Resource-Aware Functional Programming in the Automotive Domain

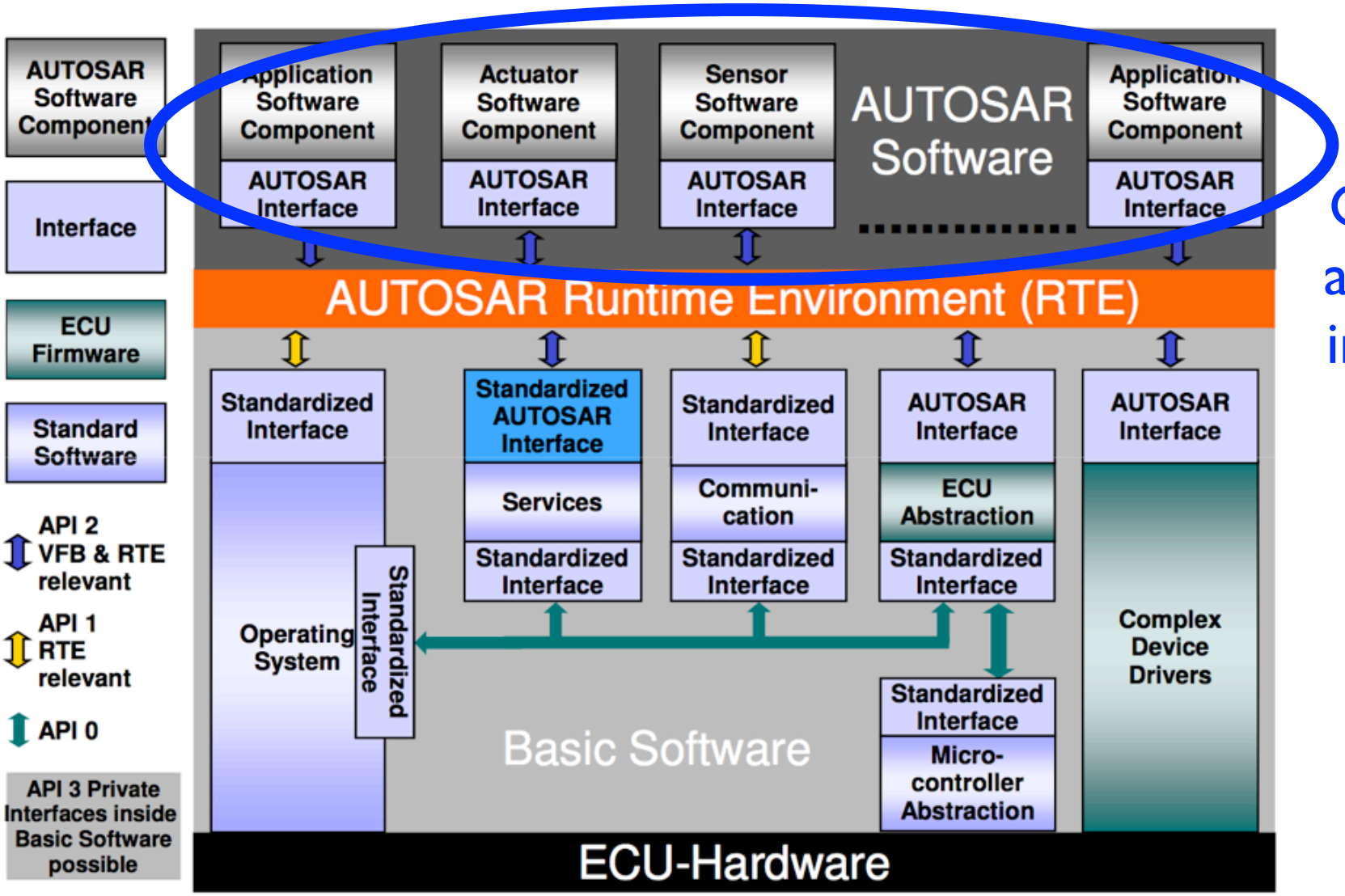
Applying *Domain-Specific Embedded Language* (DSEL) technology to concurrent, distributed and real-time software under the AUTOSAR standard

SSF project *RAW FP*
Automotive track status report
November 2013

AUTOSAR

- A vendor-independent software architecture standard for the automotive industry
- Platform-independent application layer
- Standardized APIs / Basic Software modules
- Standardized system constraint formats
- Extensive tool support for semi-automatic system configuration and code generation
- A detailed design-step methodology

AUTOSAR architecture



Captured as a DSEL in Haskell

An AUTOSAR DSEL
Why?

1. *The importance of Swedish automotive industry*

   - 110 000 people employed, 12% of export value, 25% of manufacturing industry R&D, 13% of industrial investments (2012) (2008)
   - 40% of modern car production costs pertain to electronics & software
   - Software amounts to 50-70% of electronic system development costs

An AUTOSAR DSEL
Why?

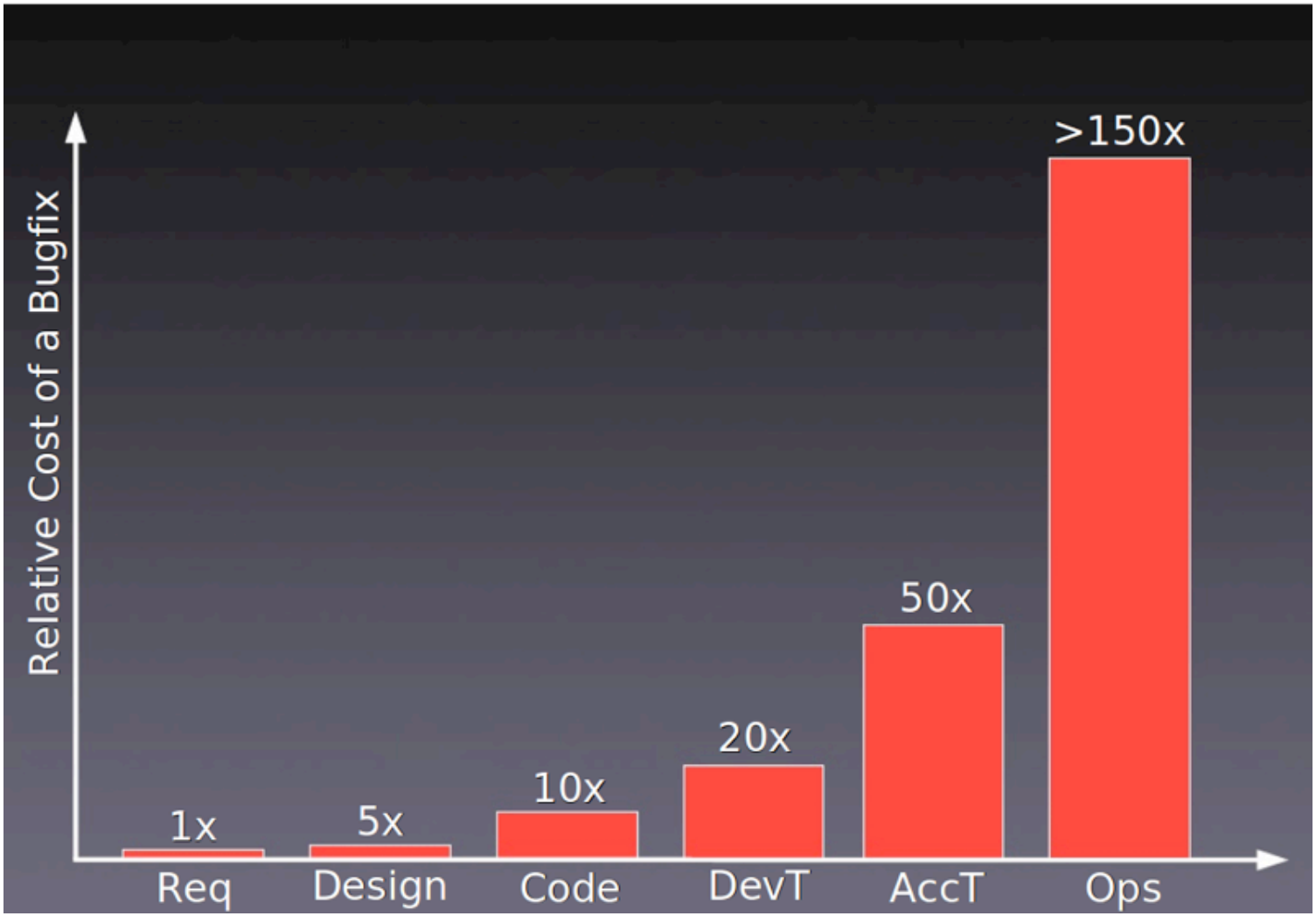
2. *The specific problem of testing automotive software*
   - > 70 ECUs, 5 busses, > 10 000 000 lines of code
   - Tight integration and interdependencies between subsystems
   - Security concerns ⇒ mandatory resource awareness
     ⇒ platform dependencies ⇒ desktop testing unrealistic
   - Full-scale testing on a real moving car is both costly and impractical

An AUTOSAR DSEL
Why?

3. *The gaps in AUTOSAR's behavioral modeling*

- AUTOSAR only specifies program structure & APIs
- *Functional behavior* is assumed to be given in Matlab/ Simulink or as plain C code
- Software components are also expected to map onto OS tasks and low-level concepts, and the standard makes no clear separation of these abstraction levels
- Testing/simulation of *models* rather than code is thus not supported by AUTOSAR

An AUTOSAR DSEL
Why?



Sources: Stefan Priebsch, Advanced OOP and Design Patterns, Barry Boehm "EQUITY Keynote Address", March 19th, 2007

An AUTOSAR DSEL
Why?

4. *The daunting AUTOSAR standard specification*

   - Counting ~100 documents, ~12 500 pages
     (plus just as much auxiliary material)
   - ~20 documents relate to software components, with ~1600 pages in just the two primary ones
   - The contents define a complex programming model, with subtle and sometimes unclear semantic detail

An AUTOSAR DSEL
Why?

5. *The challenges of concurrency, distribution & real-time*

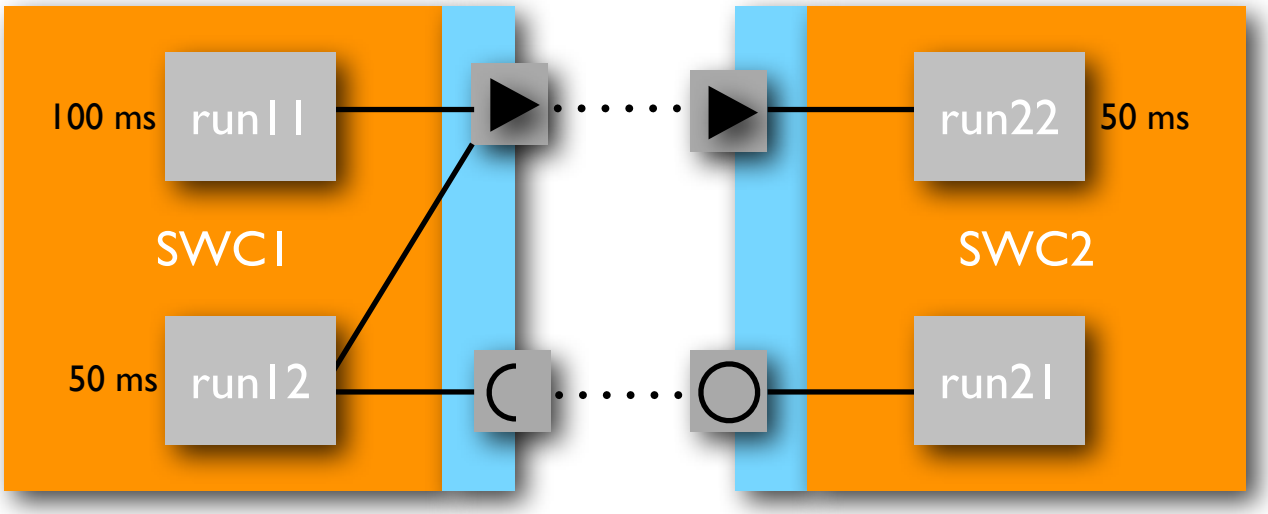
- Automotive specifics aside, the construction of concurrent, distributed & real-time software is far from a mature field
- A technology improvement in any of these dimensions is a contribution in itself

Simple example (trad)

```
FUNC(void, RTE APPL CODE) run11(void) {
    Int16 val;
    ...
    Rte_Write_pport1_intValue1(val);
    ...
}
```

```
FUNC(void, RTE APPL CODE) run22(void) {
    Int16 val;
    ...
    Rte_Read_rport2_intValue(&val);
    ...
}
```

```
TASK(Task1) {
   Rte RECount_Task1_divby2_0--;
   if ( Rte_RECount_Task1_divby2_0 == 0 ) {
      run11();
   }
   run12();
   if ( Rte_RECount_Task1_divby2_0 == 0 )
      Rte_RECount_Task1_divby2_0 = 2;
   TerminateTask();
}
```




```
FUNC(void, RTE APPL CODE) run12(void) {
    String8 val1;
    Int16 val2;
    ...
    Rte_Call_rport1_parse(val1, &val2);
    ...
    Rte_Write_pport1_intValue1(val2);
    ...
}
```

```
FUNC(void, RTE APPL CODE) run21(String8 arg, Int16 *res ) {
    ... arg ...
    ...
    *res = ...
}
```

Simple example (trad)

```
<AR-PACKAGE>
  <SHORT-NAME>swc root</SHORT-NAME>
  <ELEMENTS>
    <ATOMIC-SOFTWARE-COMPONENT-TYPE>
      <SHORT-NAME>swc1</SHORT-NAME>
      <PORTS>
        <P-PORT-PROTOTYPE>
          <SHORT-NAME>pport1</SHORT-NAME>
          <PROVIDED-INTERFACE-TREF DEST="SENDER-RECEIVER-INTERFACE">
            /interfaces/SR Int16
          </PROVIDED-INTERFACE-TREF>
        </P-PORT-PROTOTYPE>
        <R-PORT-PROTOTYPE>
          <SHORT-NAME>rport1</SHORT-NAME>
          <REQUIRED-INTERFACE-TREF DEST="CLIENT-SERVER-INTERFACE">
            /interfaces/CS string to int
          </REQUIRED-INTERFACE-TREF>
        </R-PORT-PROTOTYPE>
      </PORTS>
    </ATOMIC-SOFTWARE-COMPONENT-TYPE>
    <ATOMIC-SOFTWARE-COMPONENT-TYPE>
      <SHORT-NAME>swc2</SHORT-NAME>
      <PORTS>
        <P-PORT-PROTOTYPE>
          <SHORT-NAME>pport1</SHORT-NAME>
          <PROVIDED-INTERFACE-TREF DEST="CLIENT-SERVER-INTERFACE">
            /interfaces/CS string to int
          </PROVIDED-INTERFACE-TREF>
        </P-PORT-PROTOTYPE>
        <R-PORT-PROTOTYPE>
          <SHORT-NAME>rport1</SHORT-NAME>
          <REQUIRED-INTERFACE-TREF DEST="SENDER-RECEIVER-INTERFACE">
            /interfaces/SR Int16
          </REQUIRED-INTERFACE-TREF>
        </R-PORT-PROTOTYPE>
      </PORTS>
    </ATOMIC-SOFTWARE-COMPONENT-TYPE>
```

```
<INTERNAL-BEHAVIOR>
  <SHORT-NAME>intBehSwc1</SHORT-NAME>
  <COMPONENT-REF DEST="ATOMIC-SOFTWARE-COMPONENT-TYPE">/swc root/swc1</COMPONENT-REF>
  <EVENTS>
    <TIMING-EVENT>
      <SHORT-NAME>Time100ms</SHORT-NAME>
      <START-ON-EVENT-REF DEST="RUNNABLE-ENTITY">
        /swc root/intBehSwc1/run11
      </START-ON-EVENT-REF>
      <PERIOD>0.1</PERIOD>
    </TIMING-EVENT>
    <TIMING-EVENT>
      <SHORT-NAME>Time50ms</SHORT-NAME>
      <START-ON-EVENT-REF DEST="RUNNABLE-ENTITY">
        /swc root/intBehSwc1/run12
      </START-ON-EVENT-REF>
      <PERIOD>0.05</PERIOD>
    </TIMING-EVENT>
  </EVENTS>
  <RUNNABLES>
    <RUNNABLE-ENTITY>
      <SHORT-NAME>run11</SHORT-NAME>
      <CAN-BE-INVOKED-CONCURRENTLY>false</CAN-BE-INVOKED-CONCURRENTLY>
      <DATA-SEND-POINTS>
        <DATA-SEND-POINT>
          <SHORT-NAME>dwa1</SHORT-NAME>
          <DATA-ELEMENT-IREF>
            <P-PORT-PROTOTYPE-REF DEST="P-PORT-PROTOTYPE">
              /swc root/swc1/pport1
            </P-PORT-PROTOTYPE-REF>
            <DATA-ELEMENT-PROTOTYPE-REF DEST="DATA-ELEMENT-PROTOTYPE">
              /interfaces/SR Int16/intValue1
            </DATA-ELEMENT-PROTOTYPE-REF>
          </DATA-ELEMENT-IREF>
        </DATA-SEND-POINT>
        <DATA-SEND-POINT>
          <SHORT-NAME>dwa2</SHORT-NAME>
          <DATA-ELEMENT-IREF>
            <P-PORT-PROTOTYPE-REF DEST="P-PORT-PROTOTYPE">
              /swc root/swc1/pport1
            </P-PORT-PROTOTYPE-REF>
            <DATA-ELEMENT-PROTOTYPE-REF DEST="DATA-ELEMENT-PROTOTYPE">
              /interfaces/SR Int16/intValue2
            </DATA-ELEMENT-PROTOTYPE-REF>
          </DATA-ELEMENT-IREF>
        </DATA-SEND-POINT>
      </DATA-SEND-POINTS>
      <SYMBOL>run11</SYMBOL>
    </RUNNABLE-ENTITY>
    <RUNNABLE-ENTITY>
      <SHORT-NAME>run12</SHORT-NAME>
      <CAN-BE-INVOKED-CONCURRENTLY>false</CAN-BE-INVOKED-CONCURRENTLY>
      <DATA-SEND-POINTS>
        <DATA-SEND-POINT>
          <SHORT-NAME>dwa2</SHORT-NAME>
          <DATA-ELEMENT-IREF>
            <P-PORT-PROTOTYPE-REF DEST="P-PORT-PROTOTYPE">
              /swc root/swc1/pport1
            </P-PORT-PROTOTYPE-REF>
            <DATA-ELEMENT-PROTOTYPE-REF DEST="DATA-ELEMENT-PROTOTYPE">
              /interfaces/SR Int16/intValue1
            </DATA-ELEMENT-PROTOTYPE-REF>
          </DATA-ELEMENT-IREF>
        </DATA-SEND-POINT>
      </DATA-SEND-POINTS>
      <SERVER-CALL-POINTS>
        <SYNCHRONOUS-SERVER-CALL-POINT>
          <SHORT-NAME>sscp</SHORT-NAME>
          <OPERATION-IREFS>
            <OPERATION-IREF>
              <R-PORT-PROTOTYPE-REF DEST="R-PORT-PROTOTYPE">
                /swc root/swc1/rport1
              </R-PORT-PROTOTYPE-REF>
              <OPERATION-PROTOTYPE-REF DEST="OPERATION-PROTOTYPE">
                /interfaces/CS string to int/parse
              </OPERATION-PROTOTYPE-REF>
            </OPERATION-IREF>
          </OPERATION-IREFS>
        </SYNCHRONOUS-SERVER-CALL-POINT>
      </SERVER-CALL-POINTS>
      <SYMBOL>run12</SYMBOL>
    </RUNNABLE-ENTITY>
  </RUNNABLES>
  <SUPPORTS-MULTIPLE-INSTANTIATION>false</SUPPORTS-MULTIPLE-INSTANTIATION>
</INTERNAL-BEHAVIOR>
```

```
<INTERNAL-BEHAVIOR>
  <SHORT-NAME>intBehSwc2</SHORT-NAME>
  <COMPONENT-REF DEST="ATOMIC-SOFTWARE-COMPONENT-TYPE">/swc root/swc2</COMPONENT-REF>
  <EVENTS>
    <TIMING-EVENT>
      <SHORT-NAME>Time50ms</SHORT-NAME>
      <START-ON-EVENT-REF DEST="RUNNABLE-ENTITY">
        /swc root/intBehSwc2/run22
      </START-ON-EVENT-REF>
      <PERIOD>0.05</PERIOD>
    </TIMING-EVENT>
    <OPERATION-INVOKED-EVENT>
      <SHORT-NAME>operationInvoke</SHORT-NAME>
      <START-ON-EVENT-REF DEST="RUNNABLE-ENTITY">
        /swc root/intBehSwc2/run21
      </START-ON-EVENT-REF>
      <OPERATION-IREF>
        <P-PORT-PROTOTYPE-REF DEST="P-PORT-PROTOTYPE">
          /swc root/swc2/pport1
        </P-PORT-PROTOTYPE-REF>
        <OPERATION-PROTOTYPE-REF DEST="OPERATION-PROTOTYPE">
          /interfaces/CS string to int/parse
        </OPERATION-PROTOTYPE-REF>
      </OPERATION-IREF>
    </OPERATION-INVOKED-EVENT>
  </EVENTS>
  <RUNNABLES>
    <RUNNABLE-ENTITY>
      <SHORT-NAME>run21</SHORT-NAME>
      <CAN-BE-INVOKED-CONCURRENTLY>true</CAN-BE-INVOKED-CONCURRENTLY>
      <SYMBOL>run21</SYMBOL>
    </RUNNABLE-ENTITY>
    <RUNNABLE-ENTITY>
      <SHORT-NAME>run22</SHORT-NAME>
      <CAN-BE-INVOKED-CONCURRENTLY>false</CAN-BE-INVOKED-CONCURRENTLY>
      <DATA-RECEIVE-POINTS>
        <DATA-RECEIVE-POINT>
          <SHORT-NAME>dra1</SHORT-NAME>
          <DATA-ELEMENT-IREF>
            <R-PORT-PROTOTYPE-REF DEST="R-PORT-PROTOTYPE">
              /swc root/swc2/rport1
            </R-PORT-PROTOTYPE-REF>
            <DATA-ELEMENT-PROTOTYPE-REF DEST="DATA-ELEMENT-PROTOTYPE">
              /interfaces/SR Int16/intValue1
            </DATA-ELEMENT-PROTOTYPE-REF>
          </DATA-ELEMENT-IREF>
        </DATA-RECEIVE-POINT>
        <DATA-RECEIVE-POINT>
          <SHORT-NAME>dra2</SHORT-NAME>
          <DATA-ELEMENT-IREF>
            <R-PORT-PROTOTYPE-REF DEST="R-PORT-PROTOTYPE">
              /swc root/swc2/rport1
            </R-PORT-PROTOTYPE-REF>
            <DATA-ELEMENT-PROTOTYPE-REF DEST="DATA-ELEMENT-PROTOTYPE">
              /interfaces/SR Int16/intValue2
            </DATA-ELEMENT-PROTOTYPE-REF>
          </DATA-ELEMENT-IREF>
        </DATA-RECEIVE-POINT>
      </DATA-RECEIVE-POINTS>
      <SYMBOL>run22</SYMBOL>
    </RUNNABLE-ENTITY>
  </RUNNABLES>
  <SUPPORTS-MULTIPLE-INSTANTIATION>false</SUPPORTS-MULTIPLE-INSTANTIATION>
</INTERNAL-BEHAVIOR>
<IMPLEMENTATION>
  <SHORT-NAME>implSwc1</SHORT-NAME>
  <BEHAVIOR-REF DEST="INTERNAL-BEHAVIOR">/swc root/intBehSwc1</BEHAVIOR-REF>
  <CODE-DESCRIPTOR>
    <SHORT-NAME>src</SHORT-NAME>
    <TYPE>SRC</TYPE>
  </CODE-DESCRIPTOR>
  <PROGRAMMING-LANGUAGE>C</PROGRAMMING-LANGUAGE>
</IMPLEMENTATION>
<IMPLEMENTATION>
  <SHORT-NAME>implSwc2</SHORT-NAME>
  <BEHAVIOR-REF DEST="INTERNAL-BEHAVIOR">/swc root/intBehSwc2</BEHAVIOR-REF>
  <CODE-DESCRIPTOR>
    <SHORT-NAME>src</SHORT-NAME>
    <TYPE>SRC</TYPE>
  </CODE-DESCRIPTOR>
  <PROGRAMMING-LANGUAGE>C</PROGRAMMING-LANGUAGE>
</IMPLEMENTATION>
</ELEMENTS>
</AR-PACKAGE>
```

Simple example (*RAWFP*)

```
swc1 =  component $ do
        pport1 <- providedDataElement
        rport1 <- requiredOperation
        runnable (MinInterval 0) [Timed 0.1] (run11 pport1)
        runnable (MinInterval 0) [Timed 0.05] (run12 pport1 rport1)
        return (pport1, rport1)

swc2 = component $ do
        rport2 <- requiredDataElement
        pport2 <- providedOperation
        serverRunnable Concurrent [pport2] run21
        runnable (MinInterval 0) [Timed 0.05] (run22 rport2)
        return (pport2, rport2)

root = do
        (pdata,rop) <- swc1
        (pop,rdata) <- swc2
        connect pdata rdata
        connect rop pop
```

```
run11 pport1 = do
        ...
        rte_write pport1 val
        ...

run12 pport1 rport1 = do
        ...
        val2 <- rte_call rport1 val1
        ...
        rte_write pport1 val2

run21 arg = do
        ... arg ...
        ...
        return res

run22 rport2 = do
        ...
        val <- rte_read rport2
        ...
```

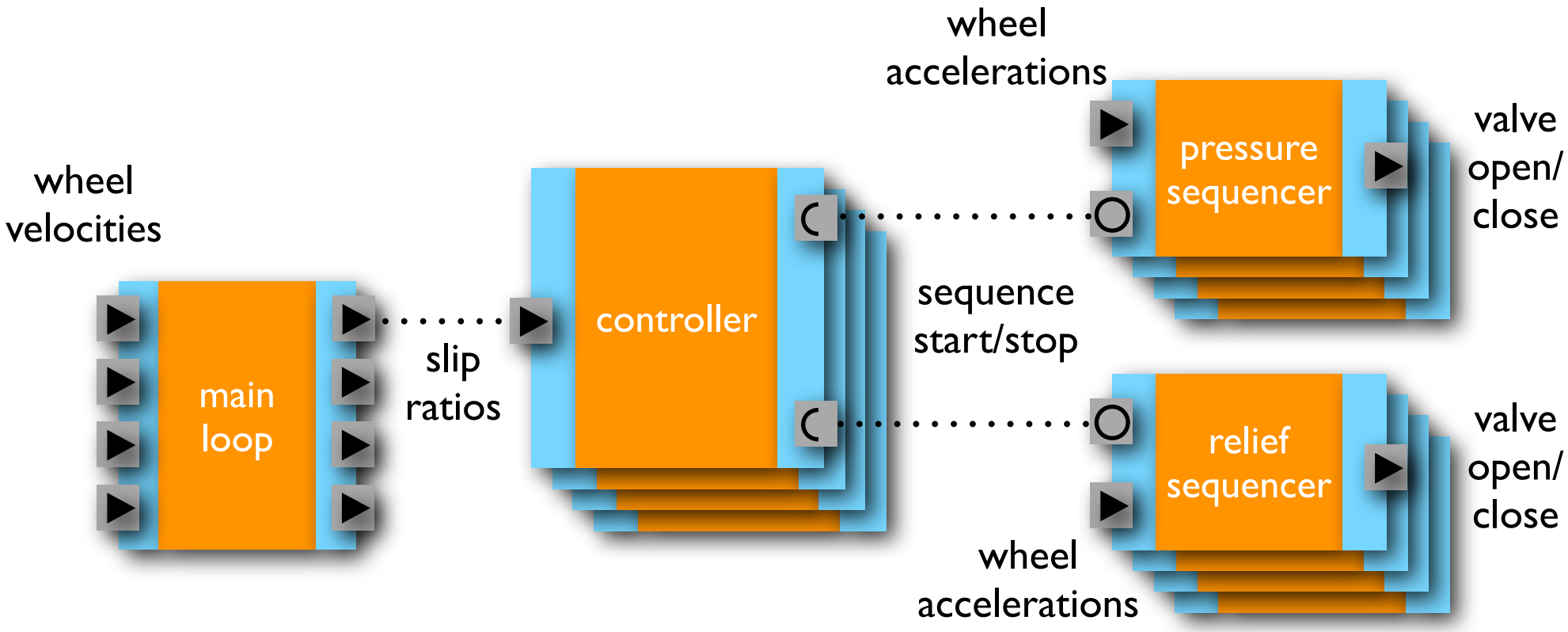
The *RAWFP* AUTOSAR DSEL

- Combines
  1. The structure of AUTOSAR software components
  2. The API of AUTOSAR's run-time environment (RTE)
  3. The functional behavior of its host language Haskell
- Formalizes
  a) The RTE semantics (concurrency, interaction & timing)
  b) Component scoping and encapsulation
  c) The potential meaning of AUTOSAR system constraints

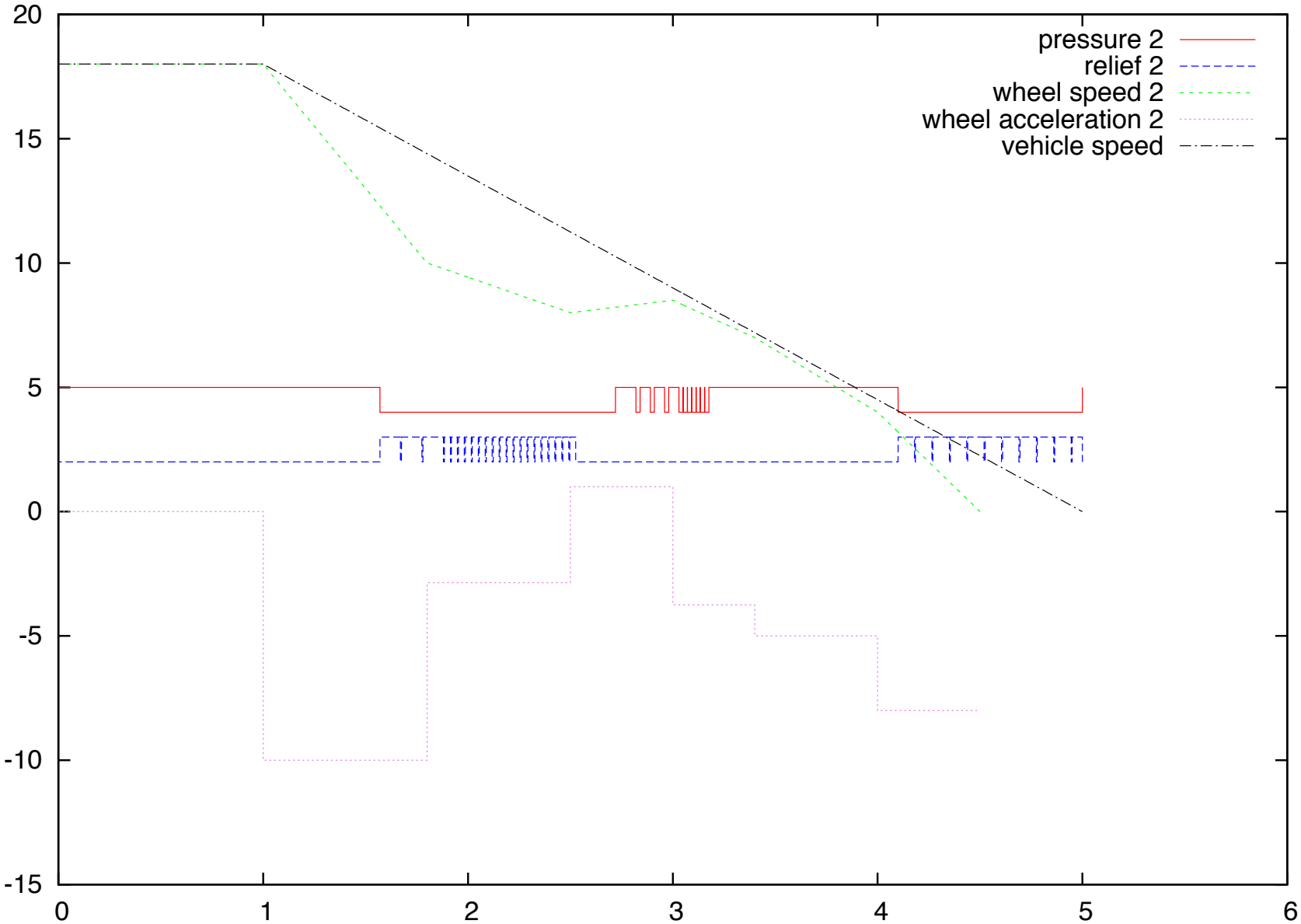
The *RAWFP* AUTOSAR DSEL

- Current achievements:
  - A simulator executing AUTOSAR systems defined entirely on the software component level
  - A modular scheduling architecture, including a fully randomized scheduler option
  - Integration with *QuickCheck* (with trace shrinking)
  - Prototype C code generation
- *Caveat: work is very much in progress!*

Demo: An ABS system



Demo: simulation output



Outlook

- Next steps:
  - Extending the AUTOSAR standard coverage
  - Improving simulator efficiency
  - Integrating code generation with industrial tools
  - Assembling and reporting semantic ambiguities found
- Long-term goals:
  - To provide a tool for truly high-level modeling and simulation of automotive software systems
  - To fully automate the translation of models to executable AUTOSAR code