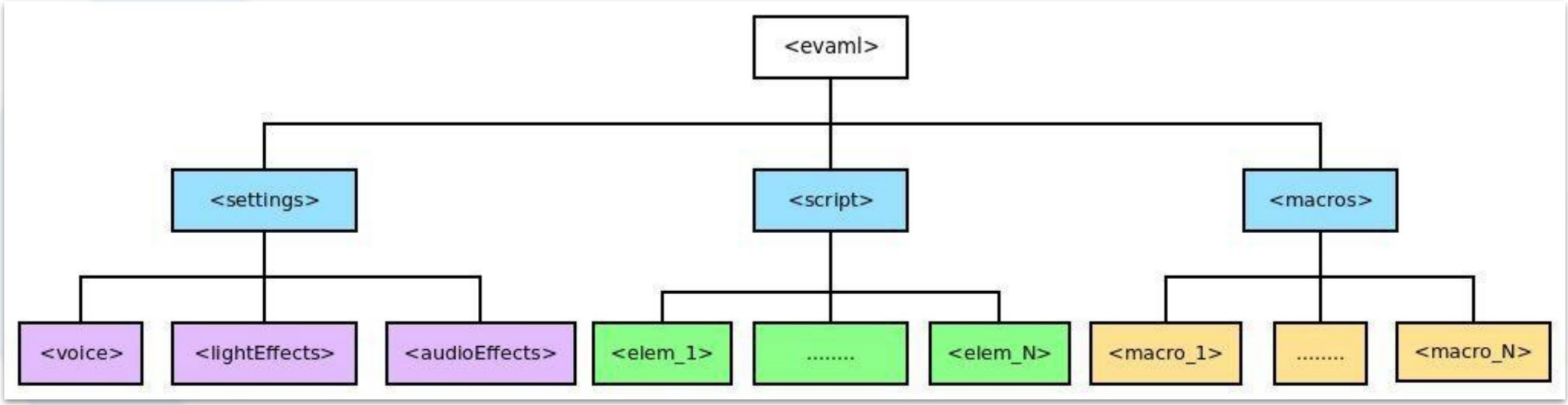
- The tag <Light> is different from the tag <light>

- Opening and closing tags should be written the same way.

  manner

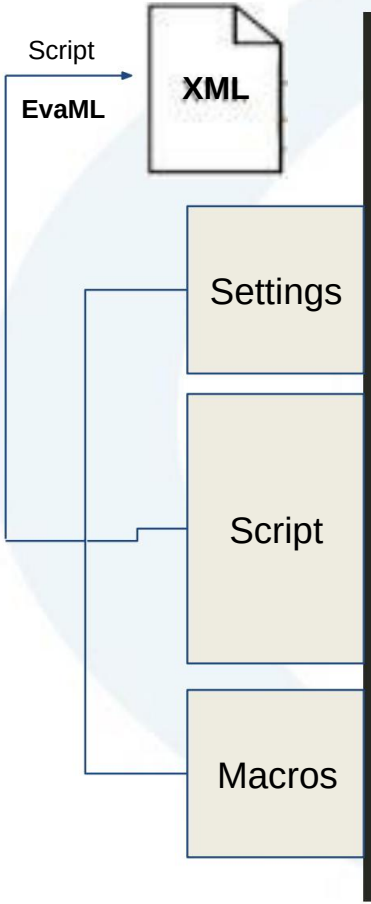
- In *XML*, the attribute values must to be between quotation marks (single) or pairs)

EvaML (Elements of an EvaML Document)



Element tree of an EvaML document

EvaML (Elements of an EvaML Document)




```xml
<?xml version="1.0" encoding="UTF-8"?>
<evaml
  name="Hackaton_Educa360"
  xsi:noNamespaceSchemaLocation="evaml-schema/evaml_schema.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <settings>
    <voice tone="pt-BR_IsabelaV3Voice" />
    <lightEffects mode="ON" />
    <audioEffects mode="ON" />
  </settings>
  <script>
    <useMacro macro="Apresenta" /> <!-- Utiliza a Macro Apresenta-->
    <light state="ON" color="BLUE"/> <!-- Acende a lâmpada inteligente e coloca na cor Azul-->
    <led animation="HAPPY" /> <!-- Acende a luz do tórax do robô com a cor referente a emoção de Felicidade-->
    <evaEmotion emotion="HAPPY" /> <!-- Faz com que no display o robô faça a expressão de feliz-->
    <audio source="efx-mario-game-intro" block="TRUE" /> <!-- Toca um áudio e aguarda o término para dar continuidade ao script-->
    <talk>Até a próxima, tchau!</talk> <!-- Se despede do usuário-->
  </script>
  <macros>
    <macro id="Apresenta">
        <talk>Olá, eu sou o robô EVA.</talk>
    </macro>
  </macros>
</evaml>
```

EvaML (Elements of an EvaML Document)

Element settings - Defines some global characteristics of the script. It is possible to define the tone of the voice and the language in which the robot will communicate. It is possible to define whether the generated code, when executed, will reproduce commands for light effects, sound effects, or even play music.

```
<settings>
  <voice tone="pt-BR_IsabelaV3Voice" />
  <lightEffects mode="ON" />
  <audioEffects mode="ON" />
</settings>
```

Example: Settings section

EvaML (Elements of an EvaML Document)

Script element - Contains the sequence of commands that the robot should execute.

```
11  <script>
12    <light state="ON" color="BLUE"/> <!-- Acende a lâmpada inteligente e coloca na cor Azul-->
13    <led animation="HAPPY" /> <!-- Acende a luz do tórax do robô com a cor referente a emoção de Felicidade-->
14    <evaEmotion emotion="HAPPY" /> <!-- Faz com que no display o robô faça a expressão de feliz-->
15    <audio source="efx-mario-game-intro" block="TRUE" /> <!-- Toca um áudio e aguarda o término para dar continuidade ao script-->
16    <talk>Até a próxima, tchau!</talk> <!-- Se despede do usuário-->
17  </script>
```

Example: Script section

EvaML (Elements of an EvaML document - Occurrence indicators)

Required attribute

Order matters!

Occurrences: 0 or 1

Elemento	Atributo	Conteúdo
evaml	name	(settings, script, macros?)
settings		(voice | lightEffects? | audioEffects?)
script		(random* | wait* | talk* | stop* | light* | goto* | userEmotion* | evaEmotion* | useMacro* | listen* | audio* | led* | counter* | switch*)
macros		(macro+)

Elements of an EvaML document (Element Content Model)

The order doesn't matter!

Occurrences: 0 or more

Creating code in EvaML

1. Open Visual Studio Code on your computer; 2. In the upper left corner, go to File, Open Folder, and open the folder. Eva-sim-2.0;
3. Open the Hackathon360.xml file; 4. Then the code below should appear on your screen.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <evaml
3    name="Hackaton360"
4    xsi:noNamespaceSchemaLocation="evaml-schema/evaml_schema.xsd"
5    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
6    <settings>
7    </settings>
8    <script>
9
10   </script>
11 </evaml>
```

Creating code in EvaML

With the environment ready to start coding, we define the settings elements.

- The language that the EVA Robot will speak is defined via the tag.
  <voice> and the tone attribute;
- The next step is to define whether you will use the light effect, through the tag.
  <lightEffects>;
- Finally, it is necessary to define whether the audio effect will be used, that is, the singing during the application sounds . For this, the tag < is used. audioEffects> .

Código	Gênero	Idioma
pt-BR_IsabelaV3Voice	feminino	português do Brasil
en-US_AllisonV3Voice	feminino	inglês dos EUA
en-US_EmilyV3Voice	feminino	inglês dos EUA
en-US_HenryV3Voice	masculino	inglês dos EUA
es-LA_SofiaV3Voice	feminino	espanhol latinoamericano
es-ES_EnriqueV3Voice	masculino	espanhol

Language options for the EVA Robot to speak

EvaML (Elements of an EvaML Document)

Once all the elements of the `<settings>` tag are defined, your code will look something like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<evaml
  name="Hackaton360"
  xsi:noNamespaceSchemaLocation="evaml-schema/evaml_schema.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <settings>
    <voice tone="pt-BR_IsabelaV3Voice" />
    <lightEffects mode="ON" />
    <audioEffects mode="ON" />
  </settings>
  <script>

  </script>
</evaml>
```

EvaML (<talk>)

By using the <talk> command , the robot can speak a specified text. When defining the text...
To be spoken, it is possible to use the contents of the robot's memory as part of the text, using it in the body.
from the text, the character $, which references a special area of the robot's memory, or the notation **#var** that
It refers to the content of a user-defined variable.

Element	Attribute	Content
talk	id	text

```
<script>
  <talk>Qual é o seu nome?</talk>
  <listen var="nome"/>
  <talk>Olá #nome, prazer em te conhecer</talk>
  <talk>Diga um número </talk>
  <listen />
  <talk>Você falou o número $</talk>
</script>
```

Example: talk command

EvaML (<listen>)

The robot can recognize the human voice and uses a speech-to-text conversion service to do so. (STT) from the Google Cloud API . The captured audio is sent to the cloud, processed, and then... The text resulting from the STT process is returned for use by the robot's software.

Element	Attribute	Content
listen	id, var	empty

```
<script>
  <talk>Qual é o seu nome?</talk>
  <listen var="nome"/>
  <talk>Olá #nome, prazer em te conhecer</talk>
  <talk>Diga um número </talk>
  <listen />
  <talk>Você falou o número $</talk>
</script>
```

Example: Listen command

EvaML (<evaEmotion>)

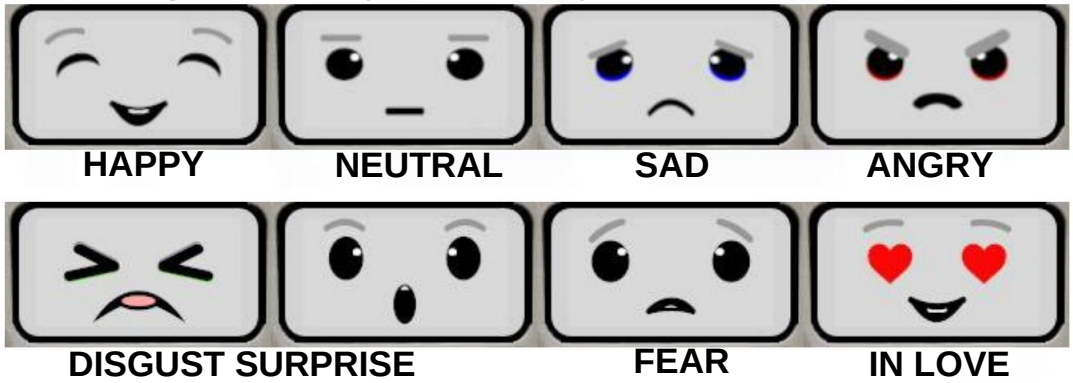
Controls the display of eye expressions on the 5.5” display. Its **emotion** attribute can have the following values: **"HAPPY", "SAD", "ANGRY", "NEUTRAL", "FEAR", "DISGUST", "SURPRISE"** and **"IN LOVE" .**

Element	Attribute	Content
evaEmotion	id, emotion	empty

```
16    <evaEmotion emotion="NEUTRAL" />
17    <evaEmotion emotion="HAPPY" />
18    <evaEmotion emotion="SAD" />
19    <evaEmotion emotion="ANGRY" />
20    <evaEmotion emotion="DISGUST" />
21    <evaEmotion emotion="SURPRISE" />
22    <evaEmotion emotion="FEAR" />
23    <evaEmotion emotion="INLOVE" />
```

Example: evaEmotion command

Expressions synthesized by the EVA robot.



Creating code in EvaML

Now that we know how to use talk, listen, and evaEmotion, let's add another snippet to our code?

• Initialize the script by assigning an emotion to the robot; • Create a sentence in which the robot introduces itself to the user; • Ask the user for their name; • Use the <listen> command and store the user's response in a... **var;**
• Set the robot's emotion to "happy" for learning the user's name; • Greet the user; • Set the robot's emotion to "neutral";

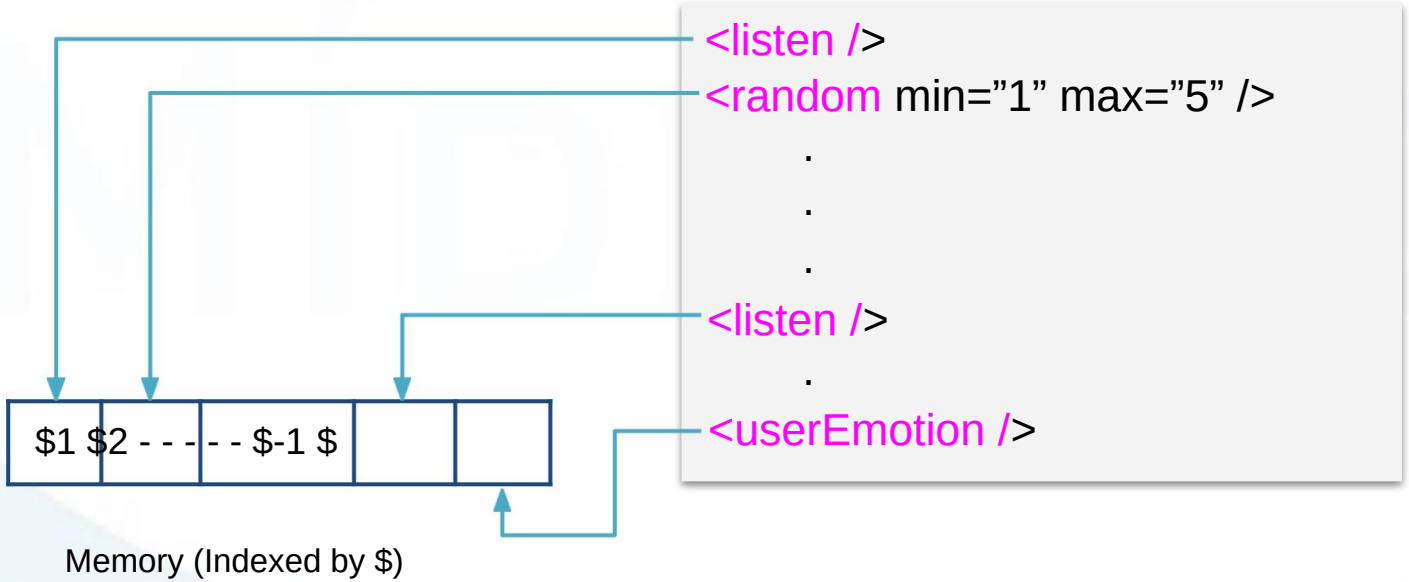
Creating code in EvaML

After completing these 7 steps, we will have code similar to this:

```
<?xml version="1.0" encoding="UTF-8"?>
<evaml
  name="Hackaton360"
  xsi:noNamespaceSchemaLocation="evaml-schema/evaml_schema.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <settings>
    <voice tone="pt-BR_IsabelaV3Voice" />
    <lightEffects mode="ON" />
    <audioEffects mode="ON" />
  </settings>
  <script>
    <evaEmotion emotion="NEUTRAL" />
    <talk>Olá, eu sou o robô EVA.</talk>
    <talk>Qual é o seu nome?</talk>
    <listen var="nome"/>
    <evaEmotion emotion="HAPPY" />
    <talk>Olá #nome, prazer em te conhecer</talk>
    <evaEmotion emotion="NEUTRAL" />
  </script>
</evaml>
```

EvaML (Accessing robot memory using the **$ character)**

The $ character , by convention, refers to the value at the end of the list in memory that stores the values. Responses returned by user interaction commands include: <listen>, <userEmotion>, <userHandPose>, <qrRead>, , <userID> , and the values generated by the <random> command.



EvaML (<wait>)

This command pauses script execution for the time interval defined in its attribute. **duration.** The unit of time used is the millisecond, that is, assigning the value 1000 to the **duration ,** This means pausing the script for 1 second.

Element	Attribute	Content
wait	ID, duration	empty

```
16      <talk>Eu consigo pausar a execução por 1 segundo </talk>
17      <wait duration="1000" />
18      <talk>Pronto</talk>
```

Example: Wait command

EvaML (<stop>)

This command is very simple and, as the name suggests, it interrupts the execution of the script.

Element	Attribute	Content
stop		empty

EvaML (<audio>)

Plays an audio file located in the *"sounds" folder.* The command has the attributes **source** and

The `source` attribute indicates the name of the file to be played. EVA only plays files in the format ...

The `block` attribute can have the following values, **"TRUE"** or **"FALSE" ,** and defines whether the execution of the file will proceed . The file must block the execution of the script; that is, the command that comes after the `<audio>` command should only...

It will run after the audio playback finishes.

Element	Attribute	Content
audio	id, source, block	empty

```
16    <light state="ON" color="PINK" />
17    <audio source="efx-mario-game-intro" block="TRUE" />
18    <light state="OFF" />
```

Example: Audio command

EvaML (<light>)

Controls the smart bulb. Its **state** attribute can take the values **"ON"** and **"OFF"** and the
The **`color`** attribute defines the lamp's color. This color can be indicated using the representation...
hexadecimal RGB "#00ff00" or one of the colors from the predefined list: "WHITE", "BLACK", "RED", "PINK",
"GREEN", "YELLOW", "BLUE".

Element	Attribute	Content
light	id, state, color	empty

```
16      <light state="ON" color="PINK" />
17      <wait duration="2000" /> <!-- Pausa a execução do script por dois segundos-->
18      <light state="ON" color="BLUE"/>
19      <light state="OFF" />
20      <stop /> <!-- Interrompe a exeução do script-->
21      <light state="ON" color="RED"/>
```

Example: Light command

Creating code in EvaML

Now that a few more commands have been presented, use your creativity and incorporate them into your code.

5 MINUTES TO BUILD THE CODE

EvaML (<motion>)

The robot can move its head, and the <motion> command is responsible for controlling this. movement. It has the attribute **type** which can take the following values: **"YES", "NO", "CENTER", "LEFT", "RIGHT", "UP", "DOWN", "ANGRY", "2UP", "2DOWN", "2LEFT"** and **"2RIGHT".**

Element	Attribute	Content
motion	id, type	empty

```
16      <talk>Eu consigo realizar alguns movimentos com a minha cabeça</talk>
17      <talk>Farei o movimento de sim</talk>
18      <motion type="YES"/>
19      <wait duration="2000" />
20      <talk>Agora farei o movimento de não</talk>
21      <motion type="NO"/>
```

Example: Motion command

EvaML (<counter>)

Creates, initializes, and performs mathematical operations on variables. The `var` attribute defines the name. variable, while the **op** attribute defines the type of operation, and can take the following values: **"="** (assignment), **"+"** (addition), **"*"** (multiplication), **"/"** (division), and **"%"** (modulo). To reference the value of A user-created variable, unlike the "$" character, requires the use of the **"#"** character. before the variable name. The **value** attribute determines the value used in the specified operation.

Element	Attribute	Content
counter	id, var, op, value	empty

```
17    <counter var="x" op="=" value="10" />
18    <counter var="x" op="+" value="10" />
19    <counter var="x" op="/" value="10" />
20    <counter var="x" op="*" value="10" />
21    <counter var="x" op="%" value="10" />
22    <talk>O valor de x é #x.</talk>
```

Example: Counter command

EvaML (<led>)

Control the animation using the LEDs on the EVA robot's chest. Its **animation** attribute can assume the following...

The following values: **"HAPPY"** (green), **"SAD"** (blue), **"ANGRY"** (red), **"STOP"** (no color/off), **"SPEAK"** (blue), **"LISTEN"** (green), and **"SURPRISE"** (yellow).

Element	Attribute	Content
LED	id, animation	empty

```
17    <led animation="HAPPY" />
18    <led animation="ANGRY" />
19    <led animation="SAD" />
```

Example: LED command

EvaML (<switch>)

Defines the variable that will be compared with the values defined in the <case> commands. For this, its **var** attribute must contain the name of the variable to be compared, for example: <switch var="$"> or <switch var="x">. The **var** attribute of the <switch> command determines which variable will be used for comparisons in <case> statements , and it can assume the following: Values **"$"** or any other variable name that has been declared previously.

Element	Attribute	Content
switch	id, ~~var~~	(case+, default?)

EvaML (<case>)

The `<case>` statement specifies a sequence of commands that will be executed if the condition defined in its attributes is true. The ` <case> ` statement has the **`op` attribute,** which defines the type of comparison or logical operator that will be processed, and the **`value` attribute,** which contains the value to be compared with the variable defined in the **`var`** attribute of the ` <switch>` statement.

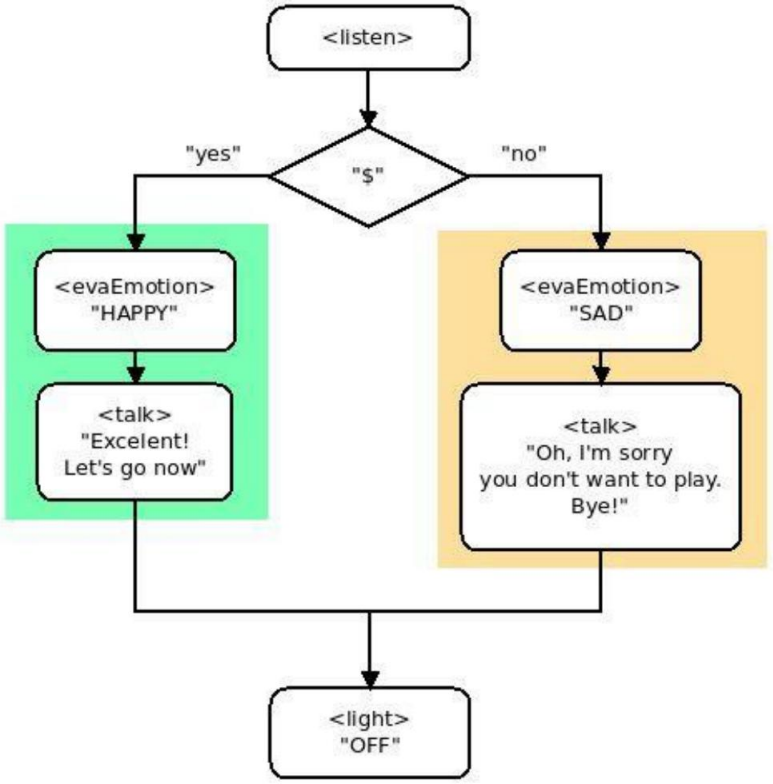
case	op, value	(random* | wait* | talk* | stop* | light* | goto* | userEmotion* | userHandPose* | userID* | qrRead* | loop* | evaEmotion* | listen* | audio* | led* | counter* | switch* | useMacro*)

EvaML (<case>)

Attribute **var** of <switch>	Type of comparison (Attribute **op** of <case>)	Value type in the **value** attribute of <case>	Description
$	**exact**	string	Exact comparison between two strings. Not case sensitive.
	contain	string	Checks if **"$"** contains the string in "value". It is not case sensitive.
$ or "variable"	**math** ("eq", "lt", "gt", "lte", "gte", "ne")	A numeric constant , **$** or **#var.**	Numerical comparison between **"$"** or another variable, with a value *"constant",* the $, or another variable contained in the **value** attribute of <case>. Only works with whole numbers.

Types of comparison and logical operations of the <case> command

EvaML (<case>)



Example flowchart

```
<script>
<listen />
<switch var=”$”>
   <case op=”exact” value=”yes”>
      <evaEmotion emotion=”HAPPY” />
      <talk>”Excellent! Let's go now”</talk> </case>
   <case
   op=”exact” value=”no”>
      <evaEmotion emotion=”SAD” />
      <talk>”Oh, I'm sorry you want to play. Bye!”</talk> </case> </
   switch>
<light
state=”OFF” /> </script>
```

Example: Switch and case commands

Creating code in EvaML

Now that all the commands have been presented, try it.

Use them in your code to create an interactive game between the EVA robot and the user.

Suggestion:

- **A game where the robot lights up the lamp and asks what color it was.**
  lit;
- **A game where the robot displays an emotion and asks what emotion it displayed;**
- **A game where the robot says a phrase and asks the user to repeat it.**

Try using switches, cases, and audio to make the game more interactive.

Creating code in EvaML

Final code example:

```
<?xml version="1.0" encoding="UTF-8"?>
<evaml
  name="Hackaton360"
  xsi:noNamespaceSchemaLocation="evaml-schema/evaml_schema.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <settings>
    <voice tone="pt-BR_IsabelaV3Voice" />
    <lightEffects mode="ON" />
    <audioEffects mode="ON" />
  </settings>
  <script>
    <evaEmotion emotion="NEUTRAL" />
    <talk>Olá, eu sou o robô EVA.</talk>
    <talk>Qual é o seu nome?</talk>
    <listen var="nome"/>
    <evaEmotion emotion="HAPPY" />
    <talk>Olá #nome, prazer em te conhecer</talk>
    <evaEmotion emotion="NEUTRAL" />
    <talk>Vou tocar uma música e acender a lâmpada.</talk>
    <audio source="song-caneta-azul" block="TRUE" />
    <light state="ON" color="YELLOW"/>
    <useMacro macro="brincadeira" />
  </script>
```

Creating code in EvaML

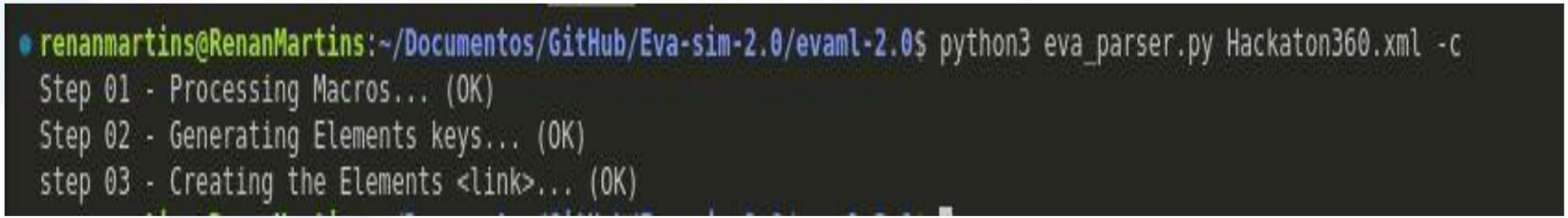
- After creating the code, perform the following steps:
- Open a terminal in VSCode via the “Terminal->New” menu.

  Terminal"

- Run the *Parser* on the EvaML code:

  python3 eva_parser.py Hackaton360.xml -c

- The terminal should display the following message:



EvaSIM

(EVA Robot Simulator Software)

EvaSIM (Graphical User Interface)

After completing all the previous steps, the next step is to test our application in the robot simulator. To do this, we will perform the following steps:

1 - In the terminal, type the following command: cd evasim

```
renanmartins@RenanMartins:~/Documentos/GitHub/Eva-sim-2.0/evaml-2.0$ cd evasim
renanmartins@RenanMartins:~/Documentos/GitHub/Eva-sim-2.0/evaml-2.0/evasim$
```

2 - The next step is to type the command: python3 eva_sim.py

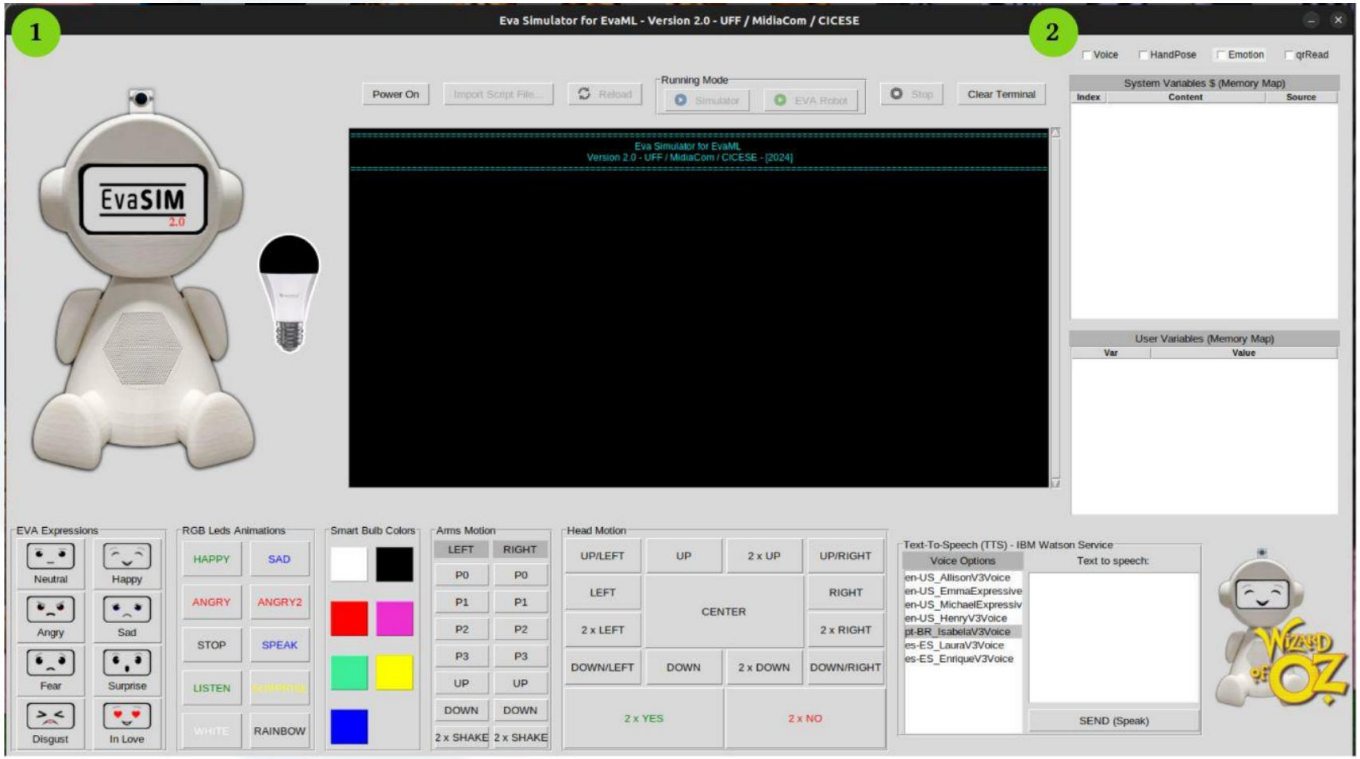
```
renanmartins@RenanMartins:~/Documentos/GitHub/Eva-sim-2.0/evaml-2.0/evasim$ python3 eva_sim.py

Linux platform identified. Loading GUI formatting for Linux.

A fake mqtt client was created!
```

3 - Then the simulator will open.

EvaSIM (Graphical User Interface)



EvaSIM Simulator

EvaSIM (Graphical User Interface)

Indication on the terminal, the movement of the robot's head.

EvaSIM (Graphical User Interface)

To run the file, follow these steps:

1. Click the **“Import script file”** button.
2. Select the file **“Hackaton360_EvaML.xml”**
3. Finally, select the **“Simulator”** button and see your code in action.

Thanks!

Renan Martins CS de Lima - renanmali@midiacom.uff.br
Marcelo Marques da Rocha - marcelo_rocha@midiacom.uff.br
Débora Christina Muchaluat Saade - debora@midiacom.uff.br
João Cecim - joaocecim@id.uff.br
Pedro Lucas - pl_silva@id.uff.br
Iasmim Santos - iasmim17santoss@gmail.com
Juliana Arraes - julianaatds@id.uff.br