AQDT-GEN: AADL and QEMU-based Digital Twin generation for IoT testing

Mónica M. Villegas, Hernán Astudillo, Fáber Giraldo,
monica.villegas@sansano.usm.cl, hernan.astudillo@unab.cl , fdgiraldo@uniquindio.edu.co
ADEPT 2024

UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Table of contents

Introduction

- Digital twin = simulated and emulated environment that closely mimics the behavior of real-world Internet of Things (IoT) deployments
- Digital twins allow to build scenarios to analyze and test products before their physical implementation
- IoT digital twins help to identify potential defects in systems through an end-to-end implementation and specific analysis in each part of the system

Introduction

- AQDT-GEN = AADL[1] and QEMU[2]-based Digital Twin GENerator
- Designed specifically to test IoT systems
- AQDT-GEN builds the Digital Twin of an IoT system using the configurations, parameters and specifications of its AADL model

[1] http://www.openaadl.org/

[2] https://www.qemu.org/

Using AQDT-GEN

1. Create the AADL model of the IoT system to be tested, specifying components, connections and properties
2. Generate the required deployment files, parameters, and launching scripts, for both emulation and simulation, using the AADL model and the generation rules
3. Deploy the digital twin, by launching the emulation and simulation process

AQDT-GEN Components

1. **Deployment files generator**: takes the AADL model of the embedded system, and generates the deployment files and codes
2. **Simulator/Emulator**: receives the generated files and codes, and triggers the emulation/simulation process

AQDT-GEN QEMU Parameters

- The launch parameters of the QEMU emulator (OS, image path and name, kernel image, CPU type, and RAM size) are configured in the annex properties of the AADL model

AQDT-GEN Files (1/2)

AQDT-GEN has four elements:

- **AQDTGENGUI.py**: GUI to guide the generation of the required files for the simulation
- **initsimulation**: script that starts the simulation inside the QEMU emulator
- **QemuCommand**: script generated by the `sim-generator.py` that modifies the Raspbian OS image and launches QEMU with the modified image
- **rpi_stable.img**: image of Raspbian OS used for simulation

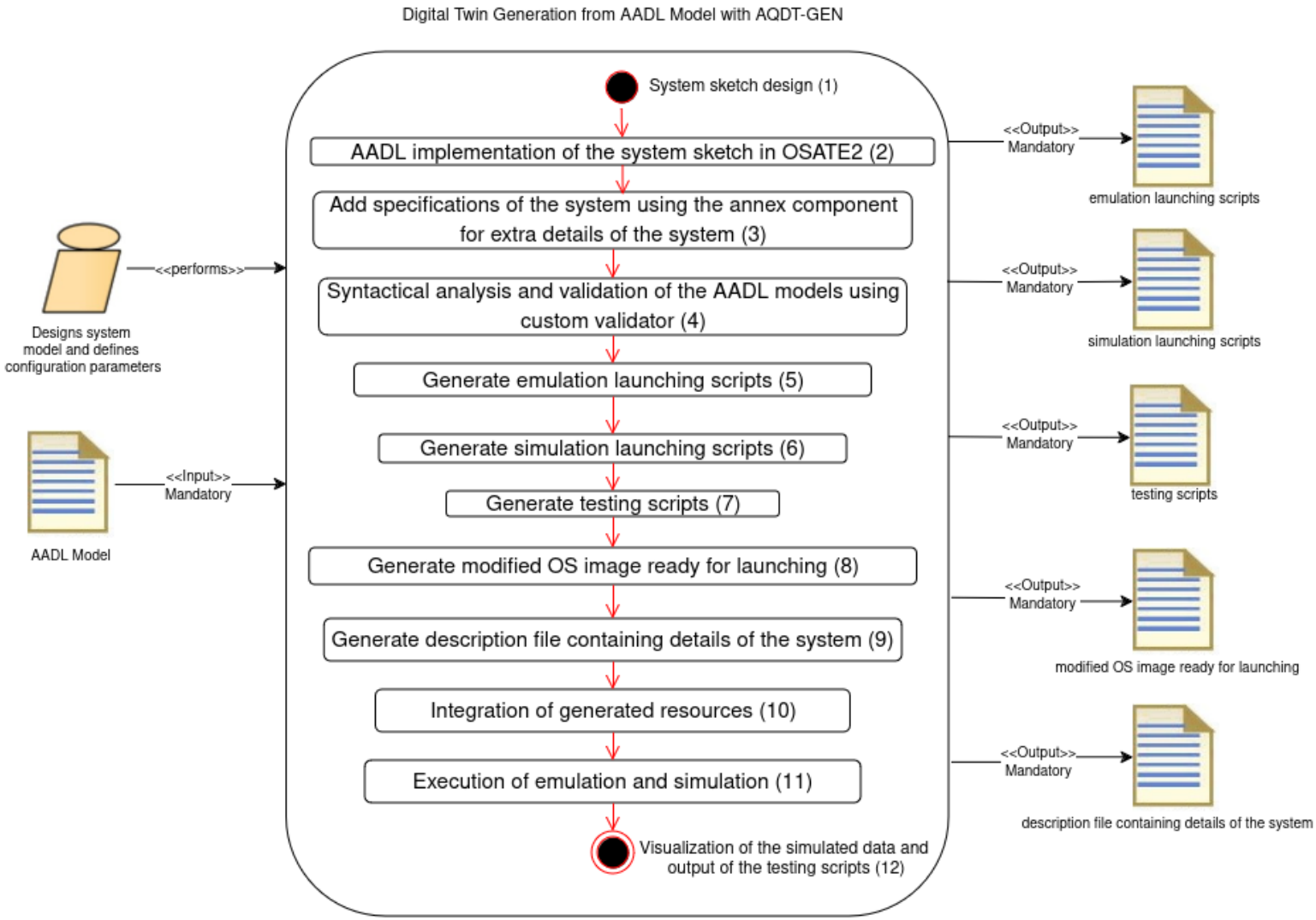
AQDT-GEN Files (2/2)

- **rules.json**: configuration rules of the QEMU commands and AADL language
- **sim-generator.py**: Python program that takes the AADL model and `rules.json`, and generates QemuCommand and `simulation.conf`
- **simulation.conf**: configuration file that describes the sensors modeled in AADL
- **simulation.c**: program launched from `initsimulation` on the QEMU emulator, that uses the data from `simulation.conf` to simulate the sensors

AQDT-GEN detailed execution

1. Design system sketch
2. Use OSATE2 to define AADL model
3. Add system specs in AADL model annex properties
4. Use custom validator to analyze the AADL model annex
5. Generate emulation launching scripts
6. Generate simulation launching scripts
7. Generate testing scripts
8. Generate modified OS image, ready for launching
9. Generate description file containing system details
10. Integrate generated resources
11. Execute emulation and simulation
12. Visualize the simulated data and testing output

Steps for DT generation



Digital Twin Generation Tool

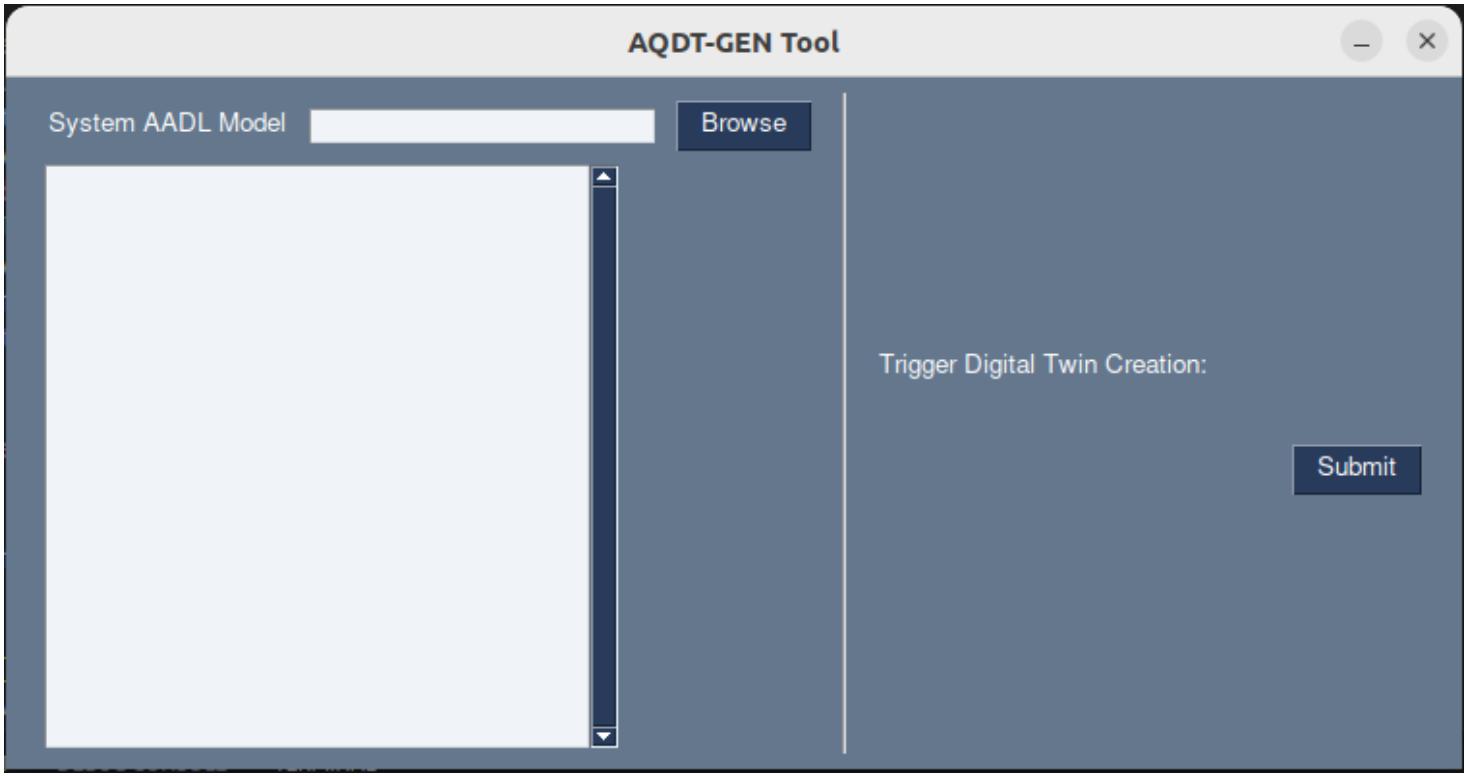


Figure 1: AQDT-GEN GUI

Deployed Digital Twin

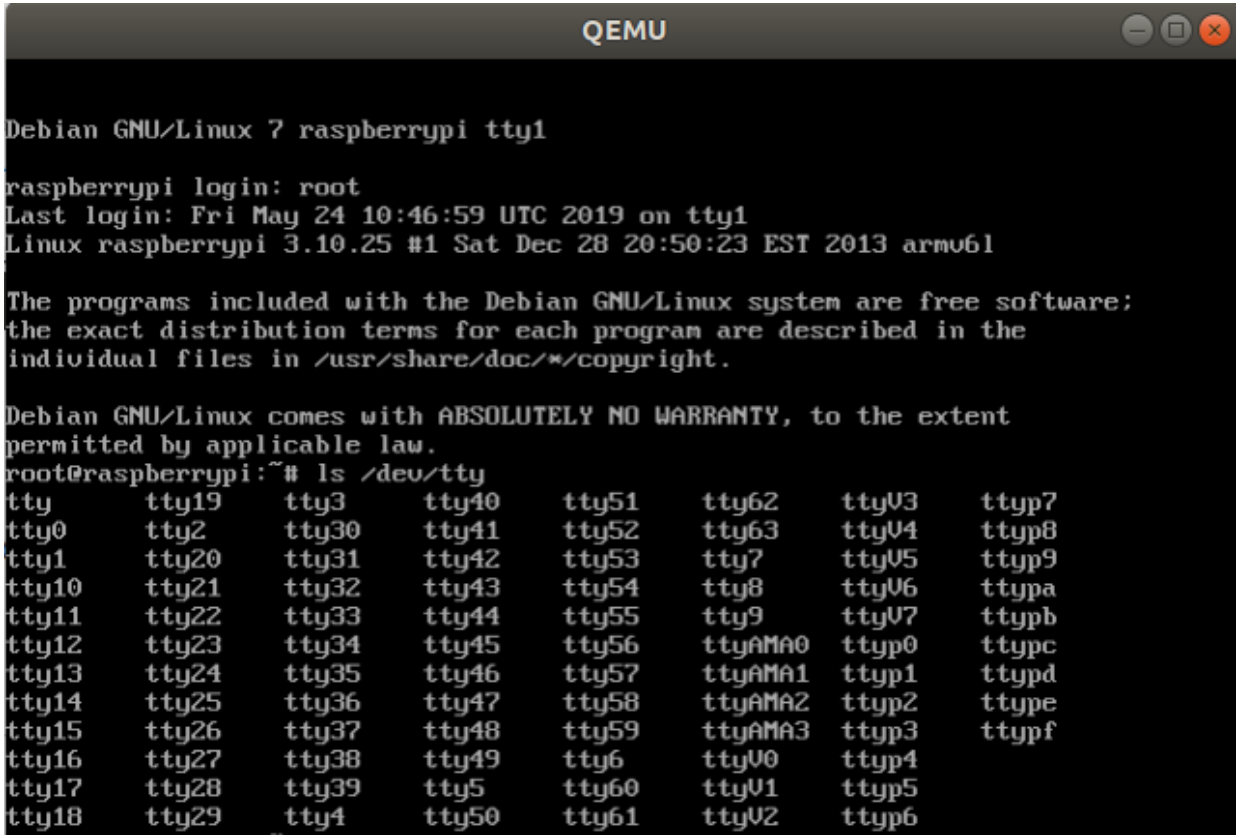


Figure 2: DT deployment

Case study - Weather station

- Data output format: c000s000g000t086r000p000h53b10020
- c000: air direction in degrees
- s000: air speed in miles per hour
- g000: air speed in miles per hour
- t086: temperature in Fahrenheit
- r000: rainfall of 1 hour in inches
- p000: rainfall of 24 hours in inches
- h53: about humidity value
- b10020: atmosphere in hPa

First weather station

```
device weatherStation1                -- MODULE 1
    features
        data_pk_out1:out data port;
    properties
        Period=>2000ms;
    annex extras {** MAC = "FF0211"; mean = "20"; deviation = "20" **};
end weatherStation1;
```

Figure 3: AADL description of the first weather station

Last weather station

```
-------------------------------------------------------------------------
device weatherStation15                 -- MODULE 15
    features
        data_pk_out1:out data port;
    properties
        Period=>1000ms;
    annex extras {** MAC = "FF02F1"; mean = "2"; deviation = "20" **};
end weatherStation15;
-------------------------------------------------------------------------
```

Figure 4: AADL description of the last weather station

Serial receiver

```
-----------------------------------------------------------
device SerialRadio1
    features
        data_pk_sensor1:in data port;
        data_pk_sensor2:in data port;
        data_pk_sensor3:in data port;
        data_pk_sensor4:in data port;
        data_pk_sensor5:in data port;
        data_pk_sensor6:in data port;
        data_pk_sensor7:in data port;
        data_pk_sensor8:in data port;
        data_pk_sensor9:in data port;
        data_pk_sensor10:in data port;
        data_pk_sensor11:in data port;
        data_pk_sensor12:in data port;
        data_pk_sensor13:in data port;
        data_pk_sensor14:in data port;
        data_pk_sensor15:in data port;
        bus_serial1:requires bus access SerialConnection1;
end SerialRadio1;
-----------------------------------------------------------

bus SerialConnection1 extends Buses::UART::UART
end SerialConnection1;
```

Figure 5: AADL description of the serial receiver

Serial receiver

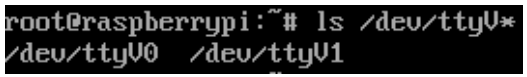


Figure 6: Serial interface of the simulated serial receiver

Weather stations output

```
root@raspberrypi:~# cat /dev/ttyV0
FF02D1c67s59g68t62r63p16h94b3
FF02C1c7s82g25t35r13p98h76b81
FF0261c88s2g43t47r71p65h99b1
FF02E1c28s79g72t89r16p29h74b74
FF0291c79s50g55t76r79p75h87b48
FF02A1c16s83g5t57r86p49h4b9
FF02B1c14s3g62t79r39p70h61b65
FF0281c15s34g56t94r36p64h22b16
FF0231c91s9g64t80r11p32h16b69
FF0241c48s25g5t57r52p20h91b9
FF0271c18s93g15t32r49p78h12b88
FF02F1c25s89g96t16r51p12h48b14
FF0251c44s64g83t34r18p53h79b34
FF02C1c95s13g41t62r93p66h3b89
FF02D1c12s50g17t12r27p22h69b80
FF0261c10s63g51t48r91p15h50b60
FF02E1c53s82g73t97r47p56h83b17
FF0291c35s1g9t30r66p50h44b59
FF02B1c68s99g0t2r6p53h37b79
FF02A1c87s86g53t49r1p57h49b44
FF0281c24s51g5t4r81p92h57b90
FF0211c93s18g82t13r17p34h16b23
FF0231c39s5g54t41r41p94h79b94
FF0221c51s45g77t76r48p34h32b29
FF0241c26s89g72t0r12p17h18b46
FF0271c80s98g59t19r3p65h12b44
FF02F1c59s44g39t7r76p42h52b5
```

Figure 7: Output of weather stations connected to serial receiver

Contact us

Mónica M. Villegas, Hernán Astudillo, Fáber Giraldo
monica.villegas@sansano.usm.cl, hernan.astudillo@unab.cl , fdgiraldo@uniquindio.edu.co
Valparaíso, Chile