+chrona

Simulation with Chrona's Validator

Wolfgang Pree

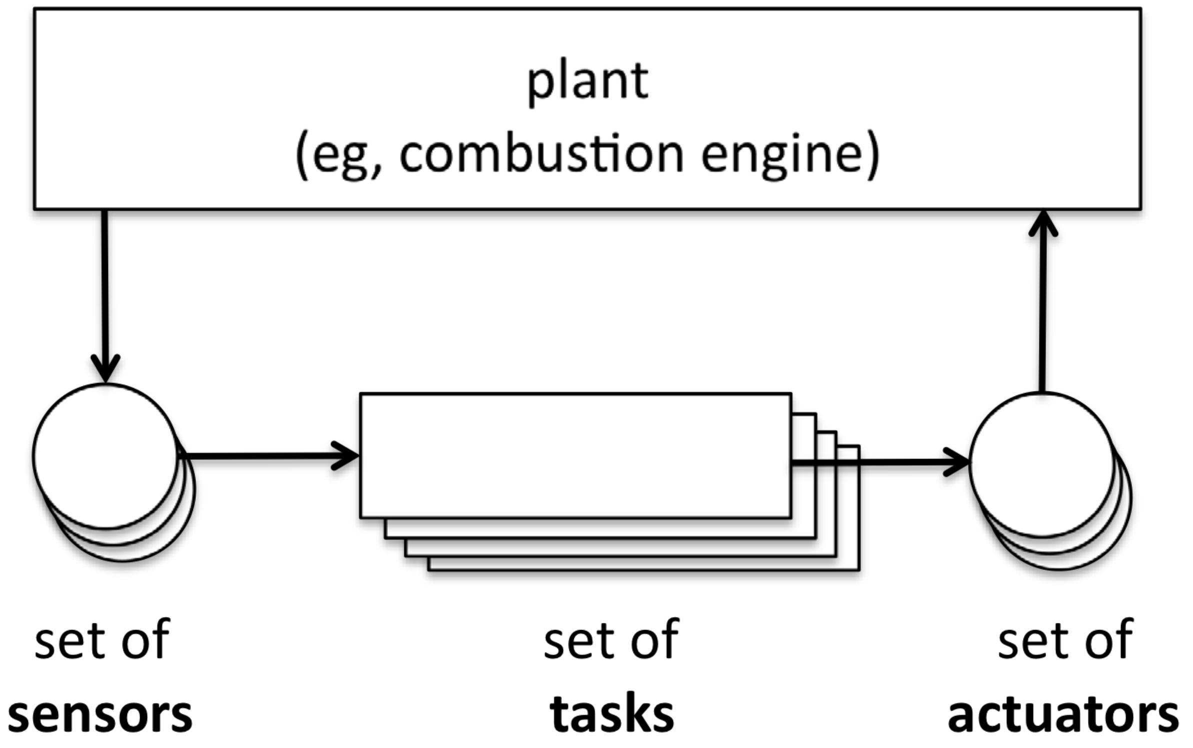
chrona.com

Overview

- Filling the gap between conventional SIL and HIL simulations
- Validator concepts and architecture
- validation and verification scenarios
  - advanced debugging
  - migration of legacy systems

Filling the gap between SIL and HIL

What should be simulated?



Why do we need improved simulation support? (I)

- SIL simulation not sufficient for verification and validation:
- simulated (functional) behavior ≠ actual behavior on execution platform

SIL simulation

conventional
SIL simulation

- test functionality

Instruction Set Simulator (ISS)

- test real-time behavior and functionality

HIL simulation

- test real-time behavior and functionality

SIL simulation

conventional
SIL simulation

- test functionality

+ cheap

+ fast

– imprecise

Instruction Set Simulator (ISS)

- test real-time behavior and functionality

HIL simulation

- test real-time behavior and functionality

SIL simulation

conventional SIL simulation

- test functionality

+ cheap

+ fast

– imprecise

Instruction Set Simulator (ISS)

- test real-time behavior and functionality

HIL simulation

- test real-time behavior and functionality

+ precise

– tedious

– expensive

SIL simulation

conventional SIL simulation

- test functionality

+ cheap

+ fast

– imprecise

Instruction Set Simulator (ISS)

- test real-time behavior and functionality

+ precise

– tedious and extremely slow

– expensive

HIL simulation

- test real-time behavior and functionality

+ precise

– tedious

– expensive

SIL simulation		HIL simulation
conventional SIL simulation	Instruction Set Simulator (ISS)	
• test functionality	• test real-time behavior and functionality	• test real-time behavior and functionality
+ cheap + fast – imprecise	+ precise – tedious and extremely slow – expensive	+ precise – tedious – expensive

+chrona

SIL simulation

conventional
SIL simulation

- test functionality

+ cheap

+ fast

– imprecise

?

HIL simulation

- test real-time behavior and functionality

+ precise

– tedious

– expensive

SIL simulation

conventional
SIL simulation

- test functionality

+ cheap

+ fast

– imprecise

?

± reasonable costs

+ fast

+ precise

HIL simulation

- test real-time behavior and functionality

+ precise

– tedious

– expensive

SIL simulation

HIL simulation

conventional
SIL simulation

- test functionality

+ cheap

+ fast

– imprecise

Validator simulation

- test real-time behavior and functionality
- **advanced debugging**

± reasonable costs

+ fast

+ precise

- test real-time behavior and functionality

+ precise

– tedious

– expensive

SIL simulation

conventional SIL simulation

- test functionality

+ cheap
+ fast
– imprecise

Validator simulation

- test real-time behavior and functionality
- **advanced debugging**

± reasonable costs
+ fast
+ precise

HIL simulation

- test real-time behavior and functionality

+ precise
– tedious
– expensive

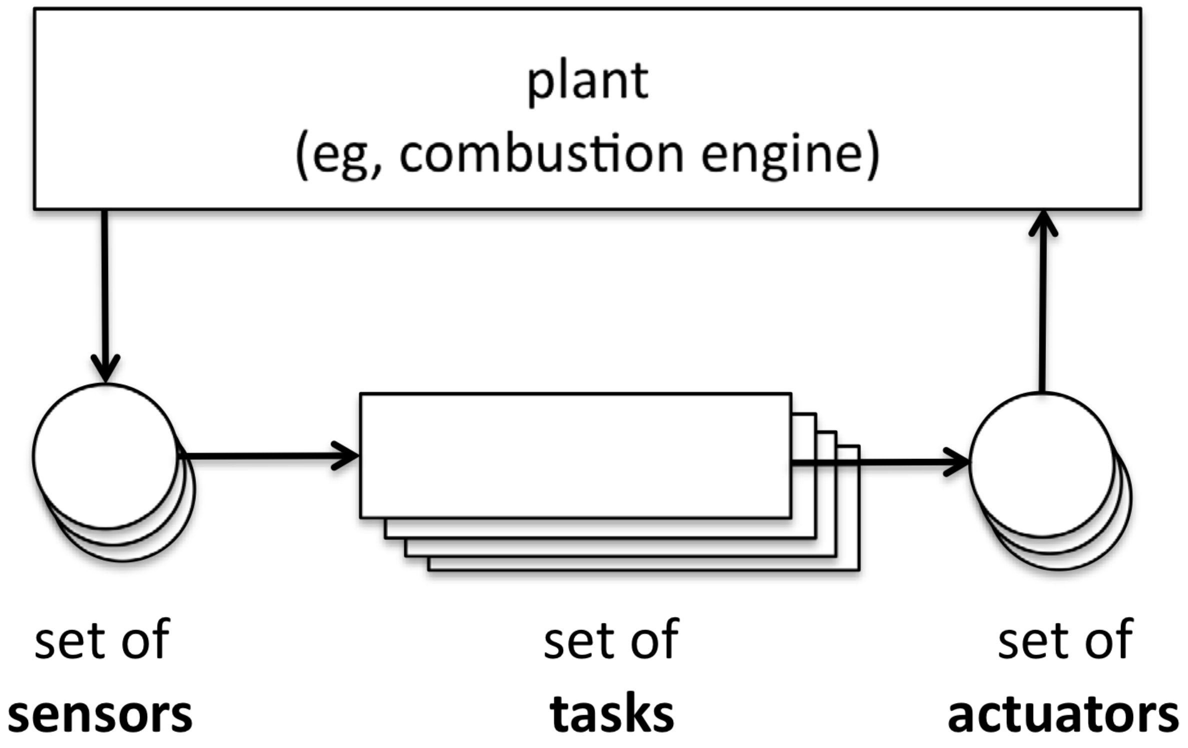
SIL simulation

HIL simulation

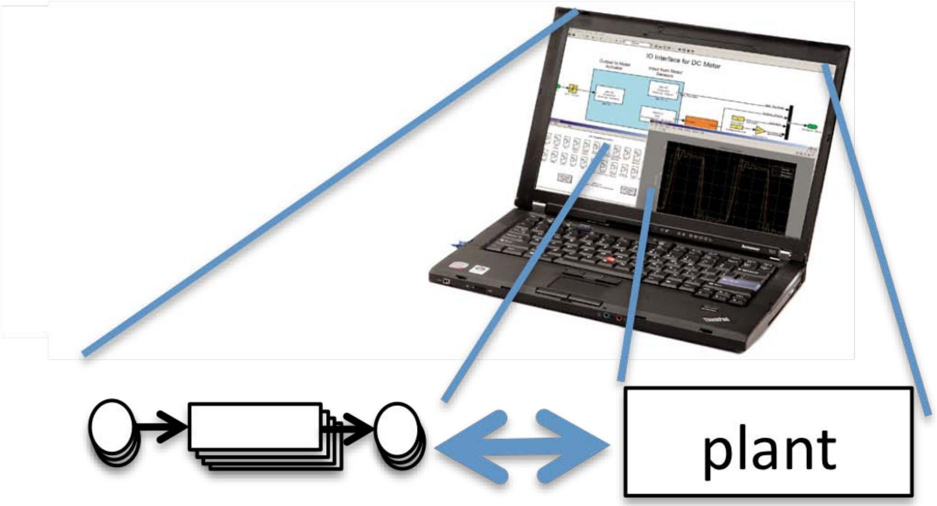
conventional
SIL simulation

Validator simulation

What do we simulate?



Co-simulation of plant and controller tasks



plant

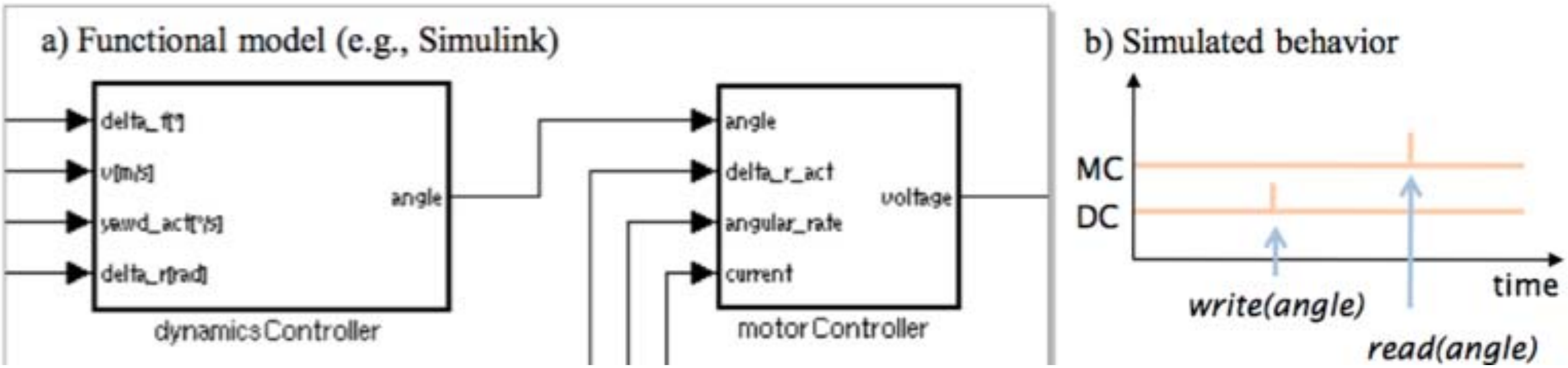
protocol,
eg via TCP/IP

- separate simulations, but typically on same PC:
  - plant simulation: eg, MATLAB/Simulink
  - controller task simulation: Validator

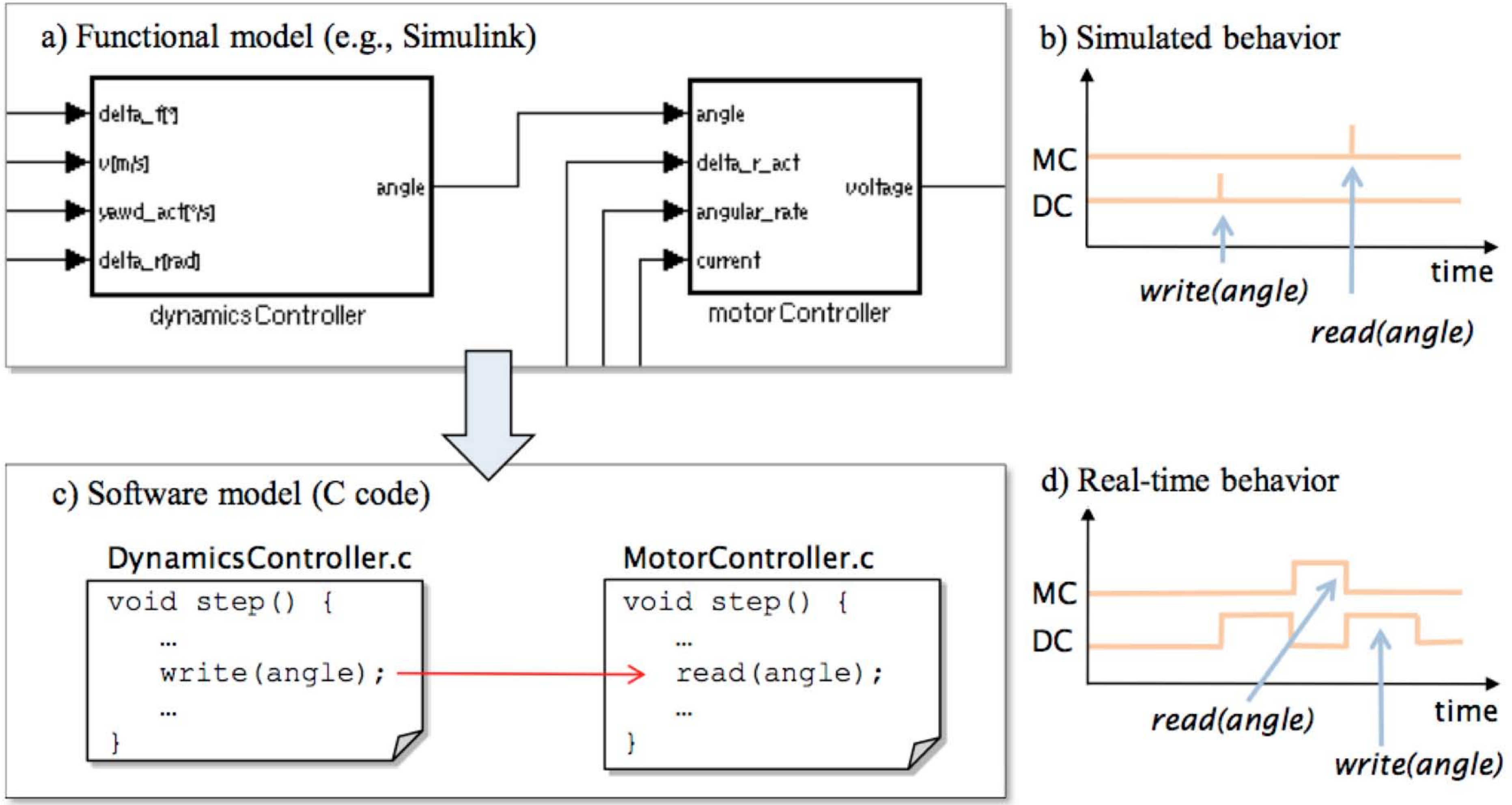
Why do we need improved simulation support? (I)

- simulated (functional) behavior ≠ actual behavior on execution platform

Why do we need improved simulation support? (II)



Why do we need improved simulation support? (III)



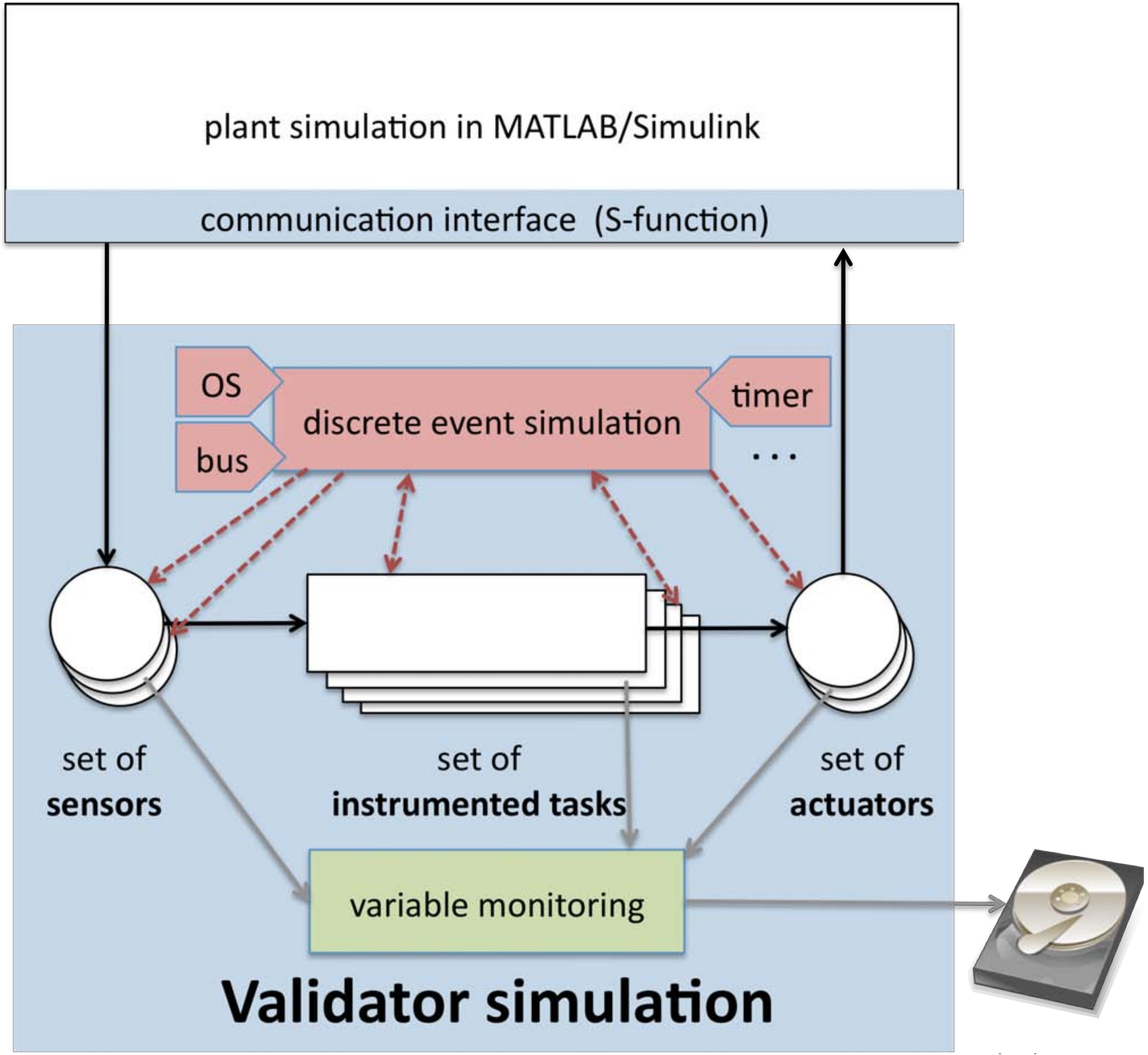
DynamicsController.c

```
void step() {
   …
   write(angle);
   …
}
```

MotorController.c

```
void step() {
   …
   read(angle);
   …
}
```

Validator architecture



plant simulation in MATLAB/Simulink

communication interface (S-function)

OS

bus

discrete event simulation

timer

. . .

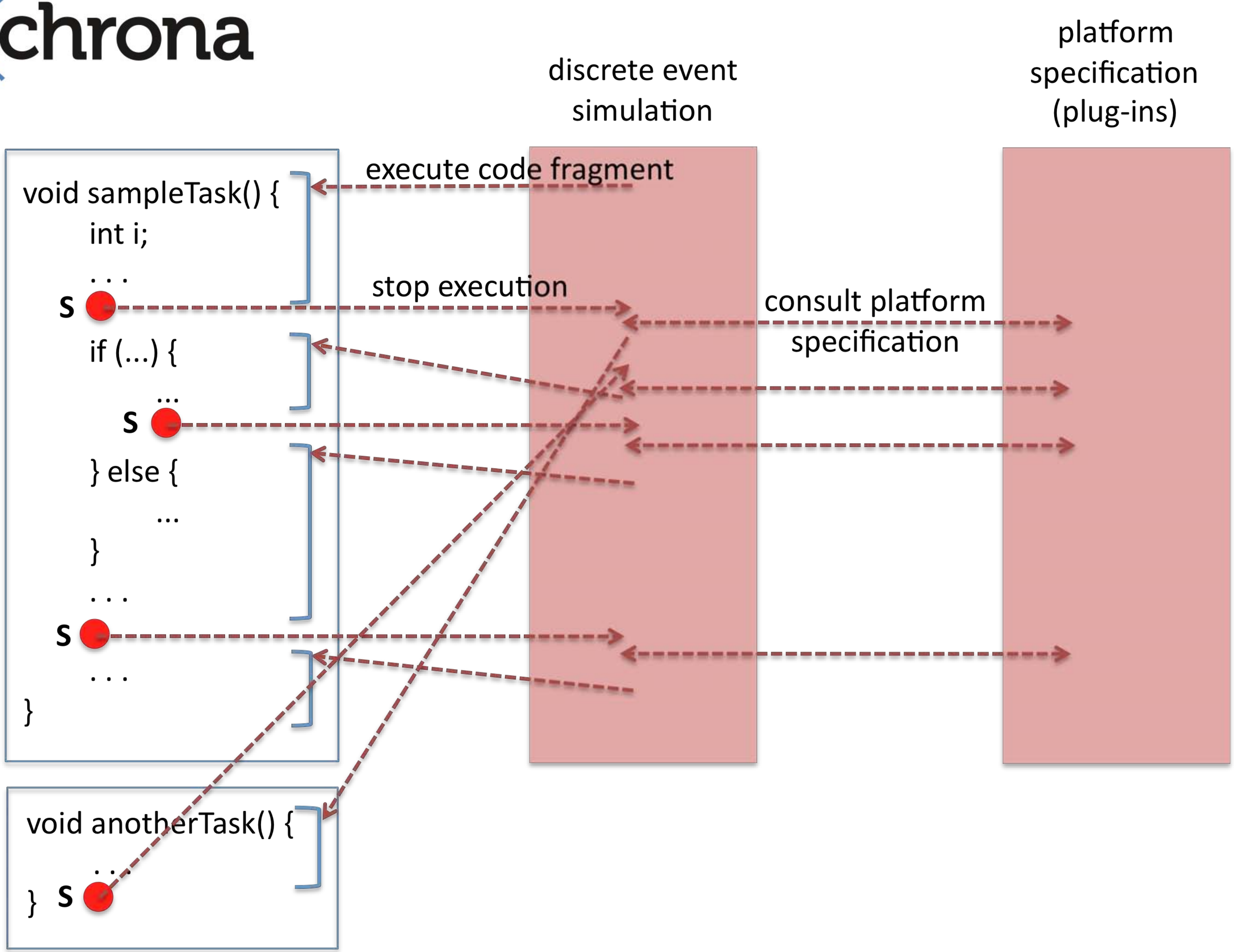
set of **sensors**

set of **instrumented tasks**

set of **actuators**

variable monitoring

Validator simulation



discrete event simulation

platform specification (plug-ins)

execute code fragment

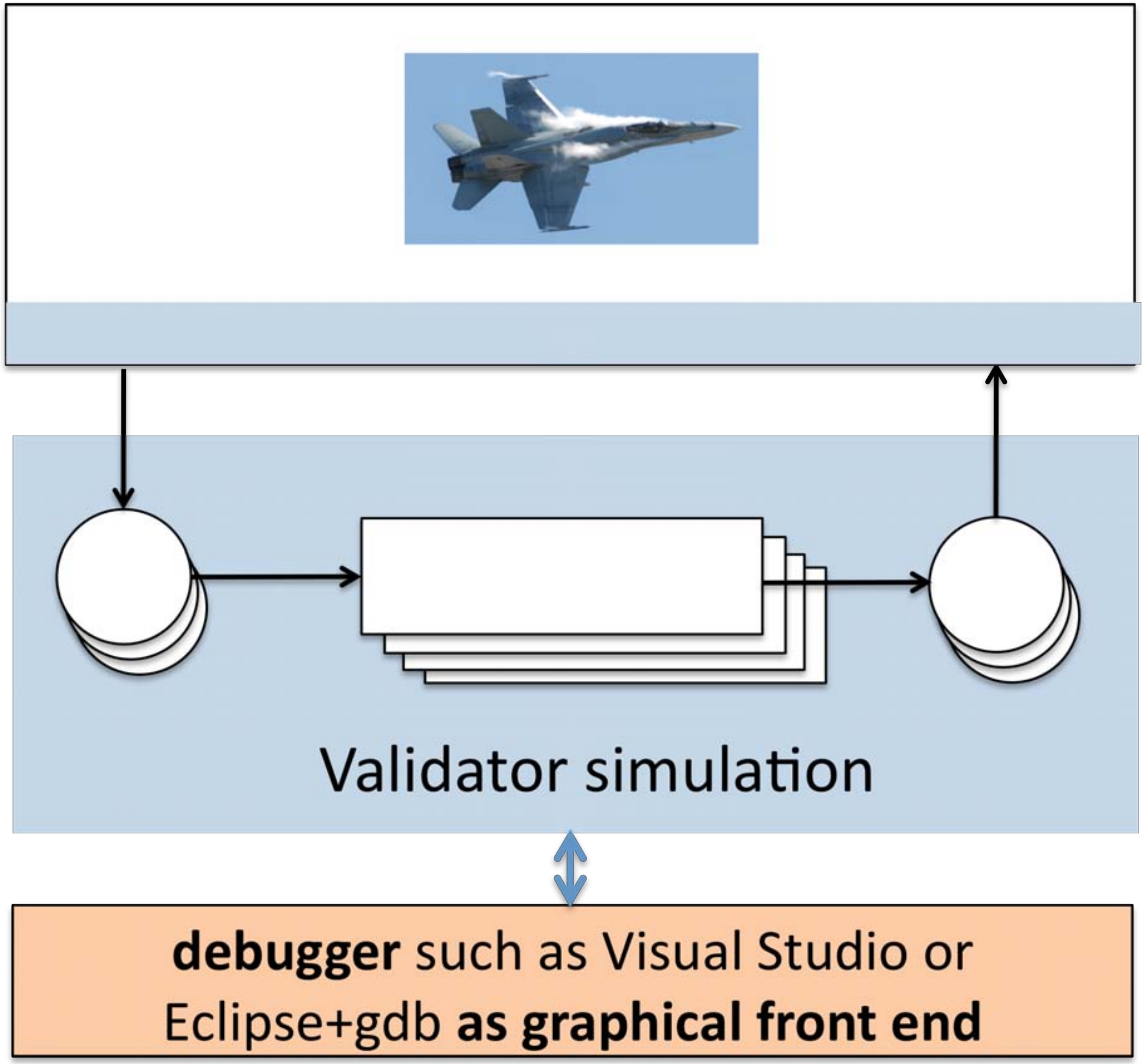
stop execution

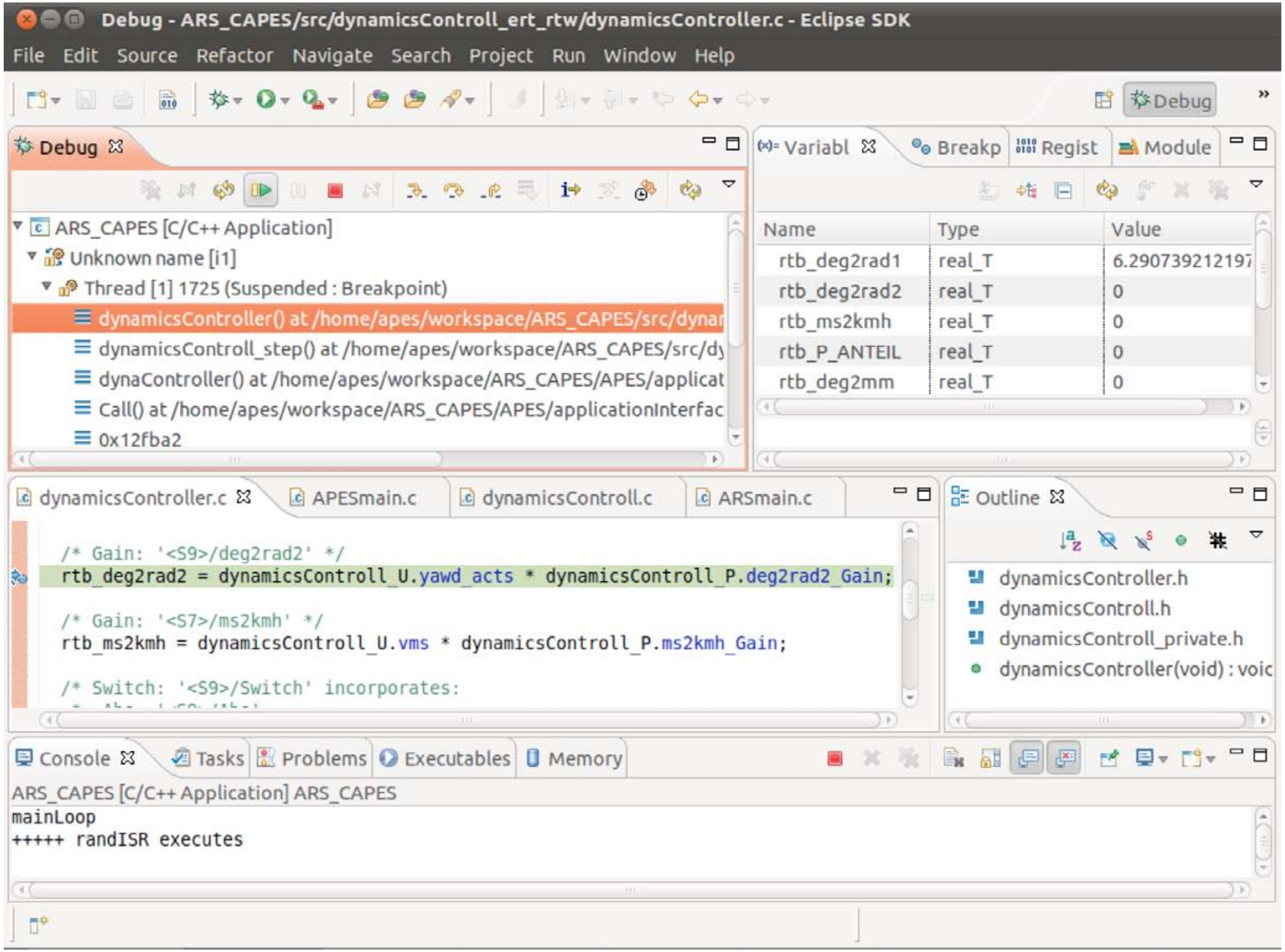
consult platform specification

```
void sampleTask() {
        int i;
        . . .
    S ●
        if (...) {
              ...
          S ●
        } else {
              ...
        }
        . . .
    S ●
        . . .
}
```

```
void anotherTask() {
        . . .
}  S ●
```

Validator usage scenarios


Validator simulation
debugger such as Visual Studio or Eclipse+gdb **as graphical front end**


Debug - ARS_CAPES/src/dynamicsControll_ert_rtw/dynamicsController.c - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Debug
Debug
Variabl
Breakp
Regist
Module
dynamicsController.c
APESmain.c
dynamicsControll.c
ARSmain.c
Outline
Console
Tasks
Problems
Executables
Memory

ARS_CAPES [C/C++ Application]
- Unknown name [i1]
  - Thread [1] 1725 (Suspended : Breakpoint)
    - dynamicsController() at /home/apes/workspace/ARS_CAPES/src/dynai
    - dynamicsControll_step() at /home/apes/workspace/ARS_CAPES/src/dy
    - dynaController() at /home/apes/workspace/ARS_CAPES/APES/applicat
    - Call() at /home/apes/workspace/ARS_CAPES/APES/applicationInterfac
    - 0x12fba2

Name	Type	Value
rtb_deg2rad1	real_T	6.290739212197
rtb_deg2rad2	real_T	0
rtb_ms2kmh	real_T	0
rtb_P_ANTEIL	real_T	0
rtb_deg2mm	real_T	0

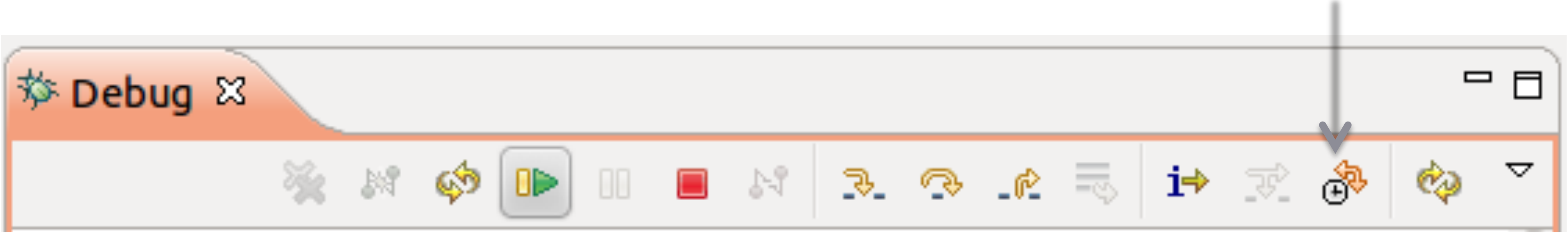
```
  /* Gain: '<S9>/deg2rad2' */
  rtb_deg2rad2 = dynamicsControll_U.yawd_acts * dynamicsControll_P.deg2rad2_Gain;

  /* Gain: '<S7>/ms2kmh' */
  rtb_ms2kmh = dynamicsControll_U.vms * dynamicsControll_P.ms2kmh_Gain;

  /* Switch: '<S9>/Switch' incorporates:
```

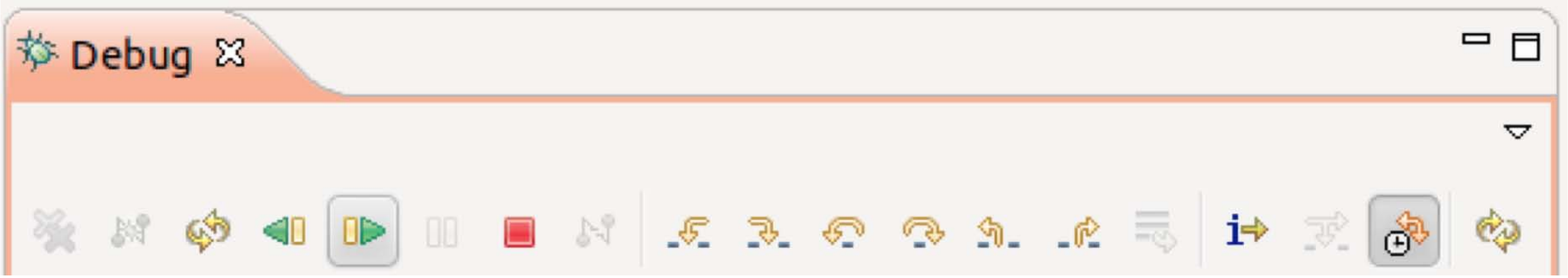
- dynamicsController.h
- dynamicsControll.h
- dynamicsControll_private.h
- dynamicsController(void) : voic

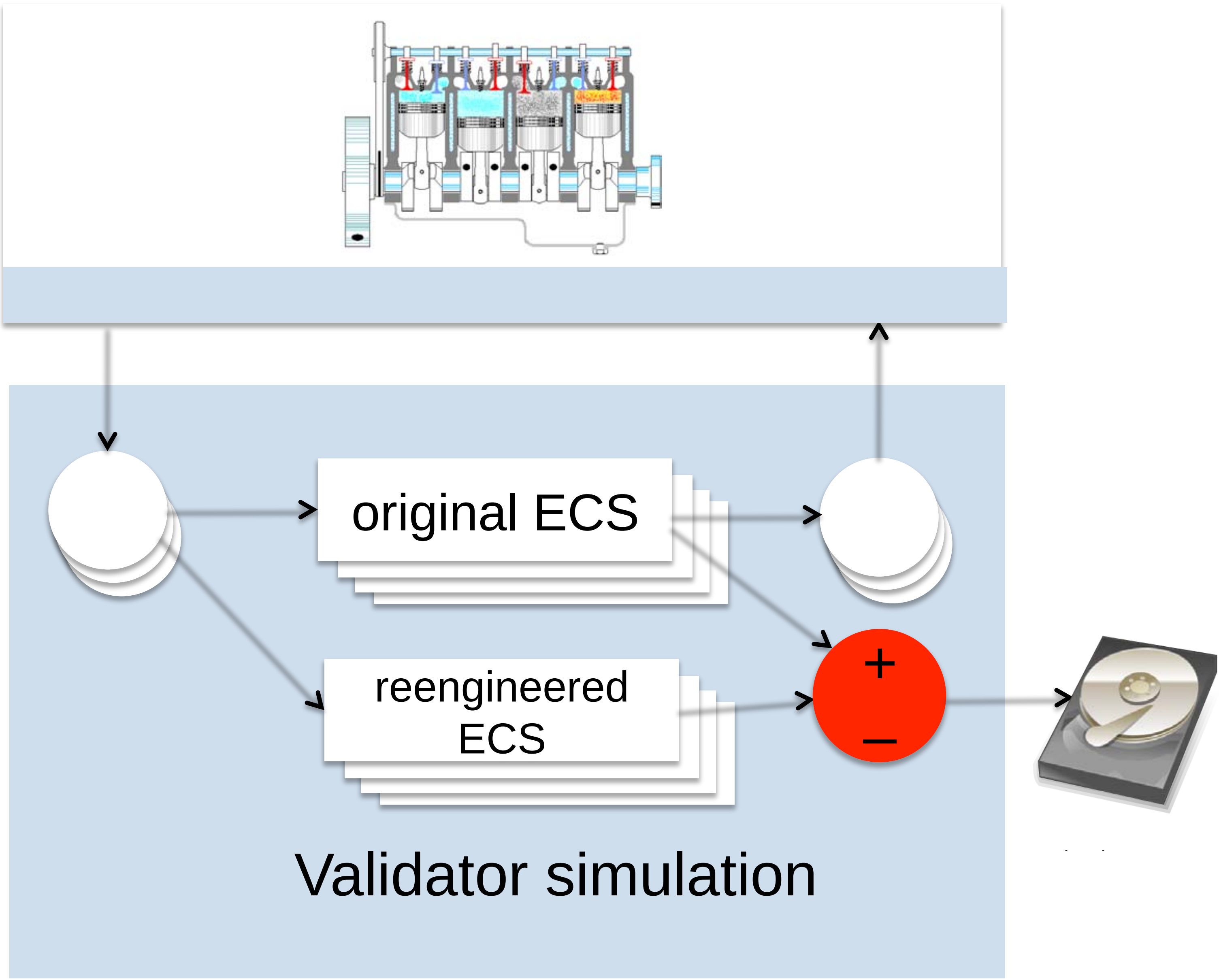
ARS_CAPES [C/C++ Application] ARS_CAPES

```
mainLoop
+++++ randISR executes
```

turn reverse debugging on/off




Debug



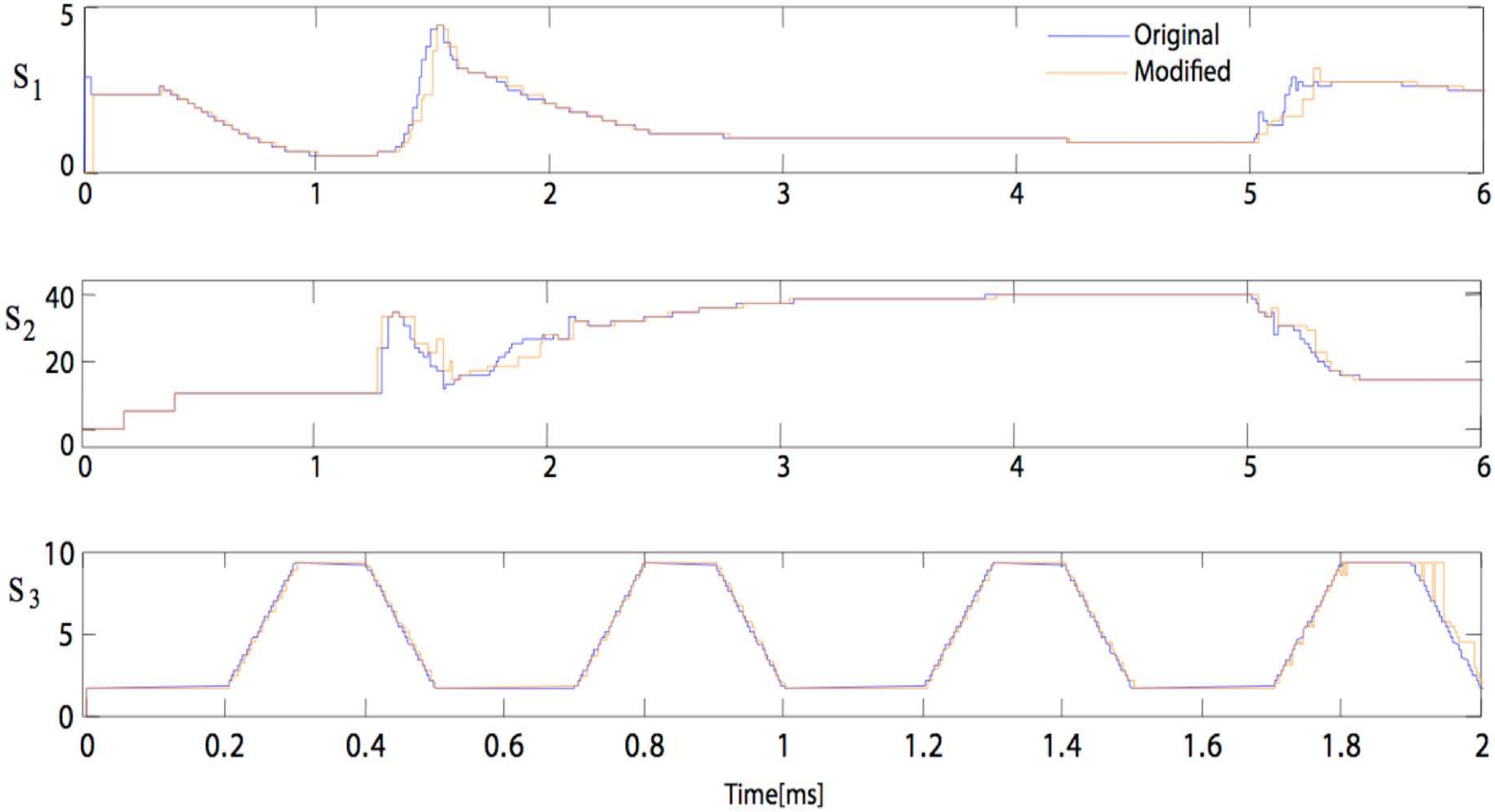
original ECS

reengineered ECS

+

–

Validator simulation


S1
5
0
0
1
2
3
4
5
6
Original
Modified
S2
40
20
0
0
1
2
3
4
5
6
S3
10
5
0
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
Time[ms]

What it takes: set-up of a Validator simulation

Currently supported ...

- co-simulation of
  - a plant represented as variable-step model in **MATLAB/Simulink** or **Ptolemy II**
  - controller software written in **C**

Platform specification

- operating system: scheduling, resource management and communication between tasks
  - Validator library: OSEK specification
- functionality and timing of common hardware parts such as interrupt controller, timers, bus controllers, hardware sensors and hardware actuators

Execution time analysis and source code annotations

- Execution time analysis of the application code, eg, with program analysis tools such as AbsInt's Advanced Analyzer (a3) tool.
- **Instrumentation of the code with execution time information.**
- **Instrumentation with callbacks at spots** to pass control to the Validator simulation engine for the execution of the tasks.
- Generation of what we call the Validator interface code between the Validator simulation engine and the tasks.

Execution time analysis and source code annotations

mostly automated

- Execution time analysis of the application code, eg, with program analysis tools such as AbsInt's Advanced Analyzer (a3) tool.
- **Instrumentation of the code with execution time information.**
- **Instrumentation with callbacks at spots** to pass control to the Validator simulation engine for the execution of the tasks.
- Generation of what we call the Validator interface code between the Validator simulation engine and the tasks.

Validator summary

- **accurate simulation**
  - simulated behavior = behavior on execution platform
- **fast**
- **advanced debugging**, including reverse debugging across preemption points
- **straight-forward to set up:** mostly automated

- allows **solid verification and validation of real-time embedded systems**

Thank you for your attention!