OpenDuT – Test Definition Language and Test Execution Architecture

Eclipse SDV Community Day

March 19, 2024, Graz, Austria

Stefan Marksteiner

Stefan Marksteiner

AVL

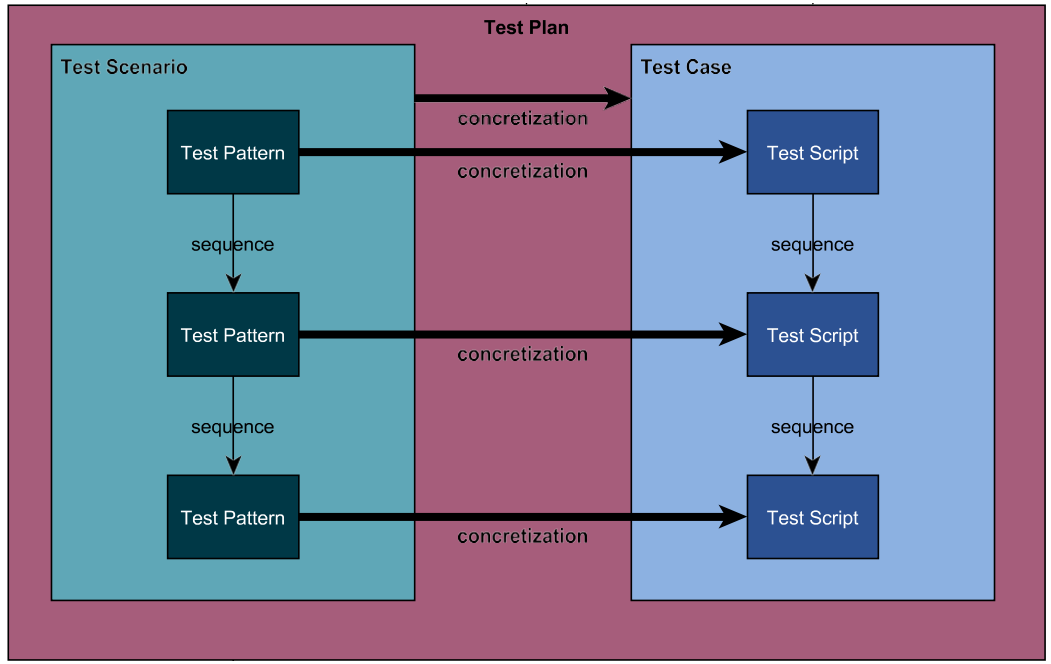
ALIA - Agnostic Language for Implementing Attacks

Test Abstraction and Execution Layers

Test Reuse on Different Systems

Abstracting Test Patterns

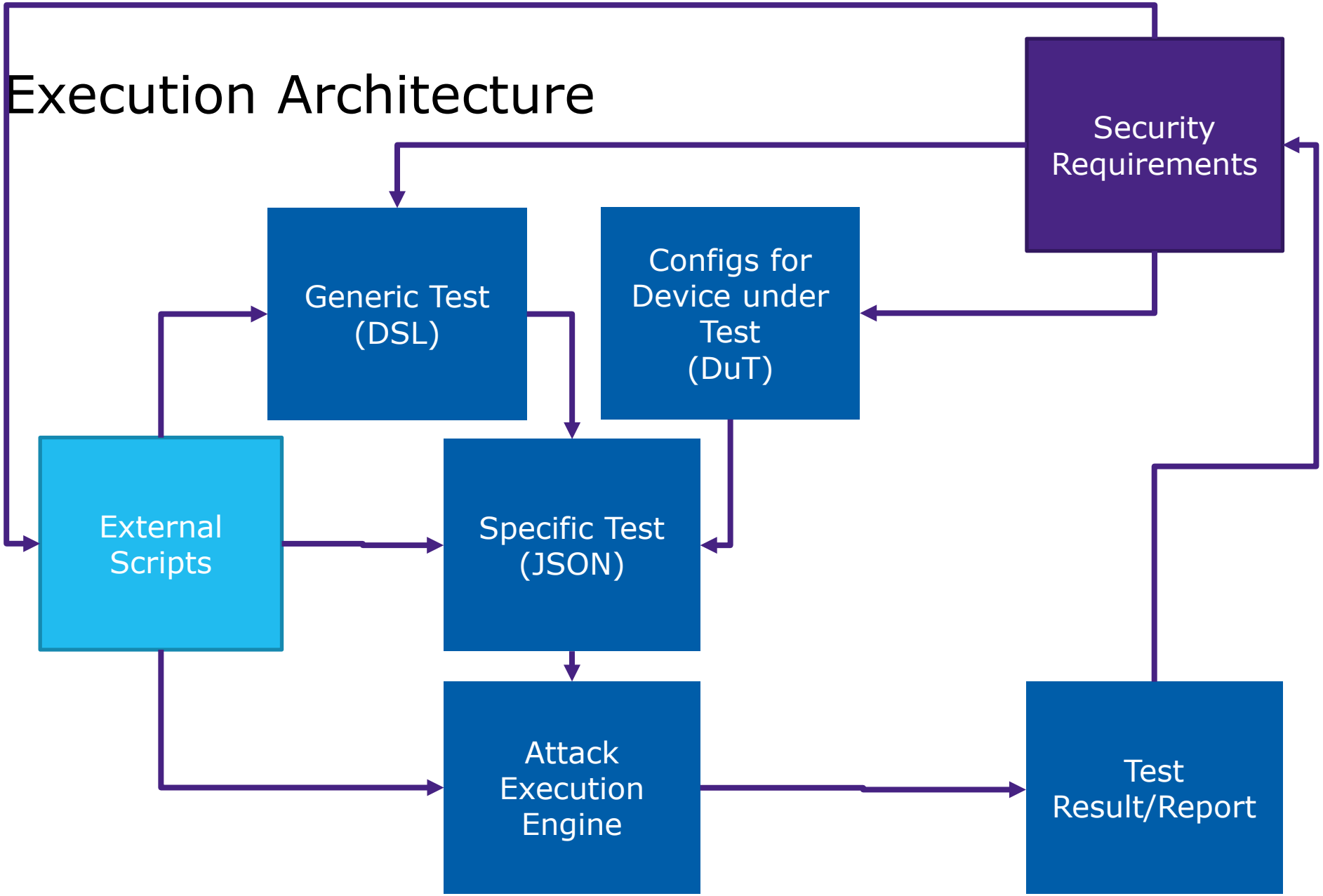
- The purpose is portability through generalization
- Test patterns are system-agnostic test building blocks that form a scenario [1]
- Scenarios are concretized for specific SUTs as test cases (consisting of single scripts)
- The test scenarios are stored in an own developed attack description domain-specific language (ALIA) [2]



[2] S. Marksteiner *et al.*, "A Process to Facilitate Automated Automotive Cybersecurity Testing," in *2021 IEEE 93rd Vehicular Technology Conference (VTC Spring)*, New York, NY, USA: IEEE, 2021.

[3] C. Wolschke, S. Marksteiner, T. Braun, and M. Wolf, "An Agnostic Domain Specific Language for Implementing Attacks in an Automotive Use Case," in *The 16th International Conference on Availability, Reliability and Security*, in ARES 2021. New York, NY, USA: Association for Computing Machinery, Aug. 2021, pp. 1–9. doi: 10.1145/3465481.3470070.

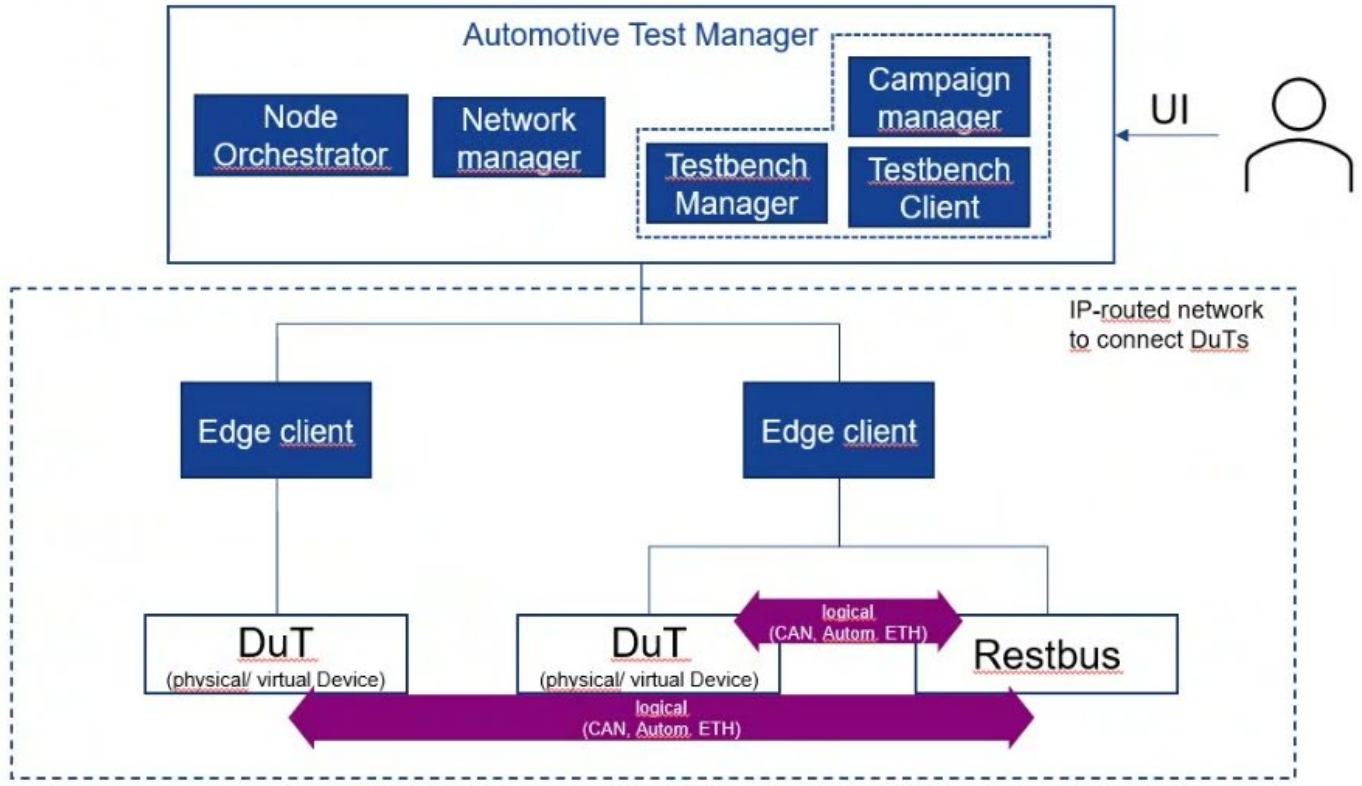
Current Execution Architecture



OpenDuT Execution Architecture

Eclipse openDuT: Building Blocks

Physical and logical connections



AVL

ALIA - Agnostic Language for Implementing Attacks

A DSL for Cyber Security Testing

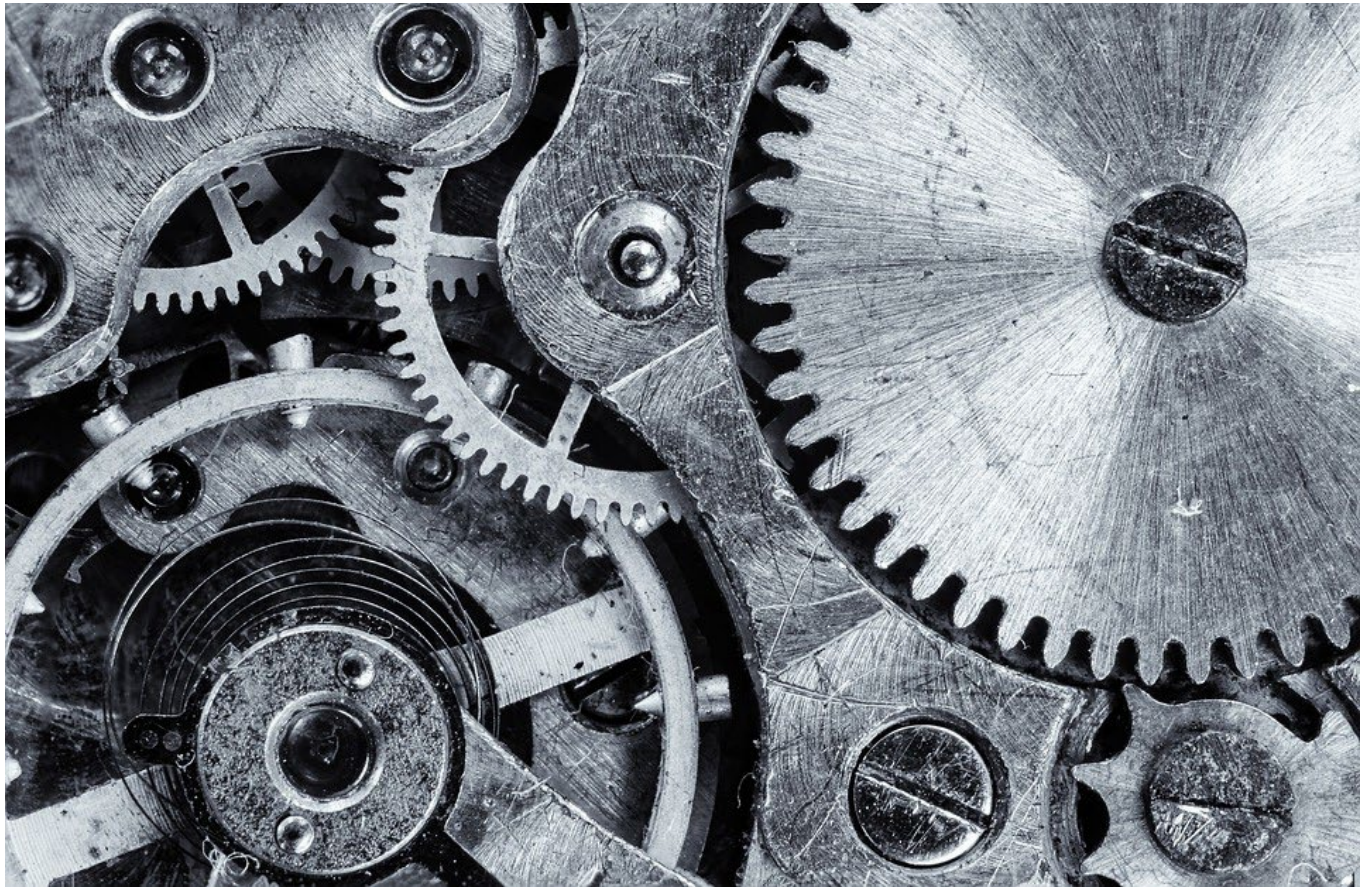
Public

Example: UDS Session Scan

```
PreConditions:

Actions:
      setIFDOWN : execute(tool: "ip", params:["link", "set", "can0", "down"])

      consetIFDownfigureIF: execute(tool:"ip", params:["link", "set", "can0", "type", "can",
      "bitrate", "500000"])

      setIFUP: execute(tool:ip", params:["link", "set", "can0", "up"])

      UDSSessionScan: execute(tool:"python3", params:["/home/kali/CAN_tools/sessionscan.py", "can0",
      "714", "7EE", "/home/kali/CAN_tools/sessionoutput.txt"])

      PrintSessionOutput: execute(tool:"cat", params:["/home/kali/CAN_tools/sessionoutput.txt "])

PostConditons:
```

Actions

- Implement the steps of a test that are executed sequentially (or in parallel)
- Consist of an identifier and an *execute* function
- Identifiers can be used in later actions to reference a previous result
- A function contains specific params that are used to parametrize the used tool
- A function executes a single tool with specified parameters
- Comments with „#“ or „//“



Flow Control

Conditionals:

```
if( <conditon> )
    <action>
elseif(<conditon>)
    <action>
else
    <action>
endif
```

Loops:

```
while(<condition>)
    <action>
endwhile
```

Example Output

- The output of the DSL-Script is parsed to a JSON Object
- This object is sent to the Execution Engine on the Attackbox for execution


```
{
   "name": „UDSSessionScan.json",
   "preconditions":[],
   "actions":[
               {"identifier":"setIFDown", "result":"setIFDown", "DSLStep":"setIFDown: execute(tool: \"ip\", params:[\"link\", \"set\",
\"{canIF}\",\"down\"])" , "commands":[{"name":"setIFDown", "tool": "ip", "parameters":["link", "set", "can0", "down"]}]},
               {"identifier":"configureIF", "result":"configureIF", "DSLStep":"configureIF: execute(tool:\"ip", params:[\"link\", \"set\",
\"can0\", \"type\", \"can\", \"bitrate\", \"500000\"])" , "commands":[{"name":"configureIF", "tool":"ip", "parameters":["link", "set",
"can0", "type", "can", "bitrate", "500000"]}]},
               {"identifier":"setIFUP", "result":"setIFUP", "DSLStep":"setIFUP: execute(tool:\"ip", params:[\"link\", \"set\", \"can0\",
\"up\"])" , "commands":[{"name":"setIFUP", "tool":"ip", "parameters":["link", "set", "can0", "up"]}]},
               {"identifier":"UDSSessionScan", "result":"UDSSessionScan", "DSLStep":"UDSSessionScan: execute(tool:\"python3\",
params:[\"/home/kali/CAN_tools/sessionscan.py\",\"can0\",\"714\",\"7EE\",\"/file.txt\"])" , "commands":[{
   "name":"UDSSessionScan", "tool":"python3"
   , "parameters":["/home/kali/CAN_tools/sessionscan.py", "can0", "714", "7EE", "file.txt"]}]},
               {"identifier":"PrintSessionOutput", "result":"PrintSessionOutput", "DSLStep":"PrintSessionOutput: execute(tool:\"cat\",
params:[\"/home/kali/CAN_tools/sessionoutput.txt \"])" , "commands":[{"name":"PrintSessionOutput", "tool":"cat",
"parameters":["/home/kali/CAN_tools/sessionoutput.txt"]}]}
   ],
   "postconditions":[]
}
```

Acknowledgement

openDuT

Federal Ministry
Republic of Austria
Climate Action, Environment,
Energy, Mobility,
Innovation and Technology

Federal Ministry
Republic of Austria
Labour and Economy

FFG
Promoting Innovation.

austria
wirtschafts
service
aws

IPCEI Microelectronics and
Communication Technologies

- The present work was made possible by contribution of the *Austrian Federal Ministry of Climate Action, Environment, Energy, Mobility, Innovation and Technology* (BMK), and the *Austrian Federal Ministry of Labour and Economy* (BMAW)*;* supported by the *Austrian Research Promotion Agency* (FFG) and *Austrian Promotional Bank* (aws).

- Contribution framework:

IPCEI ME/CT

Important Projects of Common European Interest on Micro-Electronics and Communication Technologies

Approved by the European Commission to support research, innovation and initial industrial application in the areas of Micro-Electronics and Communication Technologies across the entire value chain.