EARS-CTRL: The Demonstration

Levi Lúcio[1], Salman Rahman[1], Saad Bin Abid[1], and Alistair Mavin[2]

[1] fortiss GmbH
Guerickestraße 25, 80805 München
{lucio,abid}@fortiss.org, salman.rahman@tum.de
[2] Rolls-Royce, PO Box 31, Derby, UK
alistair.mavin@rolls-royce.com

Abstract. In this part of the paper, we demonstrate our tool in practice with the example case study of a sliding door controller. We build controller specification in natural language, automatically generate the controller and then perform verification of the generated controller by utilizing testing and simulation techniques. The demo video can be found at: [URL: youtubevideo]

1 Introduction

In this demonstration, we focus on providing a guidence to the users of our tool. Our EARS-CTRL tool is a github project [1]. An overview of the demonstration steps is as follows,

- Progressively build a set of requirements for the controller written in english like sentences using EARS lanaguage [3]
- Automatic realisation of the controller by analyzing if the specification is complete (our one click approach)
- Simulation and test cases for performing conformance analysis between the EARS-based specification and the generated controller.

2 The Running Example: Automatic Sliding Door

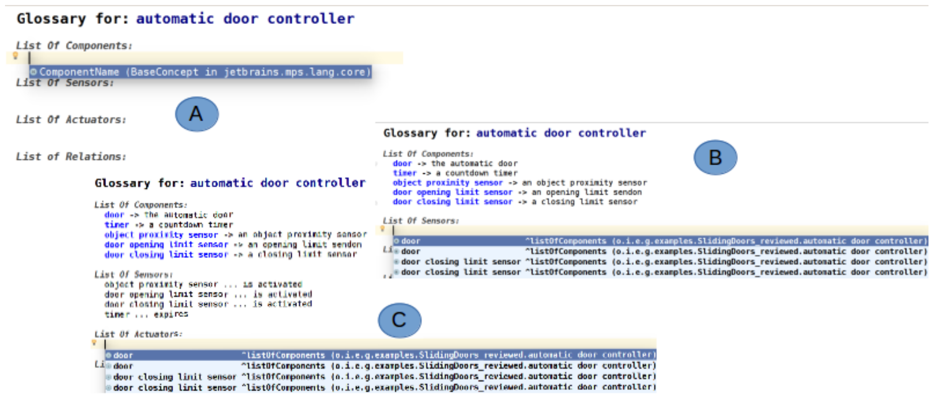
Our running example for this demo is a sliding door system of PLC based controller. The behavior of the sliding door controller is supposed to be as follows: the sliding door opens if somebody enters by sensing the object using the infrared sensor. The sliding door opens until the opening limits are reached (also detected by the sensor). Upon reaching the opening limits the count down timer starts. and the controller then closes the door when the timer expires until the closing limit is reached detected by sensor.

3 Tool Demonstration

Discuss the main steps of process that needs to be followed for synthezing and performing the conformance analysis,that are,

3.1 Glossary building and terms definition

The first step towards writing the controller specifications as a natural language in EARS-CTRL is to define the glossary terms. The user is provided with an editor (figure 1) to define glossary terms for the following, 1) components that interface with the controller, 2) sensors and actuators those components make available to the controller and 3) rules between signals definitions.



```
Glossary for: automatic door controller

List Of Components:

  ComponentName (BaseConcept in jetbrains.mps.lang.core)
List Of Sensors:

List Of Actuators:

List of Relations:
```

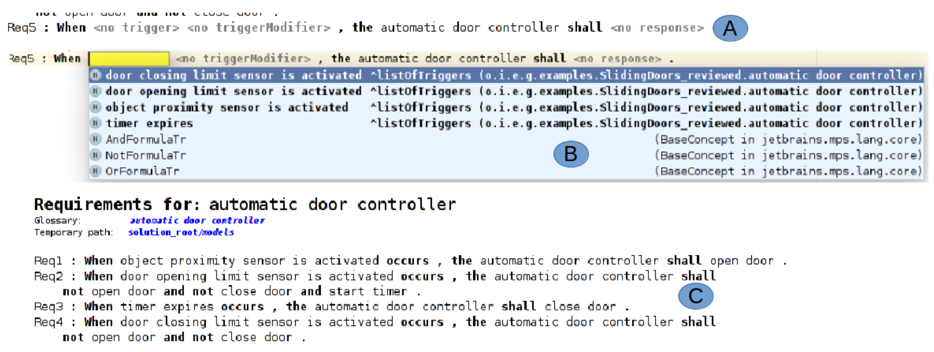
```
Glossary for: automatic door controller

List Of Components:
  door -> the automatic door
  timer -> a countdown timer
  object proximity sensor -> an object proximity sensor
  door opening limit sensor -> an opening limit sendon
  door closing limit sensor -> a closing limit sensor

List Of Sensors:

  door                      ^listOfComponents (o.i.e.g.examples.SlidingDoors_reviewed.automatic door controller)
  door                      ^listOfComponents (o.i.e.g.examples.SlidingDoors_reviewed.automatic door controller)
  door closing limit sensor ^listOfComponents (o.i.e.g.examples.SlidingDoors_reviewed.automatic door controller)
  door closing limit sensor ^listOfComponents (o.i.e.g.examples.SlidingDoors_reviewed.automatic door controller)
```

```
Glossary for: automatic door controller

List Of Components:
  door -> the automatic door
  timer -> a countdown timer
  object proximity sensor -> an object proximity sensor
  door opening limit sensor -> an opening limit sendon
  door closing limit sensor -> a closing limit sensor

List Of Sensors:
  object proximity sensor ... is activated
  door opening limit sensor ... is activated
  door closing limit sensor ... is activated
  timer ... expires

List Of Actuators:

  door                      ^listOfComponents (o.i.e.g.examples.SlidingDoors_reviewed.automatic door controller)
  door                      ^listOfComponents (o.i.e.g.examples.SlidingDoors_reviewed.automatic door controller)
  door closing limit sensor ^listOfComponents (o.i.e.g.examples.SlidingDoors_reviewed.automatic door controller)
  door closing limit sensor ^listOfComponents (o.i.e.g.examples.SlidingDoors_reviewed.automatic door controller)
```

Fig. 1. step-by-step glossary building for sliding door controller: (*A*) Components definition, (*B*) Sensors definition and (*C*) Actuators definition

3.2 EARs-based requirements building for the controller

The user is provided with an editor built in MPS is provided to start specifying the controller behavior. The user can input the requirements simply by filling in the instance of the EARS template with placeholders in the editor. Figure 2 provides a step-by-step guidence to write the EARS requirements, In the projectional editor the user is provided with the option (i.e., using CTRL+Space keys on keyboard) to intantiate the EARS-based template with placeholders (*Step A*). After obtaining an instance of the EARS template, the user starts filling in the placeholders with the information (i.e., glossary definitions) (*Step B*). *Step C* of the figure 2 depicts a complete sliding door controller specification.



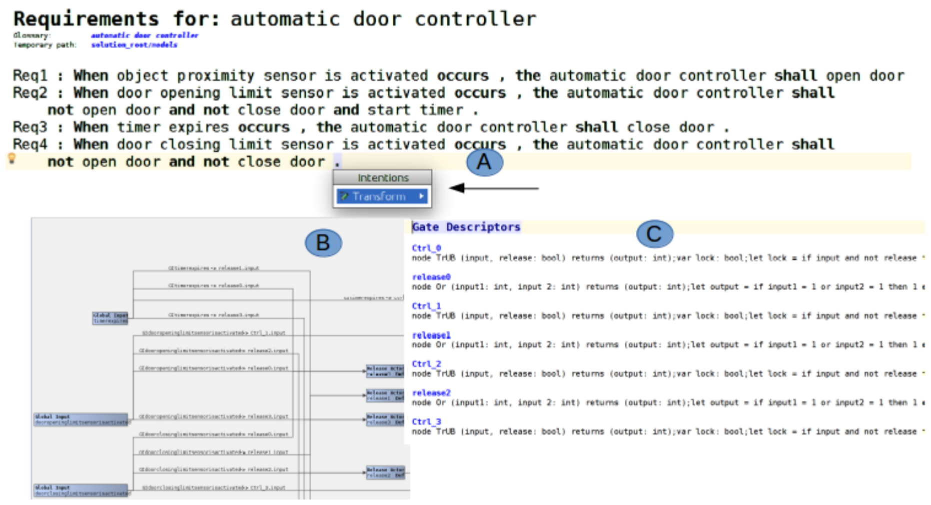
```
Req5 : When <no trigger> <no triggerModifier> , the automatic door controller shall <no response>

Req5 : When            <no triggerModifier> , the automatic door controller shall <no response> .
             door closing limit sensor is activated ^listOfTriggers (o.i.e.g.examples.SlidingDoors_reviewed.automatic door controller)
             door opening limit sensor is activated ^listOfTriggers (o.i.e.g.examples.SlidingDoors_reviewed.automatic door controller)
             object proximity sensor is activated   ^listOfTriggers (o.i.e.g.examples.SlidingDoors_reviewed.automatic door controller)
             timer expires                          ^listOfTriggers (o.i.e.g.examples.SlidingDoors_reviewed.automatic door controller)
             AndFormulaTr                                                         (BaseConcept in jetbrains.mps.lang.core)
             NotFormulaTr                                                         (BaseConcept in jetbrains.mps.lang.core)
             OrFormulaTr                                                          (BaseConcept in jetbrains.mps.lang.core)
```

```
Requirements for: automatic door controller
Glossary:        automatic door controller
Temporary path:  solution_root/models

Req1 : When object proximity sensor is activated occurs , the automatic door controller shall open door .
Req2 : When door opening limit sensor is activated occurs , the automatic door controller shall
    not open door and not close door and start timer .
Req3 : When timer expires occurs , the automatic door controller shall close door .
Req4 : When door closing limit sensor is activated occurs , the automatic door controller shall
    not open door and not close door .
```

Fig. 2. step-by-step EARS-CTRL for controller requirements: (*A*) Empty instance of EARS template with placeholders, (*B*) Filling instance of an EARS template with information and (*C*) Completed EARS Specification

3.3 Realizing the EARs-based requirements

The user of the tool can attempt to realize (by applying the *Transform* intention) the controller once the controller is completely specified. This will be acheived by applying the intention (applying *Alt+Enter* keys) on the root of the specification shown in *part A* of figure 3. The generated output of the transformation comprised of, 1) Controller Holder: Data Flow Diagram with blocks (*part B* of figure 3) and 2) Gate Descriptors: Pseudo code representing the behavior of the blocks (*part C* of figure 3).



```
Requirements for: automatic door controller
Glossary:         automatic door controller
Temporary path:   solution_root/models

Req1 : When object proximity sensor is activated occurs , the automatic door controller shall open door
Req2 : When door opening limit sensor is activated occurs , the automatic door controller shall
    not open door and not close door and start timer .
Req3 : When timer expires occurs , the automatic door controller shall close door .
Req4 : When door closing limit sensor is activated occurs , the automatic door controller shall
    not open door and not close door .
```

```
Gate Descriptors

Ctrl_0
node TrUB (input, release: bool) returns (output: int);var lock: bool;let lock = if input and not release

release0
node Or (input1: int, input 2: int) returns (output: int);let output = if input1 = 1 or input2 = 1 then 1

Ctrl_1
node TrUB (input, release: bool) returns (output: int);var lock: bool;let lock = if input and not release

release1
node Or (input1: int, input 2: int) returns (output: int);let output = if input1 = 1 or input2 = 1 then 1

Ctrl_2
node TrUB (input, release: bool) returns (output: int);var lock: bool;let lock = if input and not release

release2
node Or (input1: int, input 2: int) returns (output: int);let output = if input1 = 1 or input2 = 1 then 1

Ctrl_3
node TrUB (input, release: bool) returns (output: int);var lock: bool;let lock = if input and not release
```

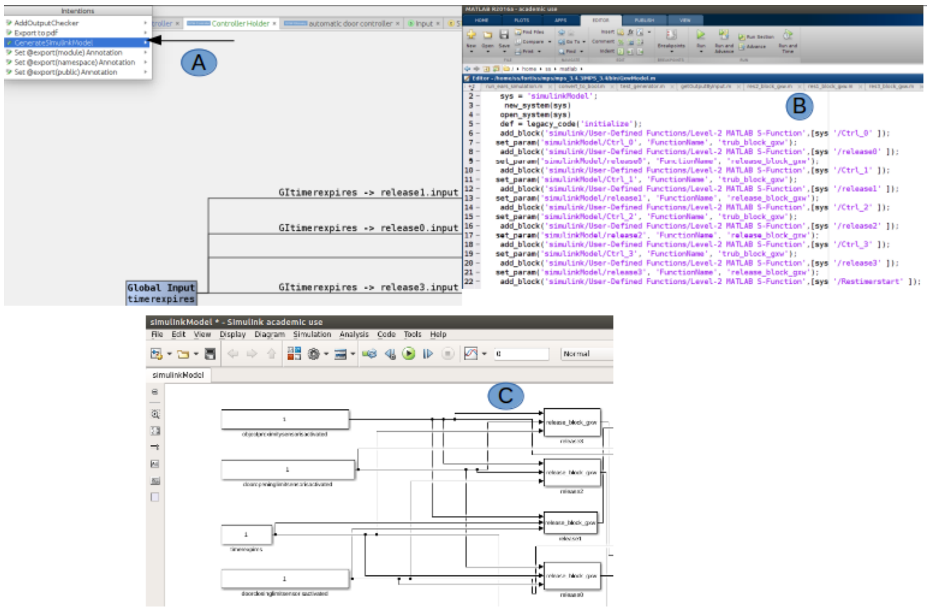
Fig. 3. Controller generation: (*A*) Applying Intention (Alt+Enter), (*B*) Data Flow Diagram of the controller (an excerpt) and (*C*) Pseudo code representing the behavior of the blocks (an excerpt)

3.4 Simulation and Test Cases for Analysis

Analysis for verification of the generated controller with respect to its specification can be performed manually by simulating the behavior in Matlab Simulink [2]. The analysis can be performed as follows,

Generation of Simulink model The user first applies the intention *GenerateSimulinkModel* on the root of the *Controller Holder* model generated earlier (i.e., part *A* of figure 4). The result of applying the intention is *.m* Simulink extention file (i.e., part *B* of figure 4) with Simulink information for block diagram generation. The user then clicks the Run button and as a result gets the Simulink block diagram (i.e., part *C* of figure 4).

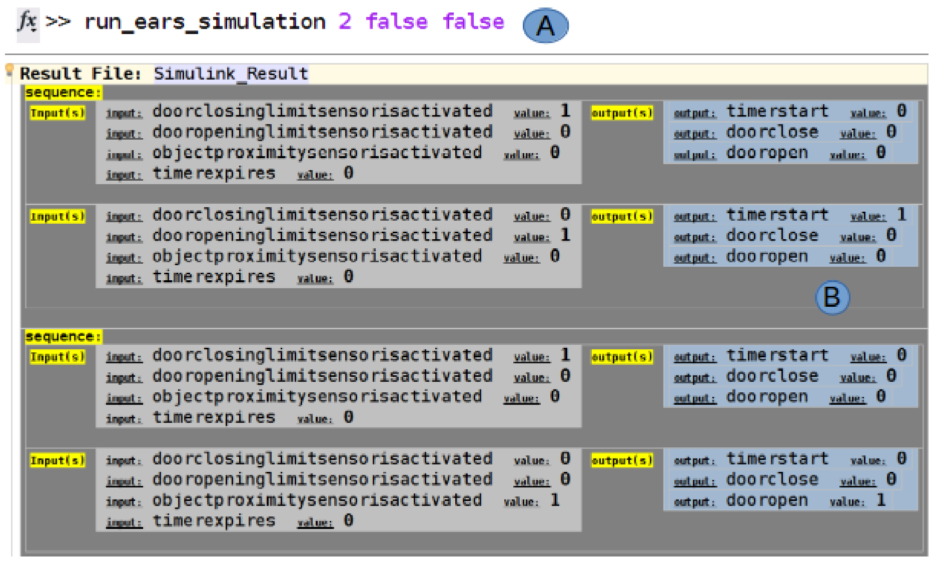
Analysis using Simulation and Test cases The user can automatically generate test cases and simulate the behavior in order to perform the analysis of the controller (as shown in figure 5). In order to automatically perform the testing and collecting the simulated data, the user can utilize the Simulink console and execute the following command in the Simulink console "run_ears_simulation 2 false false" in order to simulate the controller behavior in Simulink. The result


Intentions
AddOutputChecker
Export to pdf
GenerateSimulinkModel
Set @export(module) Annotation
Set @export(namespace) Annotation
Set @export(public) Annotation
Controller Holder
automatic door controller
A
GItimerexpires -> release1.input
GItimerexpires -> release0.input
GItimerexpires -> release3.input
Global Input
timerexpires
MATLAB R2016a - academic use
B
simulinkModel * - Simulink academic use
File Edit View Display Diagram Simulation Analysis Code Tools Help
Normal
simulinkModel
C
release_block_gxw

```
    sys = 'simulinkModel';
     new_system(sys)
    open_system(sys)
    def = legacy_code('initialize');
    add_block('simulink/User-Defined Functions/Level-2 MATLAB S-Function',[sys '/Ctrl_0' ]);
   set_param('simulinkModel/Ctrl_0', 'FunctionName', 'trub_block_gxw');
    add_block('simulink/User-Defined Functions/Level-2 MATLAB S-Function',[sys '/release0' ]);
   set_param('simulinkModel/release0', 'FunctionName', 'release_block_gxw');
    add_block('simulink/User-Defined Functions/Level-2 MATLAB S-Function',[sys '/Ctrl_1' ]);
   set_param('simulinkModel/Ctrl_1', 'FunctionName', 'trub_block_gxw');
    add_block('simulink/User-Defined Functions/Level-2 MATLAB S-Function',[sys '/release1' ]);
   set_param('simulinkModel/release1', 'FunctionName', 'release_block_gxw');
    add_block('simulink/User-Defined Functions/Level-2 MATLAB S-Function',[sys '/Ctrl_2' ]);
   set_param('simulinkModel/Ctrl_2', 'FunctionName', 'trub_block_gxw');
    add_block('simulink/User-Defined Functions/Level-2 MATLAB S-Function',[sys '/release2' ]);
   set_param('simulinkModel/release2', 'FunctionName', 'release_block_gxw');
    add_block('simulink/User-Defined Functions/Level-2 MATLAB S-Function',[sys '/Ctrl_3' ]);
   set_param('simulinkModel/Ctrl_3', 'FunctionName', 'trub_block_gxw');
    add_block('simulink/User-Defined Functions/Level-2 MATLAB S-Function',[sys '/release3' ]);
   set_param('simulinkModel/release3', 'FunctionName', 'release_block_gxw');
    add_block('simulink/User-Defined Functions/Level-2 MATLAB S-Function',[sys '/Restimerstart' ]);
```

Fig. 4. Simulink Model generation for analysis: (*A*) Applying GenerateSimulinkModel Intention, (*B*) Open generated .m file in Simulink and (*C*) Generated Simulink block diagram

of the simulation is saved as a text file that contains the inputs and generated outputs as a set of sequences. The generated text file is parsed in MPS. Model is generated in MPS when the user applies the intention *Produce Simulink Result Model*.



```
fx >> run_ears_simulation 2 false false
```

```
Result File: Simulink_Result
sequence:
Input(s)  input: doorclosinglimitsensorisactivated  value: 1   output(s)  output: timerstart  value: 0
          input: dooropeninglimitsensorisactivated  value: 0              output: doorclose   value: 0
          input: objectproximitysensorisactivated   value: 0              output: dooropen    value: 0
          input: timerexpires  value: 0
Input(s)  input: doorclosinglimitsensorisactivated  value: 0   output(s)  output: timerstart  value: 1
          input: dooropeninglimitsensorisactivated  value: 1              output: doorclose   value: 0
          input: objectproximitysensorisactivated   value: 0              output: dooropen    value: 0
          input: timerexpires  value: 0
sequence:
Input(s)  input: doorclosinglimitsensorisactivated  value: 1   output(s)  output: timerstart  value: 0
          input: dooropeninglimitsensorisactivated  value: 0              output: doorclose   value: 0
          input: objectproximitysensorisactivated   value: 0              output: dooropen    value: 0
          input: timerexpires  value: 0
Input(s)  input: doorclosinglimitsensorisactivated  value: 0   output(s)  output: timerstart  value: 0
          input: dooropeninglimitsensorisactivated  value: 0              output: doorclose   value: 0
          input: objectproximitysensorisactivated   value: 1              output: dooropen    value: 1
          input: timerexpires  value: 0
```

Fig. 5. Simulation and Analysis: (*A*) Simulink function to Simulate the Controller Behavior and (*B*) Model in MPS after parsing the Simulation results

Acknowledgements

This work presented in this paper was developed in the context of the "IETS3" research project, funded by the German Federal Ministry of Education and Research under code 01IS15037A/B.

References

1. Ears-ctrl github project. `https://github.com/levilucio/EARS-CTRL.git/`.
2. Matlab2016a. `https://de.mathworks.com//`.
3. A. Mavin, P. Wilkinson, A. Harwood, and M. Novak. Easy approach to requirements syntax (ears). In *2009 17th IEEE International Requirements Engineering Conference*, pages 317–322, Aug 2009.