Une école de l'IMT

Model-Driven Engineering for Safety, Security and Performance: SysML-Sec

Ludovic APVRILLE
ludovic.apvrille@telecom-paristech.fr

InS3PECT'2017

Outline

Context: Security for Embedded Systems

Embedded systems

SysML-Sec

Method

SysML-Sec

Case study and Demo

Case Study and Demo

Conclusion

Conclusion, future work and references

Examples of Threats

Transport systems

- Use of exploits in Flight Management System (FMS) to control ADS-B/ACARS [Teso 2013]
- Remote control of a car through Wifi [Miller 2015] [Tecent 2017]

Wired
TAKING ACTION FOR YOU
TAKING ACTION FOR YOU
HACKERS USE WIFI TO TAKE CONTROL OF VEHICLES
INTRUDERS CAN DISABLE BRAKES, STEER & KILL ENGINE
abc ACTION NEWS
11:04 84°

(C) Wired - ABC News

Medical appliances

- Infusion pump vulnerability, April 2015. http://www.scip.ch/en/?vuldb.75158

(C) Hospira

Examples of Threats (Cont.)

Internet of Things

- Proof of concept of attack on IZON camera [Stanislav 2013]

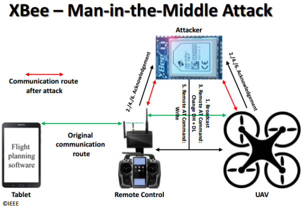
- Vulnerability on fitbit [Apvrille 2015]

Geek usages for your Fitbit Flex Tracker
Et si je cours à 4 pattes, je fais plus de pas ?
Fitness tracker
A. Apvrille
FortiGuard Labs
Fortinet
Hack.Lu
October 2015
FORTINET

A. Apvrille, Hack.lu'2015

(C) beforeitnews

- Hacking a professional drone [Rodday 2016]

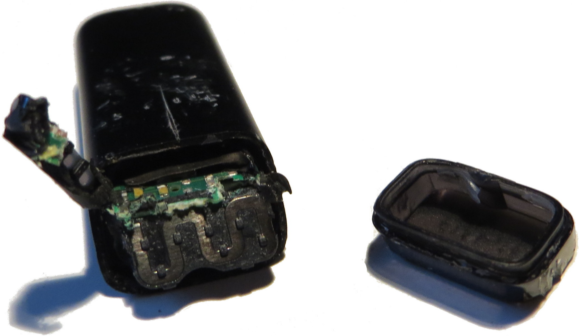

XBee – Man-in-the-Middle Attack
Attacker
Communication route after attack
2./4./6. Acknowledgement
2./4./6. Acknowledgement
5. Remote AT Command: Write
3. Remote AT Command: Change DH + DL
1. Broadcast
Original communication route
Flight planning software
Tablet
Remote Control
UAV
©IEEE

N. Rodday, BlackHat Asia'2016

Finding Vulnerabilities on IoTs

What's inside? Let's look together!

Inside a Fitbit

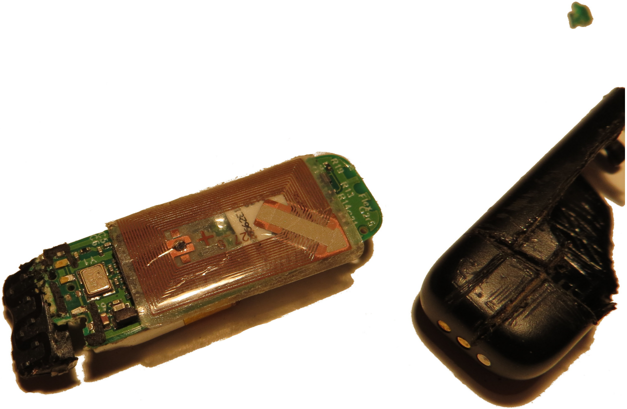


Don't try this at home!

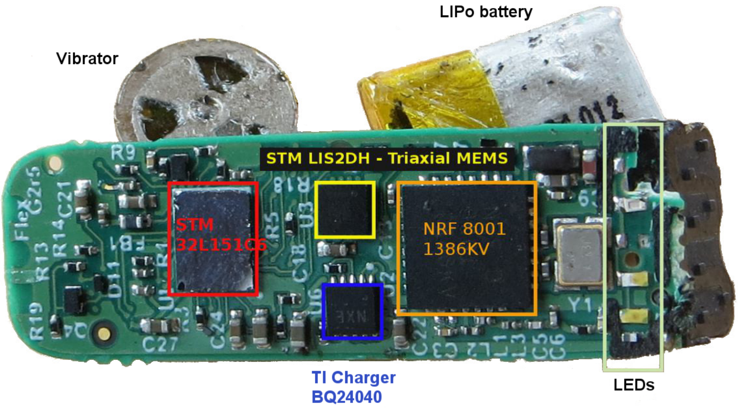
Inside a Fitbit (Cont.)

Again: don’t try this at home!

Inside a Fitbit (Cont.)



Fitbit: Hardware Components



Then, How to Identify Vulnerabilities?

Investigations

- JTAG interface
- Testing ports
- Firmware analysis
- Memory dump
- . . .

You want to better resist this?

Develop your system with security in mind from the very beginning

Our solution: SysML-Sec, supported by TTool

Designing Safe and Secure Embedded Systems: SysML-Sec

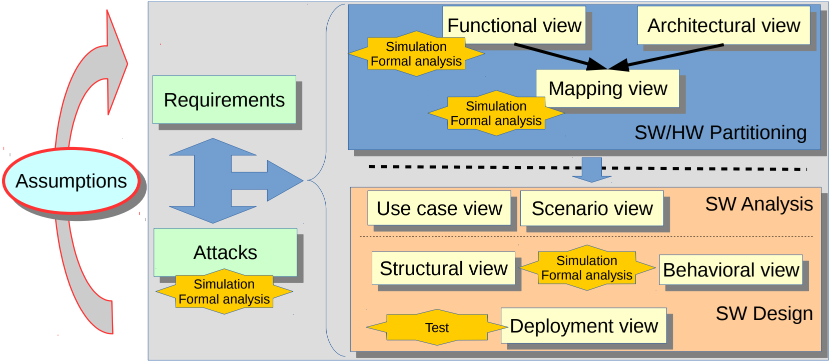
Main idea

- **Holistic approach**: bring together experts in embedded systems, system architects, system designers and security experts

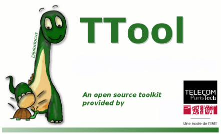
Common issues (addressed by SysML-Sec):

- Adverse effects of security over safety/real-time/performance properties
  - Commonly: only the design of security mechanisms
- Hardware/Software partitioning
  - Commonly: no support for this in tools/approaches in MDE and security approaches

SysML-Sec: Methodology



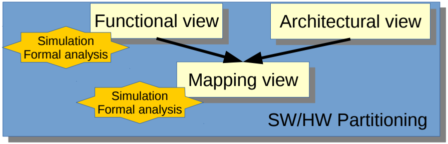
Fully supported by TTool



Partitioning

Before mapping

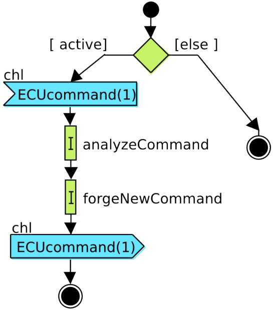
- Security mechanisms can be captured but not verified

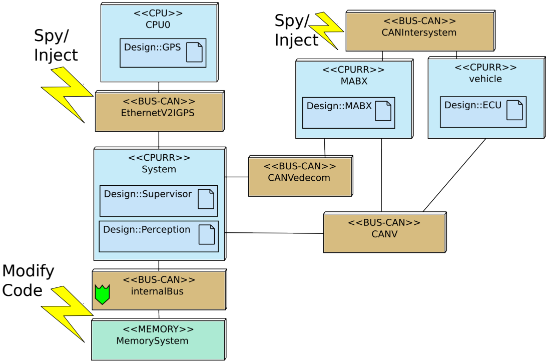


After mapping

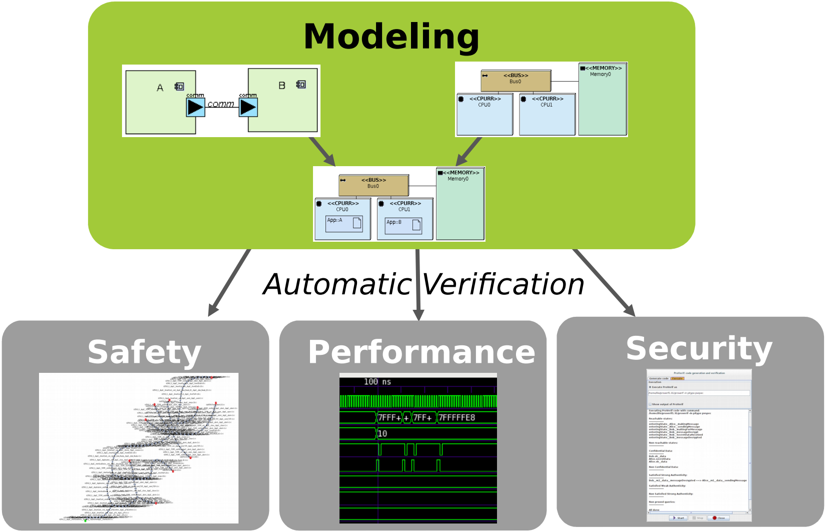
- Impact of security mechanisms on performance and safety
  - e.g. increased latency when inserting security mechanisms
- Verify security (confidentiality, authenticity) according to possible attacks
  - Depends on the attacker capabilities
  - Whether different HW elements are or not on the same die
  - Where to store the cryptographic materials (keys)
  - Where to perform encrypt/decrypt operations

Attacker Model





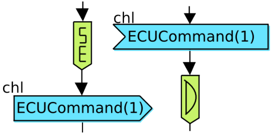
Partitioning Verification



```
Executing ProVerif code with command:
/home/le/proverif1.91/proverif -in pitype pvspec

Reachable states:
--------------
enteringState__Alice__makingMessage
enteringState__Alice__sendingMessage
enteringState__Bob__waitingForMessage
enteringState__Bob__messageDecrypt
enteringState__Bob__SecretDataReceived
enteringState__Bob__messageDecrypted

Non reachable states:
--------------

Confidential Data:
--------------
Bob.sk__data
Alice.secretData
Alice.sk__data

Non Confidential Data:
--------------

Satisfied Strong Authenticity:
--------------
Bob__m2__data__messageDecrypted ==> Alice__m1__data__sendingMessage

Satisfied Weak Authenticity:
--------------

Non Satisfied Strong Authenticity:
--------------

Non proved queries:
--------------

All done
```

Safety and Security Mechanisms

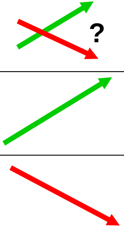
Data Encryption/ Authentication



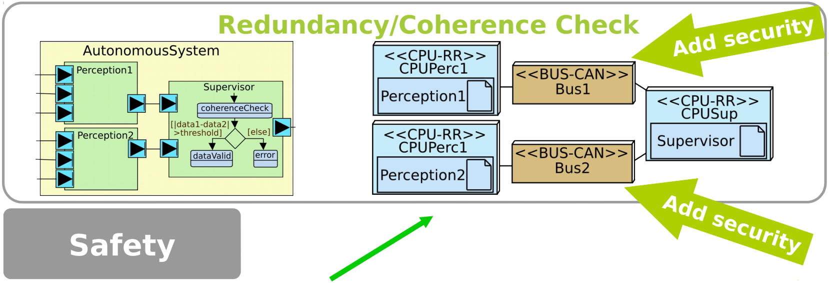
Safety

Security

Performance



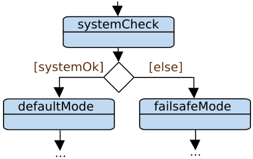
Safety and Security Mechanisms



Safety	↗
Security	↗↘ ?
Performance	↗↘ ?

Safety and Security Mechanisms

Failsafe mode

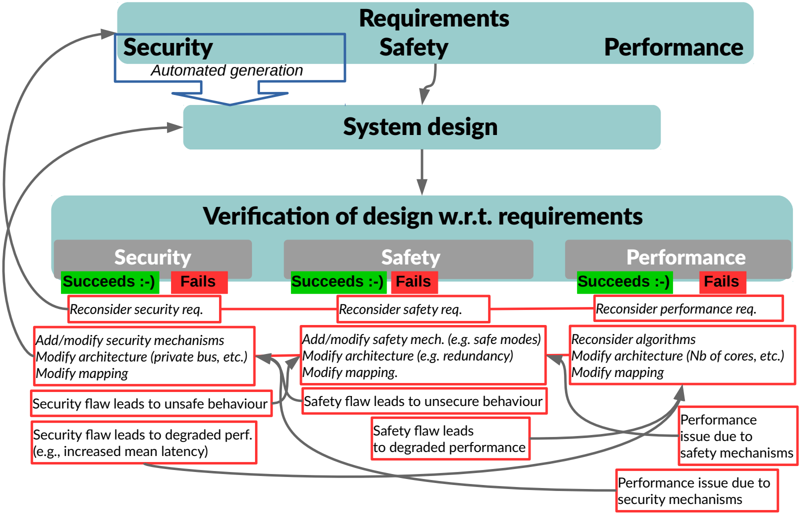


Safety ↗

Security ↗ ↘ ?

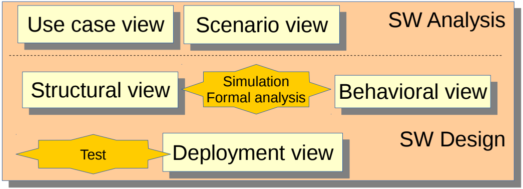
Performance ↗ ↘ ?

Partitioning Approach



Requirements
Security
Automated generation
Safety
Performance
System design
Verification of design w.r.t. requirements
Security
Succeeds :-)
Fails
Safety
Succeeds :-)
Fails
Performance
Succeeds :-)
Fails
Reconsider security req.
Reconsider safety req.
Reconsider performance req.
Add/modify security mechanisms
Modify architecture (private bus, etc.)
Modify mapping
Add/modify safety mech. (e.g. safe modes)
Modify architecture (e.g. redundancy)
Modify mapping.
Reconsider algorithms
Modify architecture (Nb of cores, etc.)
Modify mapping
Security flaw leads to unsafe behaviour
Safety flaw leads to unsecure behaviour
Security flaw leads to degraded perf. (e.g., increased mean latency)
Safety flaw leads to degraded performance
Performance issue due to safety mechanisms
Performance issue due to security mechanisms

SysML-Sec: SW Design

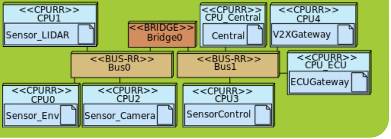


- Precise model of security mechanisms (security protocols)
- Proof of security properties : confidentiality, authenticity
- Channels between software blocks can be defined as private or public
  - This should be defined according to the hardware support defined during the partitioning phase

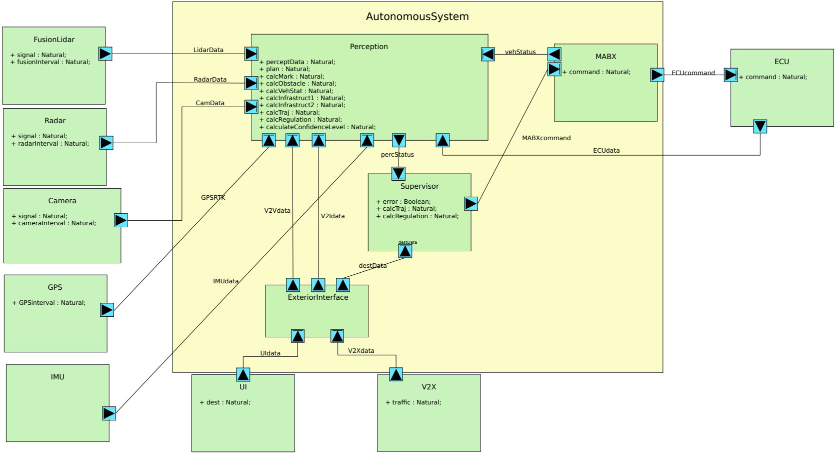
Case Study: Autonomous Vehicle

Verification

Model



Functional View



AutonomousSystem

FusionLidar
+ signal : Natural;
+ fusionInterval : Natural;

Radar
+ signal : Natural;
+ radarInterval : Natural;

Camera
+ signal : Natural;
+ cameraInterval : Natural;

GPS
+ GPSinterval : Natural;

IMU

Perception
+ perceptData : Natural;
+ plan : Natural;
+ calcMark : Natural;
+ calcObstacle : Natural;
+ calcVehStat : Natural;
+ calcInfrastruct1 : Natural;
+ calcInfrastruct2 : Natural;
+ calcTraj : Natural;
+ calcRegulation : Natural;
+ calculateConfidenceLevel : Natural;

MABX
+ command : Natural;

ECU
+ command : Natural;

Supervisor
+ error : Boolean;
+ calcTraj : Natural;
+ calcRegulation : Natural;

ExteriorInterface

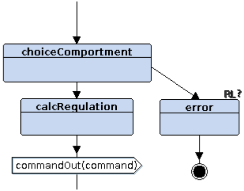
UI
+ dest : Natural;

V2X
+ traffic : Natural;

Safety Verification (Before Mapping)

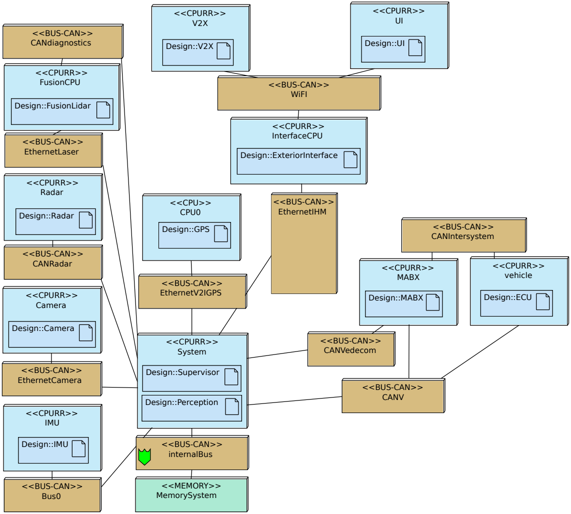
Reachability/Liveness

Queries



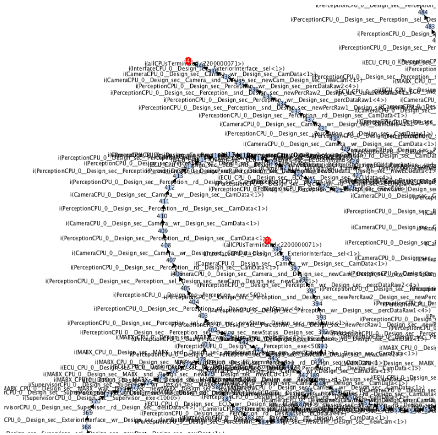
Safety Pragma
A[] Supervisor.running
Perception.distance<threshold -->
Supervisor.brakingOrder

Architecture and Mapping Views

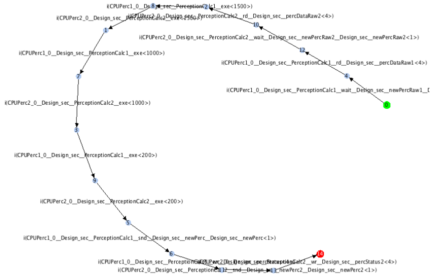


Safety Verification (After Mapping)

Reachability Graph

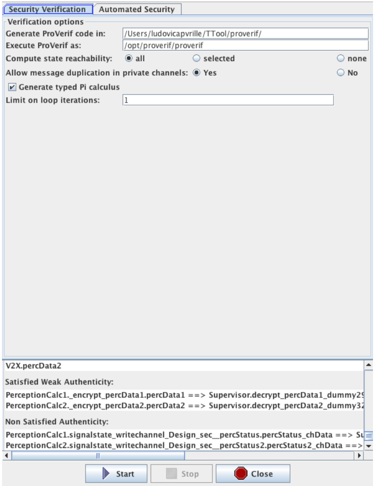

i(allCPUsTerminated<2200000071>)
i(InterfaceCPU_0__Design_sec__ExteriorInterface__sel<1>)
i(CameraCPU_0__Design_sec__Camera__wr__Design_sec__CamData<1>)
i(CameraCPU_0__Design_sec__Camera__snd__Design_sec__newCam__Design_sec__newCam<1>)
i(PerceptionCPU_0__Design_sec__Perception__wr__Design_sec__percDataRaw2<4>)
i(PerceptionCPU_0__Design_sec__Perception__snd__Design_sec__newPercRaw2__Design_sec__newPercRaw2<1>)
i(PerceptionCPU_0__Design_sec__Perception__wr__Design_sec__percDataRaw1<4>)
i(PerceptionCPU_0__Design_sec__Perception__snd__Design_sec__newPercRaw1__Design_sec__newPercRaw1<1>)
i(PerceptionCPU_0__Design_sec__Perception__rd__Design_sec__CamData<1>)
i(CameraCPU_0__Design_sec__Camera__wr__Design_sec__CamData<1>)
i(PerceptionCPU_0__Design_sec__Perception__exe<500>)
i(SupervisorCPU_0__Design_sec__Supervisor__exe<1000>)
i(PerceptionCPU_0__Design_sec__Perception__wr__Design_sec__newStatus<4>)
484
483
431
430
413
412
411
410
409
408
407
406
405
404
403
401
399
398
397
396
395
394
393
391
390
389
370
369
368

Minimized RG


i(CPUPerc1_0__Design_sec__PerceptionCalc1__exe<1500>)
i(CPUPerc2_0__Design_sec__PerceptionCalc2__exe<1500>)
i(CPUPerc2_0__Design_sec__PerceptionCalc2__rd__Design_sec__percDataRaw2<4>)
i(CPUPerc2_0__Design_sec__PerceptionCalc2__wait__Design_sec__newPercRaw2__Design_sec__newPercRaw2<1>)
i(CPUPerc1_0__Design_sec__PerceptionCalc1__rd__Design_sec__percDataRaw1<4>)
i(CPUPerc1_0__Design_sec__PerceptionCalc1__wait__Design_sec__newPercRaw1__[
i(CPUPerc1_0__Design_sec__PerceptionCalc1__exe<1000>)
i(CPUPerc2_0__Design_sec__PerceptionCalc2__exe<1000>)
i(CPUPerc1_0__Design_sec__PerceptionCalc1__exe<200>)
i(CPUPerc2_0__Design_sec__PerceptionCalc2__exe<200>)
i(CPUPerc1_0__Design_sec__PerceptionCalc1__snd__Design_sec__newPerc__Design_sec__newPerc<1>)
i(CPUPerc2_0__Design_sec__PerceptionCalc2__wr__Design_sec__percStatus2<4>)
i(CPUPerc2_0__Design_sec__PerceptionCalc2__snd__Design_sec__newPerc2__Design_sec__newPerc2<1>)
0
4
12
10
2
8
1
7
3
9
5
6
11
13
14

Security Verification

Dialog window



Verification options

Generate ProVerif code in: /Users/ludovicapvrille/TTool/proverif/

Execute ProVerif as: /opt/proverif/proverif

Compute state reachability: ● all ○ selected ○ none

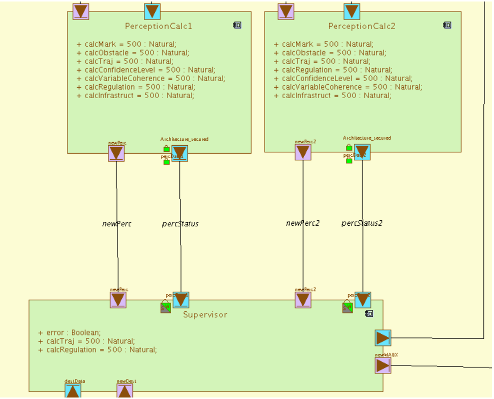
Allow message duplication in private channels: ● Yes ○ No

☑ Generate typed Pi calculus

Limit on loop iterations: 1

```
V2X.percData2

Satisfied Weak Authenticity:
PerceptionCalc1._encrypt_percData1.percData1 ==> Supervisor.decrypt_percData1_dummy2
PerceptionCalc2._encrypt_percData2.percData2 ==> Supervisor.decrypt_percData2_dummy3

Non Satisfied Authenticity:
PerceptionCalc1.signalstate_writechannel_Design_sec__percStatus.percStatus_chData ==> Su
PerceptionCalc2.signalstate_writechannel_Design_sec__percStatus2.percStatus2_chData ==>
```

Backtracing



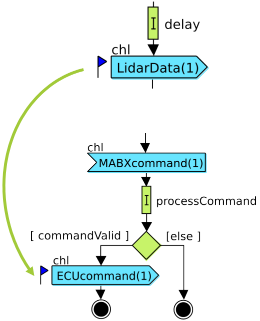
```
PerceptionCalc1
+ calcMark = 500 : Natural;
+ calcObstacle = 500 : Natural;
+ calcTraj = 500 : Natural;
+ calcConfidenceLevel = 500 : Natural;
+ calcVariableCoherence = 500 : Natural;
+ calcRegulation = 500 : Natural;
+ calcInfrastruct = 500 : Natural;
```

```
PerceptionCalc2
+ calcMark = 500 : Natural;
+ calcObstacle = 500 : Natural;
+ calcTraj = 500 : Natural;
+ calcRegulation = 500 : Natural;
+ calcConfidenceLevel = 500 : Natural;
+ calcVariableCoherence = 500 : Natural;
+ calcInfrastruct = 500 : Natural;
```

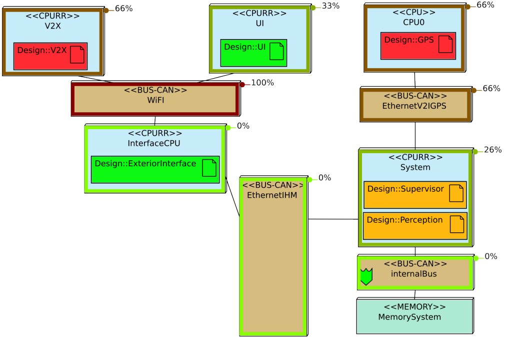
```
Supervisor
+ error : Boolean;
+ calcTraj = 500 : Natural;
+ calcRegulation = 500 : Natural;
```

Performance Verification

Latency



Bus/CPU Load



Conclusion and Future Work

Achievements: SysML-Sec

- Methodology for designing safe and secure embedded systems
- Fully supported by TTool
- Applied to different domains, e.g., automotive systems, IoTs, malware

Future work

- Security risk assistance and backtracing
- Improve security provers
- Assistance to handle conflicts between security/safety/performance
    - Design space exploration

To Go Further ...

Web sites

- https://sysml-sec.telecom-paristech.fr
- https://ttool.telecom-paristech.fr

References

- Ludovic Apvrille, Yves Roudier, "SysML-Sec: A SysML Environment for the Design and Development of Secure Embedded Systems", Proceedings of the INCOSE/APCOSEC 2013 Conference on system engineering, Yokohama, Japan, September 8-11, 2013.
- Ludovic Apvrille, Yves Roudier, "Designing Safe and Secure Embedded and Cyber-Physical Systems with SysML-Sec", Chapter in Model-Driven Engineering and Software Development, p293–308, Springer International Publishing, 2015