INTELLIGENT TEST-CASE GENERATION FOR AUTOMATED VALIDATION OF TCUs

Lionel Belmon, Technical director
Global Crown Technology

Yijia Xu, Software Engineer
DCT Project engineering
SAGW/SAIC group

11th International CTI Symposium, December 2012, Berlin
- Lionel Belmon – Global Crown Technology, Yijia Xu – SAGW (SAIC group)

Outline

- Introduction, context, motivation
- Automatic validation with TestWeaver
- Implementation for a Simulink setup
- Applications and use cases
- TestWeaver and HiL
- Conclusions and limitations

Context and introduction

- DCT project at SAIC-SAGW
- *Model based development* of the TCU Application software
  - Auto-code of the software from *models*
- Testing and validation of the application software

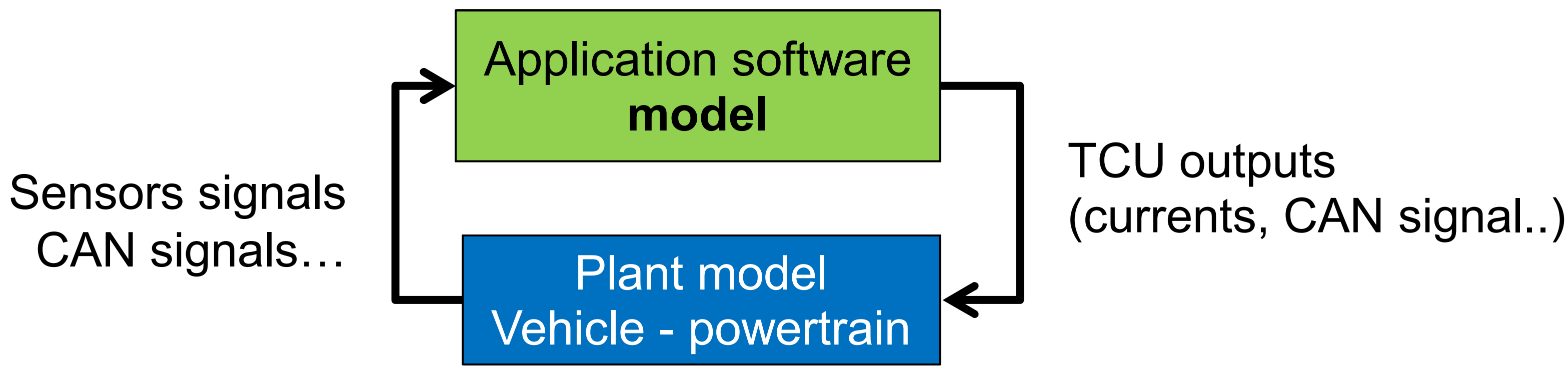
Challenges and motivation

- TCU software is in closed loop control with the vehicle
  - Open loop / module testing is not representative

- Number of test cases to be covered is huge
  - Gearshifts, drive inputs, environment…
  - How to generate quickly high coverage ?
  - How to efficiently analyze test results ?

System Under Test

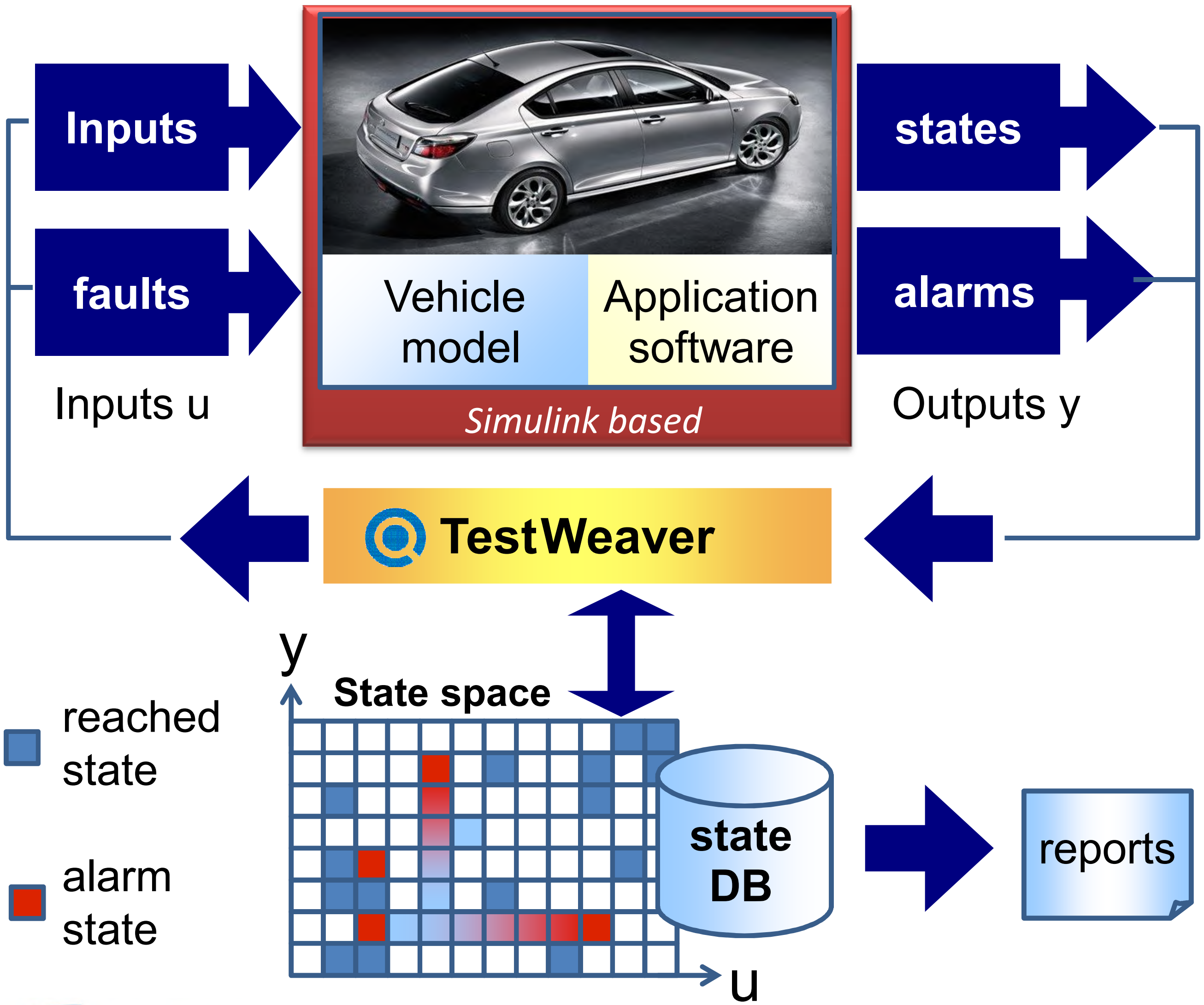
Model-in-the-Loop
Simulink based



- Application Software *model* is used for code generation

- Realistic plant model

Principles for test generation



Goals

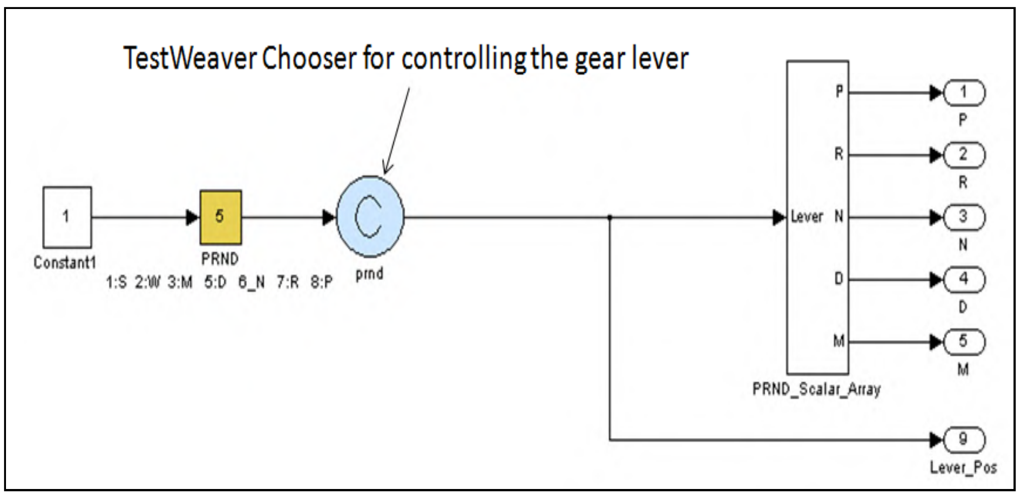
Coverage :
Drive the system to states that have not been explored before.

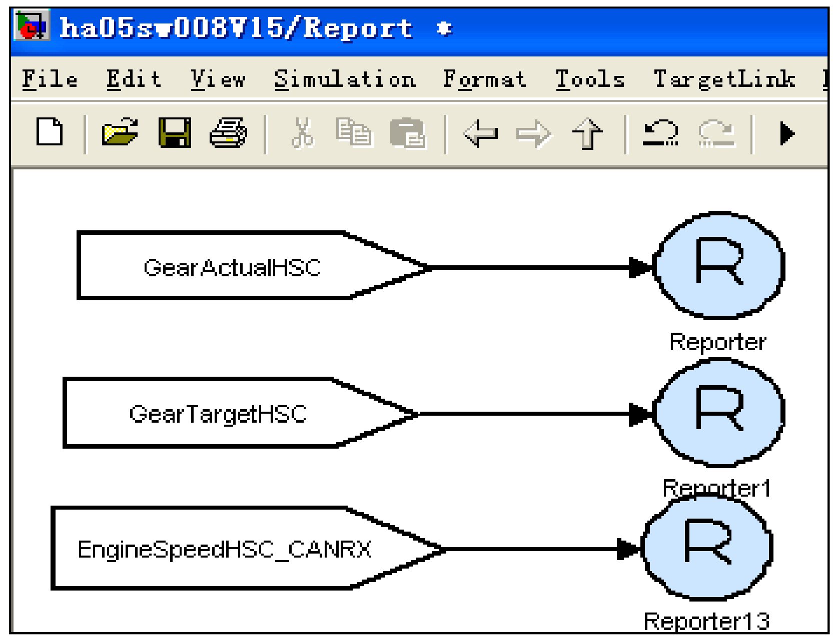
Find bugs and issues :
Change sub-optimal scenarios to generate worst-case situations.

Automation and execution speed:
Generate and analyze automatically 1000’s of scenarios

Implementation in Simulink

Addition of TestWeaver instruments inside the Simulink model
Avoid modifications of the original model
Packaging of instruments into a separate subsystem





Compilation of the SUT – Simulation speed

Use of the grt.tlc Simulink target to compile the model to a *instrumented_model.exe* through Simulink Coder / RTW

The .exe contains TCP connection to TestWeaver

The .exe is executed 1000's of times by TestWeaver

Compiled Simulink model runs around 50x times faster than interpreted model

2000 driving sequences of 1 minute each
generated/evaluated in less then 2 hours.

Reporting system

Reported variables :

Actual Gear
Target Gear
Car speed, acceleration
Engine speed/torques
Shafts speeds/torques
Gearshift time counters
Clutches temperature
Synchronizers positions
…
Around 15 reports analyzing tested scenarios

State coverage report

currentGear	targetGear	slope	engineTorque	scenarios
		flat	medium	s4
	1	downhill	brake	s2
		flat	low	s0
		uphill		s1
		flat	medium	s0, s4, s3
		flat	high	s7, s6
	2	downhill	brake	s2
		downhill	aroundZero	s2
		downhill	low	s2
		flat	high	s7, s6
2	1	downhill	brake	s2
		downhill	aroundZero	s2
		downhill	low	s2
		downhill	medium	s2
	2	downhill	brake	s2
		flat	high	s7, s6
	3	downhill	brake	s2
		flat	high	s7, s6
3	2	downhill	brake	s2
	3	downhill	brake	s2
		flat	high	s7, s6
	4	downhill	brake	s2
		flat	high	s7, s6
4	3	downhill	brake	s2
	4	downhill	brake	s2
		flat	high	s7, s6
	5	downhill	brake	s2
		flat	high	s7, s6

Reporting system – Switch of test paradigm

Script based testing :
How to write a script that will go from 0% accelerator position to 100% accelerator position during a gearshift ?

TestWeaver approach :
write a query which will find *all* generated scenarios where such case happen

```
where
        state.targetGear.value != state.currentGear.value // detect a gearshift
and
        state.accel Pedal.value='0'  // current position is 0%
and
        nextState.accelPedal.value='100'  //next state is 100% pedal
```

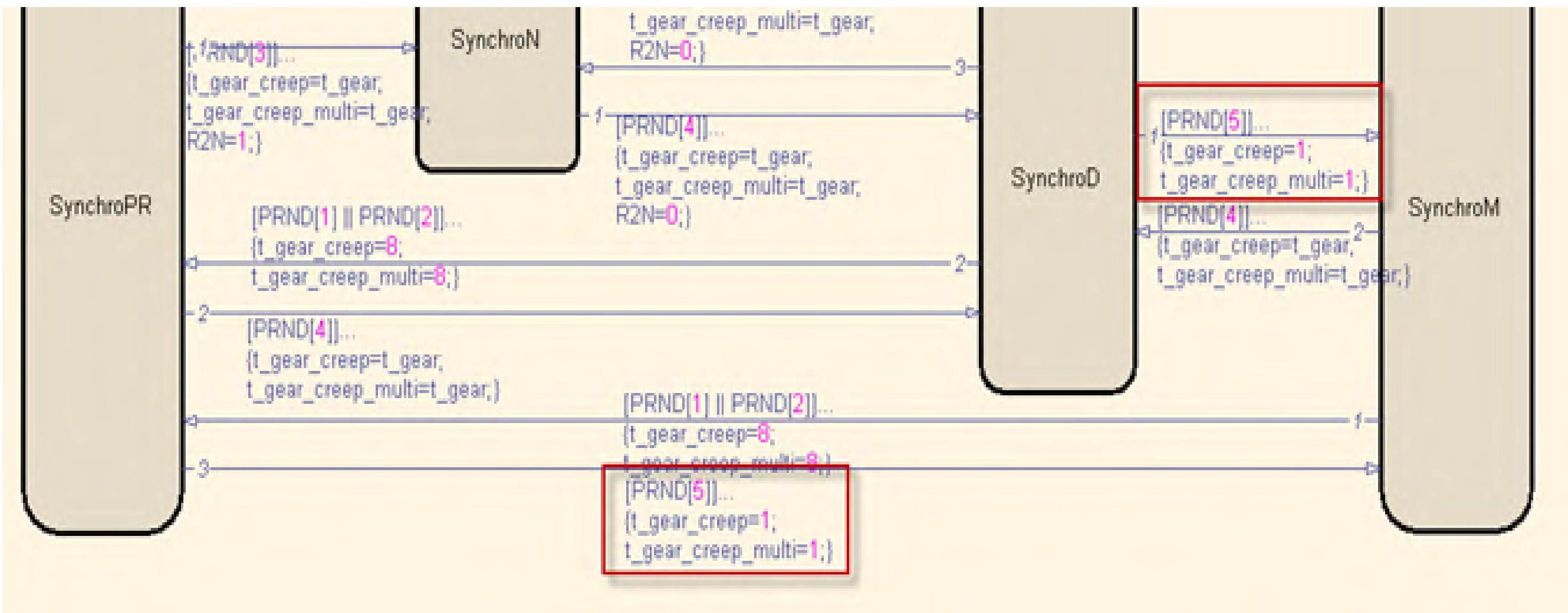
Application and use cases - debugging

TestWeaver found engine speed above prescribed limit of 7300 rpm after a sequence of changes in PRND.

time	Input sequence	faults	Actual gear	Tgt gear	Odd shaft	Even shaft	alarms	duration	Car speed	Car acceleration	Engine speed
31.470	-	(none)	5	4	5	6	(none)	0.030	40..60	-0.2..0.5	1400..6010
31.500	prnd=R	(none)	5	4	5	6	(none)	0.020	40..60	-0.2..0.5	1400..6010
31.520	-	(none)	5	4	5	6	(none)	0.080	40..60	-0.2..0.5	1400..6010
31.600	slope=-20 prnd=W	(none)	5	R	5	6	(none)	0.005	40..60	-0.2..0.5	1400..6010
31.605	-	(none)	N	R	5	6	(none)	0.015	40..60	-0.2..0.5	1400..6010
31.620	-	(none)	N	R	5	6	(none)	0.010	40..60	-0.2..0.5	1400..6010
31.630	-	(none)	N	4	5	6	(none)	0.010	40..60	-0.2..0.5	1400..6010
31.640	-	(none)	N	4	5	6	(none)	0.010	40..60	-0.2..0.5	1400..6010
31.650	-	(none)	N	4	5	0	(none)	0.050	40..60	-0.2..0.5	1400..6010
31.700	AccelPedal=0 prnd=M	(none)	N	4	5	0	(none)	0.020	40..60	-0.2..0.5	1400..6010
31.720	-	(none)	N	4	5	0	(none)	0.020	40..60	-0.2..0.5	1400..6010
31.740	-	(none)	N	4	5	0	(none)	0.010	40..60	-0.2..0.5	1400..6010
31.750	-	(none)	N	1	5	0	(none)	0.055	40..60	-0.2..0.5	1400..6010
31.805	-	(none)	N	1	5	4	(none)	0.190	40..60	-0.2..0.5	1400..6010
31.995	-	(none)	N	1	0	4	(none)	0.272	40..60	-0.2..0.5	1400..6010
32.267	-	(none)	N	1	0	4	(none)	0.931	60..80	-0.2..0.5	1400..6010
33.198	-	(none)	N	1	0	4	(none)	0.001	60..80	1..1.5	1400..6010
33.199	-	(none)	N	1	0	4	(none)	0.001	60..80	-0.2..0.5	1400..6010
33.200	-	(none)	N	1	1	4	(none)	0.190	60..80	-0.2..0.5	1400..6010
33.390	-	(none)	N	1	1	0	(none)	0.385	60..80	-0.2..0.5	1400..6010
33.775	-	(none)	N	1	1	0	(none)	0.105	60..80	-5...-0.2	1400..6010
33.880	-	(none)	N	1	1	2	(none)	0.323	60..80	-5...-0.2	1400..6010
34.203	-	(none)	N	1	1	2	(none)	0.041	40..60	-5...-0.2	1400..6010
34.244	-	(none)	N	1	1	2	EngineSpeedHSC_CANRX=6010..7310 nEng_SENS_PT_OUT=6010..7310	0.204	40..60	-5...-0.2	6010..7310
34.448	-	(none)	N	1	1	2	-	0.123	40..60	-0.2..0.5	6010..7310
34.571	-	(none)	N	1	1	2	EngineSpeedHSC_CANRX=7310..10000 nEng_SENS_PT_OUT=7310..10000	0.034	40..60	-0.2..0.5	7310..1000
34.605	-	(none)	1	1	1	2	-	0.225	40..60	-0.2..0.5	7310..1000
34.830	-	(none)	1	2	1	2	-	0.242	40..60	-0.2..0.5	7310..1000

Application and use cases - debugging

Replay of the scenario in Simulink for debugging
Bug is localized as an incorrect state transition
Correction implementation
Replay of the scenario with corrected logic
Problem solved



TestWeaver and HiL

TestWeaver can be connected to various HiL systems (dSPACE, ETAS, NI…) to generate and analyze scenarios

HiL+TestWeaver setup is more complex than MiL

XCP/UDS protocol to the ECU, read/reset scripts

Additional interfaces with the HiL model

HiL setups are *slow (real-time)*. MiL can be much faster.

Generating 2000 scenarios on HiL would take 40hours

Generating 2000 scenarios on MiL takes 2 hours

For this project, TestWeaver was not used for test generation on the HiL. Instead MiL scenarios were exported to HiL.

Motivation of exporting scenarios for HiL

Back to back testing : Verification of the real TCU behavior (instead of model) for specific scenarios

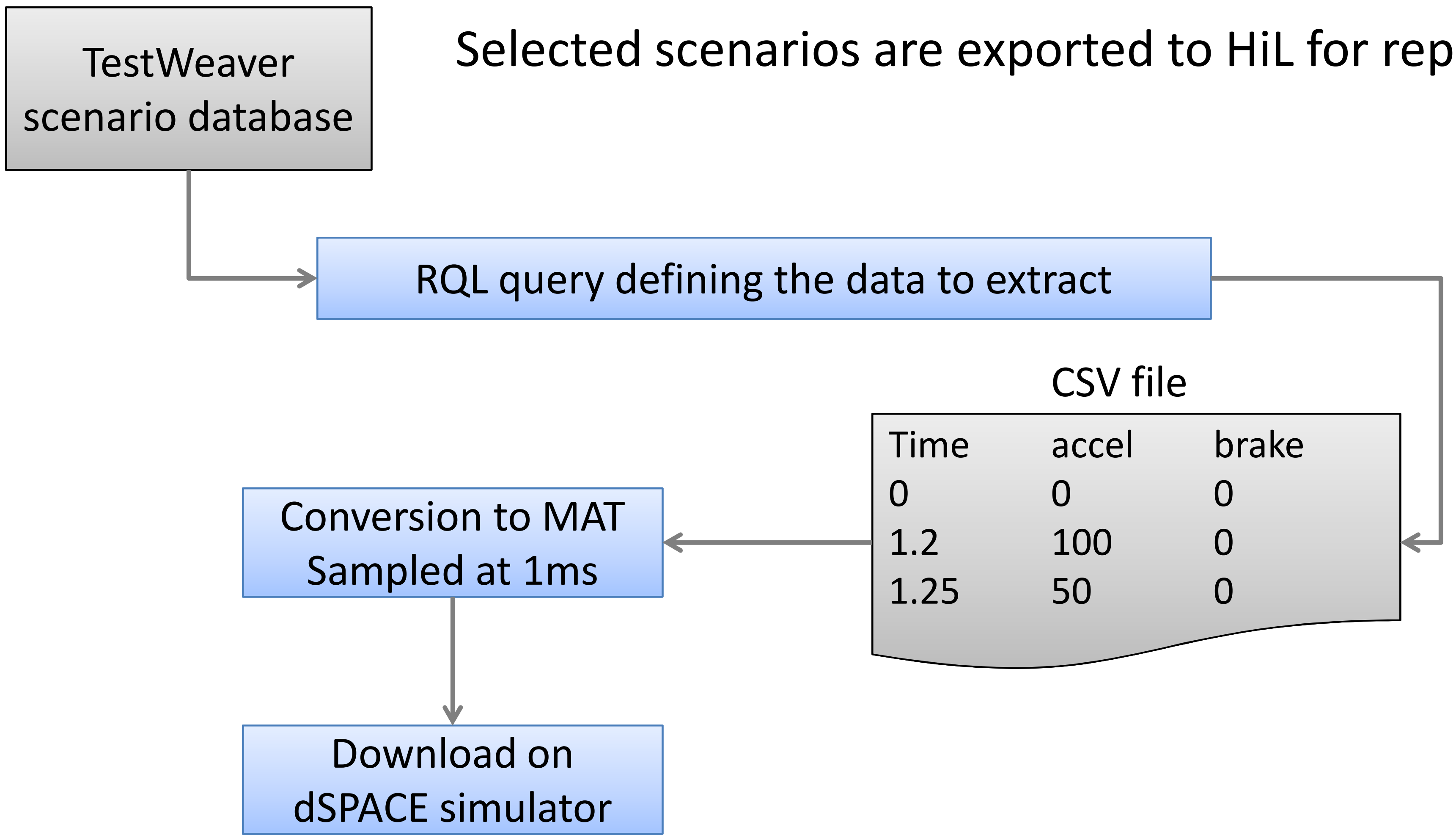
Ready-made scenarios database for HiL, reduce HiL scripting work

HiL setup replays much faster than Simulink interpreted model

Analyzing data on CANape is much easier than looking at scopes on Simulink

Exporting a scenario for HiL replay

Selected scenarios are exported to HiL for replay



Conclusions and perspectives

A method for automatic large coverage testing of Transmission Control Unit has been established, helpful for debugging and validating complex controls

Limitations :

Calibration parameters are not easily handled in the Simulink model.
The TCU model is tested but not the TCU **production c-code**.
System state coverage can be measured but not the **code coverage.**
Replaying is slow due to the interpreted Simulink model.
No connection with **measurement and calibration tools** such as CANape, no convenient writing/reading of measurements files (.mdf)
CAN configurations (dbc files) are not included in the Simulink-based test

Above issues can be solved with the
Virtual ECU Simulation using **QTronic Silver**