Aprendendo a desenvolver Aplicações Interativas com o Robô EVA

Renan Martins C. S. de Lima
Marcelo Marques da Rocha
Débora Christina Muchaluat Saade

Roteiro

- **Desenvolvimento de Aplicações Interativas para o Robô EVA**
  - EvaML – Linguagem baseada em XML
  - EvaSIM - Simulador

EvaML

(Uma Linguagem Baseada em XML para a Especificação de Sessões Interativas com o Robô EVA)

EvaML (Objetivos)

- Maior controle na entrada e edição dos comandos e seus parâmetros
- Abstração de elementos de linguagens de programação (Macros)
- Possibilitar a criação de scripts independente da interface de controle do robô
- Linguagem criada para facilitar o desenvolvimentos de aplicações interativas utilizando o robô EVA.

EvaML

- Comandos que controlam os elementos de interação multimodal do robô estão presentes: componentes **Light**, **userEmotion, qrRead,** e **userID**
- Elementos para *criação* e *manipulação* de **variáveis**
- Geração de números aleatórios **random**
- Controles condicionais usando elementos **switch** e **case**
- Outros...

EvaML (Noções básicas sobre XML)

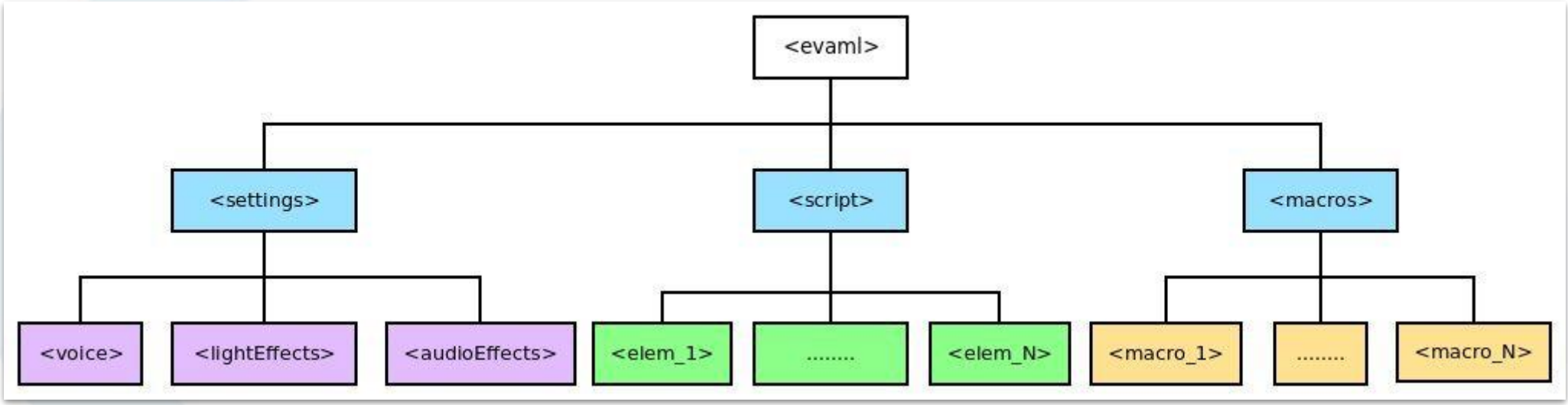
- *XML* - eXtensible Markup Language
- Documentos *XML* são formados como “árvores” de elementos
- Uma árvore *XML* começa em um elemento raiz e se ramifica da raiz para os elementos filhos

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

EvaML (Noções básicas sobre XML)

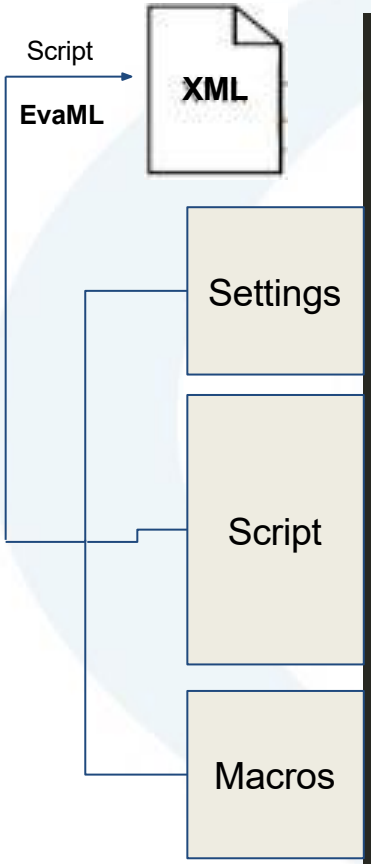
- Todos os elementos *XML* devem ter uma tag de fechamento
- As tags *XML* diferenciam letras maiúsculas de minúsculas
- A tag <Light> é diferente da tag <light>
- As tags de abertura e fechamento devem ser escritas da mesma maneira
- Em *XML*, os valores dos atributos devem estar entre aspas (simples ou duplas)

EvaML (Elementos de um Documento EvaML)



Árvore de elementos de um documento EvaML

EvaML (Elementos de um Documento EvaML)




```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 (Elementos de um Documento EvaML)

Elemento settings - Define algumas características globais do script. É possível definir como será o timbre da voz e o idioma em que o robô irá se comunicar. É possível definir se o código gerado, ao ser executado, irá reproduzir comandos de efeitos de luz, efeitos sonoros ou até mesmo tocar música.

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

Exemplo: Seção settings

EvaML (Elementos de um Documento EvaML)

Elemento script - Contém a sequência de comandos que o robô deve executar.

```
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>
```

Exemplo: Seção script

EvaML (Elementos de um doc. EvaML - Indicadores de ocorrência)

Atributo obrigatório

A ordem importa!

Ocorrências: 0 ou 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+)

Elementos de um documento EvaML (Modelo de conteúdo dos Elementos)

A ordem NÃO importa!

Ocorrências: 0 ou mais

Criando um código em EvaML

1. Abra o Visual Studio Code em seu computador;
2. No canto superior esquerdo vá em arquivo, abrir pasta e abra a pasta Eva-sim-2.0;
3. Abra o arquivo Hackaton360.xml;
4. Então deverá aparecer em sua tela este código abaixo

```
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>
```

Criando um código em EvaML

Com o ambiente pronto para começar o código, definimos os elementos do settings

- Define-se qual idioma será dito pelo Robô EVA, através da tag <voice> e atributo tone;
- O próximo passo é definir se irá usar o efeito de luz, através da tag <lightEffects>;
- Por último, definir se será utilizado o efeito de audio, ou seja, sons durante a aplicação. Para isso, utiliza-se a tag <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

Opções de idioma a ser dito pelo Robô EVA

EvaML (Elementos de um Documento EvaML)

Definidos todos os elementos do <settings>, seu código ficará parecido com este aqui:

```
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      <voice tone="pt-BR_IsabelaV3Voice" />
8      <lightEffects mode="ON" />
9      <audioEffects mode="ON" />
10   </settings>
11   <script>
12
13   </script>
14 </evaml>
```

EvaML (<talk>)

Através do uso do comando <talk> o robô pode falar um texto especificado. Ao se definir o texto a ser falado é possível utilizar o conteúdo da memória do robô como parte do texto, utilizando-se, no corpo do texto, o caractere **$**, que referencia uma área especial da memória do robô ou a notação **#var** que referencia o conteúdo de uma variável definida pelo usuário.

Elemento	Atributo	Conteúdo
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>
```

Exemplo: Comando talk

EvaML (<listen>)

O robô pode reconhecer a voz humana e para isso utiliza o serviço de conversão de fala para texto (STT) da API na nuvem do *Google*. O áudio capturado é enviado para a nuvem, processado, e então, o texto resultante do processo de STT é retornado para ser utilizado pelo software do robô.

Elemento	Atributo	Conteúdo
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>
```

Exemplo: Comando listen

EvaML (<evaEmotion>)

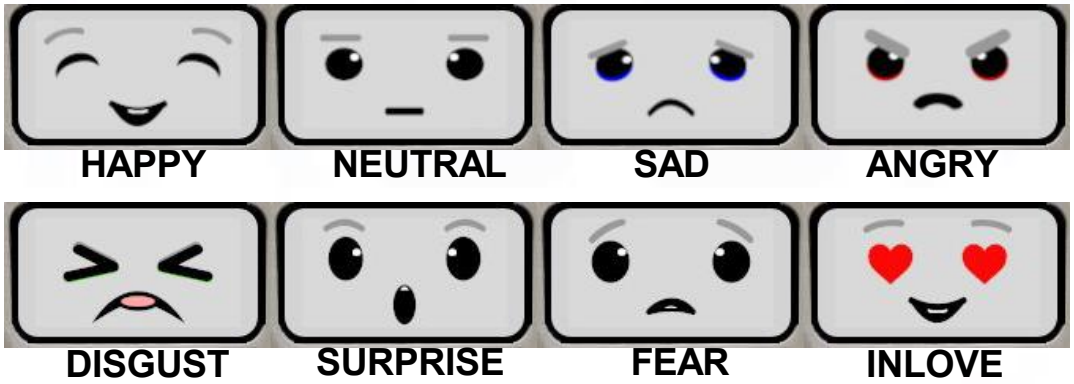
Controla a apresentação das expressões do olhar no display de 5.5”. O seu atributo **emotion** pode ter os seguintes valores: "**HAPPY**", "**SAD**", "**ANGRY**", "**NEUTRAL**", "**FEAR**", "**DISGUST**", "**SURPRISE**" e "**INLOVE”** .

Elemento	Atributo	Conteúdo
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" />
```

Exemplo: Comando evaEmotion

Expressões sintetizadas pelo robô EVA



Criando um código em EvaML

Agora que já sabemos como usar o talk, o listen e o evaEmotion, vamos adicionar mais um trecho em nosso código?

- Inicialize o script definindo uma emoção ao robô;
- Crie uma frase em que o robô se apresenta ao usuário;
- Pergunte o nome do usuário;
- Utilize o comando <listen> e armazene a resposta do usuário em um **var;**
- Defina uma emoção de alegria ao robô por saber o nome do usuário;
- Cumprimente o usuário;
- Coloque a emoção neutra no robô;

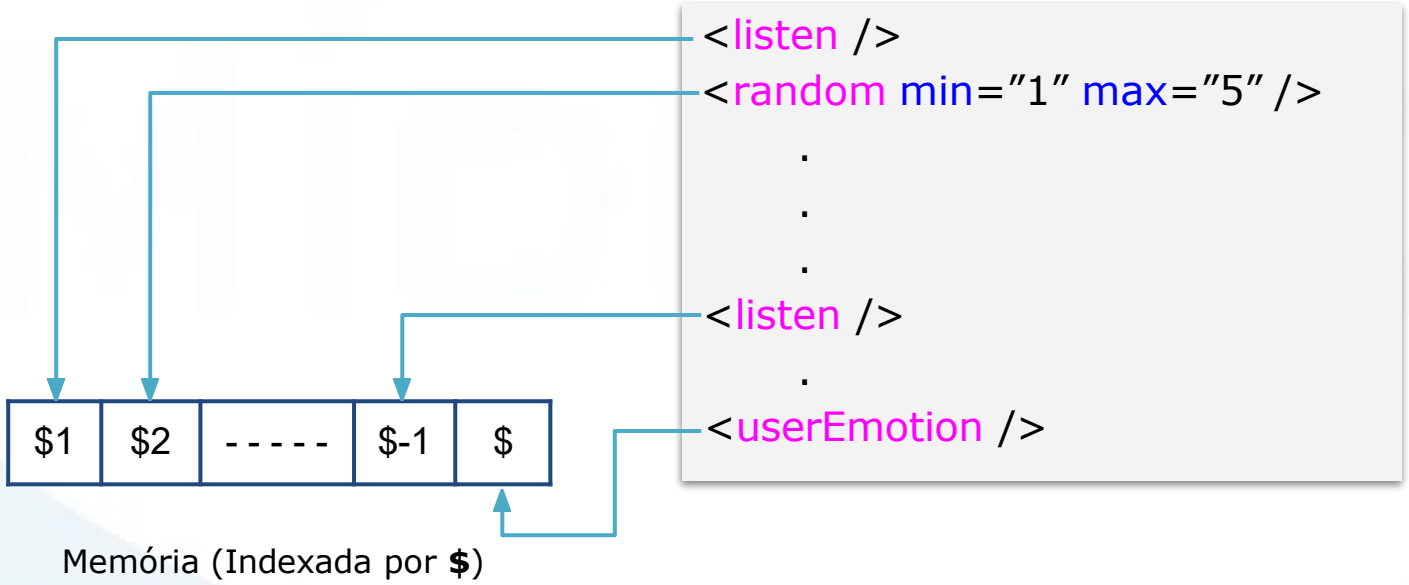
Criando um código em EvaML

Após a conclusão desses 7 passos, teremos o código parecido com este:

```
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      <voice tone="pt-BR_IsabelaV3Voice" />
8      <lightEffects mode="ON" />
9      <audioEffects mode="ON" />
10   </settings>
11   <script>
12     <evaEmotion emotion="NEUTRAL" />
13     <talk>Olá, eu sou o robô EVA.</talk>
14     <talk>Qual é o seu nome?</talk>
15     <listen var="nome"/>
16     <evaEmotion emotion="HAPPY" />
17     <talk>Olá #nome, prazer em te conhecer</talk>
18     <evaEmotion emotion="NEUTRAL" />
19   </script>
20 </evaml>
```

EvaML (Acessando a memória do robô usando o caractere **$**)

O caractere **$**, por convenção, faz referência ao valor no final da lista na memória que armazena as respostas retornadas pelos comandos de interação com o usuário, são eles: <listen>, <userEmotion> , <userHandPose>, <qrRead>, <userID> e, os valores gerados pelo comando <random>.



EvaML (<wait>)

Este comando pausa a execução do script pelo intervalo de tempo definido no seu atributo **duration**. A unidade de tempo usada é o milissegundo, ou seja, atribuindo ao **duration** o valor 1000, significa pausar o script durante 1 segundo.

Elemento	Atributo	Conteúdo
wait	id, duration	empty

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

Exemplo: Comando wait

EvaML (<stop>)

Este comando é muito simples e como o nome sugere, ele interrompe a execução do script.

Elemento	Atributo	Conteúdo
stop		empty

EvaML (<audio>)

Reproduz um arquivo de áudio contido na pasta "*sonidos*". O comando possui os atributos **source** e **block**. O atributo **source** indica o nome do arquivo a ser tocado. O EVA só reproduz arquivos no formato *.wav*. O atributo **block** pode ter os seguintes valores, "**TRUE**" ou "**FALSE**" e define se a execução do arquivo deve bloquear a execução do script, ou seja, o comando que vem após o comando <audio> só será executado após o término da reprodução do áudio.

Elemento	Atributo	Conteúdo
audio	id, source, block	empty

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

Exemplo: Comando audio

EvaML (<light>)

Controla a lâmpada inteligente. O seu atributo **state** pode assumir os valores "**ON**" e "**OFF**" e o atributo **color** define a cor da lâmpada. Essa cor pode ser indicada usando-se a representação hexadecimal RGB "#00ff00" ou alguma das cores da lista predefinida: "WHITE", "BLACK", "RED", "PINK", "GREEN", "YELLOW", "BLUE".

Elemento	Atributo	Conteúdo
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"/>
```

Exemplo: Comando light

Criando um código em EvaML

Agora que foram apresentados mais alguns comandos, utilize sua criatividade e os insira em seu código

5 MINUTOS PARA A CONSTRUÇÃO DO CÓDIGO

EvaML (<motion>)

O robô pode mover sua cabeça e o comando <motion> é responsável por controlar este movimento. Ele possui o atributo **type** que pode assumir os seguintes valores: "**YES**", "**NO**", "**CENTER**", "**LEFT**", "**RIGHT**", "**UP**", "**DOWN**", "**ANGRY**", "**2UP**", "**2DOWN**", "**2LEFT**" e "**2RIGHT**".

Elemento	Atributo	Conteúdo
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"/>
```

Exemplo: Comando motion

EvaML (<counter>)

Cria, inicializa e realiza operações matemáticas sobre variáveis. O atributo **var** define o nome variável, enquanto o atributo **op** define o tipo de operação, podendo assumir os seguintes valores: "**=**" (atribuição), "**+**" (adição), "*****" (multiplicação), "**/**" (divisão) e "**%**" (módulo). Para referenciar o valor de uma variável criada pelo usuário, diferentemente do caractere "**$**", é necessário utilizar o caractere "**#**" antes do nome da variável. O atributo **value** determina o valor usado na operação especificada.

Elemento	Atributo	Conteúdo
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>
```

Exemplo: Comando counter

EvaML (<led>)

Controla a animação com os LEDs no peito do robô EVA. O seu atributo **animation** pode assumir os seguintes valores: "**HAPPY**" (verde), "**SAD**" (azul), "**ANGRY**" (vermelho), "**STOP**" (sem cor/desligado), "**SPEAK**" (azul), "**LISTEN**" (verde) e "**SURPRISE**" (amarelo).

Elemento	Atributo	Conteúdo
led	id, animation	empty

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

Exemplo: Comando led

EvaML (<switch>)

Define a variável que será comparada com os valores definidos nos comandos <case>. Para isso, seu atributo **var** deve conter o nome da variável a ser comparada, por exemplo: <switch **var**="$"> ou <switch **var**="x">. O atributo **var** do comando <switch> determina com qual variável as comparações serão feitas nos comandos <case> e pode assumir os valores "**$**" ou qualquer outro nome de variável que tenha sido declarada anteriormente.

Elemento	Atributo	Conteúdo
switch	id, var	(case+, default?)

EvaML (<case>)

O comando <case> especifica uma sequência de comandos que serão executados se a condição definida em seus atributos for verdadeira. O comando <case> possui o atributo **op**, que define o tipo de comparação ou operador lógico que será processado, e o atributo **value**, que contém o valor a ser comparado com a variável definida no atributo **var** do comando <switch>.

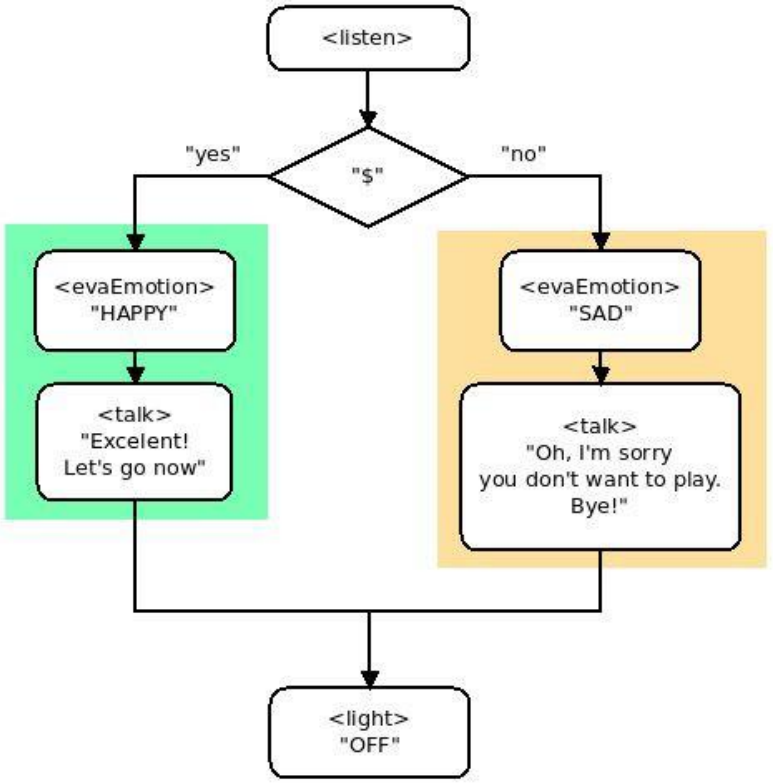
case	op, value	(random* | wait* | talk* | stop* | light* | goto* | userEmotion* | userHandPose* | userID* | qrRead* | loop* | evaEmotion* | listen* | audio* | led* | counter* | switch* | useMacro*)

EvaML (<case>)

Atributo **var** do <switch>	Tipo de comparação (Atributo **op** do <case>)	Tipo do valor no atributo **value** do <case>	Descrição
$	**exact**	string	Comparação exata entre duas strings. Não é case sensitive.
	contain	string	Verifica se "**$**" contém a string em "value". Não é case sensitive.
$ ou "variável"	**math** ("eq", "lt", "gt", "lte", "gte", "ne")	Uma constante numérica, **$** ou **#var**.	Comparação numérica entre "**$**" ou outra variável, com um valor "*constante*", o **$**, ou outra variável contida no atributo **value** de <case>. Somente trabalha com números inteiros.

Tipos de comparação e operações lógicas do comando <case>

EvaML (<case>)



Fluxograma de exemplo

```
<script>
<listen />
<switch var="$">
  <case op="exact" value="yes">
    <evaEmotion emotion="HAPPY" />
    <talk>"Excelent! 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>
```

Exemplo: Comandos switch e case

Criando um código em EvaML

Agora que foram apresentados todos os comandos, tente utilizá-los em seu código criando um jogo interativo entre o robô EVA e o usuário.

Sugestão:

- **Jogo onde o robô acende a lâmpada e pergunta qual a cor foi acesa;**
- **Jogo onde o robô faz uma emoção e pergunta qual foi a emoção feita;**
- **Jogo onde o robô diz uma frase e pede para o usuário repetir.**

Tente usar switch, case, audio para deixar o jogo mais interativo.

Criando um código em EvaML

Exemplo de código final:

```
<?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>
```

Criando um código em EvaML

- Após a criação do código, execute os seguintes passos:
- Abra um terminal no VSCode através do menu "Terminal->Novo Terminal"
- Execute o *Parser* sobre o código EvaML:

  python3 eva_parser.py Hackaton360.xml -c

- O terminal deve exibir a seguinte mensagem:

```
renanmartins@RenanMartins:~/Documentos/GitHub/Eva-sim-2.0/evaml-2.0$ python3 eva_parser.py Hackaton360.xml -c
Step 01 - Processing Macros... (OK)
Step 02 - Generating Elements keys... (OK)
step 03 - Creating the Elements <link>... (OK)
```

EvaSIM

(O Software Simulador do Robô EVA)

EvaSIM (Interface Gráfica de Usuário)

Após a conclusão de todas as etapas anteriores, o próximo passo é testar nossa aplicação no simulador do robô. Para isso, realizaremos os seguintes passos:

1 - No terminal, escreva o seguinte comando: cd evasim

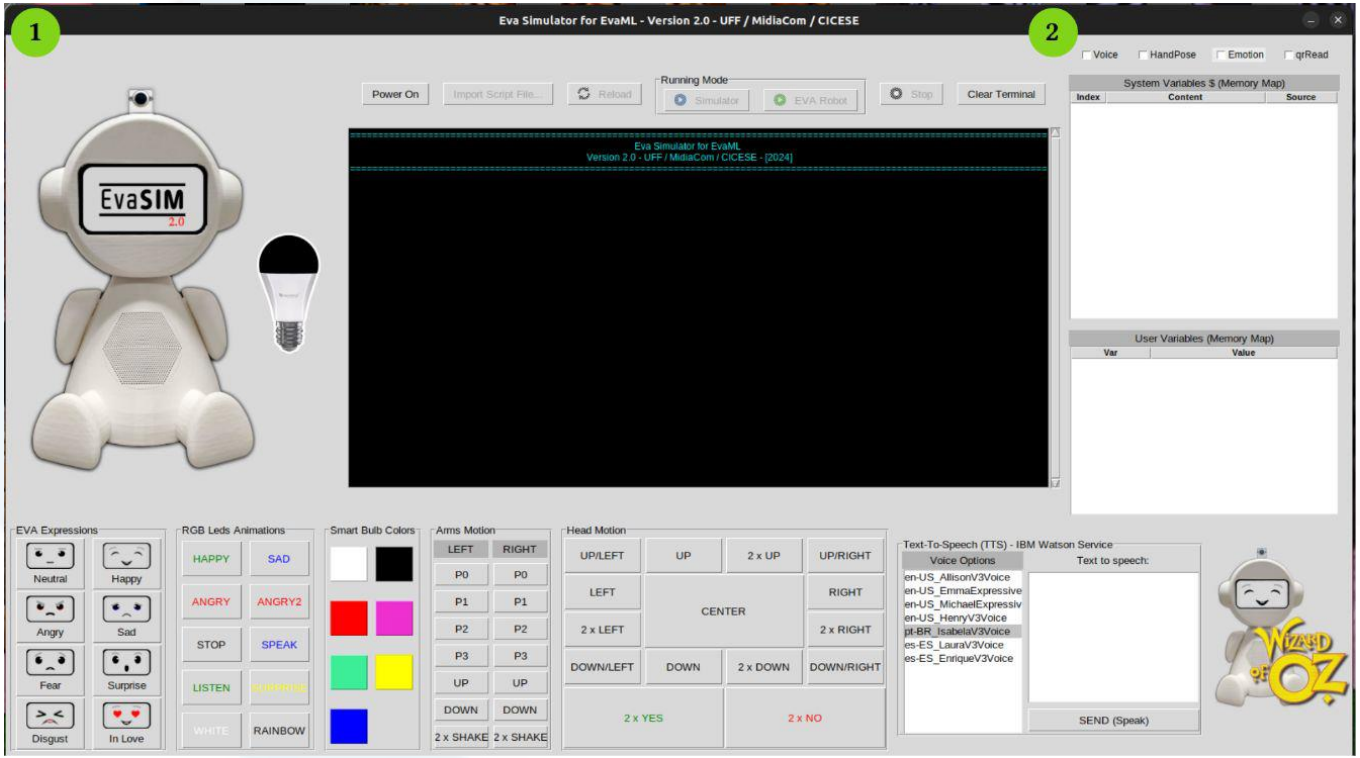
```
renanmartins@RenanMartins:~/Documentos/GitHub/Eva-sim-2.0/evaml-2.0$ cd evasim
renanmartins@RenanMartins:~/Documentos/GitHub/Eva-sim-2.0/evaml-2.0/evasim$
```

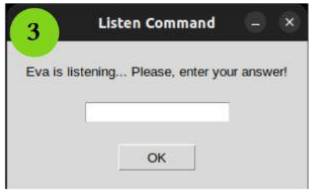
2 - O próximo passo é digitar o comando: python3 eva_sim.py

```
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 - Então será aberto o simulador

EvaSIM (Interface Gráfica de Usuário)





Simulador EvaSIM

EvaSIM (Interface Gráfica de Usuário)

```
state: Matrix Leds. Animation=STOP
state: Moving the head! Movement type: CENTER
state: Pausing. Duration=1000 ms
state: Matrix Leds. Animation=SURPRISE
state: Moving the head! Movement type: 2RIGHT
state: End of script.
```

Indicação no terminal, o movimento da cabeça do robô

EvaSIM (Interface Gráfica de Usuário)

Para executar o arquivo, faça os seguintes passos:

1. Clique no botão “**Import script file**”
2. Selecione o arquivo “**Hackaton360_EvaML.xml**”
3. Por último selecione o botão “**Simulator”** e veja seu código funcionando

Obrigado!

Renan Martins C. S. 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