“An Experiment in Requirements Engineering and Testing using EARS Notation for PLC Systems”

Mikael Ebrahimi Salari, Eduard Paul Enoiu,
Wasif Afzal, Cristina Seceleanu

Mälardalen University, Sweden

{mikael.salari, eduard.enoiu, wasif.afzal,
cristina.seleceleanu}@mdu.se

April 2023

ICST 2023
A-MOST Workshop

Introduction and Motivation

- Regulatory standards for engineering safety-critical systems often demand both traceable requirements and specification-based testing, during development.

- Engineering safety-critical systems
  - obey regulatory standards
    - traceable requirements
    - specification-based testing } required during development

- Requirements are often written in natural language, yet for specification purposes, this may be supplemented by formal or semi-formal descriptions
  - increase clarity.

- However, the choice of notation of the semi-formal descriptions is often constrained by the training, skills, and preferences of the designers.

EARS: Easy Approach to Requirements Syntax

- EARS
  - A simple notation for writing textual requirements
  - First published at Requirements Engineering Conference, RE 2009*
  - An initiative by Rolls-Royce and Intel to reduce the main problems detected in stakeholder requirements

- Addresses the inherent imprecision of natural language (NL) requirements
  - potential ambiguity and lack of accuracy.

* Alistair Mavin, Philip Wilkinson, Adrian Harwood, and Mark Novak. Easy approach to requirements syntax (ears). In 2009 17th IEEE International Requirements Engineering Conference, pages 317–322. IEEE, 2009.

Research Goal

- This work investigates
  1. Requirement specification using EARS
  2. Specification-based testing of embedded software written in two IEC 61131-3* standard languages.
     - Function Block Diagram (FBD)
     - Structured Text (ST)
     - Ladder Diagram (LD)
     - Sequence Function Chart (SFC)
     - Continuous Function Chart (CFC)

* Michael Tiegelkamp and Karl-Heinz John. IEC 61131-3: Programming industrial automation systems, volume 166. Springer, 2010.

Research Goal

- Conduct experiments to study:

  - How participants translate natural language requirements into EARS.

  - How engineers use EARS requirements to test PLC software.

Research Questions

- RQ1: How are the EARS semi-structured requirement engineering syntax and test creation applied in the context of PLC programs?
- RQ2: What EARS patterns are used during the writing of requirements?
- RQ3: What challenges are perceived during the specification of requirements and test creation using EARS?

- We report our observations during the experiments, including
  - The type of EARS patterns participants use to structure natural language requirements
  - Challenges during the specification phase
  - Present the results of testing based on EARS-formalized requirements.

What is PLC?

- Programmable Logic Controller (PLC) devices play a significant role in today’s automated industry.

- PLC devices are being widely used in safety-critical applications such as
  - Power Plants
  - Nuclear Plants
  - Cranes

- Using a semi-structured easy to understand requirement syntax such as EARS may improve the quality of PLC testing

EARS - Experimental Setup Overview

- Controlled experiment with participants
  - Write 4 given requirements using EARS syntax.
  - Freedom of choice of preferred EARS syntax template

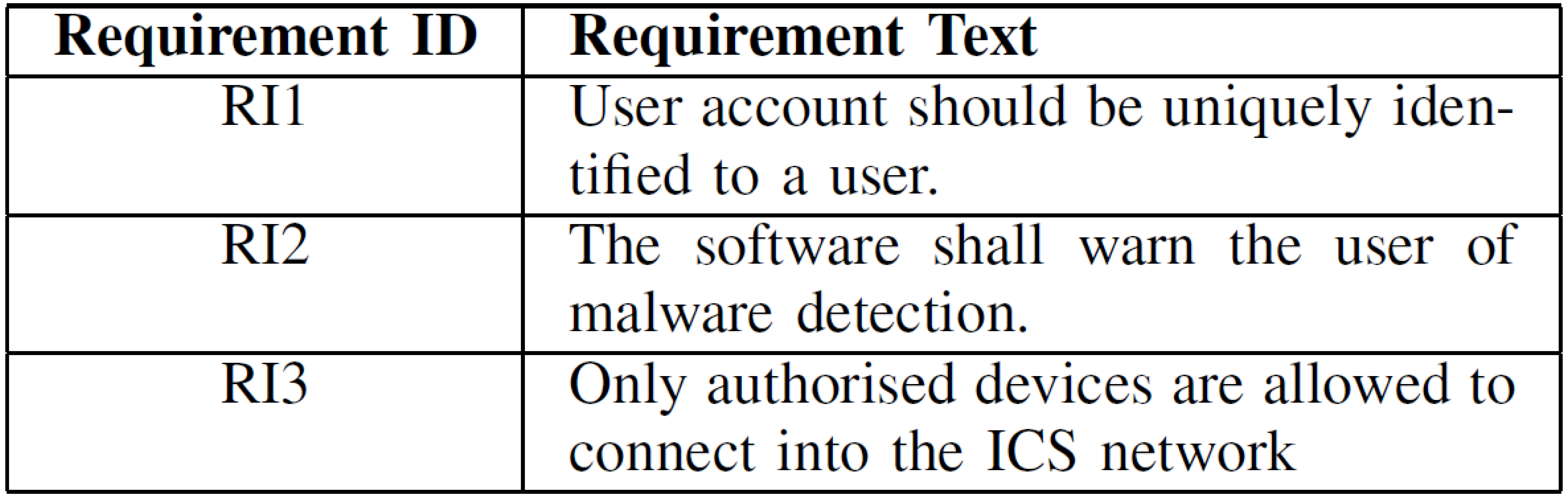
- Subjects: 10 individuals including
  - 4 experienced engineers at a large automation company in Sweden and Spain
  - 6 researchers and managers from different universities and research institutions across Europe.

EARS - Experimental Setup Overview

- Object Selection:
  - Manual choice based on the following criteria on requirements:
    - Specifications in NL should be understandable.
    - Should be sufficiently rich in detail for an engineer to write executable tests.
    - Should represent different types of real testing scenarios in different areas using IEC 61131-3 standard.
    - Should be simple to understand without any domain knowledge.
    - The resulting test cases should be executable in the CODESYS environment.

Natural Language Requirements

- Industrial libraries provided by a large company that develops and manufactures control systems
  - Identified three candidate requirements matching our criteria
- Selected high-level requirements should
  - not be trivial & fully manageable within 60 minutes
  - not require domain-specific knowledge



Requirement ID	Requirement Text
RI1	User account should be uniquely identified to a user.
RI2	The software shall warn the user of malware detection.
RI3	Only authorised devices are allowed to connect into the ICS network

EARS Templates

1. Ubiquitous requirement (U):
   - A type of requirement that is not bonded to any preconditions or triggers and is always enabled in the system.
   - **The <system name> shall <system response>**

2. Event-driven requirements (ED):
   - The event-driven requirement is used only when an event is identified in the system.
   - **WHEN <optional preconditions> <trigger> the <system name> shall <system response>**

3. Unwanted behaviours (UB):
   - refers to covering all possible situations that are not desirable and are usually a big source of omissions in preliminary requirements.
   - **IF <optional preconditions> <trigger>, THEN the <system name> shall <system response>**

EARS Templates

4. State-driven requirements (SD):

   The State-driven requirement is only active if the system is in a specific status

   WHERE <feature is included> the <system name> shall <system response>

5. Optional features (OF):

   designed to be used when the author of the requirement wants to include a specific feature in the system.

   WHERE <feature is included> the <system name> shall <system response>

Process Challenges

- Types of challenges during the use of EARS templates:

  1. Encountered during the specification of requirements

  2. When designing test cases for PLC systems

- Thematic analysis for qualitative data to extract the main themes as reflected by the input given by each participant.

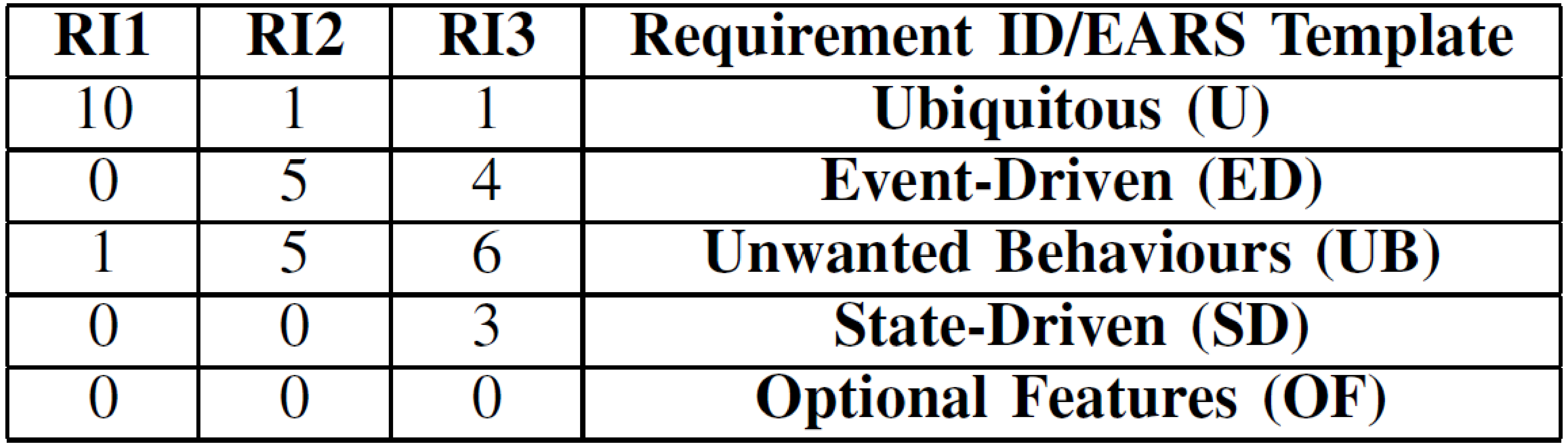
Instrumentation

- One session was organized
- The subjects were given the task to use the three requirements and rewrite these in EARS.
- The subjects were not grouped.
- Both digital and written forms for the Needed experiment document
- A short tutorial on EARS syntax was provided to the subjects (10 mins)
- Data Collection Procedure
  - As part of the instructions, subjects submitted their solutions in the form of a record documenting their work.
  - Data from this experiment session was then used for quantitative and qualitative analysis.

EXPERIMENT ANALYSIS

Requirement Engineering Results

- Results w.r.t. EARS templates used for each requirement

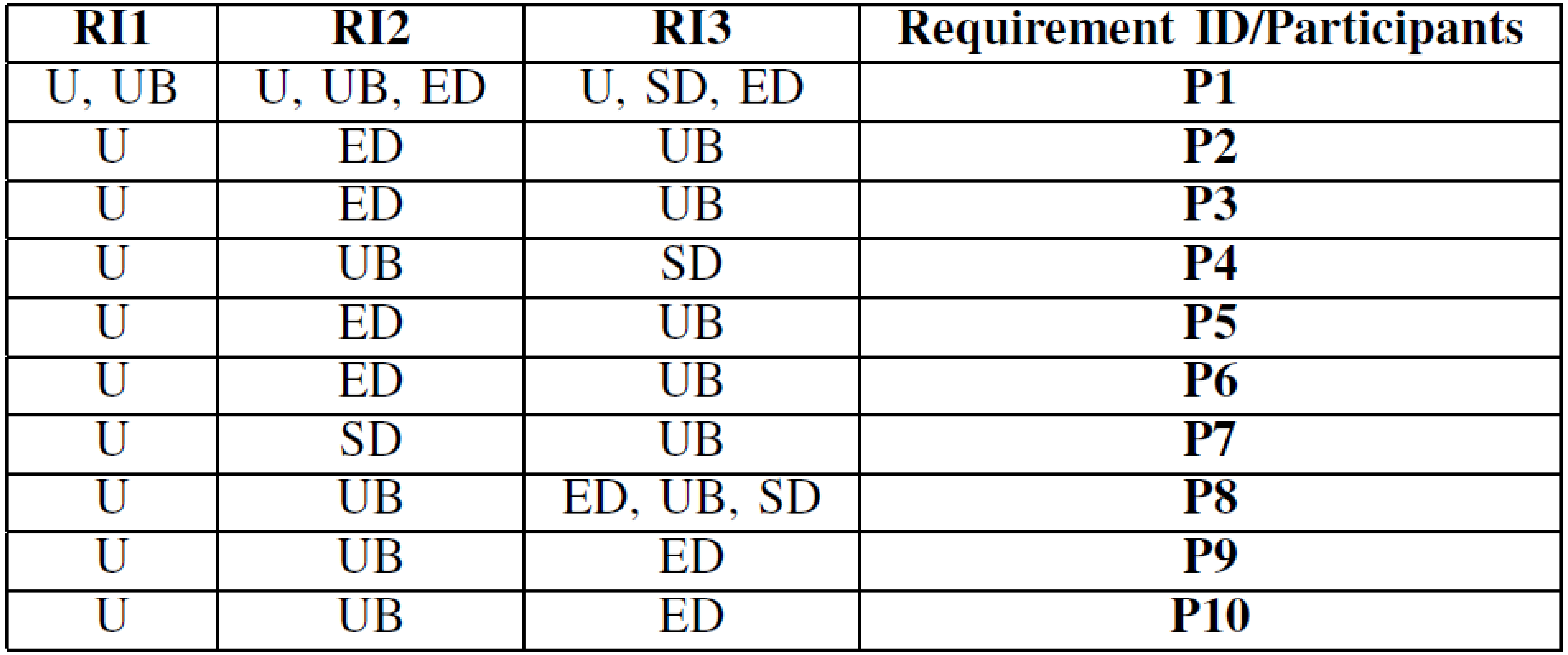


RI1	RI2	RI3	Requirement ID/EARS Template
10	1	1	Ubiquitous (U)
0	5	4	Event-Driven (ED)
1	5	6	Unwanted Behaviours (UB)
0	0	3	State-Driven (SD)
0	0	0	Optional Features (OF)

EXPERIMENT ANALYSIS

Requirement Engineering Results

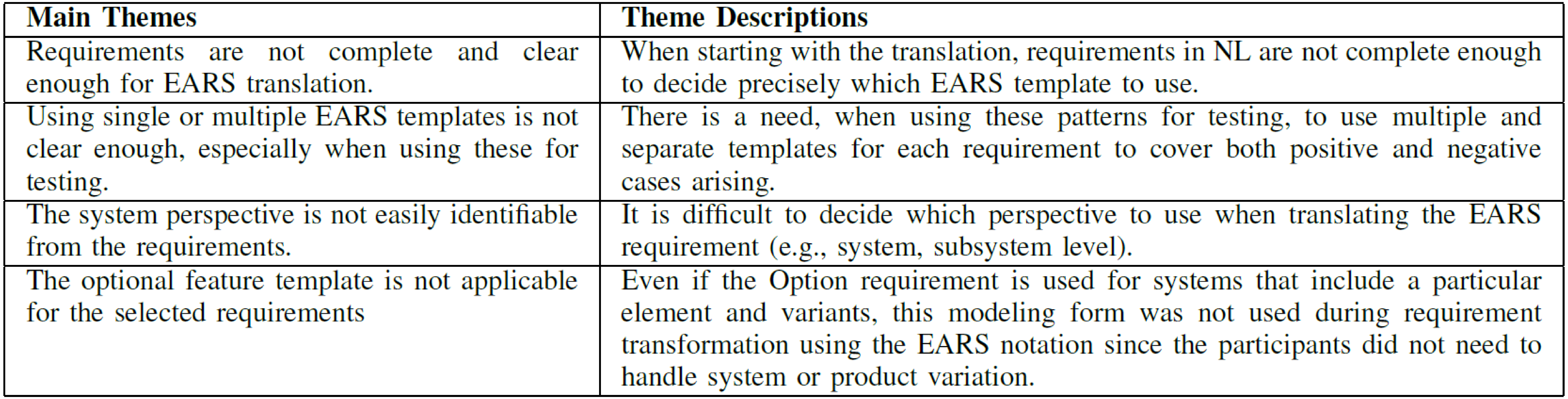
- Results of the requirements writing in terms of the templates used by each participant for each requirement



RI1	RI2	RI3	Requirement ID/Participants
U, UB	U, UB, ED	U, SD, ED	**P1**
U	ED	UB	**P2**
U	ED	UB	**P3**
U	UB	SD	**P4**
U	ED	UB	**P5**
U	ED	UB	**P6**
U	SD	UB	**P7**
U	UB	ED, UB, SD	**P8**
U	UB	ED	**P9**
U	UB	ED	**P10**

EXPERIMENT ANALYSIS
Requirement Engineering Results

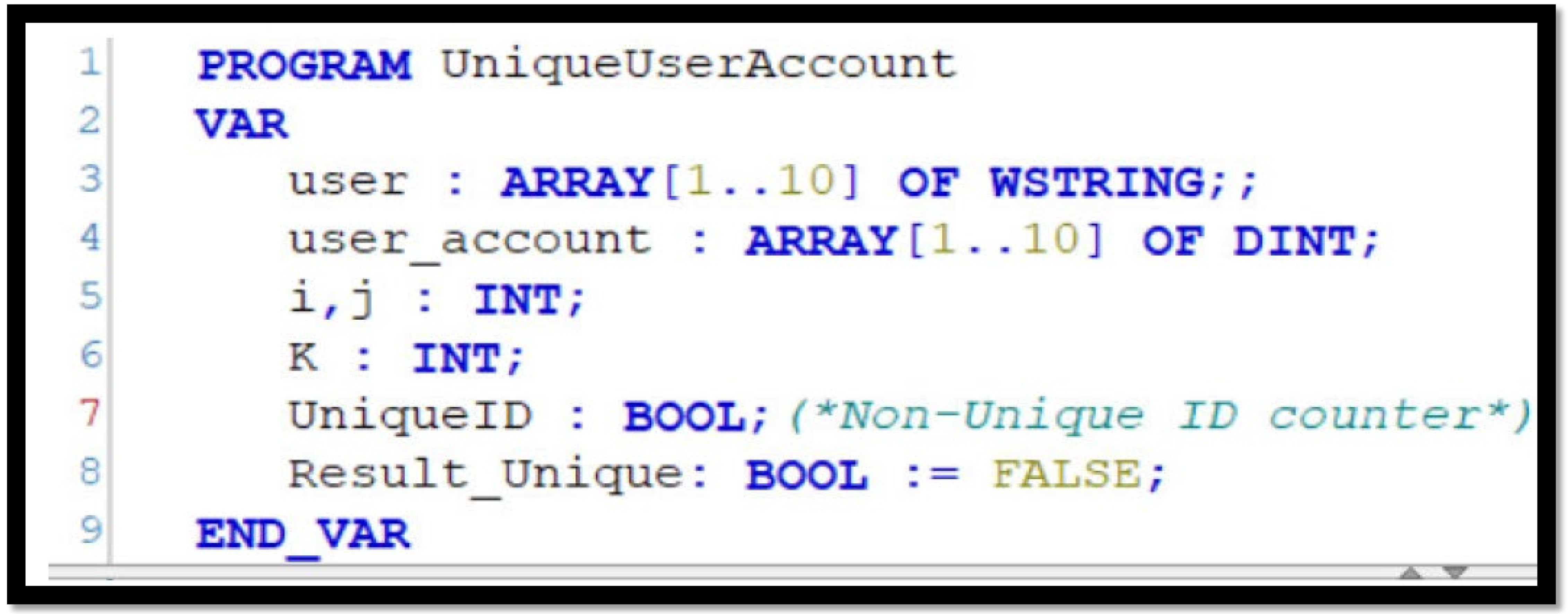
- Results showing the main themes identified related to approaches and challenges encountered during the translation process.



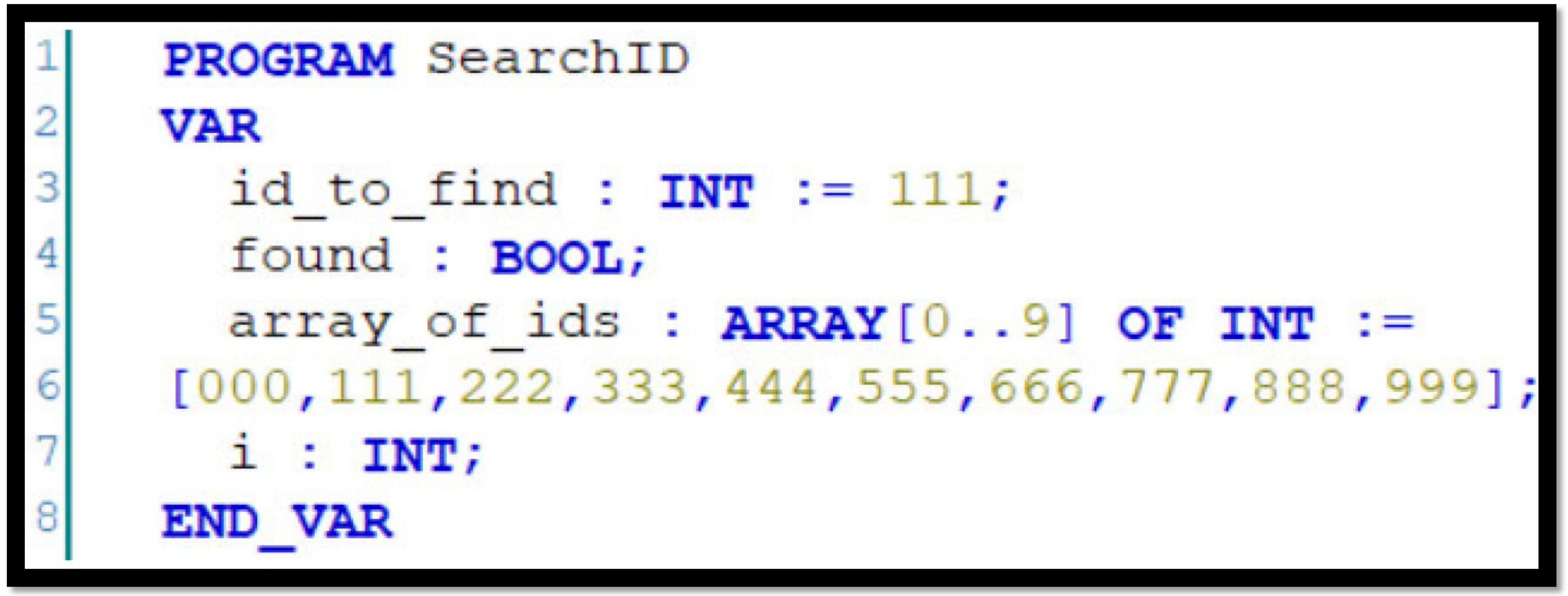
Main Themes	Theme Descriptions
Requirements are not complete and clear enough for EARS translation.	When starting with the translation, requirements in NL are not complete enough to decide precisely which EARS template to use.
Using single or multiple EARS templates is not clear enough, especially when using these for testing.	There is a need, when using these patterns for testing, to use multiple and separate templates for each requirement to cover both positive and negative cases arising.
The system perspective is not easily identifiable from the requirements.	It is difficult to decide which perspective to use when translating the EARS requirement (e.g., system, subsystem level).
The optional feature template is not applicable for the selected requirements	Even if the Option requirement is used for systems that include a particular element and variants, this modeling form was not used during requirement transformation using the EARS notation since the participants did not need to handle system or product variation.

EARS
PLC Testing Experimental Setup

- PLC Programs:
  - 3 PLC programs that implement the behavior of the selected NL requirements
  - ST language
  - PLC IDE:
    - CODESYS V3.15
  - Testing Tool:
    - CODESYS Test Manger



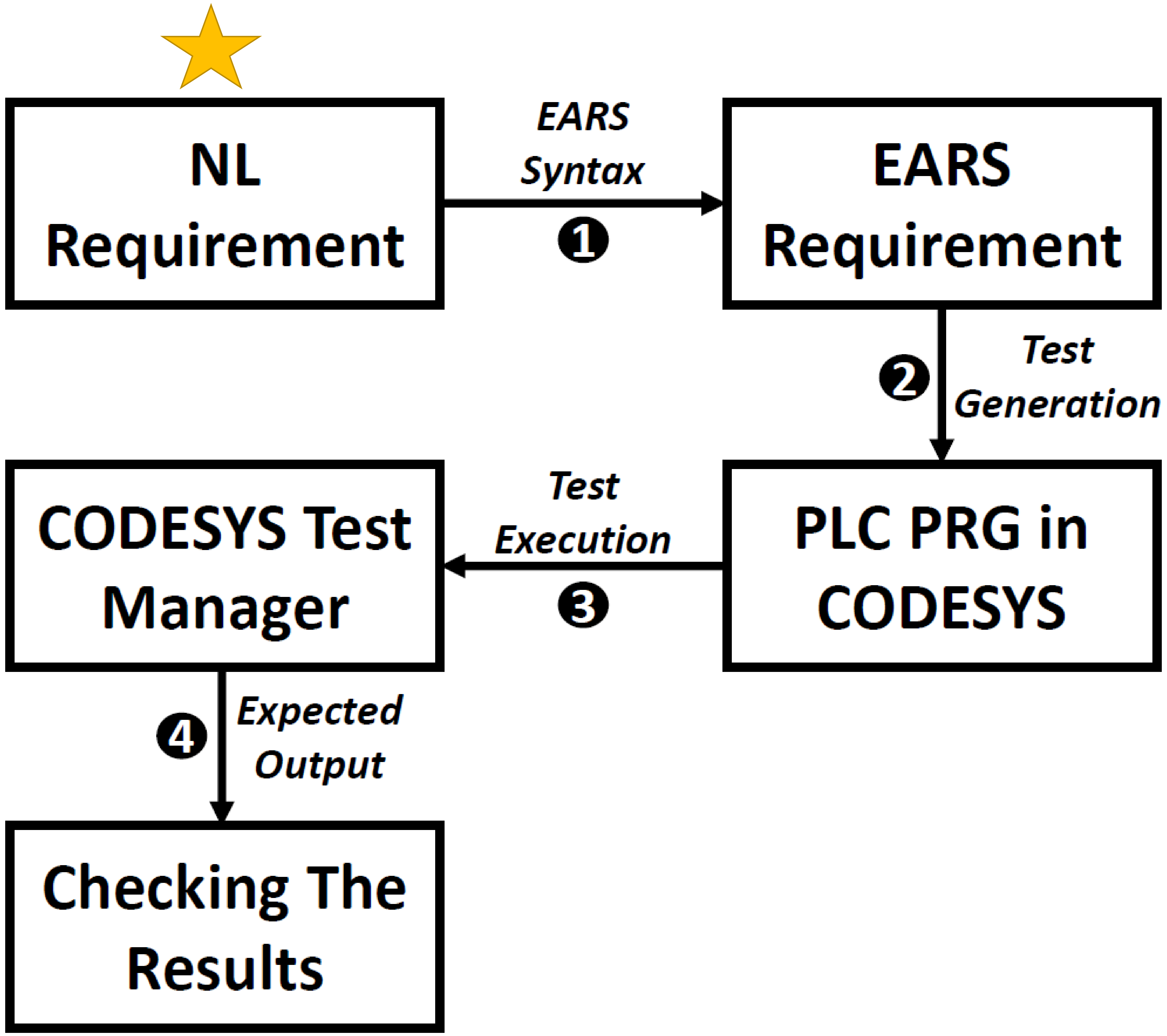
```
PROGRAM UniqueUserAccount
VAR
    user : ARRAY[1..10] OF WSTRING;;
    user_account : ARRAY[1..10] OF DINT;
    i,j : INT;
    K : INT;
    UniqueID : BOOL;(*Non-Unique ID counter*)
    Result_Unique: BOOL := FALSE;
END_VAR
```



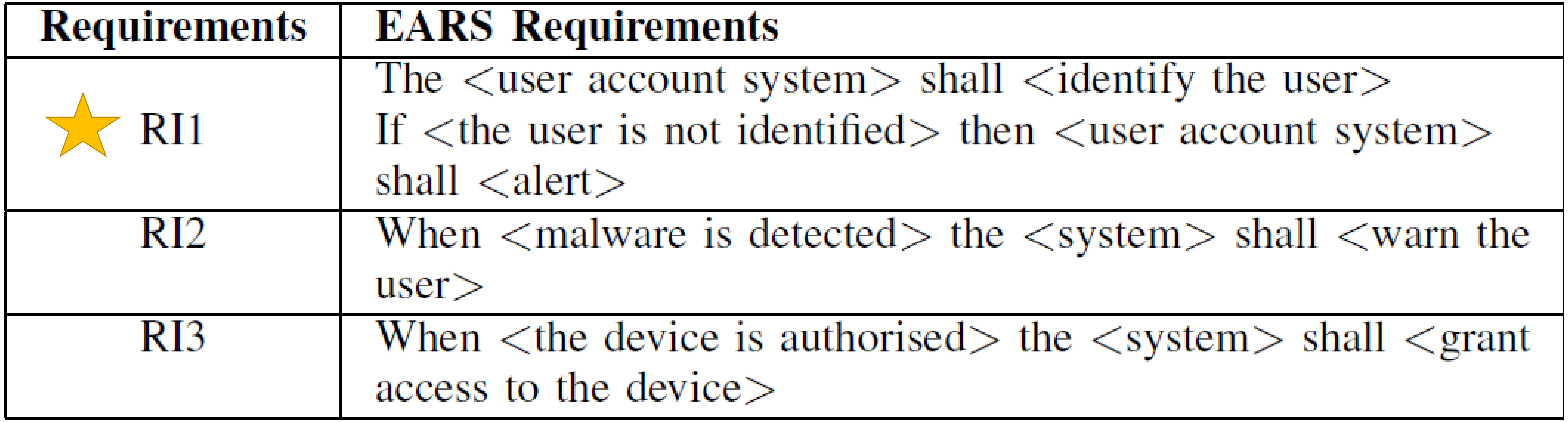
```
PROGRAM SearchID
VAR
  id_to_find : INT := 111;
  found : BOOL;
  array_of_ids : ARRAY[0..9] OF INT :=
[000,111,222,333,444,555,666,777,888,999];
  i : INT;
END_VAR
```

EARS
PLC Testing Experimental Setup

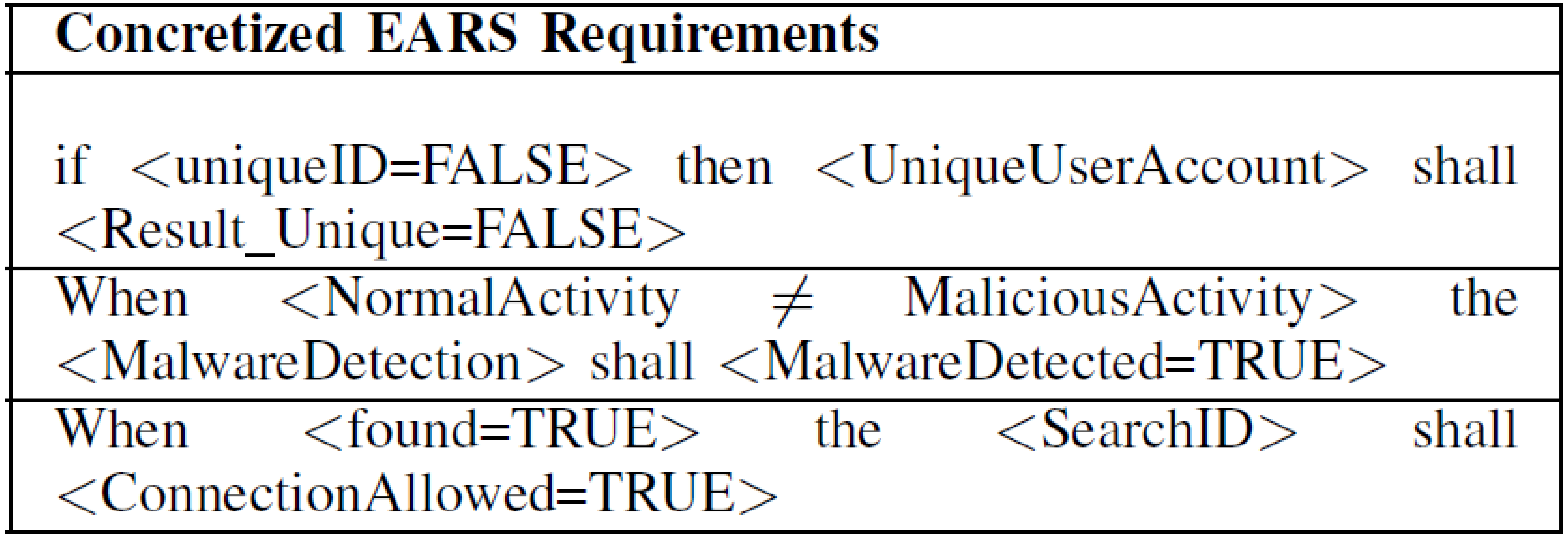
EARS to PLC Testing Workflow



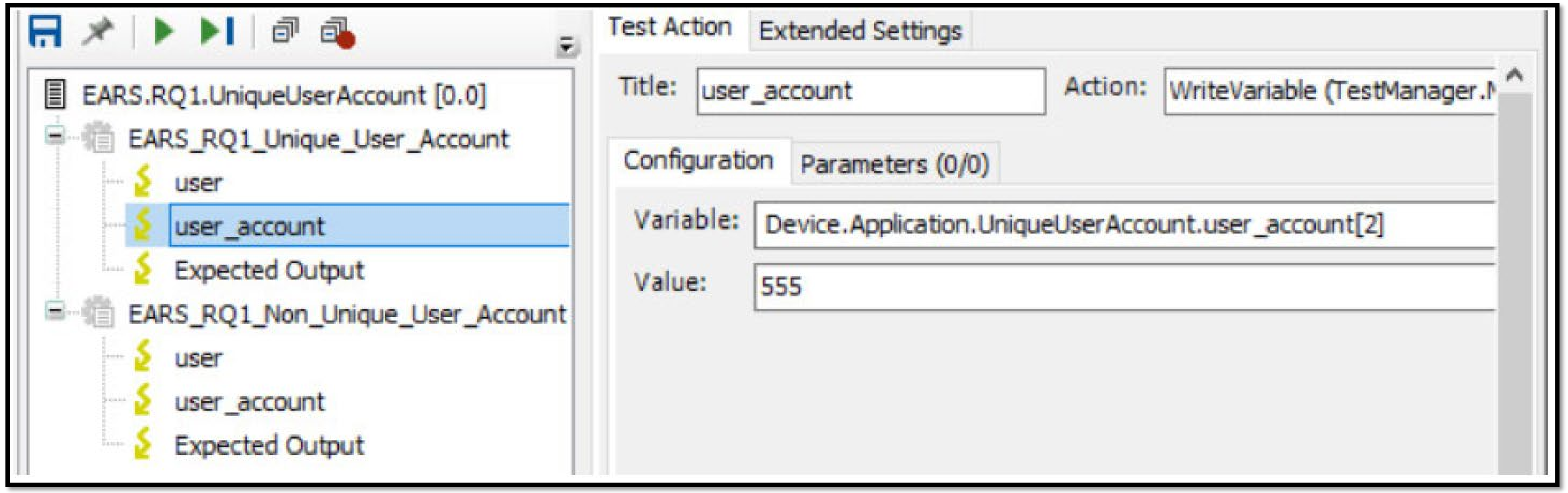
EARS
PLC Testing Experimental Setup



Requirements	EARS Requirements
★ RI1	The <user account system> shall <identify the user> If <the user is not identified> then <user account system> shall <alert>
RI2	When <malware is detected> the <system> shall <warn the user>
RI3	When <the device is authorised> the <system> shall <grant access to the device>



Concretized EARS Requirements
if <uniqueID=FALSE> then <UniqueUserAccount> shall <Result_Unique=FALSE>
When <NormalActivity ≠ MaliciousActivity> the <MalwareDetection> shall <MalwareDetected=TRUE>
When <found=TRUE> the <SearchID> shall <ConnectionAllowed=TRUE>



- EARS.RQ1.UniqueUserAccount [0.0]
  - EARS_RQ1_Unique_User_Account
    - user
    - user_account
    - Expected Output
  - EARS_RQ1_Non_Unique_User_Account
    - user
    - user_account
    - Expected Output

Test Action | Extended Settings

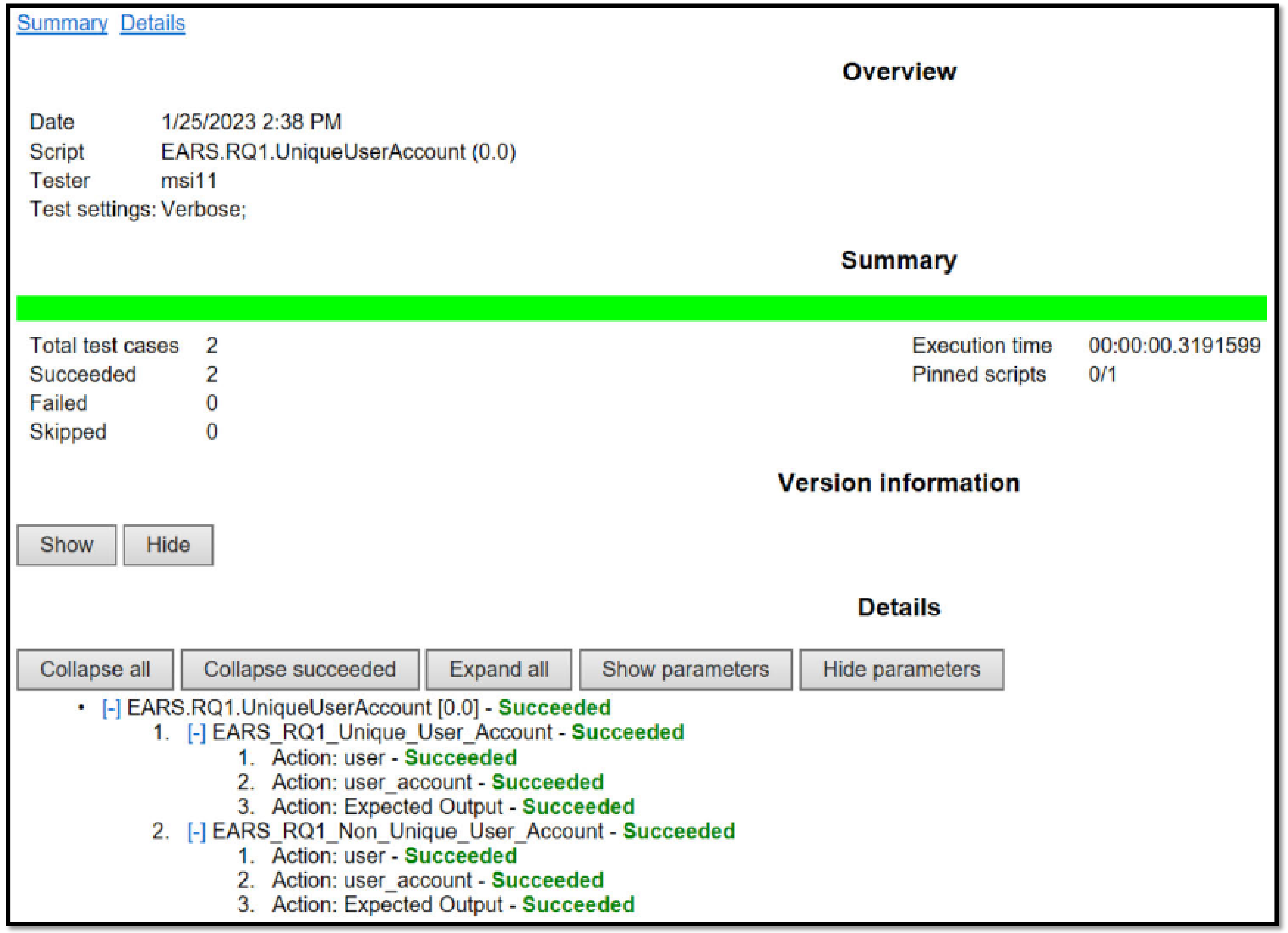
Title: user_account   Action: WriteVariable (TestManager.N

Configuration | Parameters (0/0)

Variable: Device.Application.UniqueUserAccount.user_account[2]

Value: 555

CODESYS
Test Manager



Summary Details

Overview

Date 1/25/2023 2:38 PM
Script EARS.RQ1.UniqueUserAccount (0.0)
Tester msi11
Test settings: Verbose;

Summary

Total test cases	2	Execution time	00:00:00.3191599
Succeeded	2	Pinned scripts	0/1
Failed	0		
Skipped	0		

Version information

Details

- [-] EARS.RQ1.UniqueUserAccount [0.0] - **Succeeded**
  1. [-] EARS_RQ1_Unique_User_Account - **Succeeded**
     1. Action: user - **Succeeded**
     2. Action: user_account - **Succeeded**
     3. Action: Expected Output - **Succeeded**
  2. [-] EARS_RQ1_Non_Unique_User_Account - **Succeeded**
     1. Action: user - **Succeeded**
     2. Action: user_account - **Succeeded**
     3. Action: Expected Output - **Succeeded**

EXPERIMENT ANALYSIS

Requirement Engineering Results-Testing

- Test Results of PRG1 (User Identification):
  - Two test cases to cover the user identification scenarios.
  - Each test case includes the following two test actions:
    - Two WriteVariable test actions to alter the user and user account inputs
    - One CompareVariable test action that compares the actual output with the expected one
  - Execution time: 0.3 seconds.
  - All executed test cases have successfully passed

EXPERIMENT ANALYSIS
Requirement Engineering Results-Testing

- Test Results of PRG2 (Malware Detection):
  - Considering the results of the experiment
    - Event-driven requirement pattern was used
  - Two test cases for PRG2.
  - Each test case consists of two test actions (MaliciousActivity and NormalActivity)
  - Automated test execution using CODESYS Test Manager in
  - Test execution time: 1.71 seconds.
  - All developed test cases have successfully passed.

EXPERIMENT ANALYSIS
Requirement Engineering Results-Testing

- Test Results of PRG3 (Authorised Devices):
- Program units:

1. a database of authorised device IDs
   - An array of IDs,
2. An input signal corresponding to the device ID that needs to be authorized
3. a boolean output signal (i.e., found)
   - Returns True in the case of the authorized device being allowed to connect given the ID is known

EXPERIMENT ANALYSIS
Requirement Engineering Results-Testing

- Two test cases were developed
    - Successful Authorization
    - Unsuccessful Authorization.
- Each test case includes
  - Two actions including
    - The provision of a new Input ID
    - Comparing the actual output with the expected output.
- Automated test execution using CODESYS Test Manager
  - 1.14 seconds
- All test cases have successfully passed

Conclusion

- We have conducted an experiment in requirements engineering and testing using EARS notation for PLC systems.

- In the requirement engineering part of our experiment:
  - Most participants preferred the EARS ubiquitous pattern for transforming the RI1 requirement from NL to the EARS syntax.
  - The unwanted behaviour and event-driven patterns were the most popular types for RI2 and RI3 requirement transformations.

- It was observed that different individuals used different EARS patterns for transforming the same requirement based on their personal interpretation
  - Implies an acceptable level of flexibility in EARS syntax.

Conclusion

- In the testing part of our experiment, we investigated the applicability of using the EARS patterns in terms of PLC testing.

- The gathered test execution results show that using EARS in creating requirement-based test cases for PLC programs is promising.

- EARS can benefit the PLC testers by establishing an easy-to-understand way of expressing test specifications.

Future Work

- Investigating the applicability of using EARS in PLC requirement engineering
  - on other levels of testing
  - by including more PLC programs.

- Inspection of the impact of choosing different EARS templates for describing the requirements over the quality of the generated test cases.

- We want to automate our solution and generate test cases from the created EARS requirements based on existing functional and non-functional requirements.

VeriDevOps

Thanks for your attention…

Questions?

Mälardalens universitet