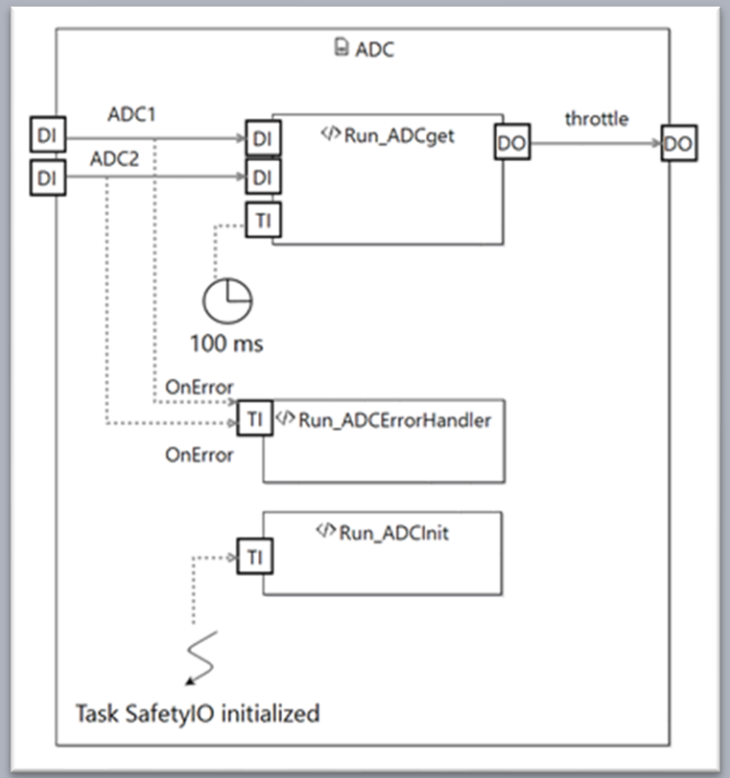
embeddedworld
Exhibition&Conference
... it´s a smarter world

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Modeling and code generation for safety critical systems

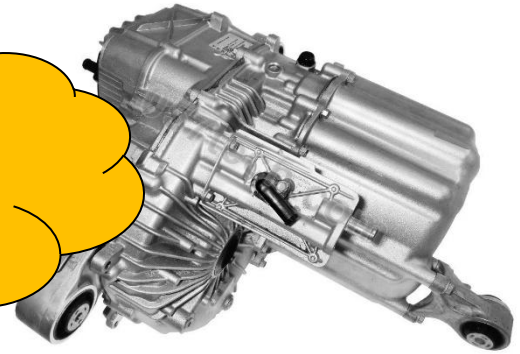
Thomas Barth (h_da)
Prof. Dr.-Ing. Peter Fromm (h_da)



Use Case example – Electronic throttle control

Synchronicity

Informal notation



Hazard & Risks

Safety Goals (SIL)

Safety Requirements

Different types of artifacts

Functional Architect

ecture

Qualification

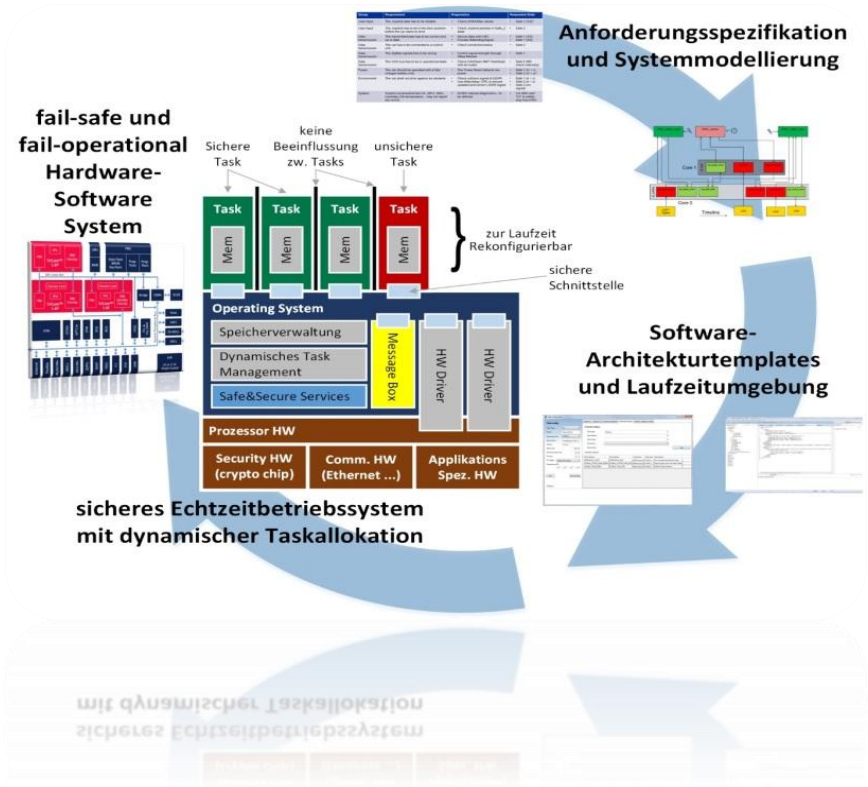
Documentation

Analysis/ Report

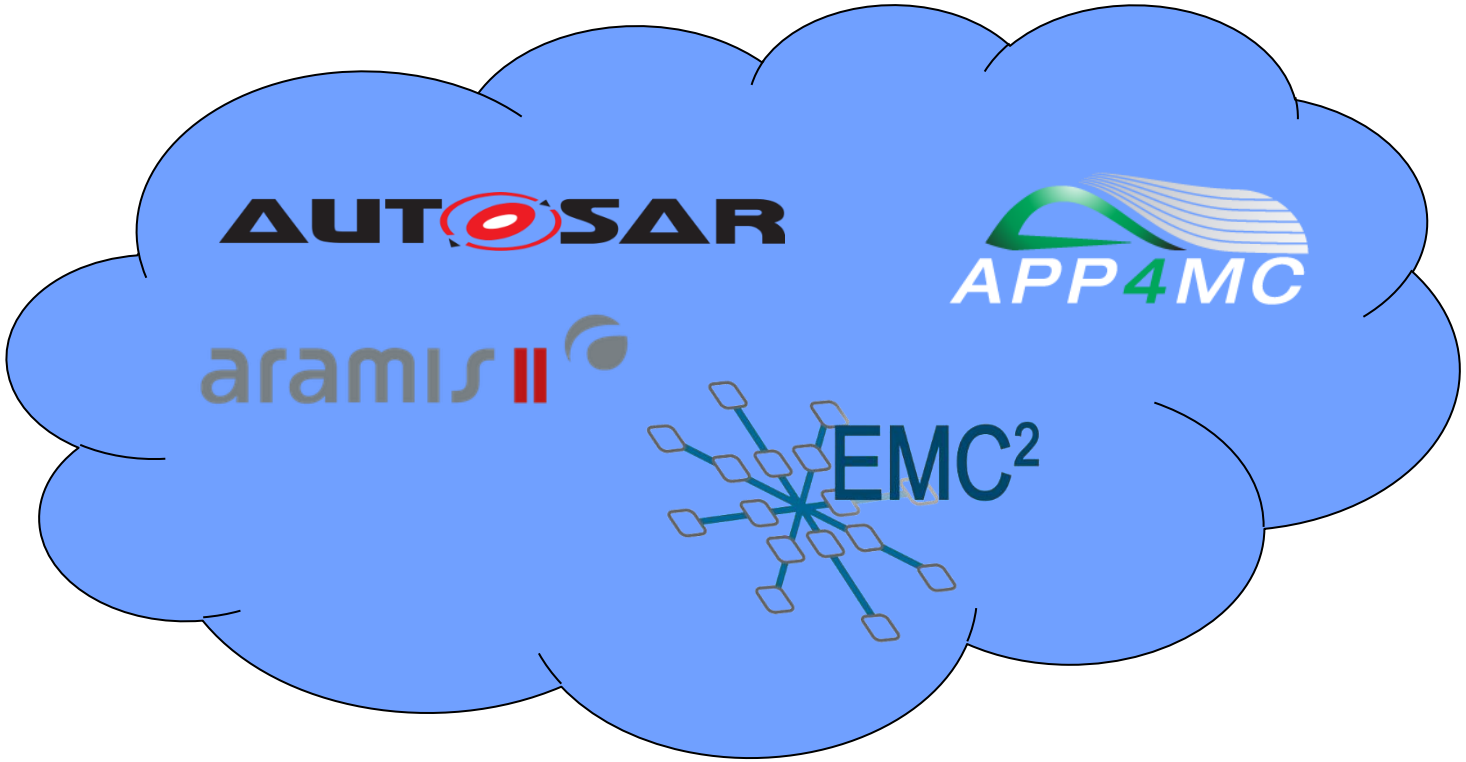
Code

Content

- Framework goals
- Tooling
  - Domain specific language
  - A common model per system
  - Views on a system
- Runtime environment
  - Signal-layer
  - Activation Engine
- Conclusion / Summary / Discussion



Goals: Differentiation from existing solutions

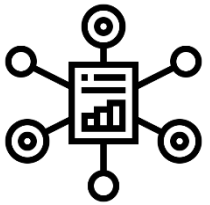


The goal of the project is a integrated solution to model, assess and generate multicore safety systems for real-time embedded applications. The tool and the generated code shall be lightweight, easy to understand and targeting small and medium sized companies as well as academia.

Goals: Overview

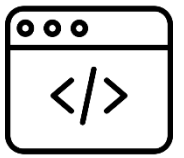
We need:

- An easily available framework to model a complex safety system and supporting the generation of code, reports and documentation
  ✓ Eclipse Modeling Framework – Covered in EWC 19



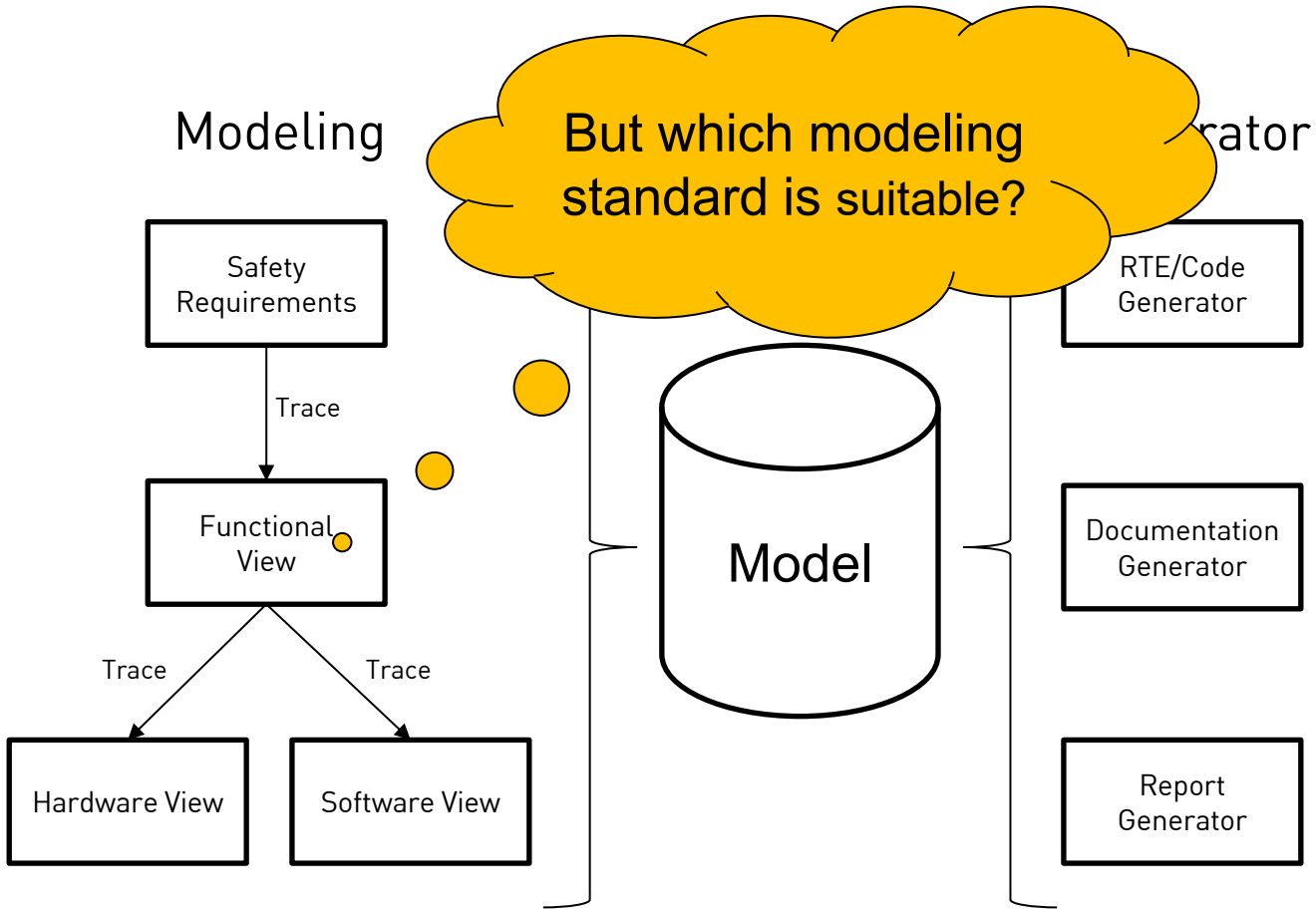
- An appropriate modelling language, supporting the modelling of signal flow and runtime behaviour on multicore systems

- An easy to understand runtime environment architecture, custom generated for each model, supporting safety multicore targets



Icons: Eucalyp, Icongeek26

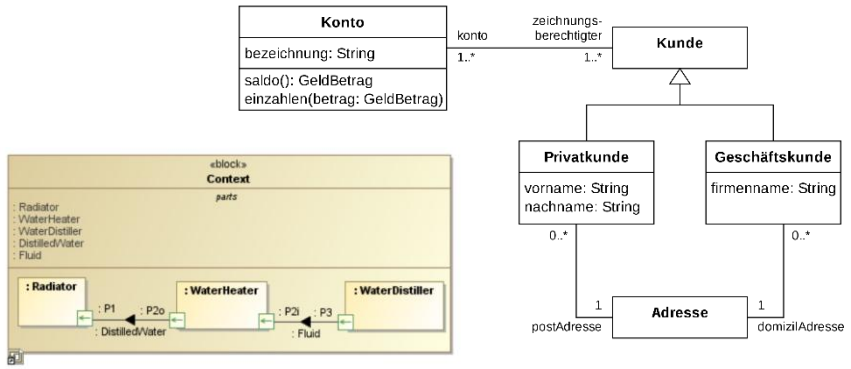
Goal: A common model for all views



Analysis of existing Modelling Notations

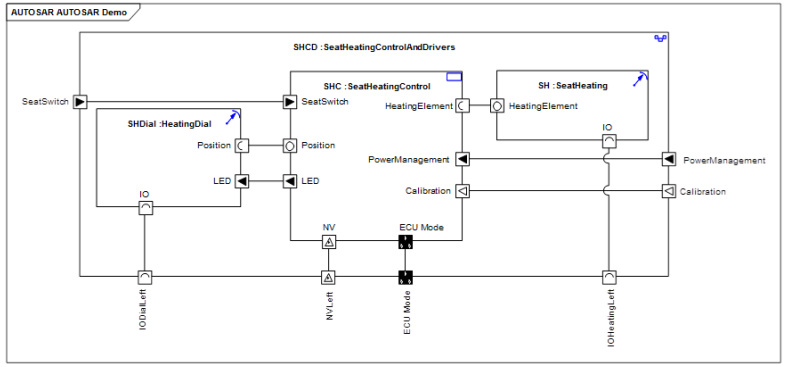
UML / SysML

- Well known
- Limited code generation ability
- No safety support



AUTOSAR

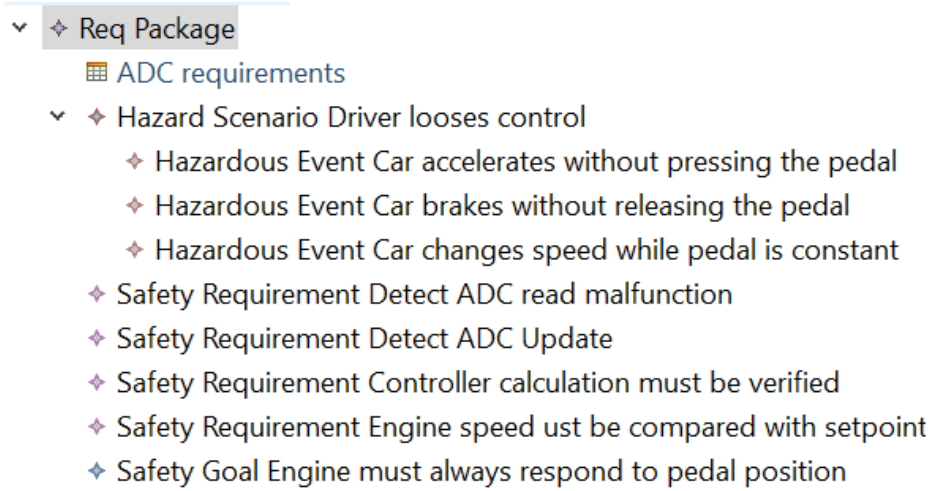
- Well established in automotive industry
- Signal driven approach
- Code generation for RTE
- Very complex



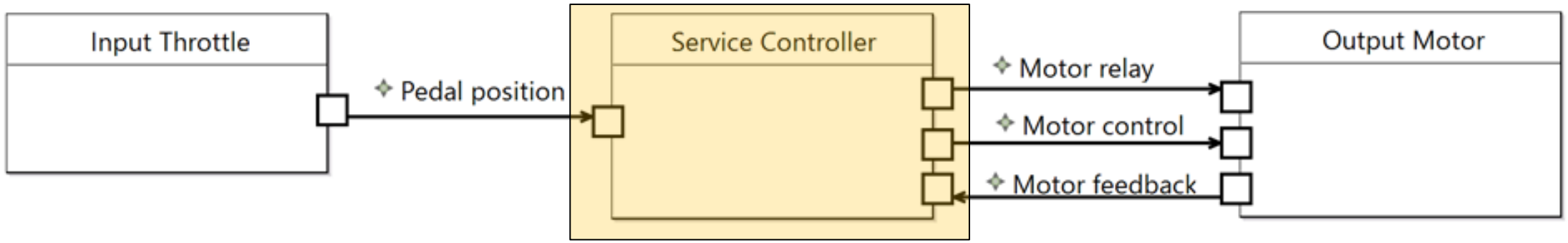
Own Domain Specific Development required.
Using best practices from existing standards.

Requirements view

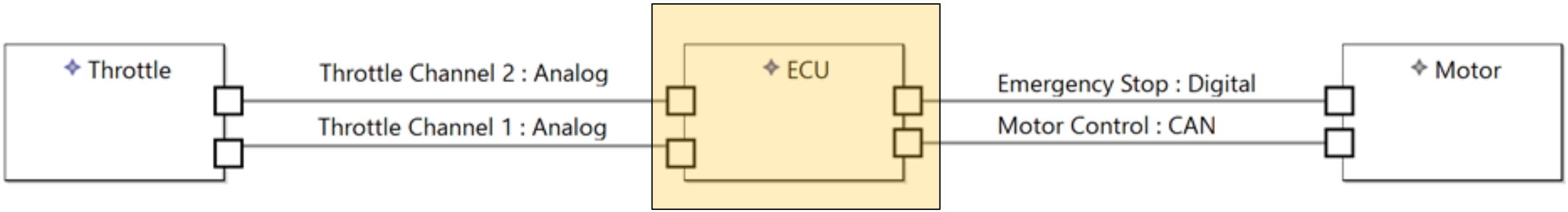
	Author	Required SIL	Status	Description
Detect ADC read malfunction	Barth	SIL-1	verifying	The ADC might fail. It needs to be ensured that such ma
Detect ADC Update	Barth	SIL-1	approving	Whenever there are new samples available, the system
Controller calculation must be verified	Fromm	SIL-1	unused	The computation of the target speed might be incorrect.



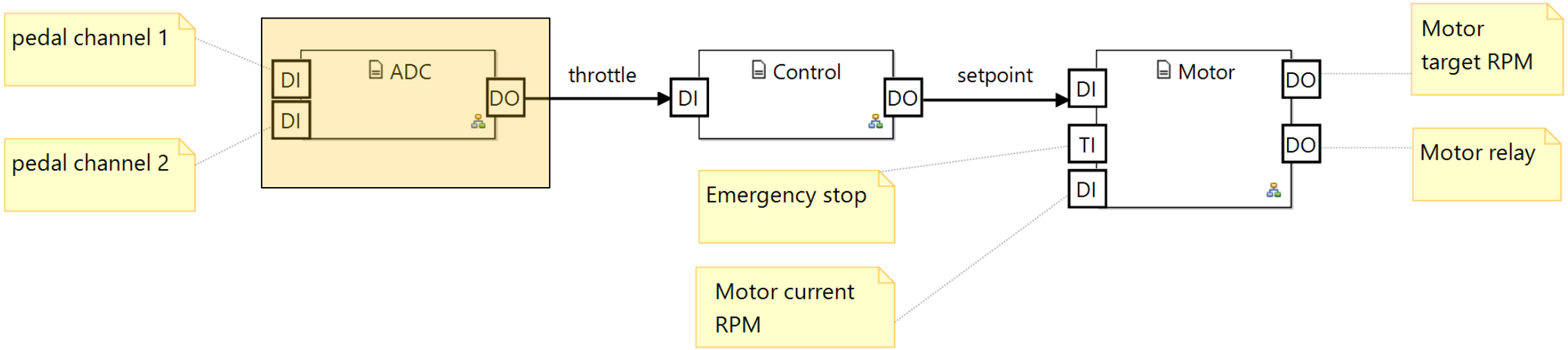
Functional view



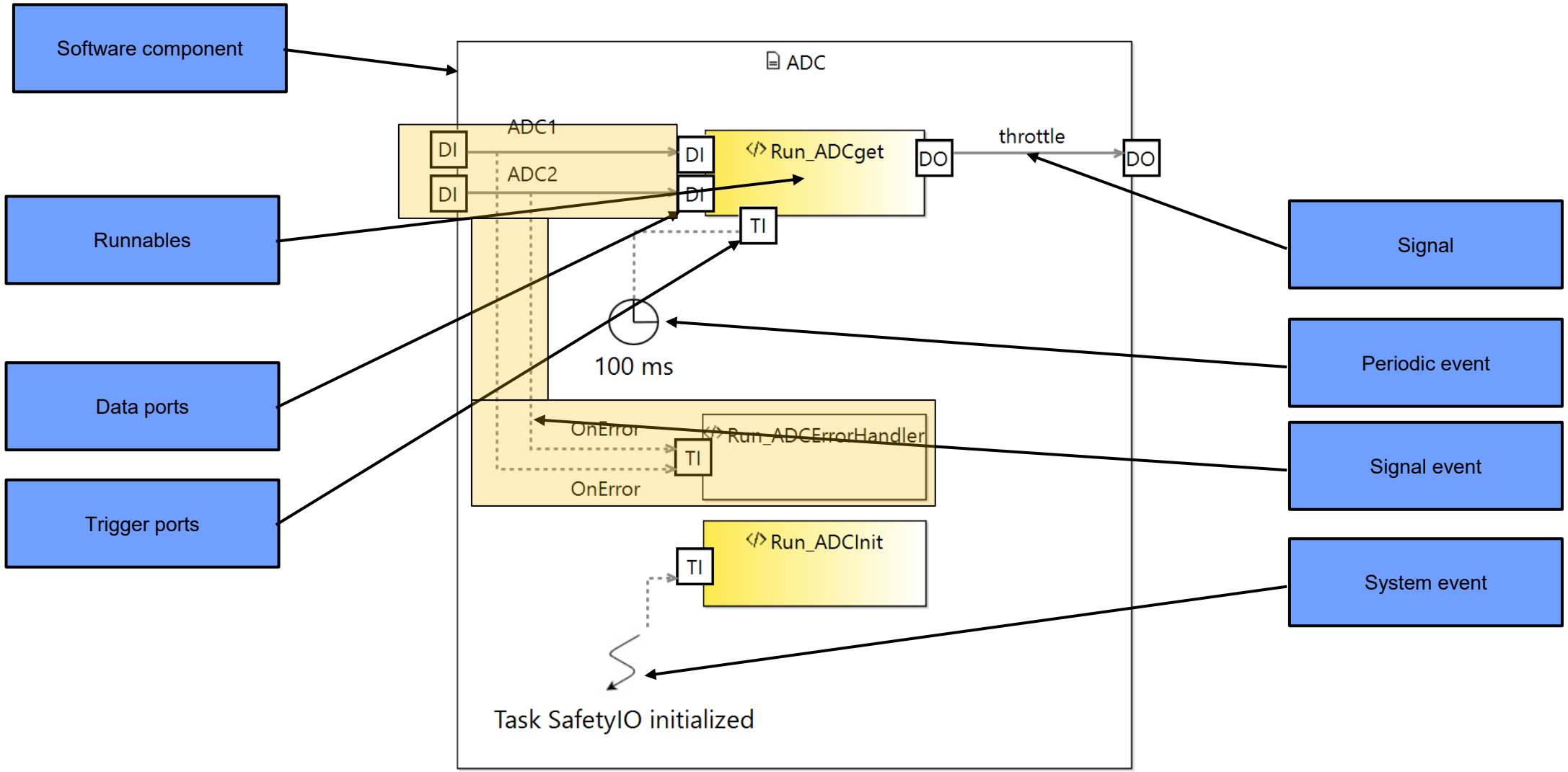
Hardware view



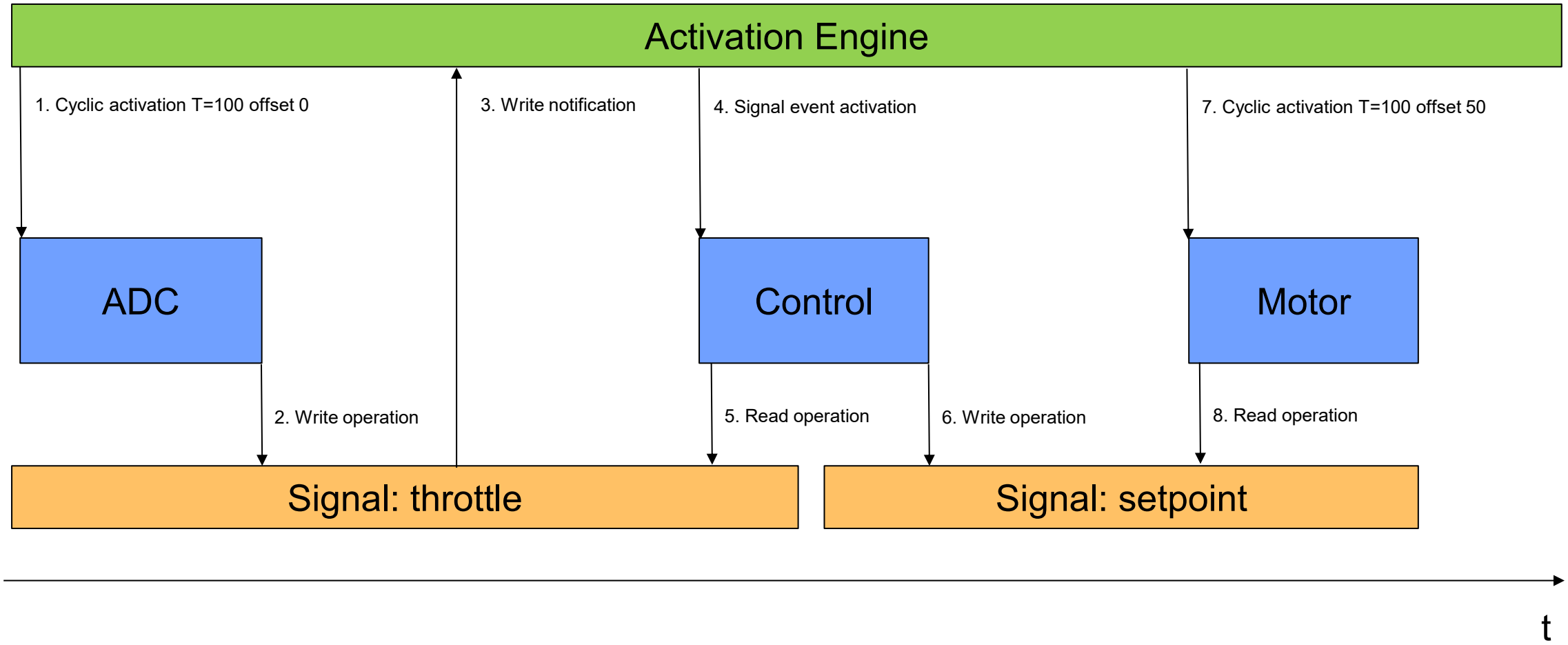
Software view – Overview



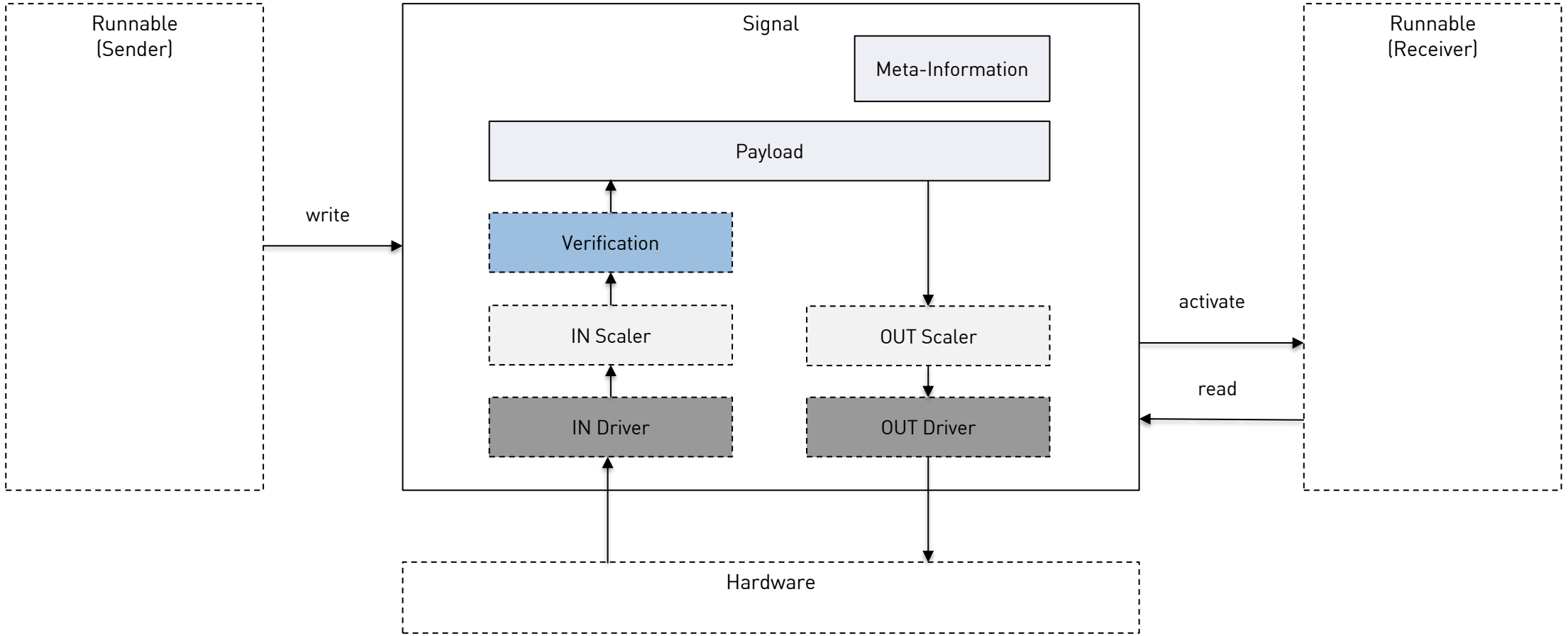
Software view – Detailed



Runtime environment – typical signal-flow



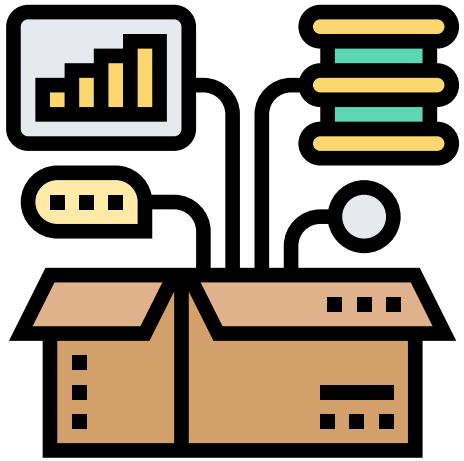
Runtime environment – signal



Principles of code generation

RTE meta-features:

- Clearly structured code with good readability (debug/test/verification)
- Model changes have very local effect on generated code (integration/migration/versioning..)
- Detailed in-code documentation based on the model (system understanding)
- Flexible and centralized configuration
- Minimal external interfaces to RTOS and ecosystem (supports various targets/RTOS)



Icon: Eucalyp

Runtime environment – snippets

Runnable outline

```
/* --- Details for runnable "Run_control" ---
 *  Description:
 *      wird aufgerufen, wenn throttle einen neuen Wert bekommen hat. […]
 *  SIL: QM
 *  System states:
 *      normal
 *  Signal association
 *       InPorts:
 *           Trigger
 *                  Trigger source throttle.onData
 *                  Task: "Control"
 […]
 */
RTE_ret_t Run_control(Signal_throttle& sig_IN_throttle, Signal_setpoint* const
p_sig_OUT_setpoint){

    //runnable return
    RTE_ret_t ret=RTE_RET_OK;

    //local IN signal value buffers
    Signal_throttle::data_t throttle_data;

//Start of user code implementation Run_control

    //get throttle value
    if(RTE_RET_OK==sig_IN_throttle.get(&throttle_data)){

        //calculate new setpoint
        setpoint_data_t new_setpoint=(throttle_data/100.0)*SETPOIN_T_MAX;

        //set setpoint
        p_sig_OUT_setpoint->set(new_setpoint);

    }else{

        //throttle is invalid, invalidate the setpoint
        p_sig_OUT_setpoint->setStatusInvalid();
        ret = RTE_RET_SIGNAL_INVALID;
    }

//End of user code
    return ret;
}
```

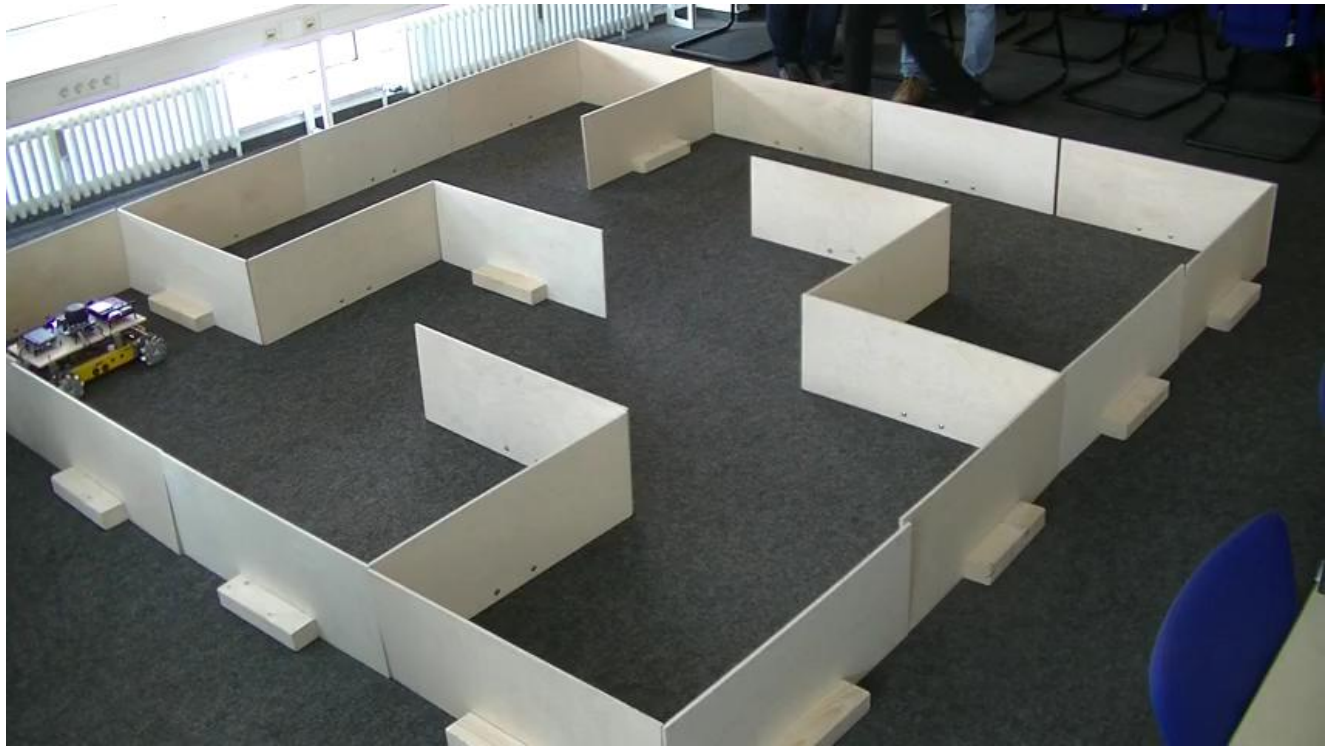
Activation engine

```
INLINE RTE_ret_t AE_ctx_SafetyIO::InvokeSignalTriggerRunnables(AE_ctx_SafetyIO_TriggerPorts_t trigger){

    // method result
    RTE_ret_t ret = RTE_RET_OK;

     //get current system state
    const AE_Ccentral::SystemState_t currentSystemState=AE_central.getSystemState();

    //invoke runnables for the current state
    switch(currentSystemState){
    case AE_Ccentral::sysState_error_detected: /* Nur der Motor-Treiber ist zyklisch aktiv […] */

        // invoke runnables depending on the trigger
        switch(trigger){
        case AE_ctx_SafetyIO_Run_MotorEMS_MotorEMS_OnTrigger:
            // ------------  Run_MotorEMS, SWC "Motor", Order 0, Guards: none
            Run_MotorEMS(&ds__MotorRelais);
            break;
        case AE_ctx_SafetyIO_Run_ADCErrorHandler_ADC1_OnError:
            // ------------  Run_ADCErrorHandler, SWC "ADC", Order 0, Guards: none
            Run_ADCErrorHandler(/* runnable is not associated with signals */);
            break;
        }
        break; /* sysState_error_detected */

     case AE_Ccentral::sysState_normal: /* Die ADC Werte werden zyklisch ausgelesen, […] */

    […]

    }
    break; /* sysState_normal */

    }

    // return result
    return ret;
}
```

Real life experience

A omnidirectional mecanum wheel robot serves as a demonstrator

In the summer term 2019, several independent student teams with no multicore experience realized an autonomous navigation algorithm



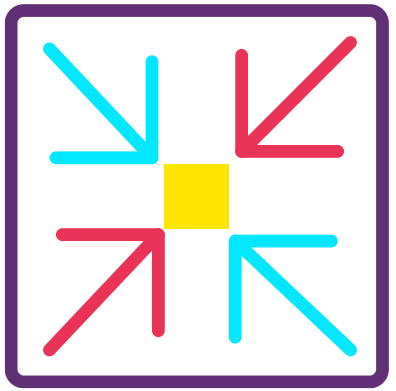
Conclusion

The tool already has proven itself in an academic environment

- Potential design flaws become visible early
- In comparison to manual approach tremendous simplification and time saving
- Requires training and experience

The generated code:

- The goals of readable code and simple structures have been met
- The system is easy to debug test and verify
- Changes in the model only have very local effects on the code
- Lightweight interface
- Task-local storage support freedom from interference
- Model, generated code and documentation ease qualification



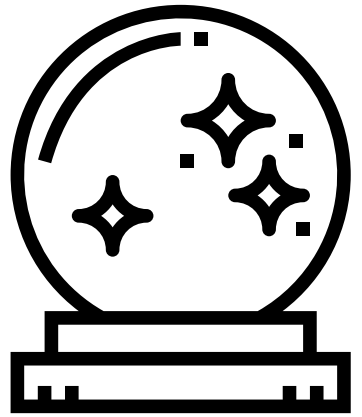
Icon: Darius Dan

Outlook

Future research will include:

- Automated assessment of the model
- Porting to other targets and RTOS (also singlecore)
- Control flow monitoring
- Test/Error-Injection mechanism
- Trace

The tool currently is a feasibility study
It will be stabilized/refined and might be published



Icon: smalllikeart

Contact

M.Sc. Thomas Barth
Prof. Dr.-Ing. Peter Fromm

Hochschule Darmstadt - University of Applied Sciences
FB Elektrotechnik und Informationstechnik (EIT)
Birkenweg 8
64295 Darmstadt

Email: thomas.barth@h-da.de; peter.fromm@h-da.de
Web: www.eit.h-da.de