Microelectronics Engineering Group

University of Cantabria

Software Simulation Technologies in Virtual Platforms

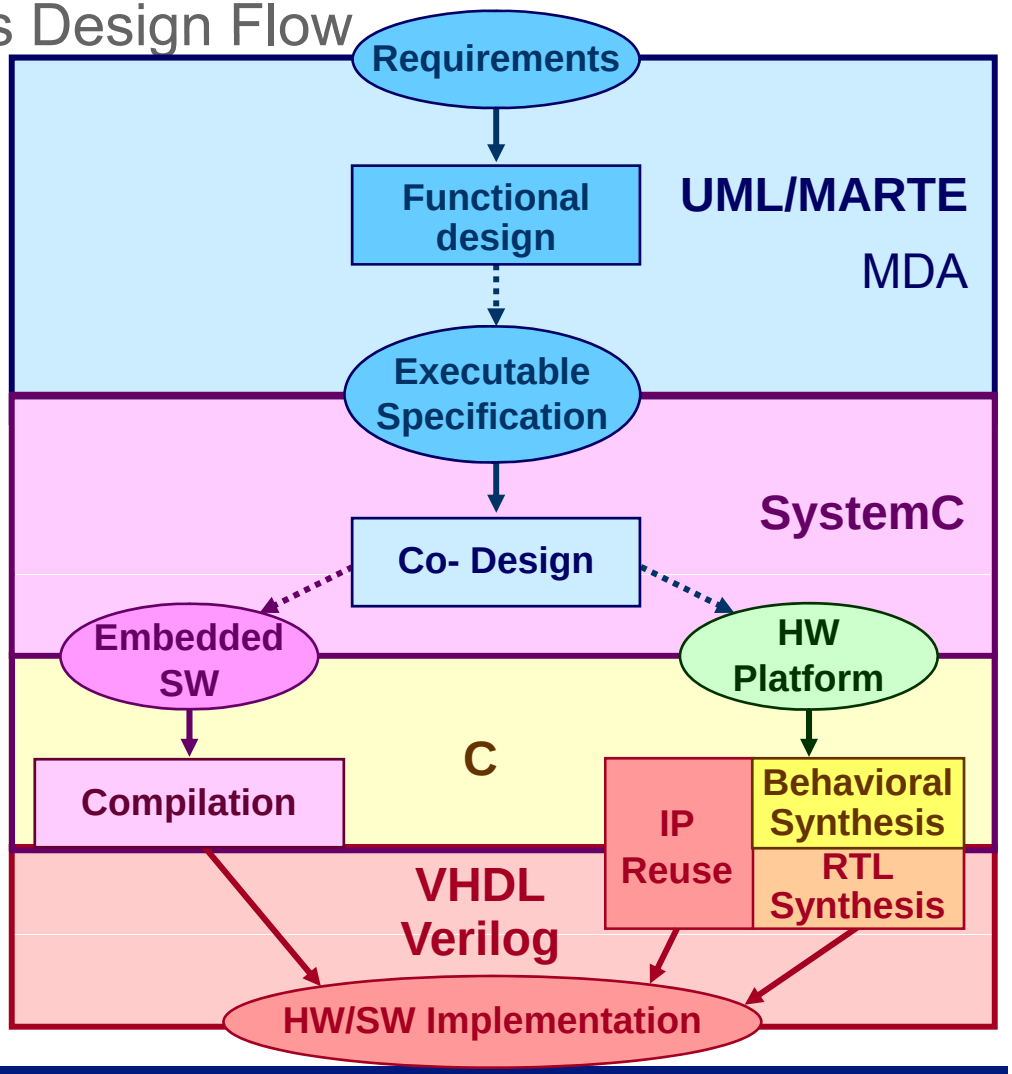
Eugenio Villar & Héctor Posadas

Microelectronics Engineering Group

University of Cantabria

Context

- HW/SW Embedded Systems Design Flow
  - HW/SW Simulation
  - Performance Analysis
    - avoiding slow design iterations
  - Design Verification
    - At the different abstraction levels



Agenda

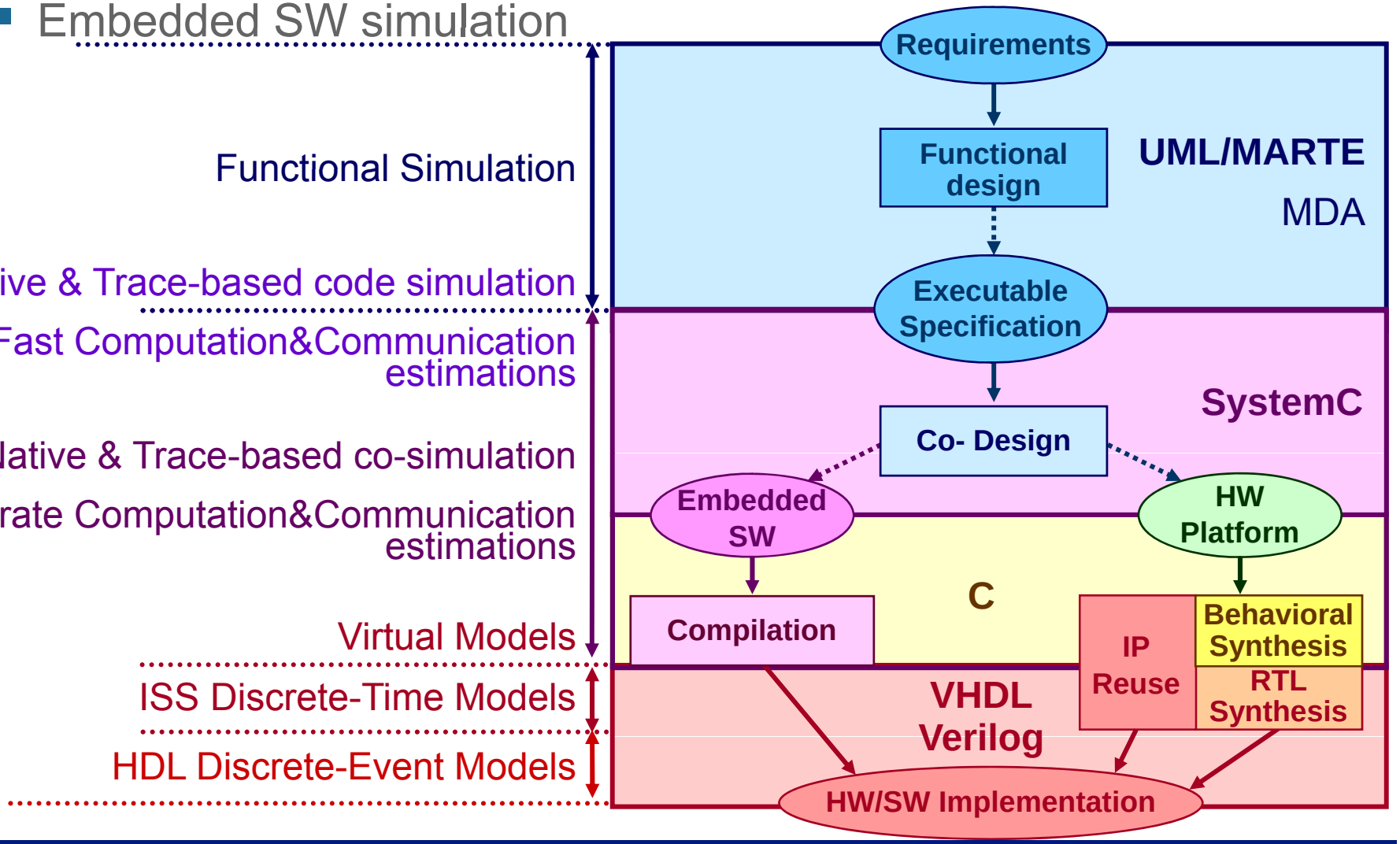
- Motivation
  - Why SW performance analysis
- Software Simulation Technologies in Virtual Platforms
  - Simulation Technologies at different abstraction levels
  - SCoPE: SW performance analysis for DSE
    - Native simulation
    - After architectural mapping
  - SCoPE^{+}: SW performance analysis for DSE
    - Compositional Native simulation
    - Before architectural mapping
    - Direct PSM from the same PIM
- Conclusions

Motivation

- The MPSoC
  - Multi-processing platform
    - ASIC
    - FPGA
    - Commercial multi-processing platform
  - SW-centric design methodology
    - Most of the functionality implemented as Embedded SW
    - With ‘some’ application-specific HW

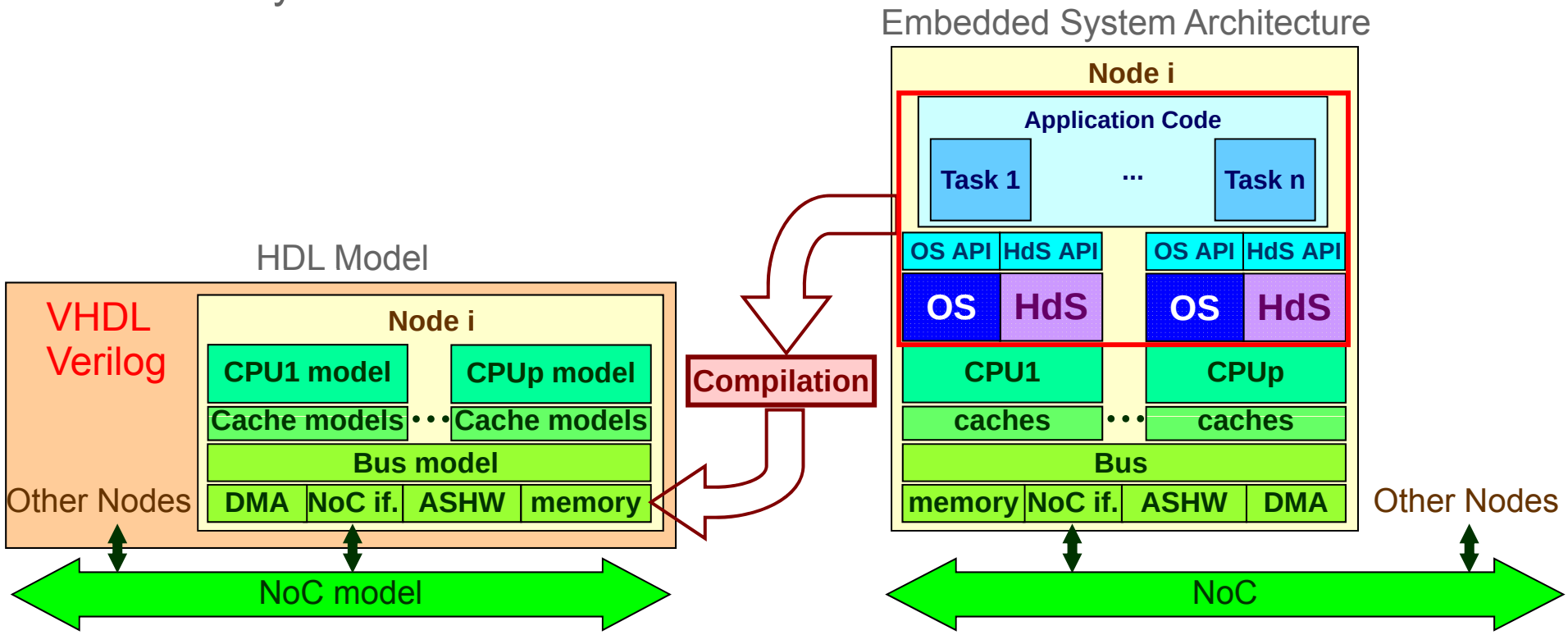
SW Simulation Technologies

- Embedded SW simulation



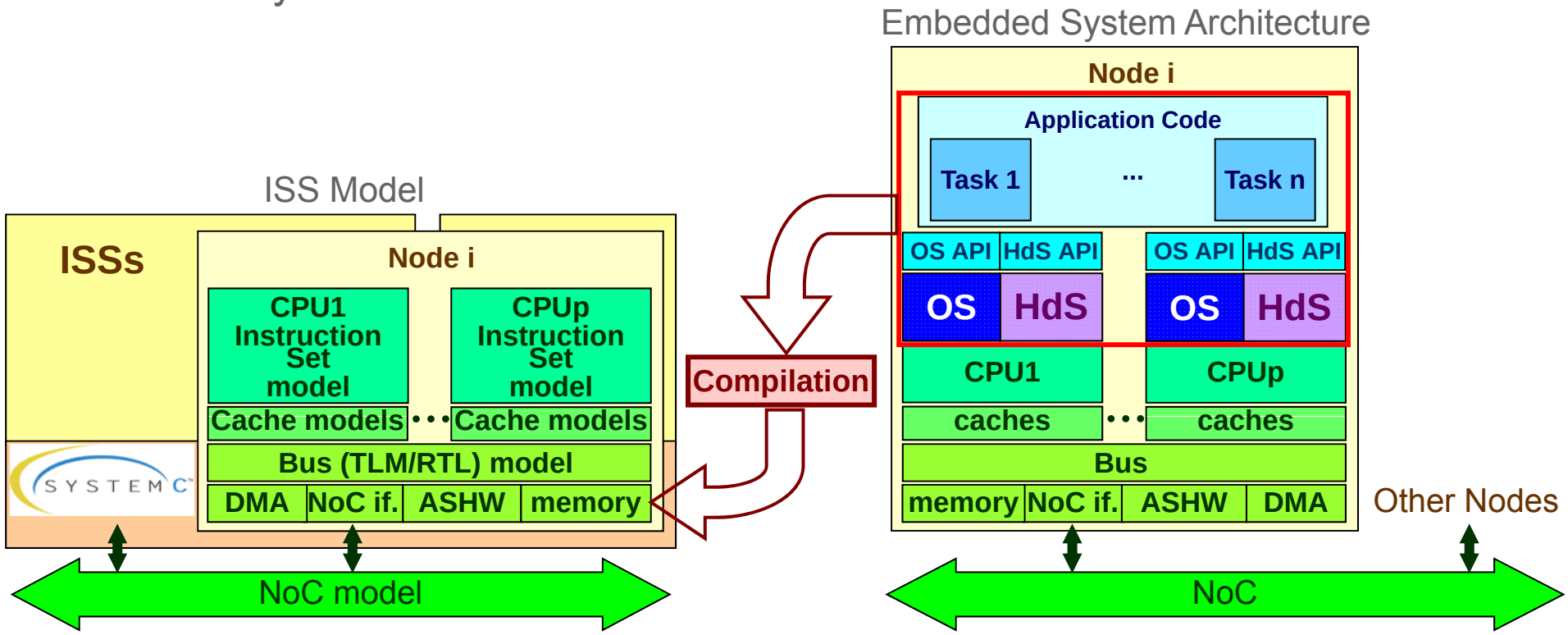
SW Simulation Technologies

- HDL simulation
  - Very detailed Model
  - Very accurate
  - Very slow



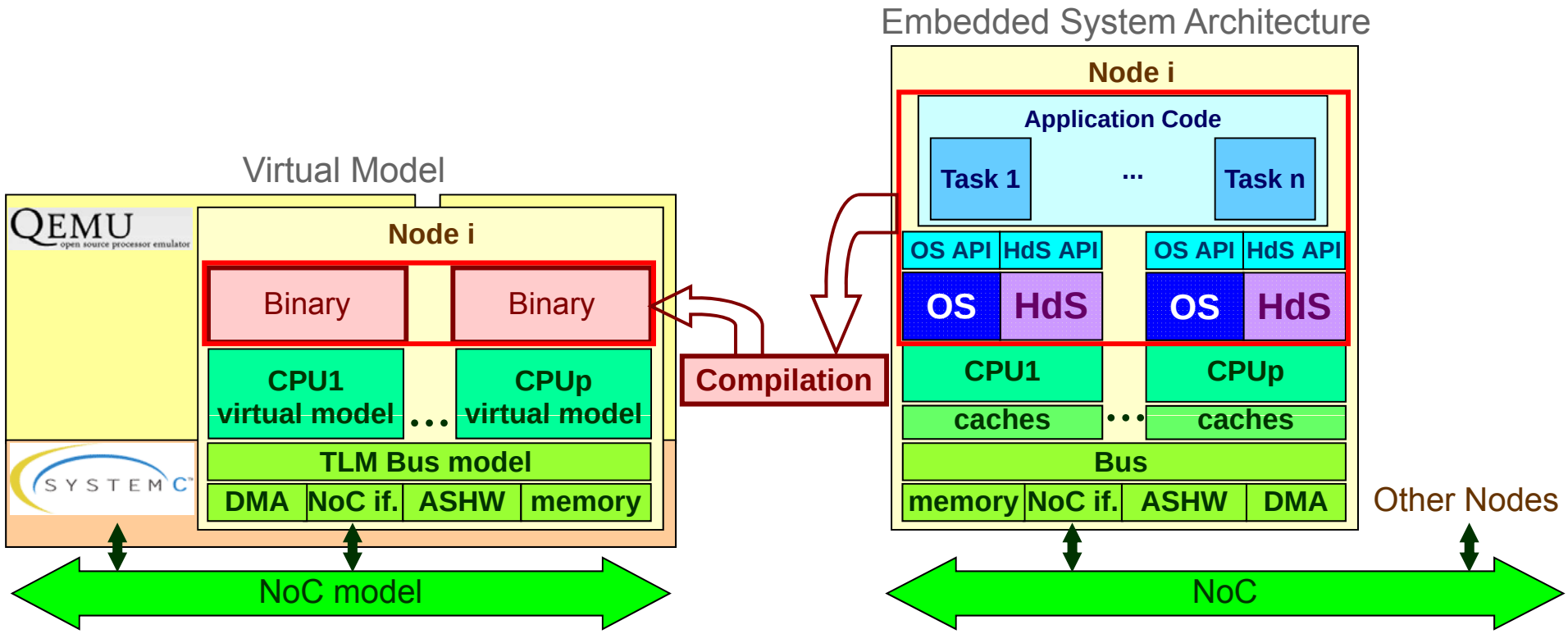
SW Simulation Technologies

- ISS simulation
  - Very detailed Model
  - Very accurate
  - Very slow



SW Simulation Technologies

- Virtualization
  - Target virtual model on host

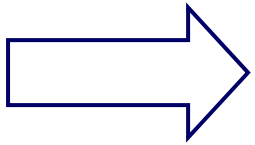


SW Simulation Technologies

- Virtualization (QEMU)
  - Detailed model
    - High modeling cost
    - Late design steps
  - Faster than ISS

PowerPC (200 MHz)

```
# r1 = r1 - 16
addi r1,r1,-16
```



Intel Core i5 (2.40 GHz)

```
# movl_T0_r1
# ebx = env->regs[1]
mov 0x4(%ebp),%ebx

# addl_T0_im -16 # ebx = ebx - 16
add $0xfffffff0,%ebx # movl_r1_T0

# env->regs[1] = ebx
mov %ebx,0x4(%ebp)
```

SW Simulation Technologies

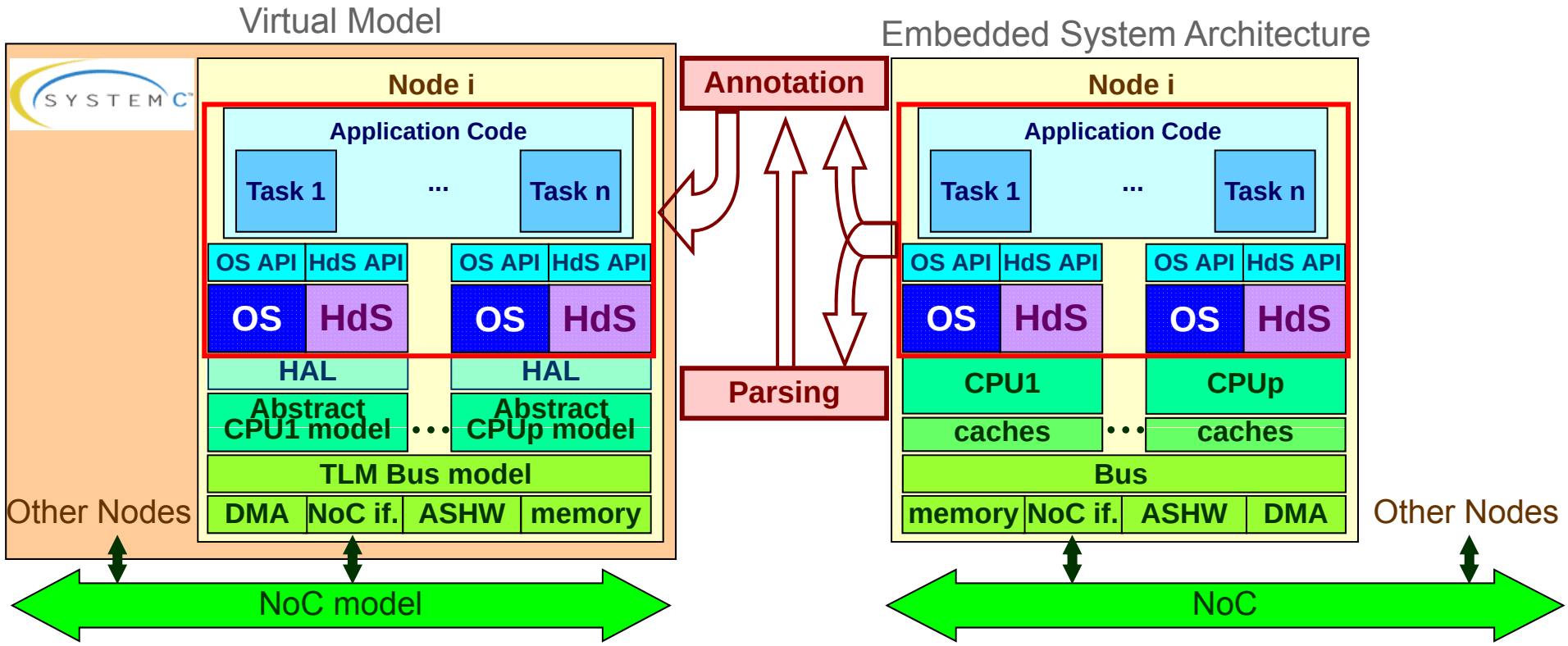
- Virtualization
  - Functional emulation
  - Rough timed simulation
    - i.e. 1 cycle per instruction
  - Additional effort needed for more accurate modeling
    - Execution times
    - Power consumption
    - Caches
    - …
  - Requires a specific Virtual Model for each processor
- Commercial tools
  - OVP, FastModels, Cadence, Carbon, Synopsys (CoWare), etc.

SW Simulation Technologies

- Native & Trace-based simulation
  - Embedded code directly executed by the host
  - Good accuracy
    - Native back-annotation
    - Trace analysis
  - Fast execution time

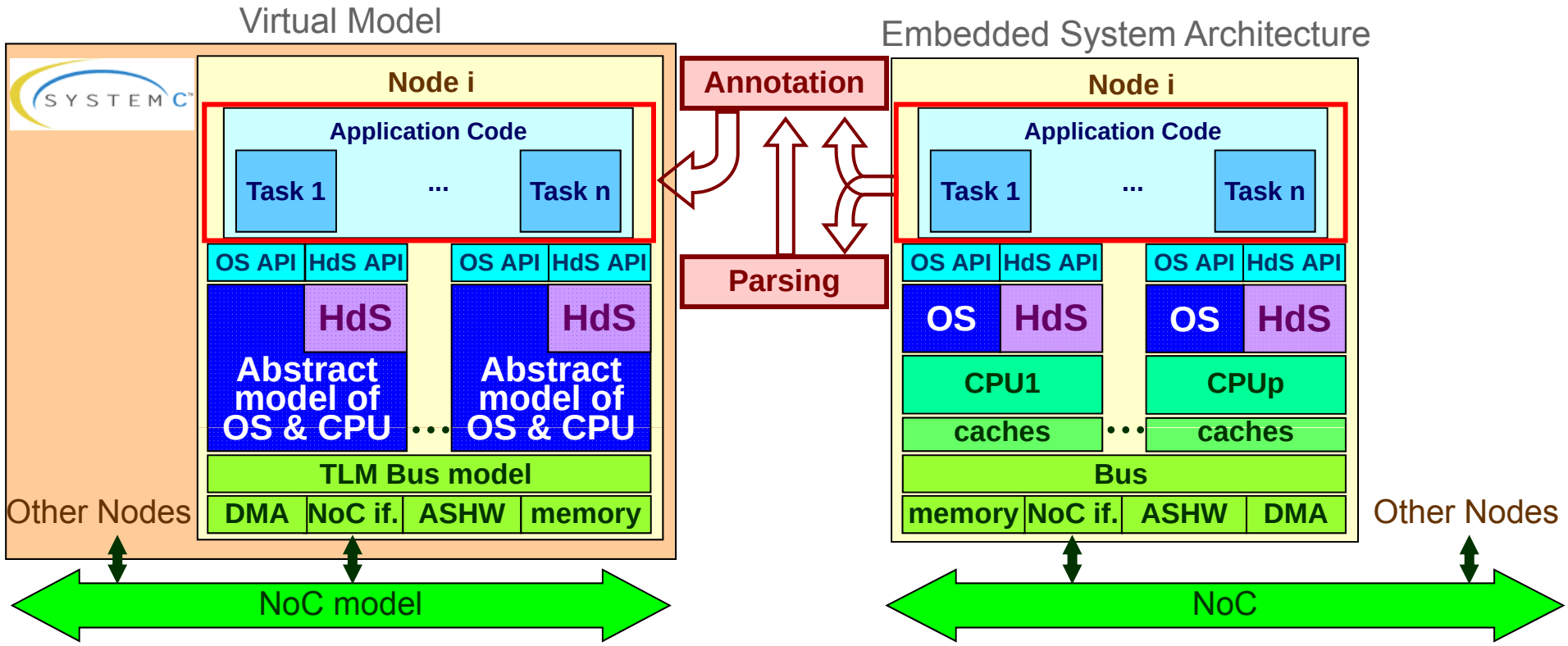
SW Simulation Technologies

- Native simulation based on HAL API
  - Abstraction of the HW platform



SW Simulation Technologies

- Native simulation based on OS API
  - Abstraction of the SW platform



SW Simulation Technologies

- Basic code annotation in native simulation

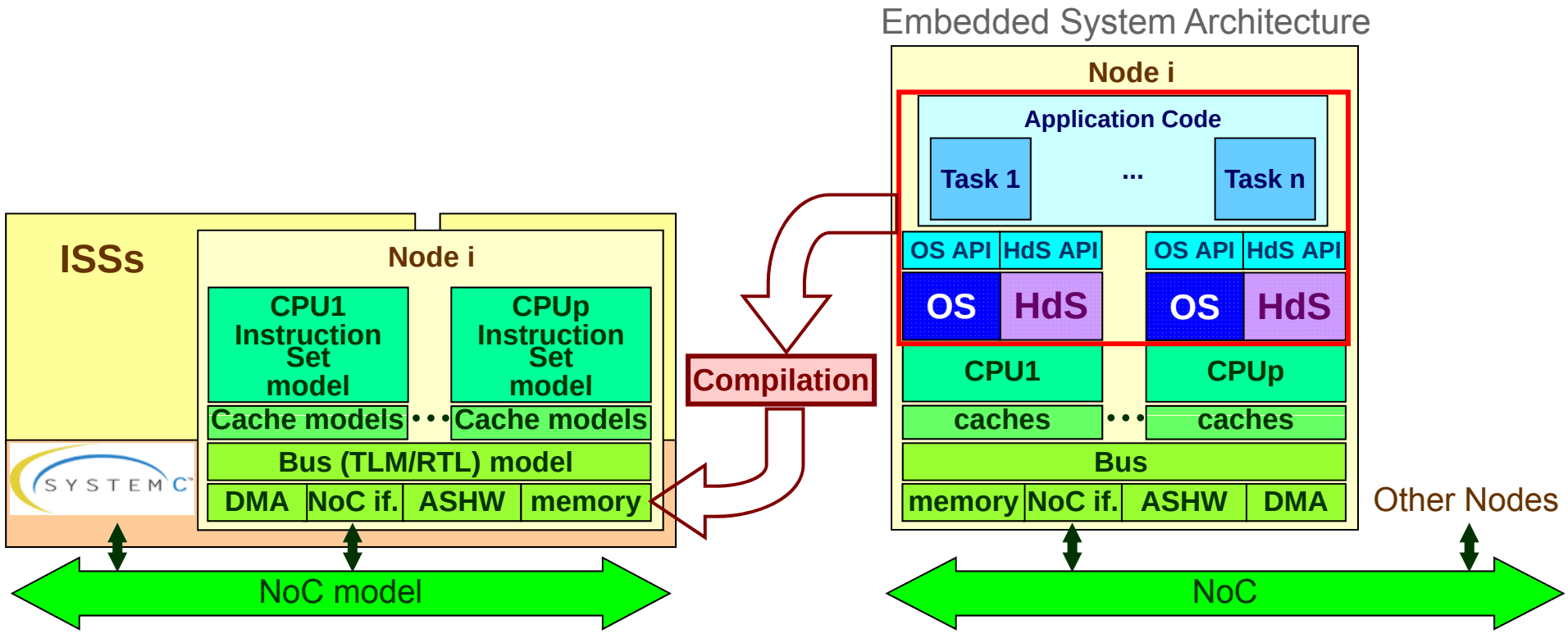
```
...
Overflow = 0;
s = 1L;
for (i = 0; i < L_subfr; i++) {
    Carry = 0;
    s = L_macNs(s, xn[i], y1[i]);
    if (Overflow != 0) {
        break;}}
if (Overflow == 0)  {
    exp_xy = norm_l(s);
    if (exp_xy<=0)
        xy = round(L_shr (s, -exp_xy));
    else
        xy = round(L_shl (s, exp_xy));}
mq_send(queue1, &xy, p, t);
...
```

Global variable
int Sim_Time = 0;

Sim_Time += 20;
Sim_Time += 25;
Sim_Time += 15;
Sim_Time += 10;
Sim_Time += 10;
Sim_Time += 10;
wait included

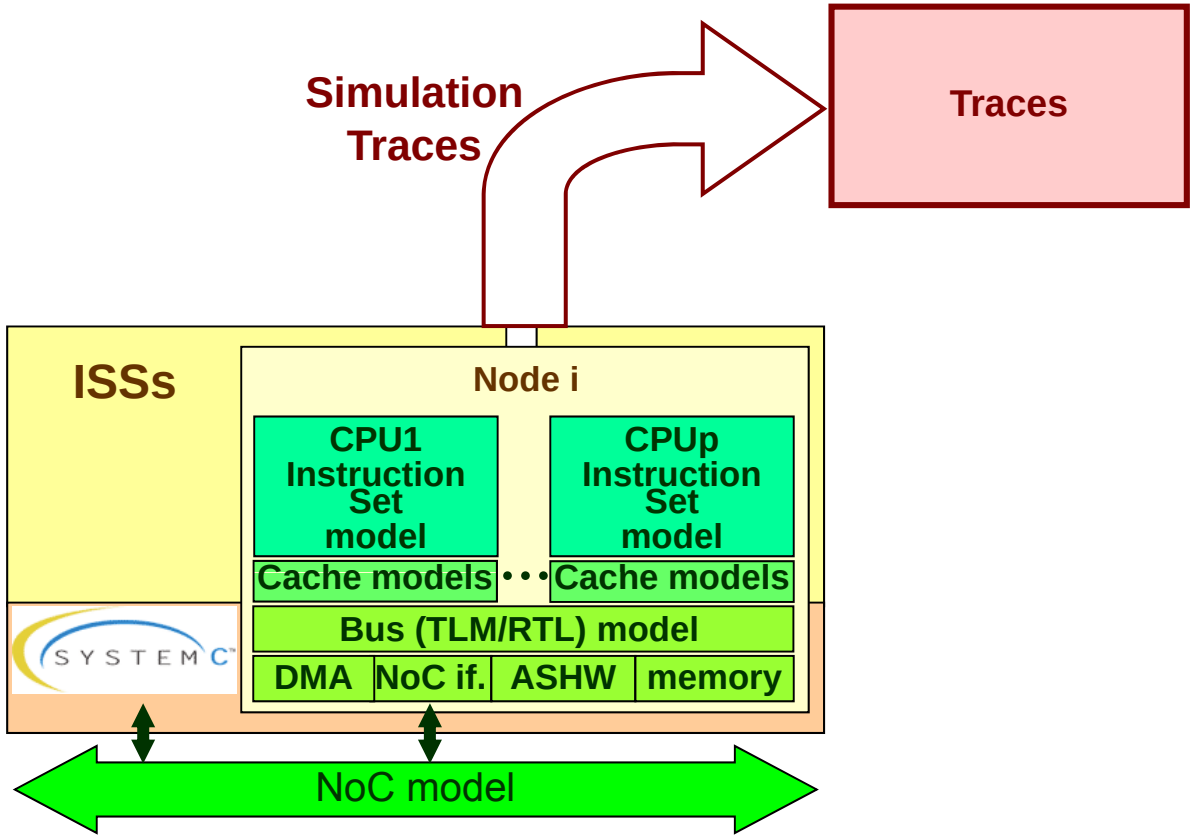
SW Simulation Technologies

- Trace-based simulation
  - Activity traces from detailed models



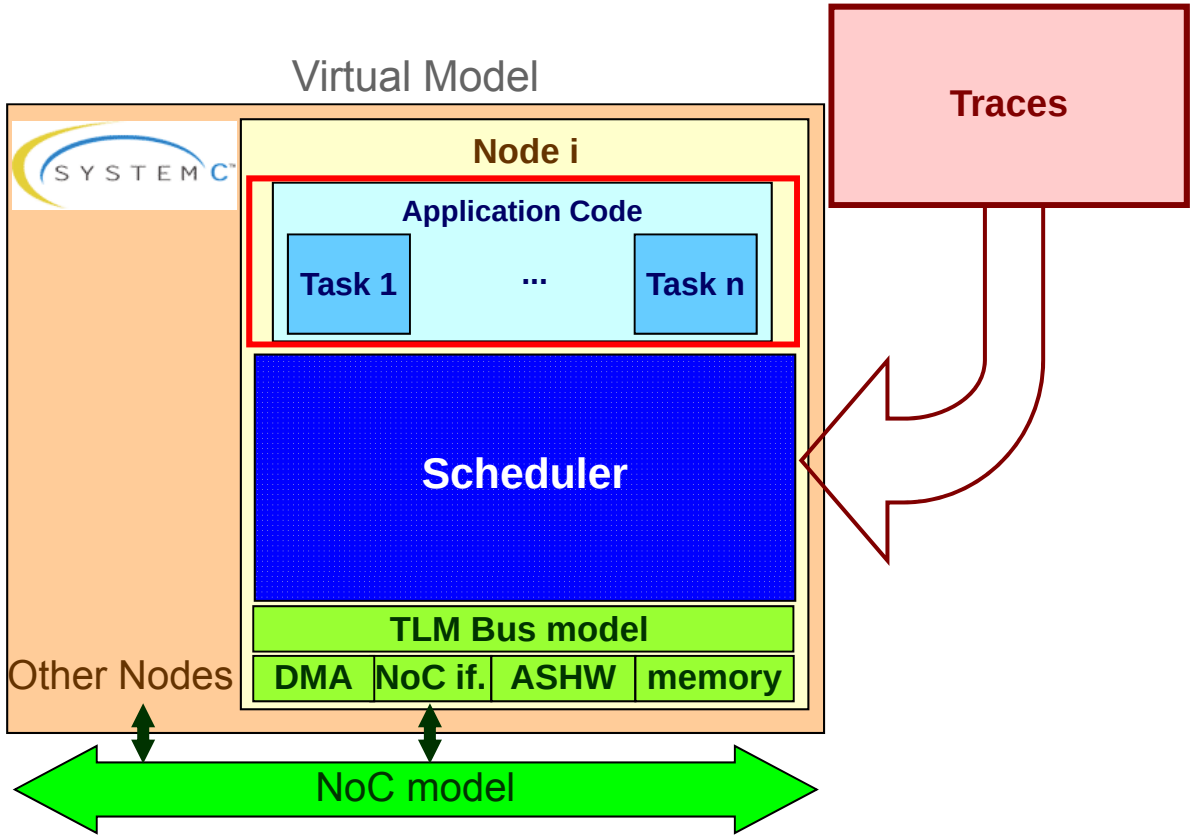
SW Simulation Technologies

- Trace-based simulation



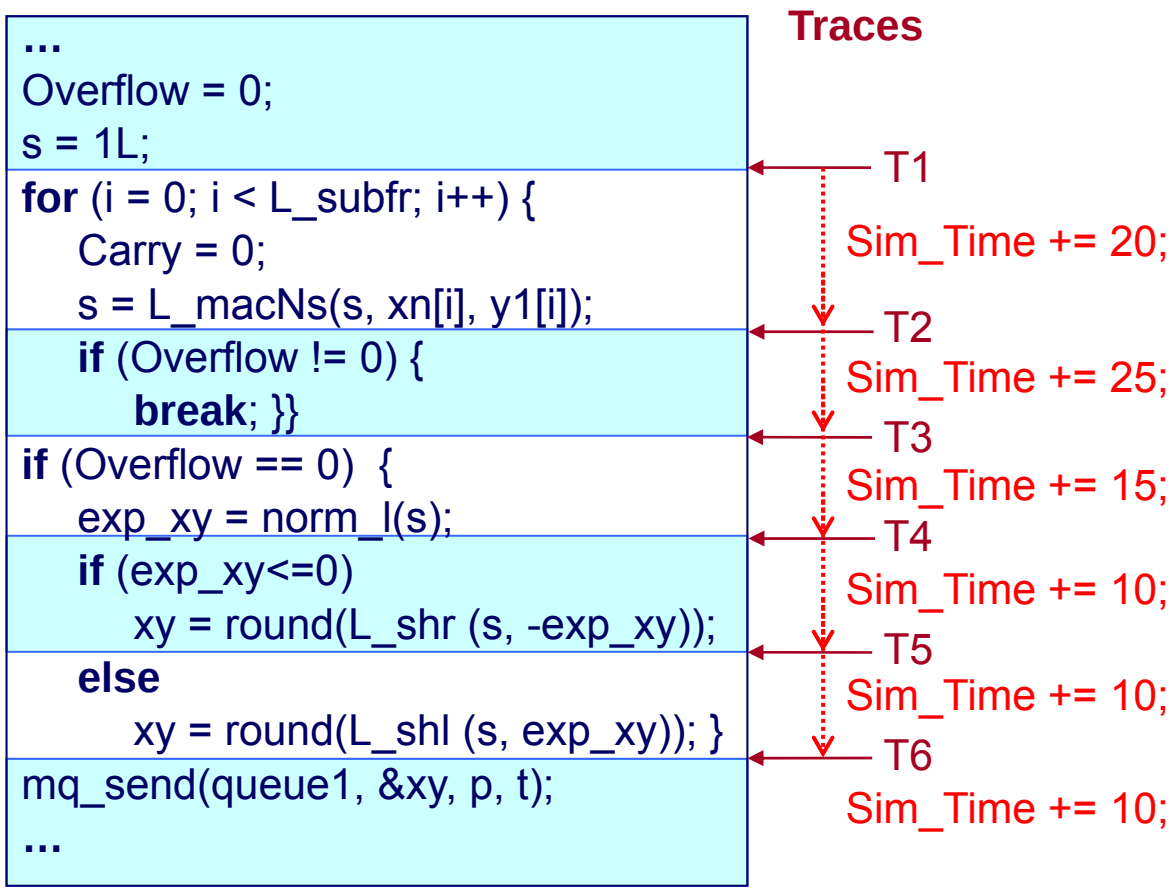
SW Simulation Technologies

- Trace-based simulation
  - Difficult scheduling in complex multi-processing systems



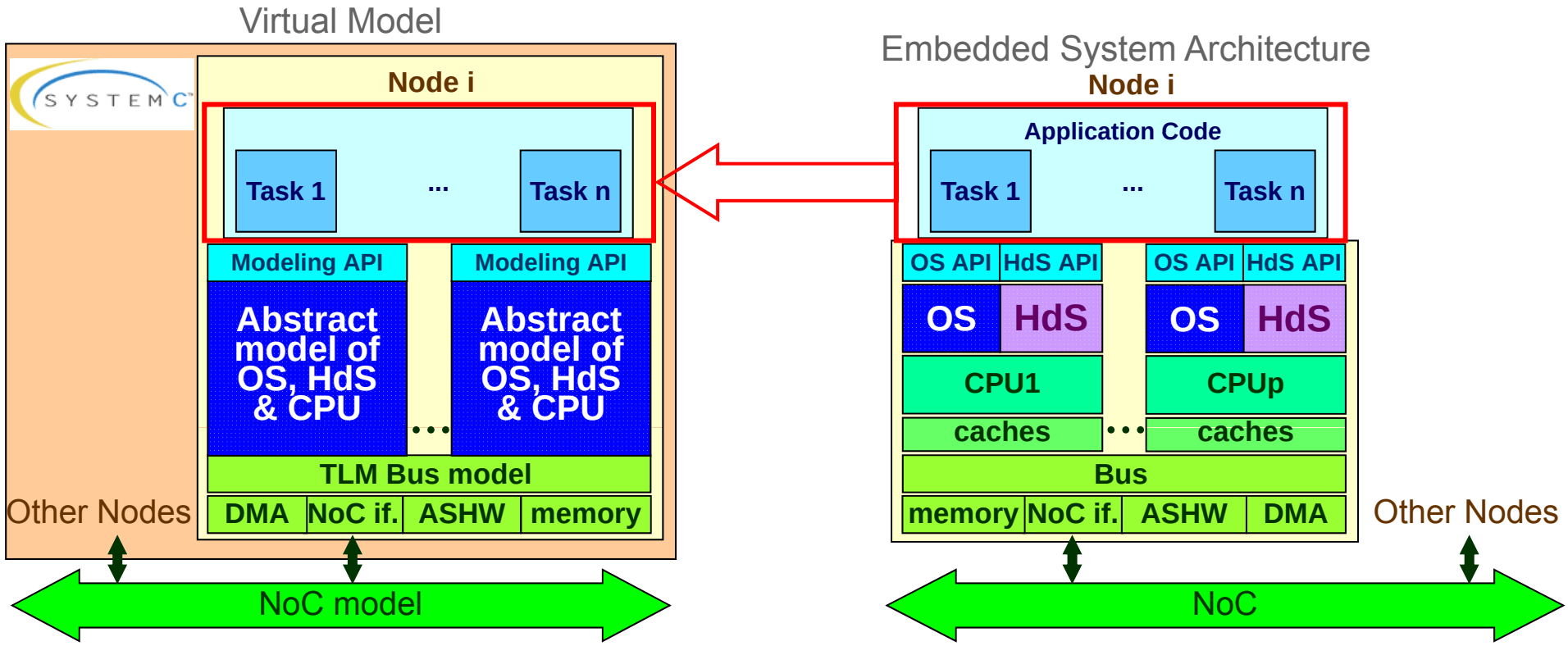
SW Simulation Technologies

- Basic code execution in trace-based simulation



SW Simulation Technologies

- Functional simulation based on code
  - Fastest but least accurate
  - Work-load analysis



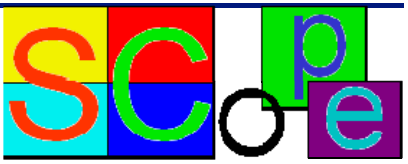
SW Simulation Technologies

- Performance/Error comparison

Technology		Time Estimation	Time & Power Estimation
Functional	Performance	5,000	N.A.
	Error	N.A.	N.A.
Native Trace-based	Performance	1,000	500
	Error	1.3	1.4
Virtualization	Performance	200	T.B.M.
	Error	1.5	T.B.M.
ISS (cycle-accurate)	Performance	10	1
	Error	1.1 (DT)	1.1
HDL	Performance	1	0.1
	Error	1 (DE)	1

  - Rough approximate figures

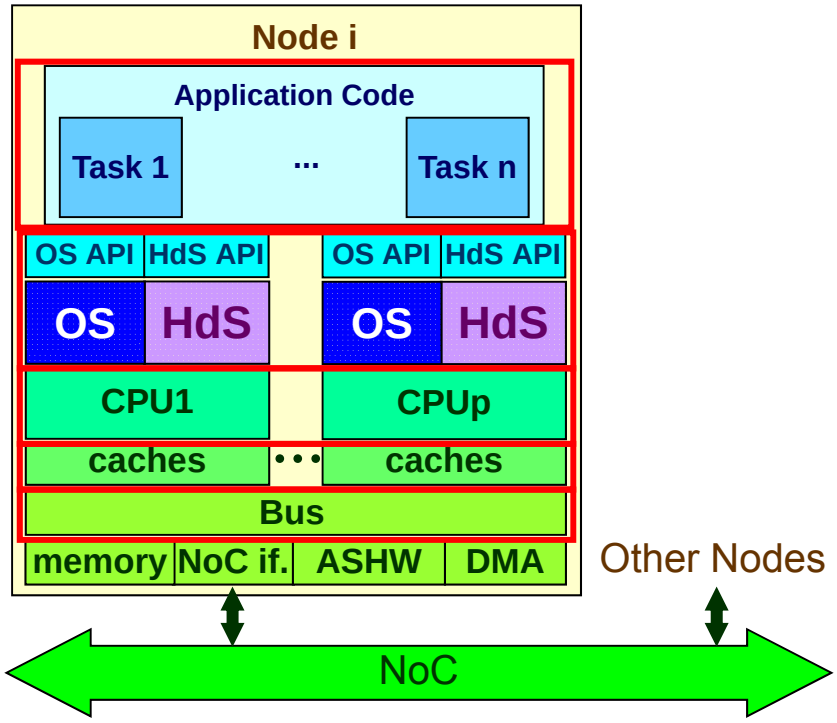
SCoPE: SW Performance Estimation for DSE



- Key features
  - Abstract OS modeling
  - Instruction cache modeling
  - Data cache modeling
  - System power estimation
- Novel features
  - Physical memory accesses
  - Separate memory spaces
  - Configurability for Design-Space Exploration
  - Dynamic Voltage-Frequency Scaling
  - Thermal modeling
  - System composition from IP-XACT components
  - Win32 API

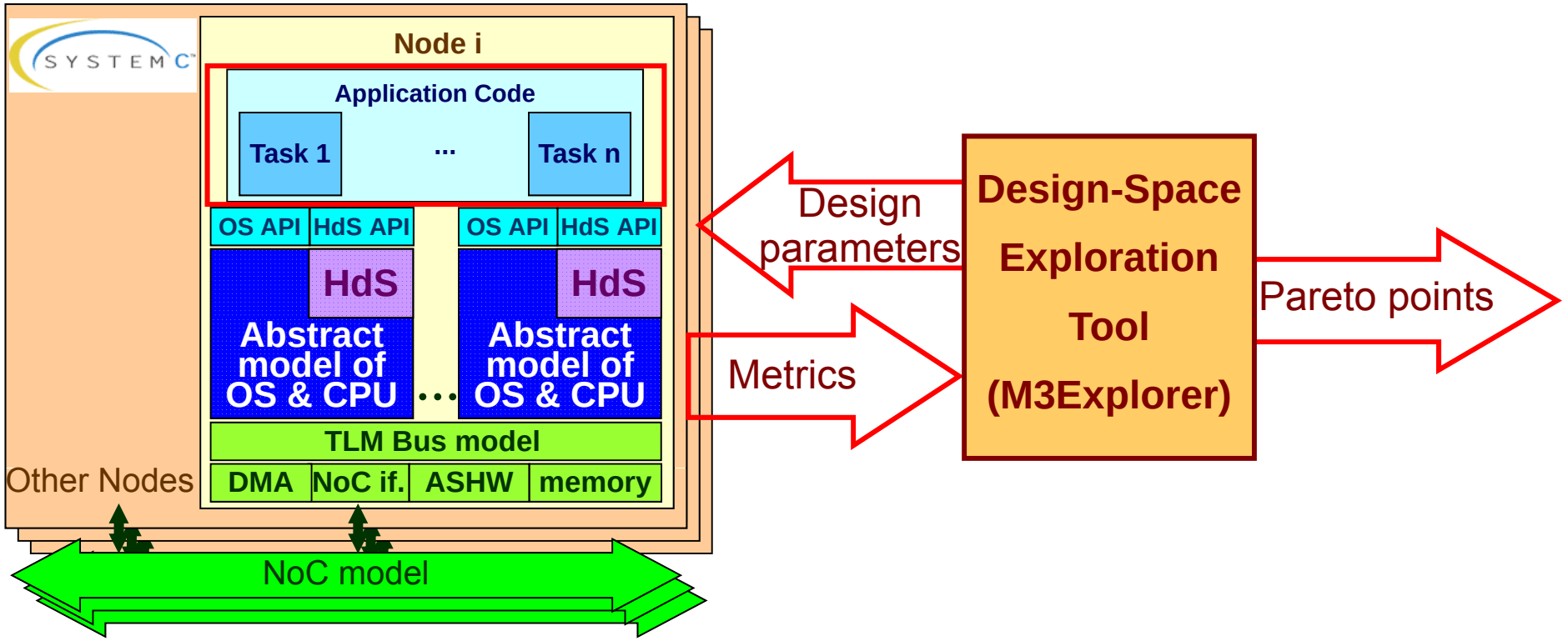
SCoPE: SW Performance Estimation for DSE

- System power estimation
  - Application code
    - Instruction counting from binary
  - OS & HW-dependent SW
    - Function power estimation
  - Caches
    - Counting memory accesses
    - Cache misses
  - Bus
    - Actual bandwidth
      - Cache misses
      - DMA accesses
      - HW accesses
  - HW & NoC
      - SystemC power models



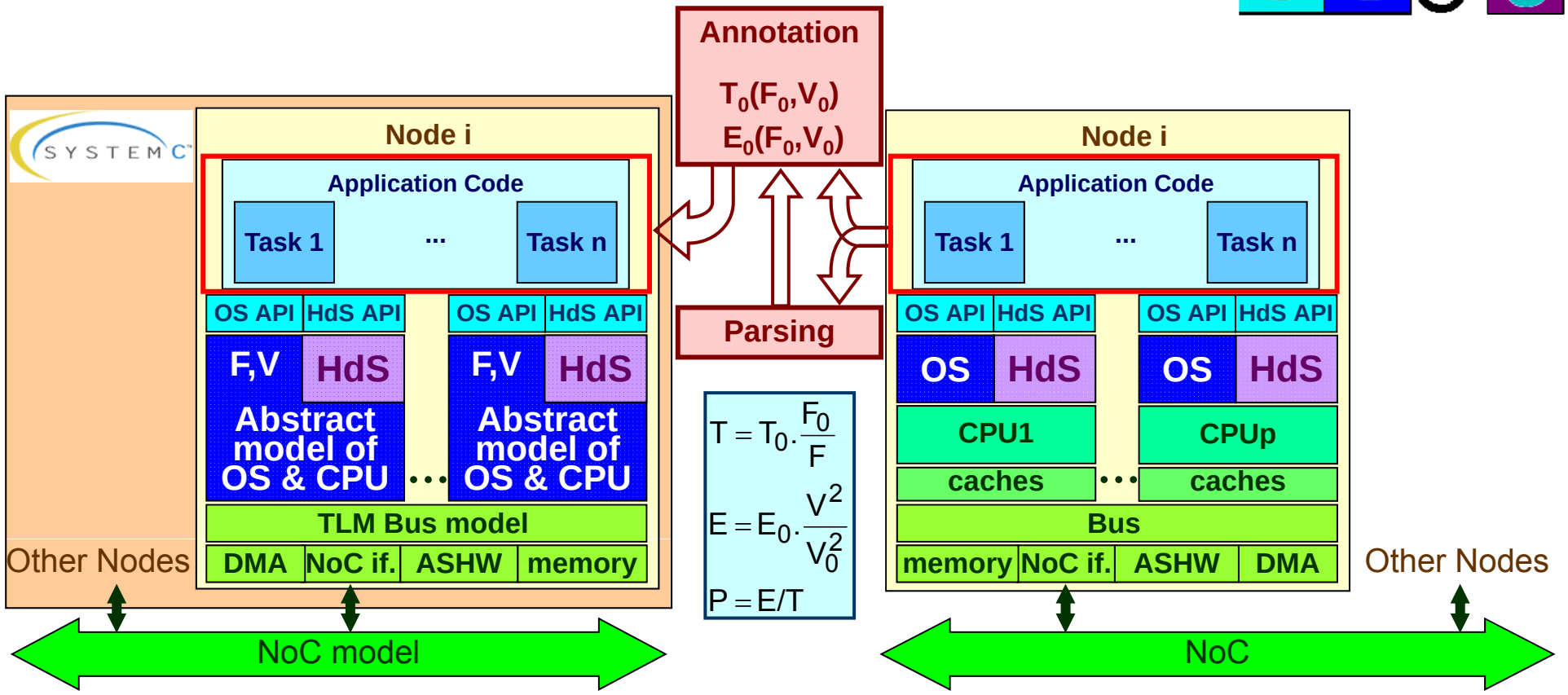
SCoPE: SW Performance Estimation for DSE

- Design-Space Exploration
  - Configurable model



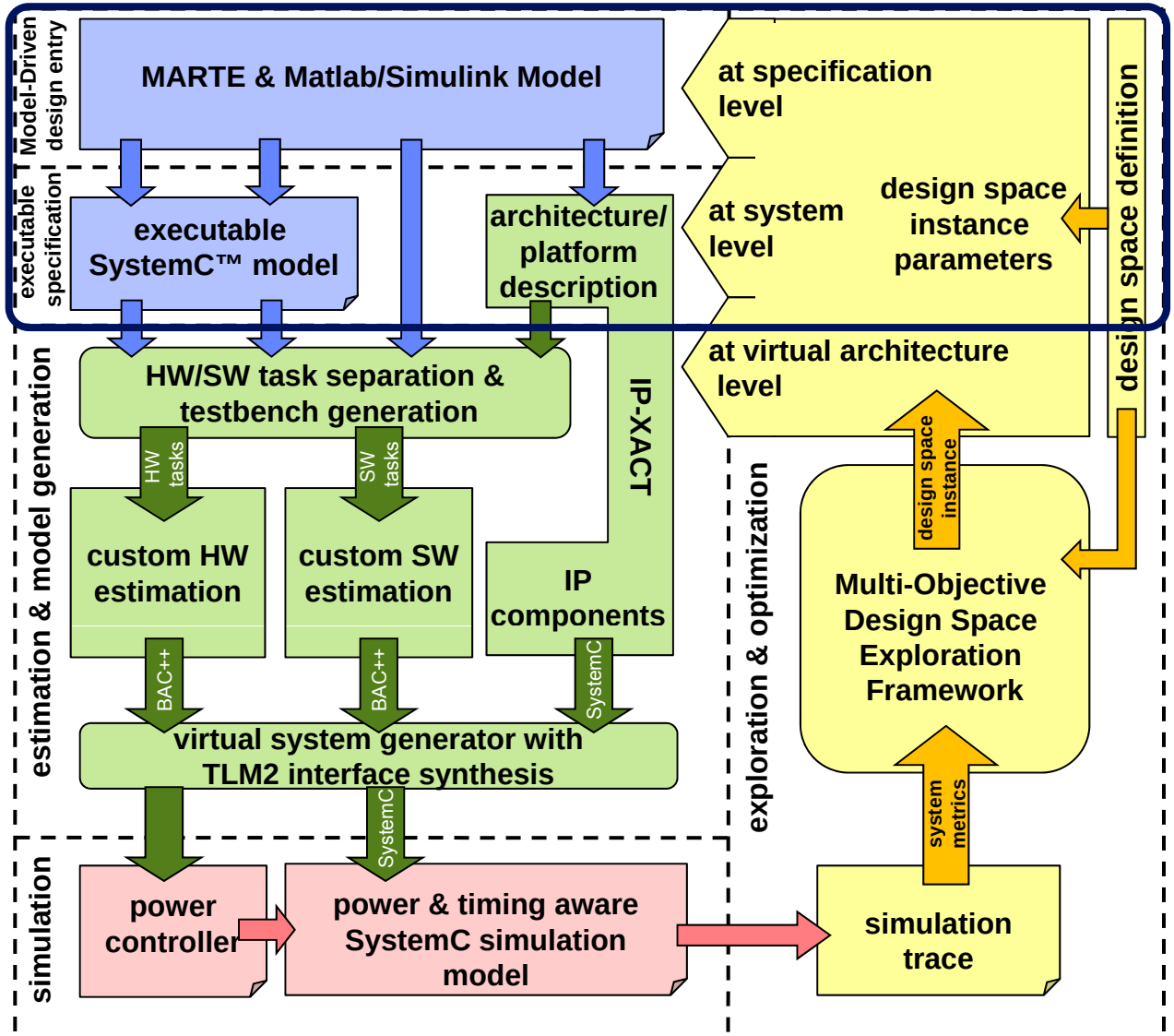
SCoPE: SW Performance Estimation for DSE

- Dynamic Voltage-Frequency Scaling



The COMPLEX Framework

- Model-Driven Design entry (MARTE/Simulink)
- Executable specification
- Power/Timing Estimation & Model Generation
- Power/Timing aware SystemC simulation
- Automatic Multi-Objective Design-Space Exploration
  - at system-level
  - at block-level



SCoPE+: Compositional Native Performance Estimation

- Implementation-Agnostic Platform Independent Frontend
  - CFAM-CM API
- Fulfilling COMPLEX UML/MARTE executive semantics
- System-Level Modeling of Multi-OS execution
- SW/SW-HW/SW-HW/HW communications
  - Architectural mapping agnostic

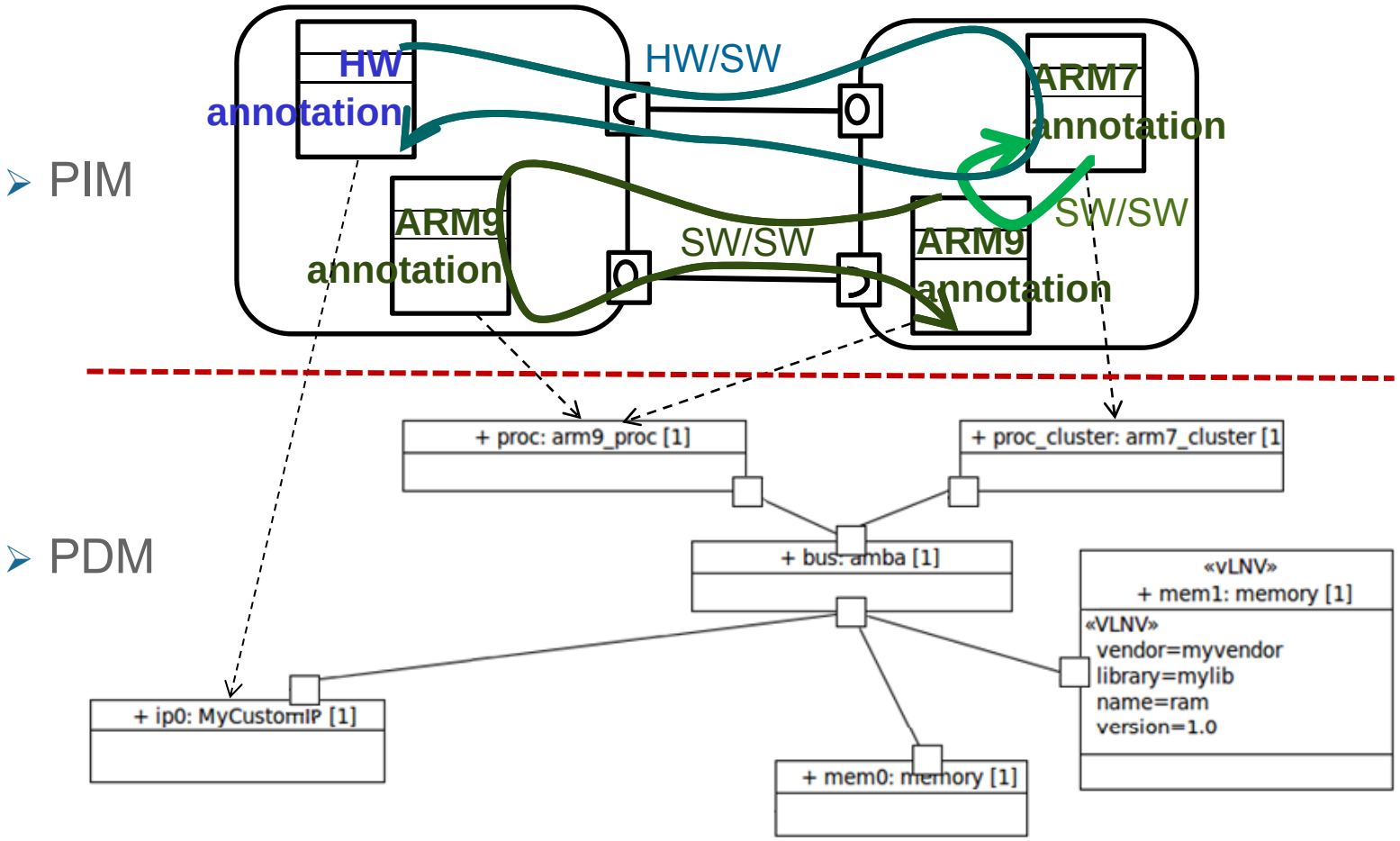
- Taking advantage of the native simulation speed*accuracy

CFAM-CM

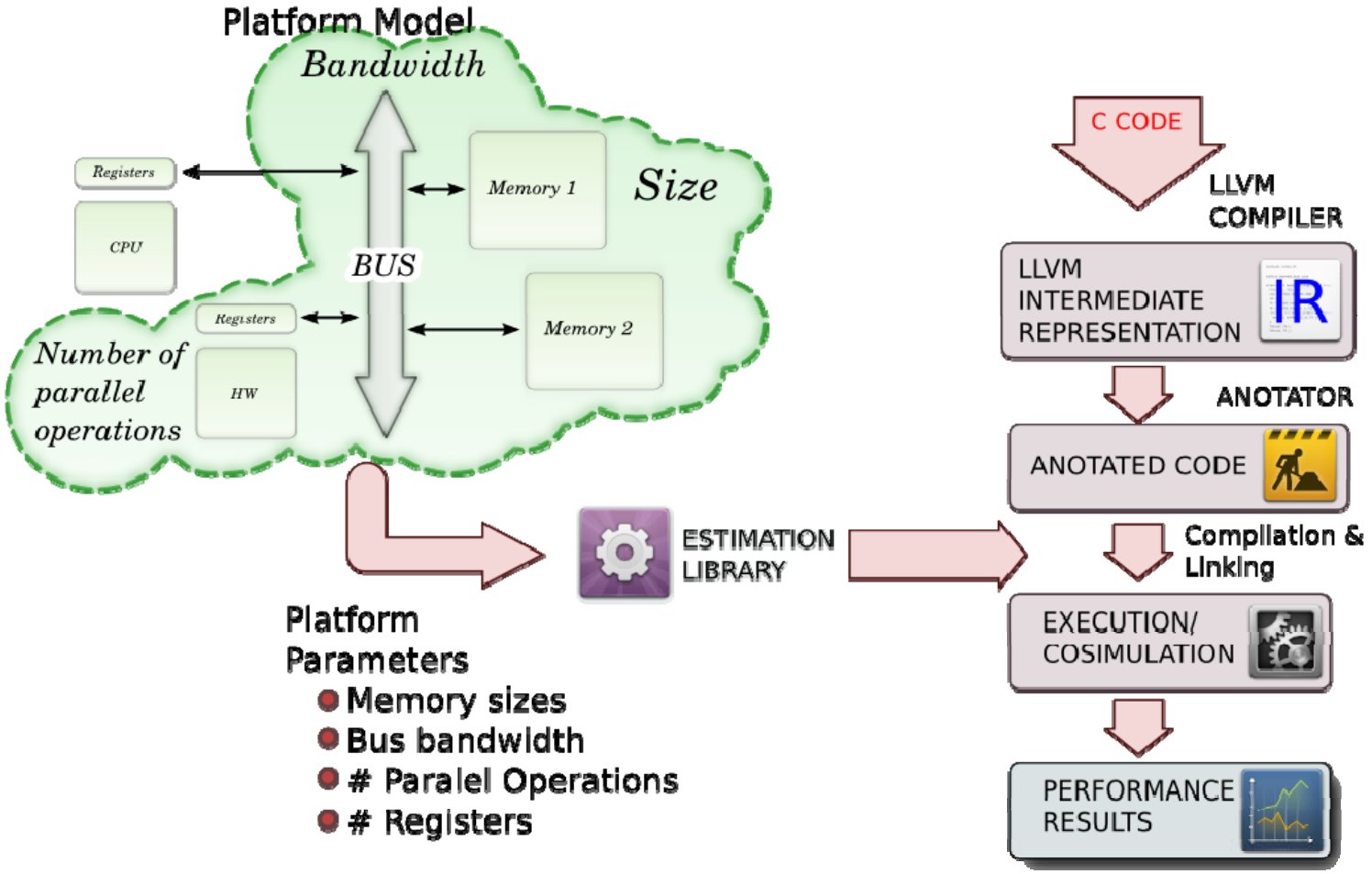
- Macros and functions
  - Concurrent Functional Application & Component Model
- Component Based PIM
- CFAM API
  - Platform services required by functional code
  - Hide RTOS specific calls

CFAM-CM

- Platform-Dependent Estimations directly on the PIM

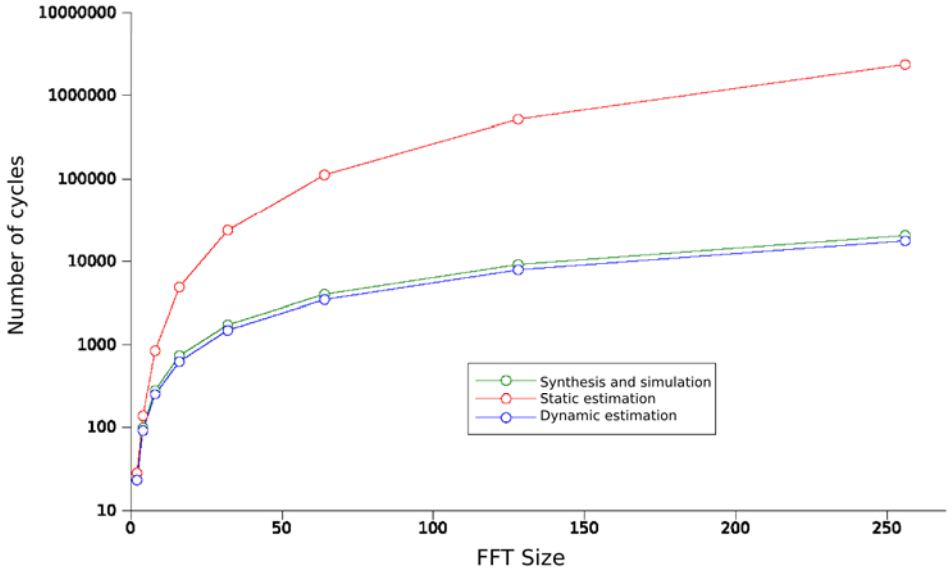


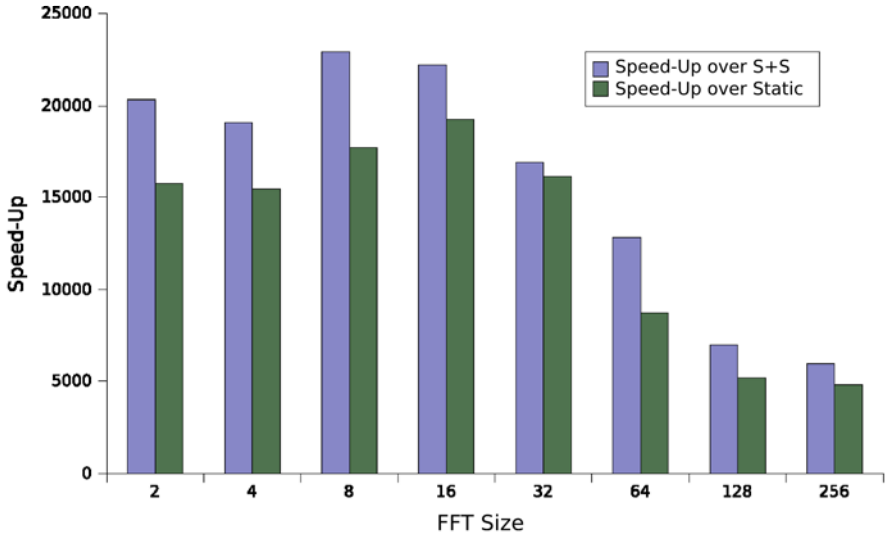
High-Level Custom HW Estimation Methodology



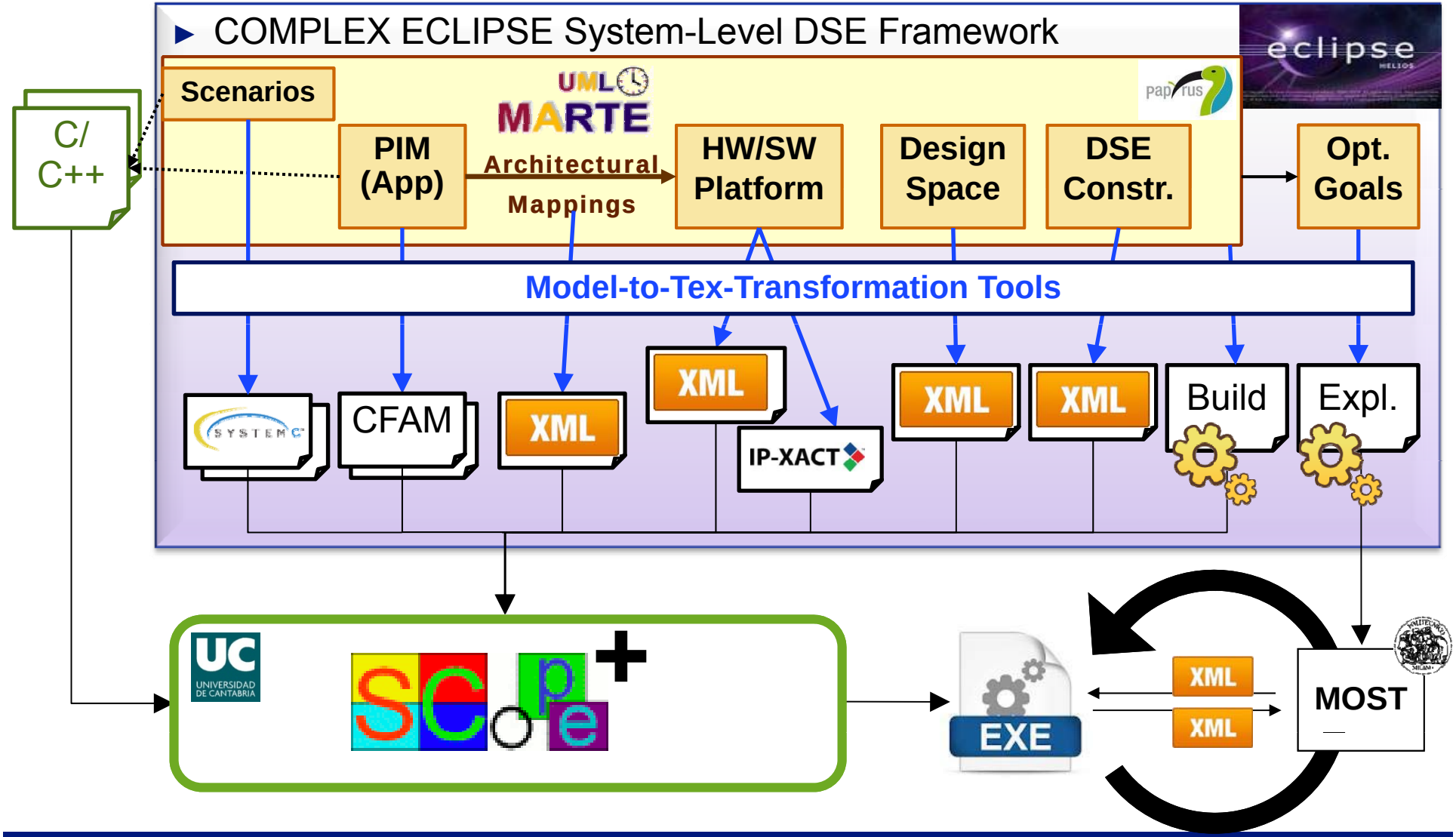
30 High-Level Custom HW Estimation Methodology

- Accuracy vs
  - HL-Synthesis and Simulation
  - Static
  - (RTL model as golden model)
- Speed-Up vs
  - HL-Synthesis and Simulation
  - Static

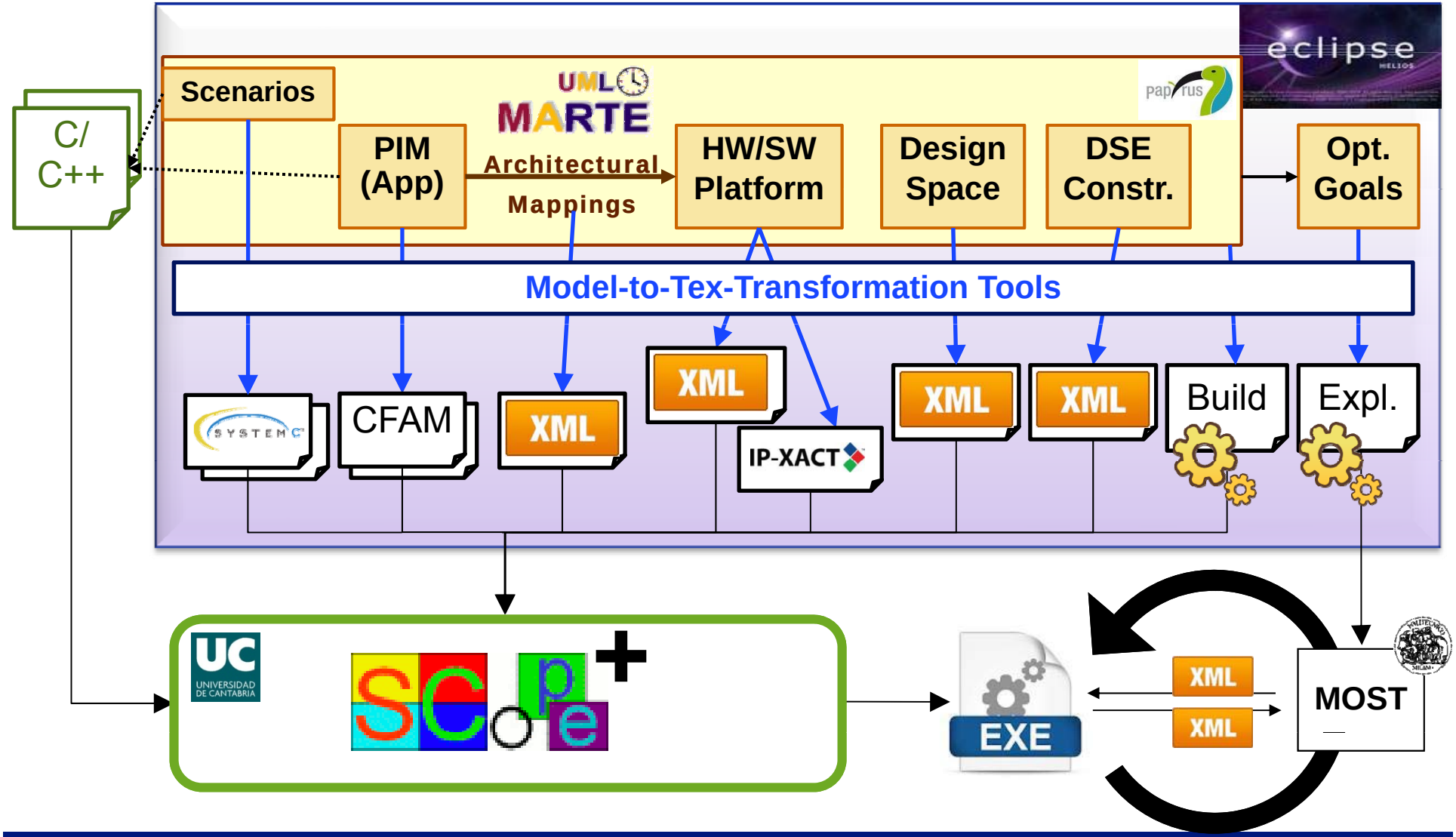




The COMPLEX Eclipse System-Level DSE Framework



The COMPLEX Eclipse System-Level DSE Framework

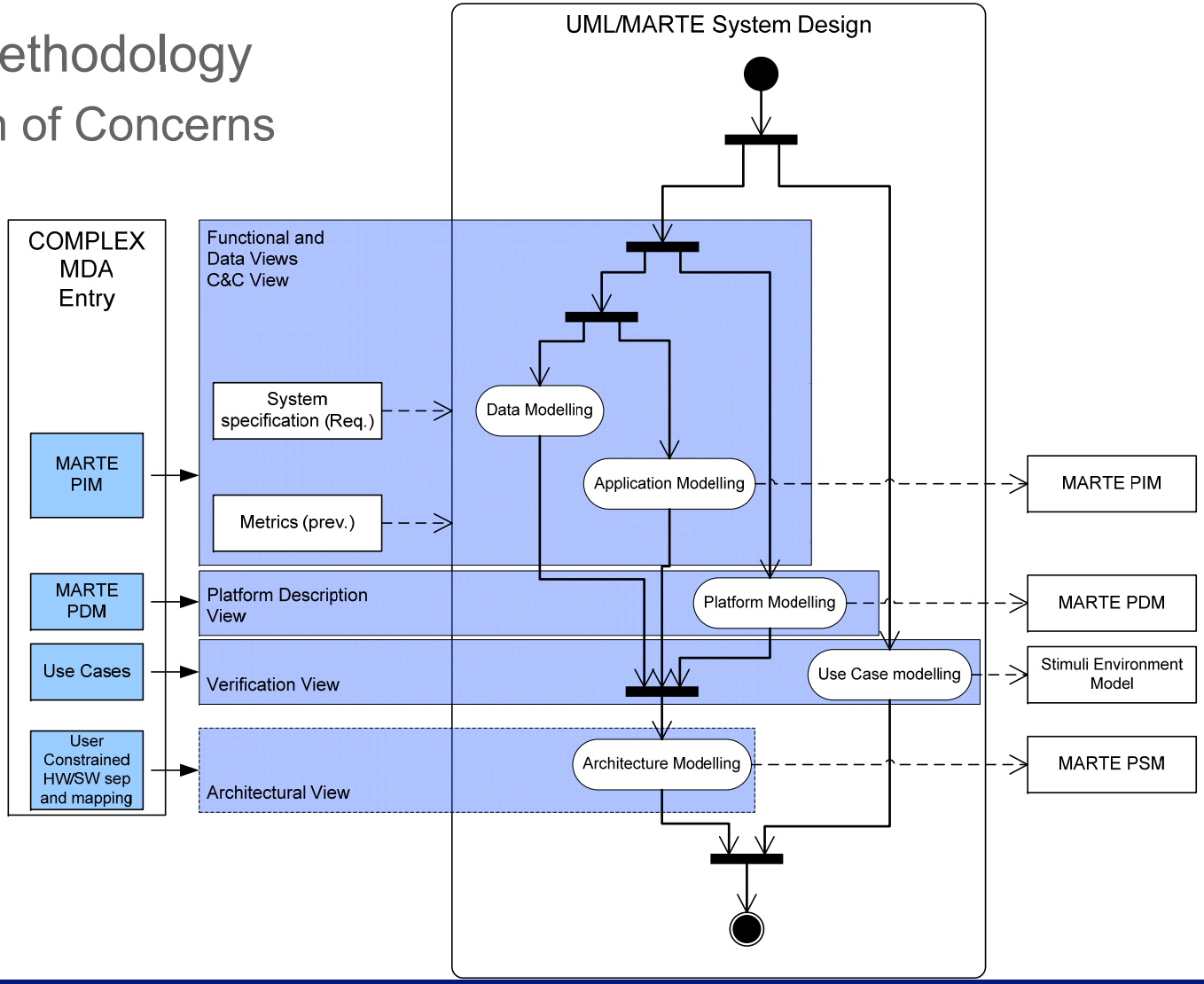


The COMPLEX Eclipse System-Level DSE Framework

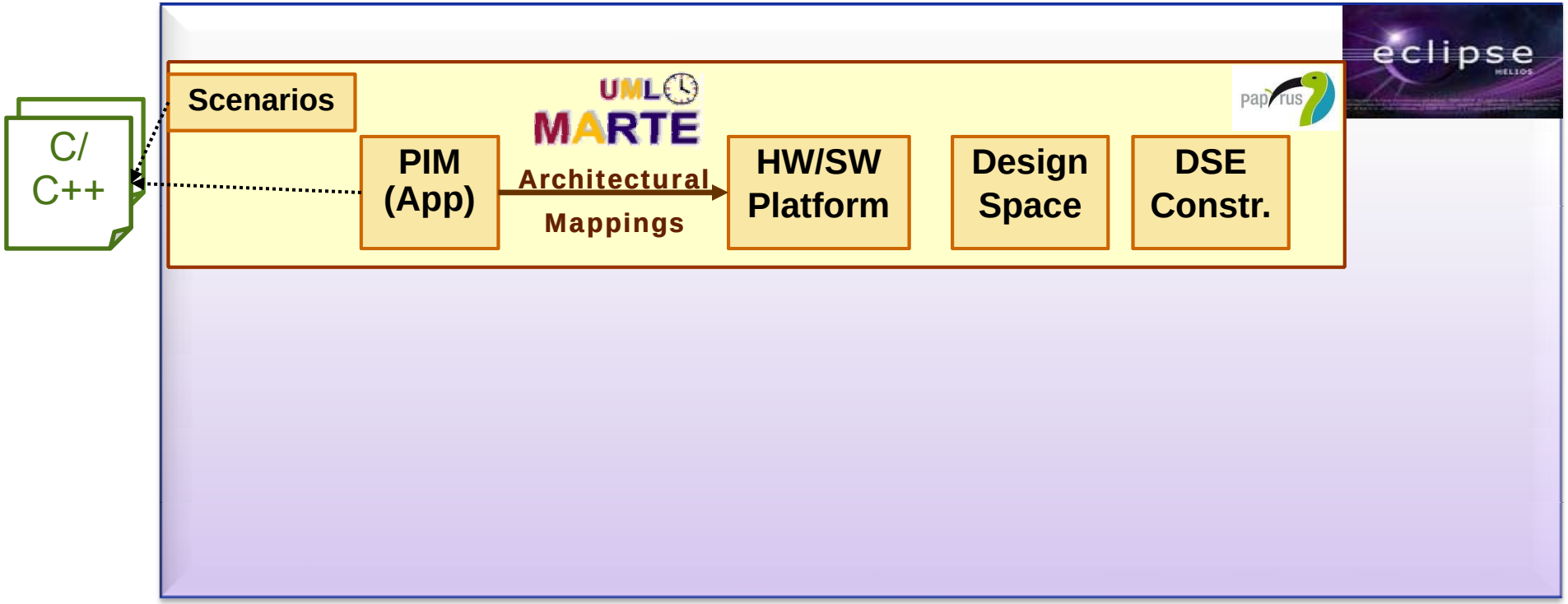
- Main features
  - MDD concepts
    - Separation of Concerns
  - CBE: Component-Based Engineering approach
  - SW centric
  - DSE oriented
  - UML-based
    - MARTE profile
      - Capture most of the RTE required semantics
    - COMPLEX profile
      - Defines DSE specific aspects not covered by MARTE

The COMPLEX Eclipse System-Level DSE Framework

- Modeling Methodology
  - Separation of Concerns



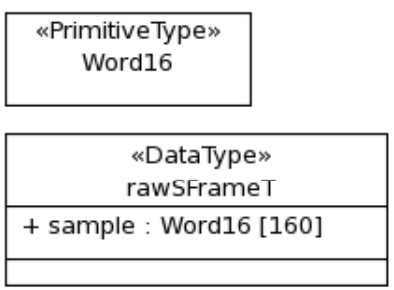
The COMPLEX Eclipse System-Level DSE Framework



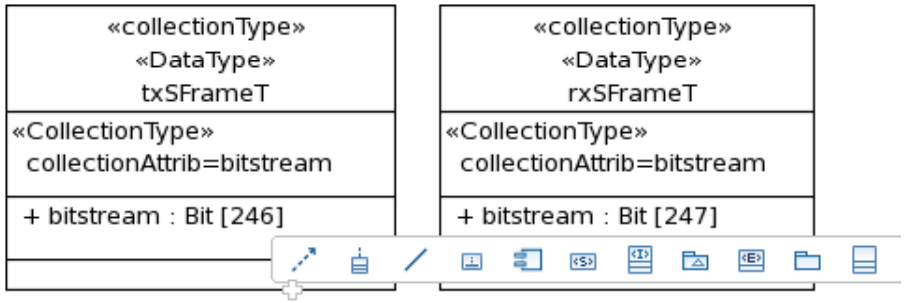
PIM Modeling: Data View

Data Types for Communication Interfaces

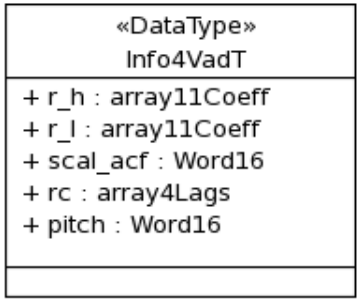
Primitive Types



Bit Arrays



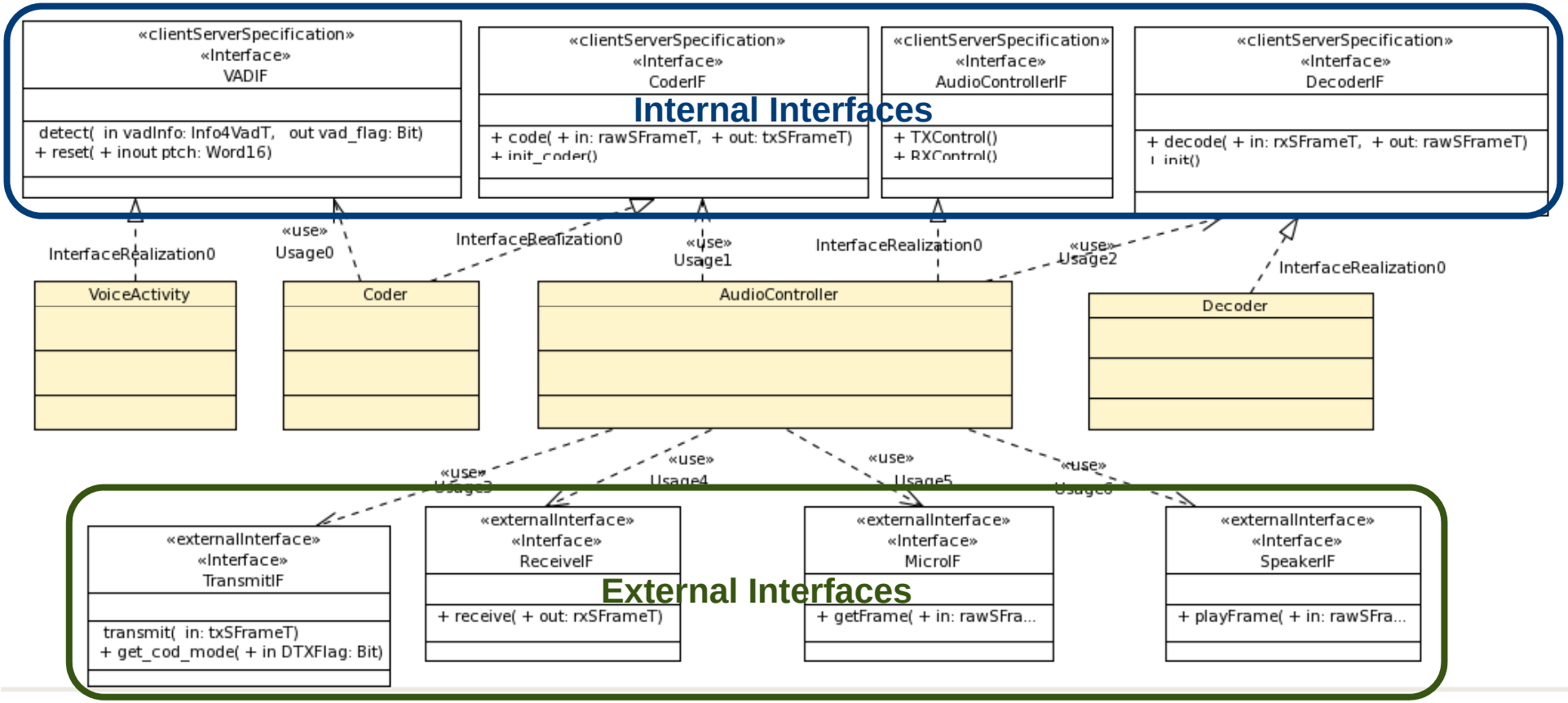
Data Structures



Arrays

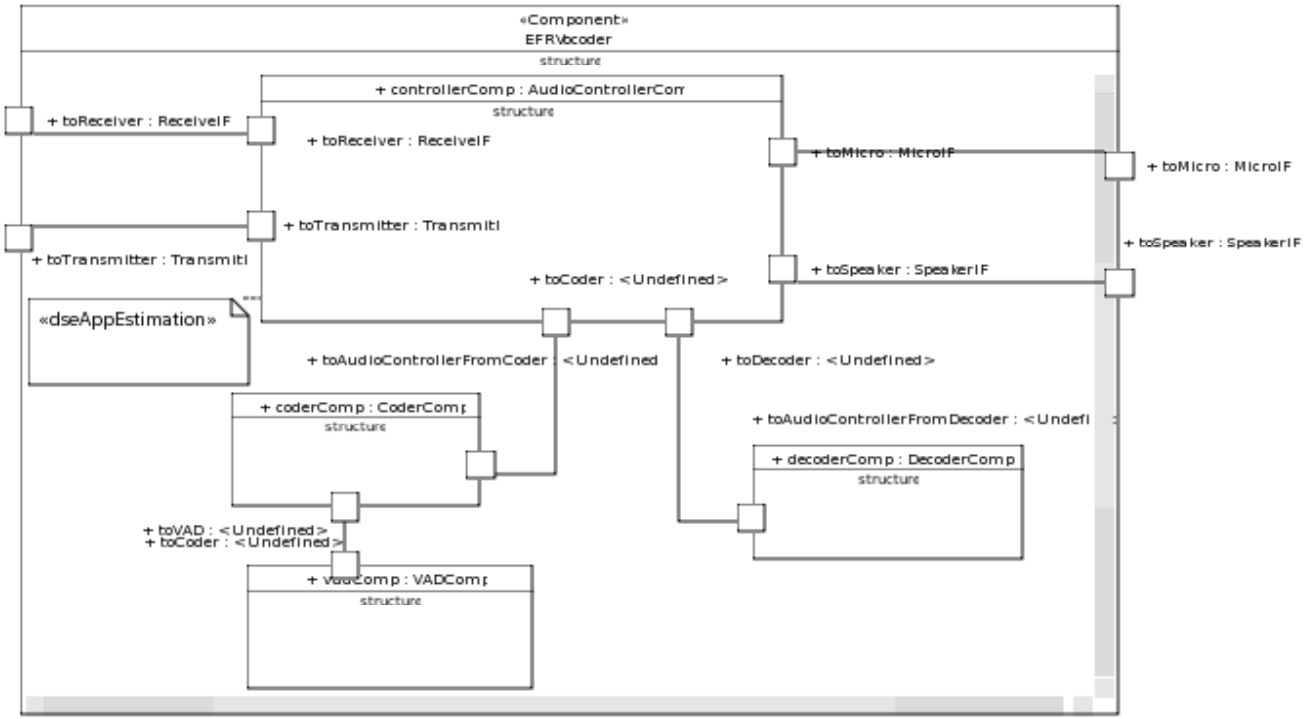
PIM Modeling: Functional View

- Data Types for Component Interfaces and Functional Classes
  - Classes implement Interfaces and require services of other interfaces

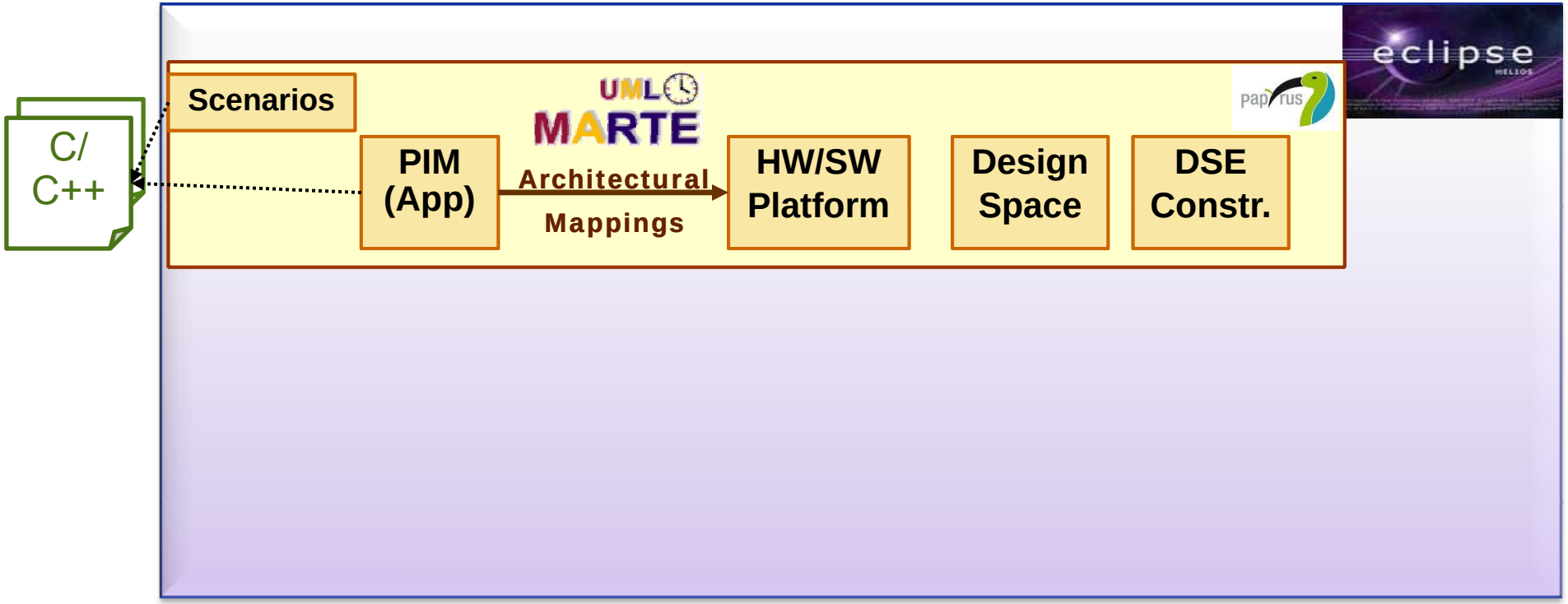


PIM Modeling: Communication & Concurrency View

- Application Component Architecture
  - As a Composite Diagram
- Application components
  - Provided and required operations



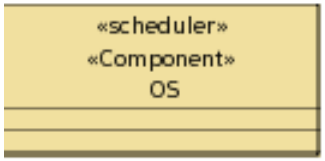
The COMPLEX Eclipse System-Level DSE Framework



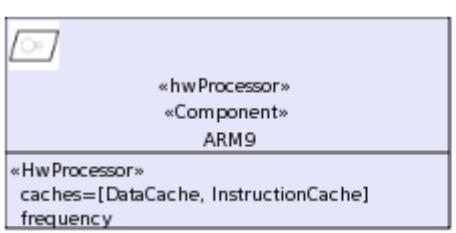
Platform View: The Platform Description Model

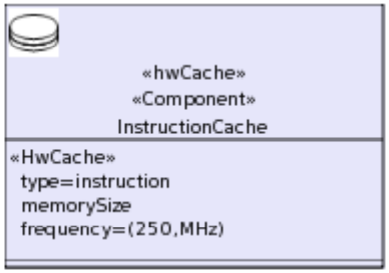
- HW/SW Components of the Platform
  - Software Components
    - OS, Drivers, …
  - Hardware Components
    - Processors, Memories, Buses, Custom HW, I/O
  - Components using MARTE stereotypes

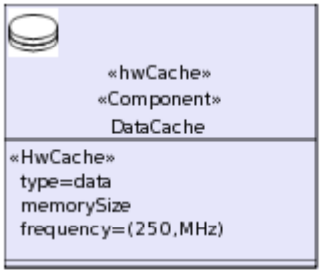
▶ SW

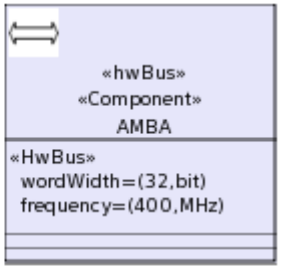


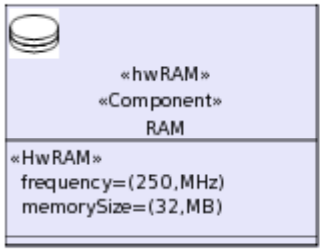
▶ HW

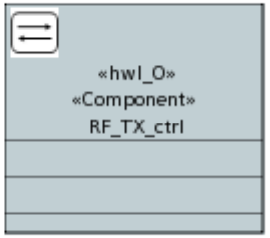


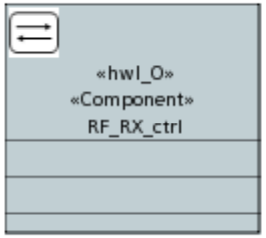


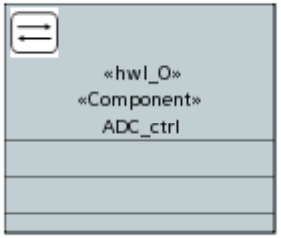


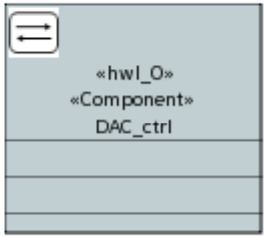




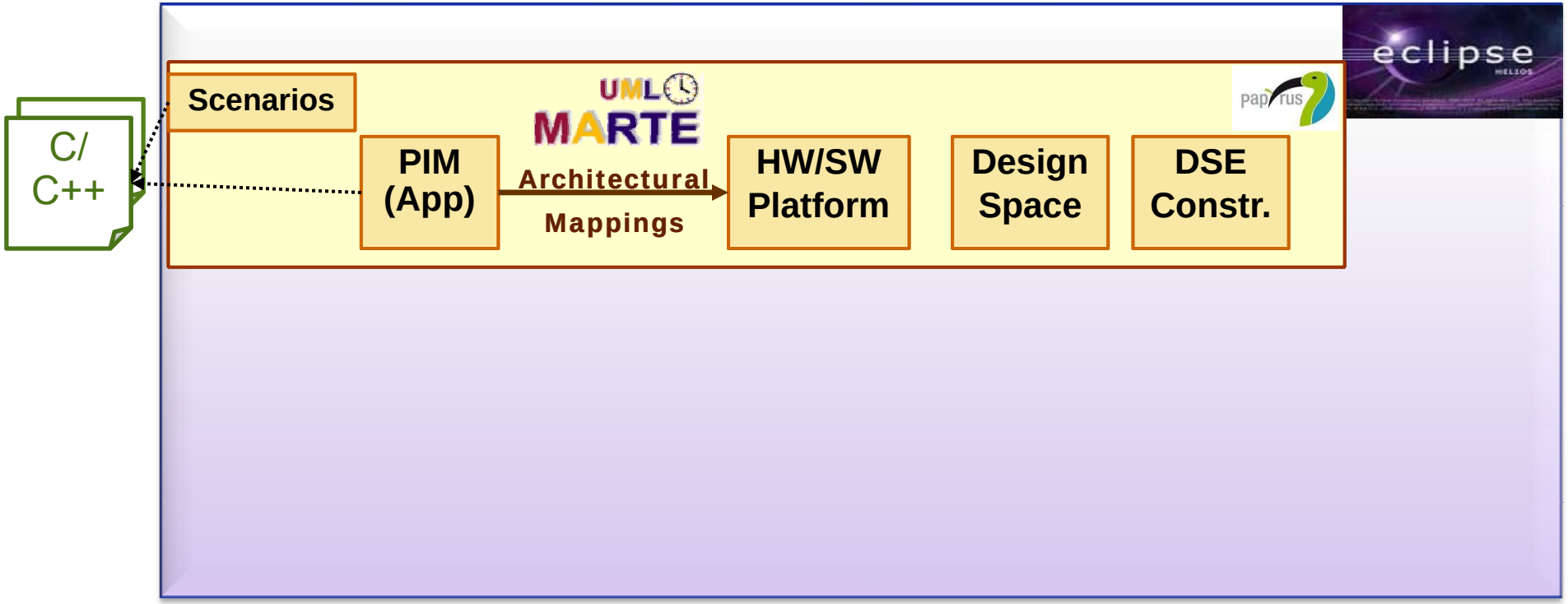






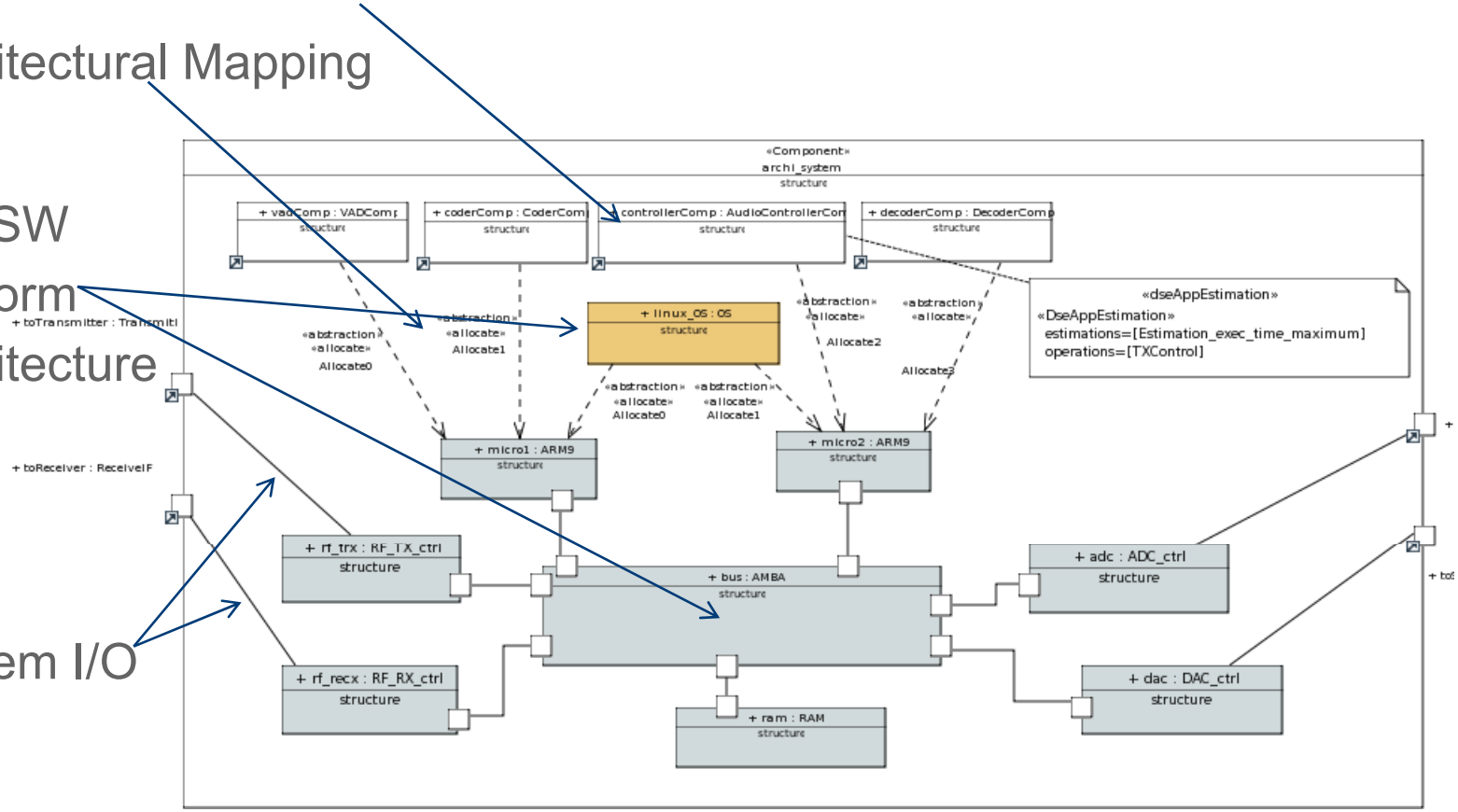


The COMPLEX Eclipse System-Level DSE Framework

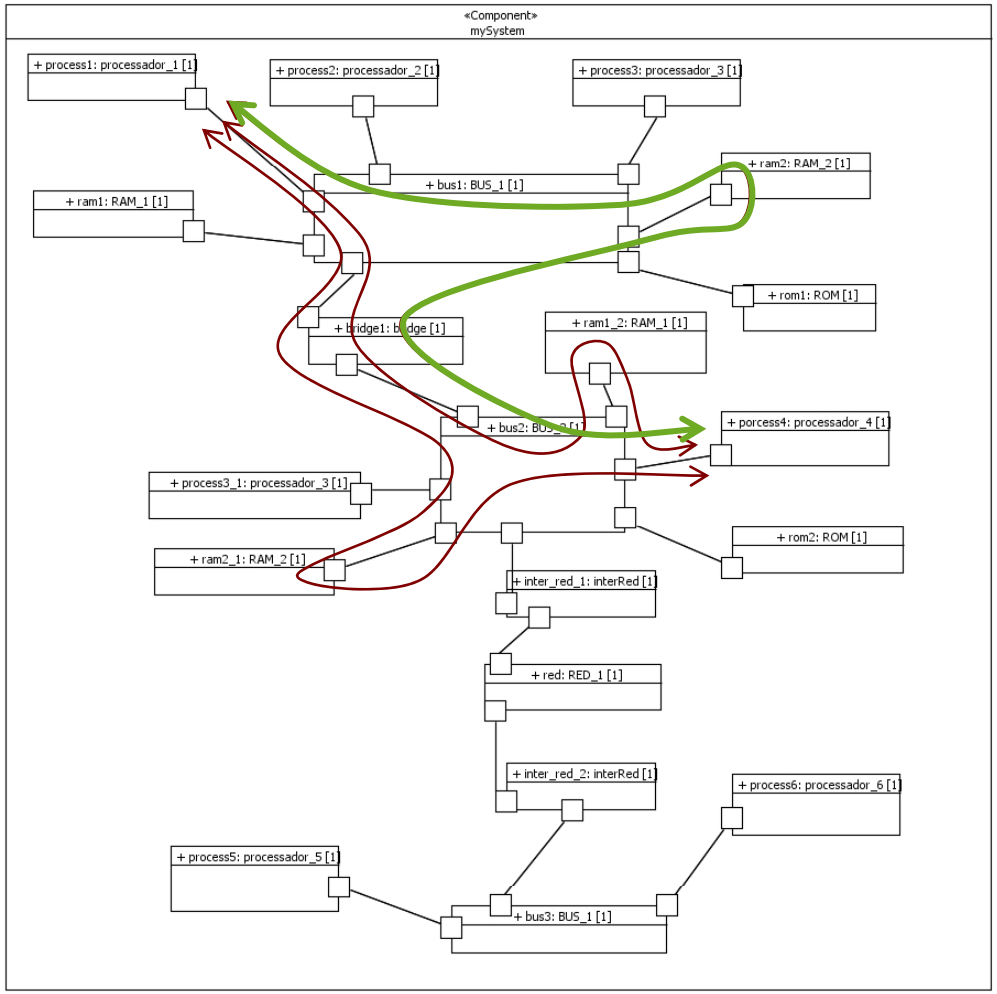


Architectural View: The Platform-Specific Model

- Composite Diagram
  - Application Component Instances
  - Architectural Mapping
  - HW/SW Platform Architecture
  - System I/O

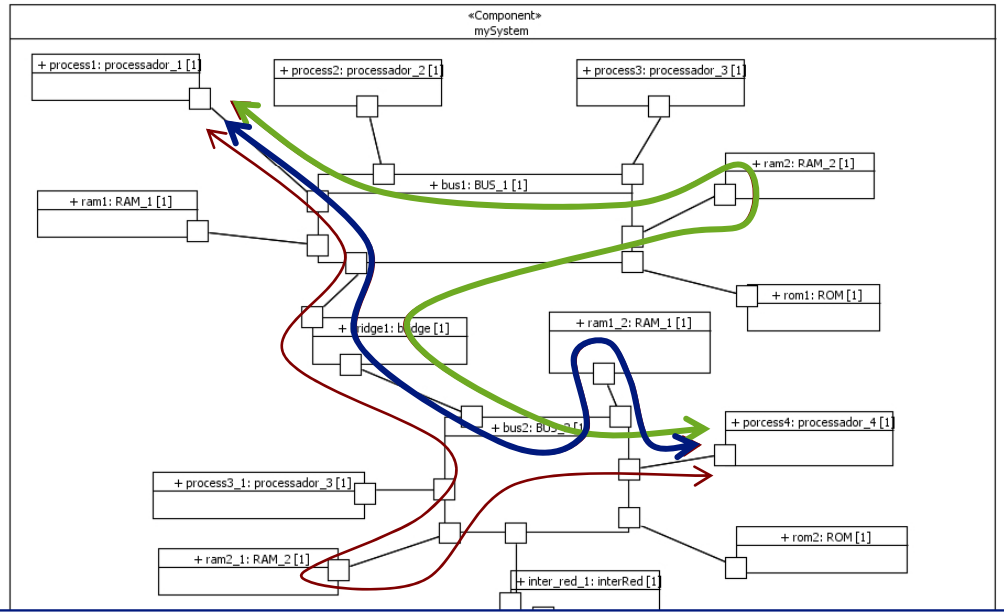


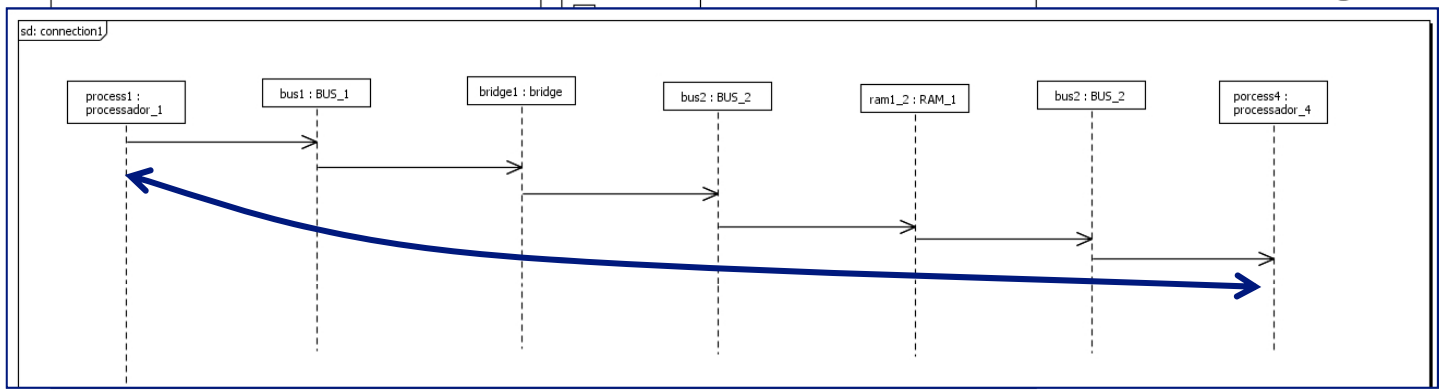
Architectural View: Data Path Alternatives



- Several routing alternatives possible
  - Which data path to communicate processors 1 and 4?
  - Impact on Performance
- Solution 1: The estimation tool (SCoPE+) selects one path:
  - Optimum: #hops, hop cost

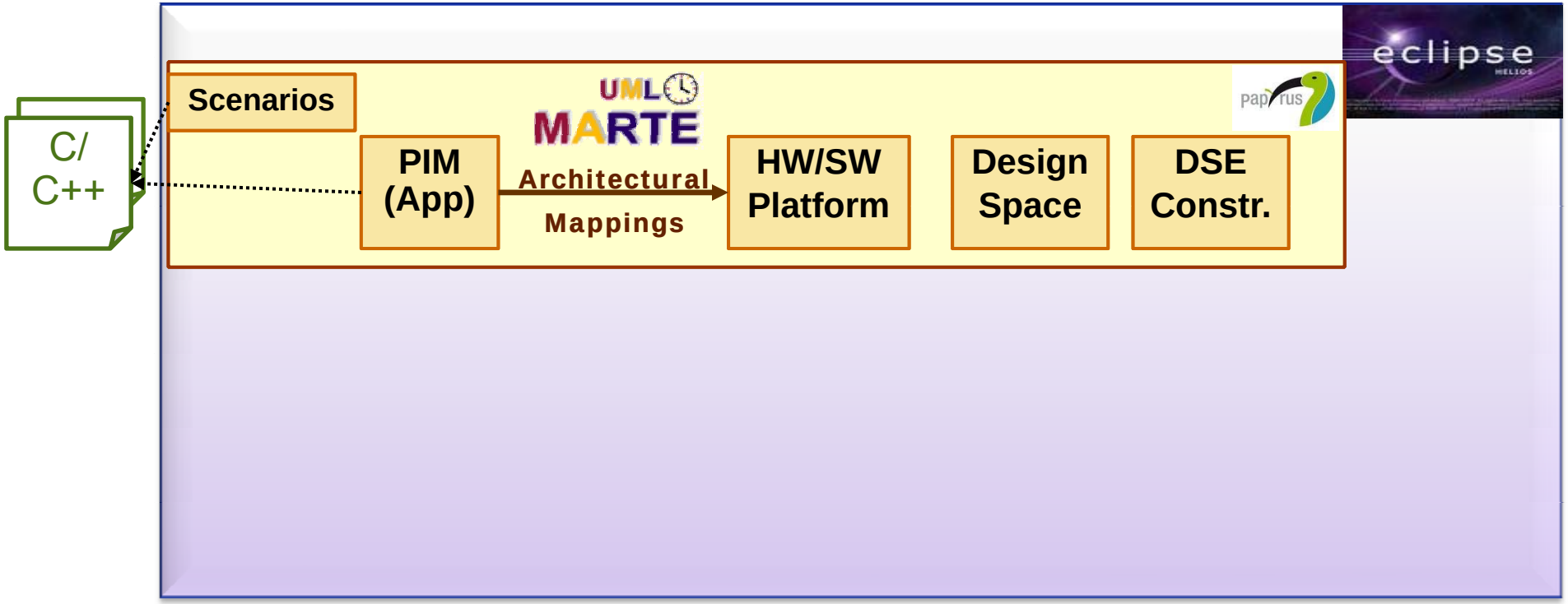
Architectural View: Data Path Alternatives





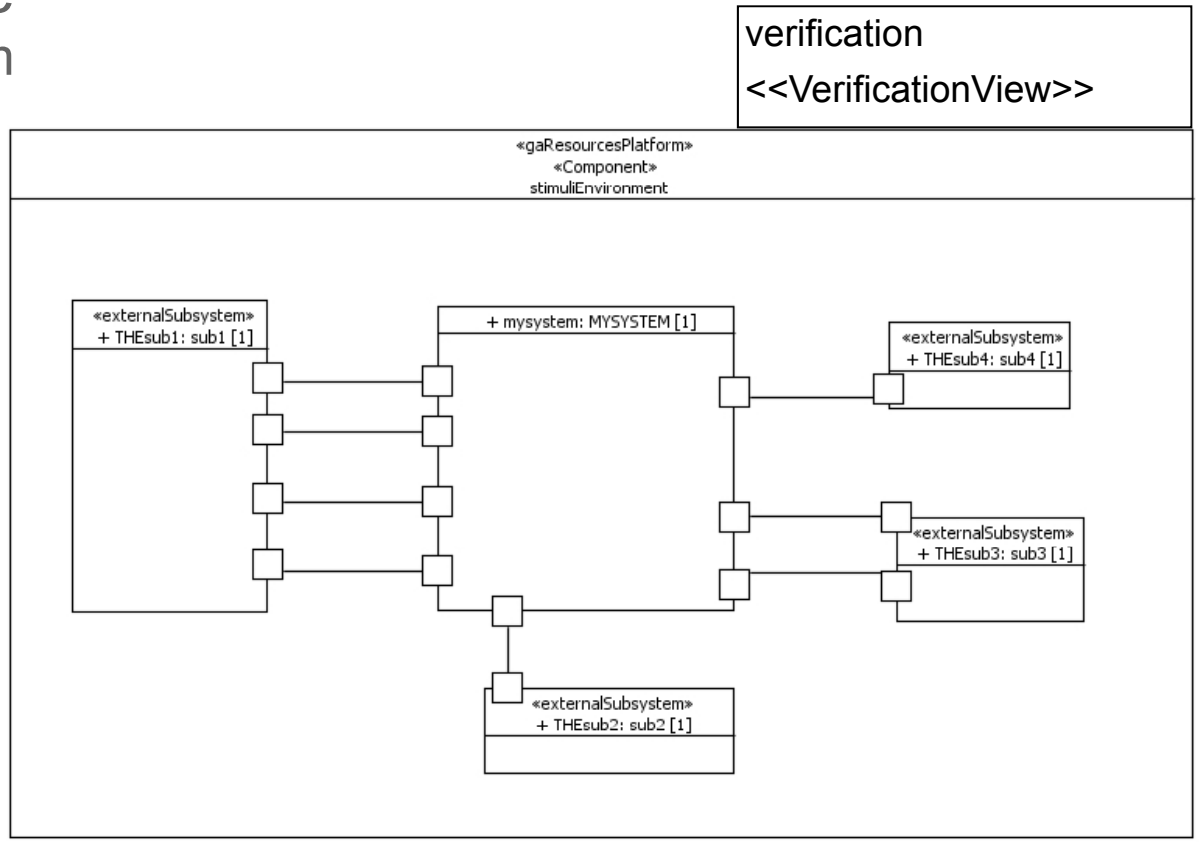
- Several routing alternatives possible
  - Which data path to communicate processors 1 and 4?
  - Impact on Performance
- Solution 2: The UML/MARTE model contains information indicating the preferred routing
- How?
  - Sequence Diagram
  - Static Routing

The COMPLEX Eclipse System-Level DSE Framework

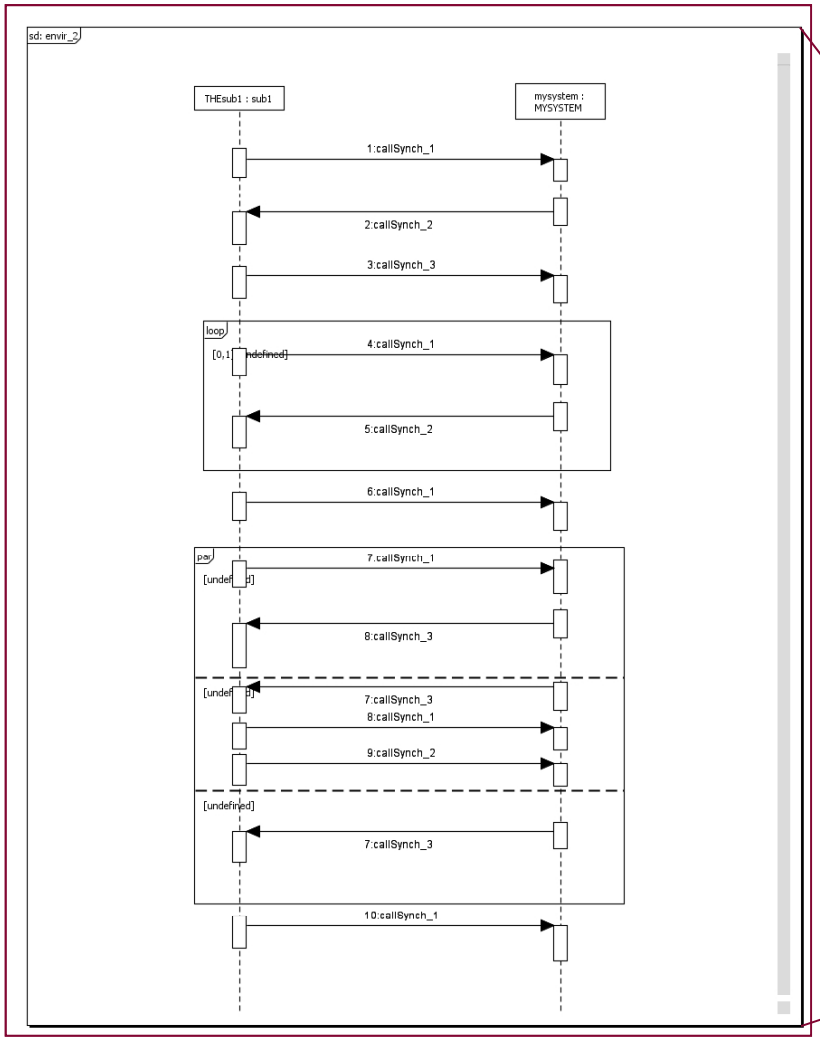


Verification View: Environment Components

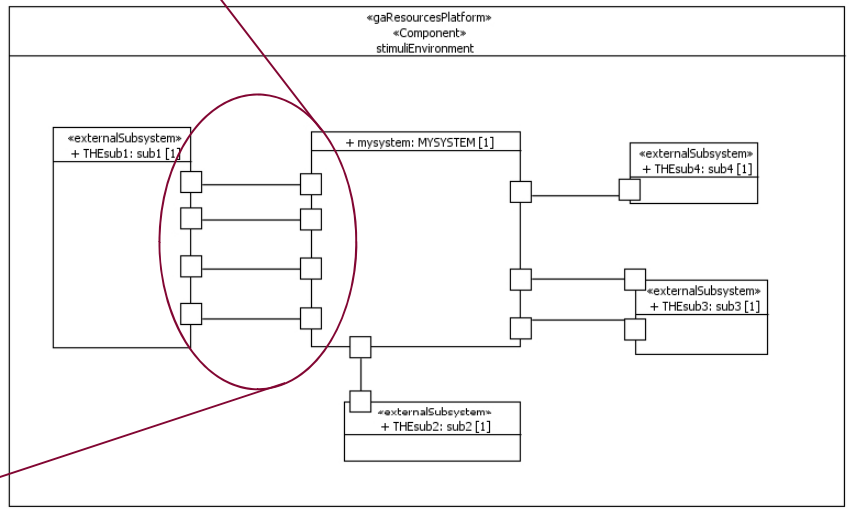
- Automatic generation of the SystemC Test-Bench
- COMPLEX <<VerificationView>> stereotype
- Homogeneous style
  - Composite diagram



Verification View: Interaction between system and environment

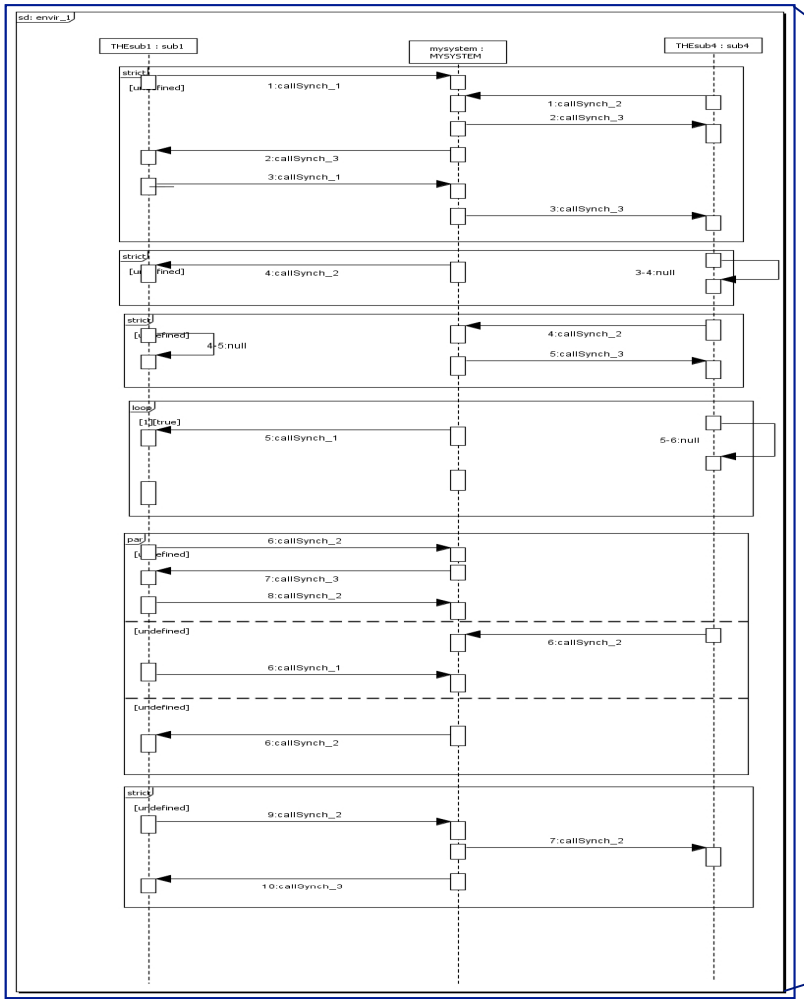


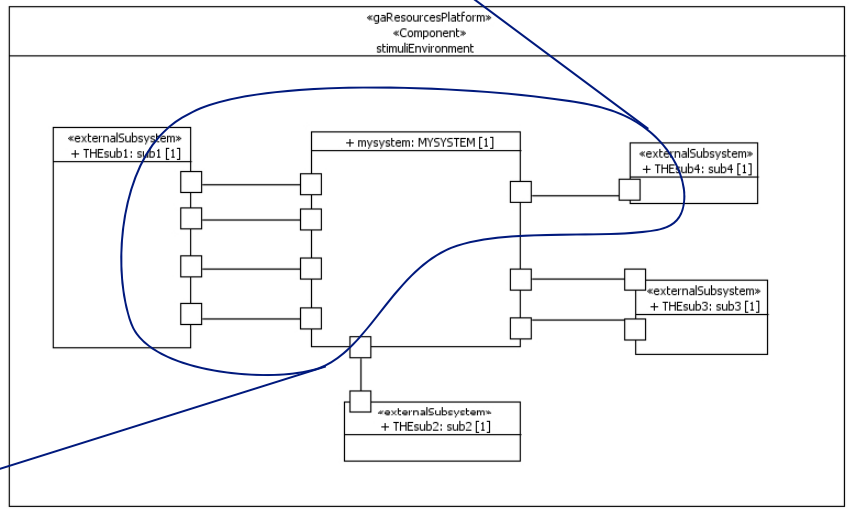
- Synchronization and Communication semantics
  - Sequence Diagram
- Extraction of code
  - behavior of environment components



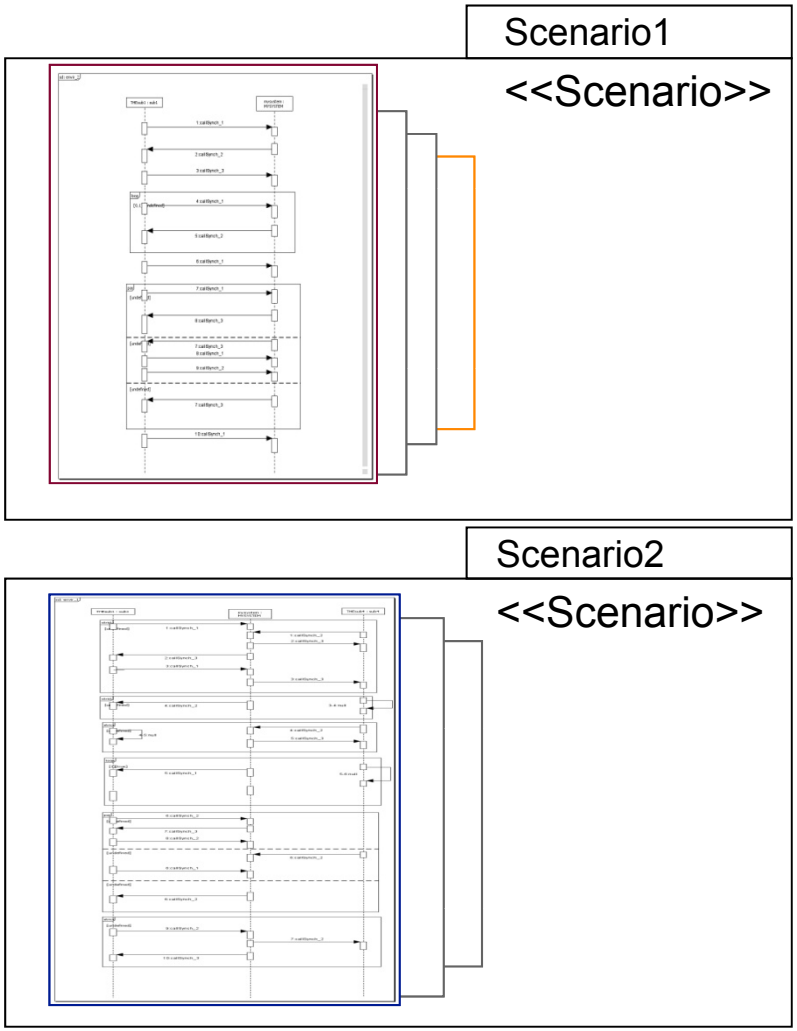
Verification View: Interaction between system and environment

- Scenario modeling
  - A single sequence diagram can cover the interaction of more than one environment component

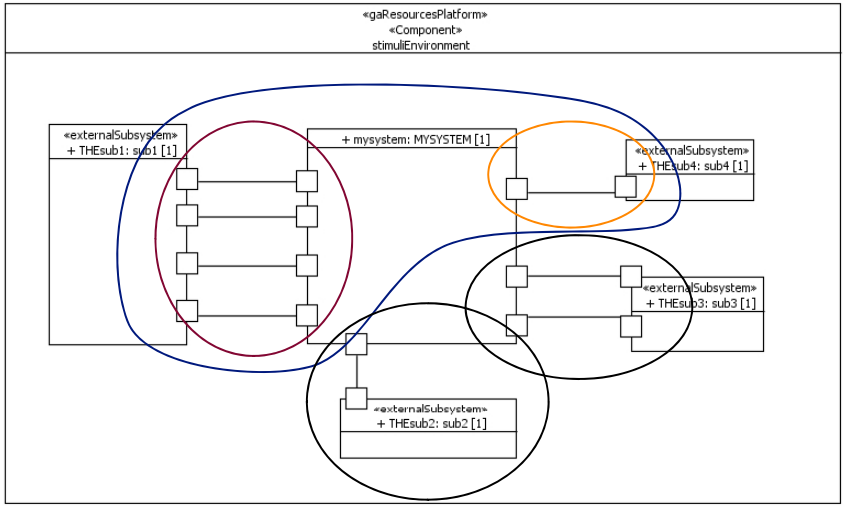




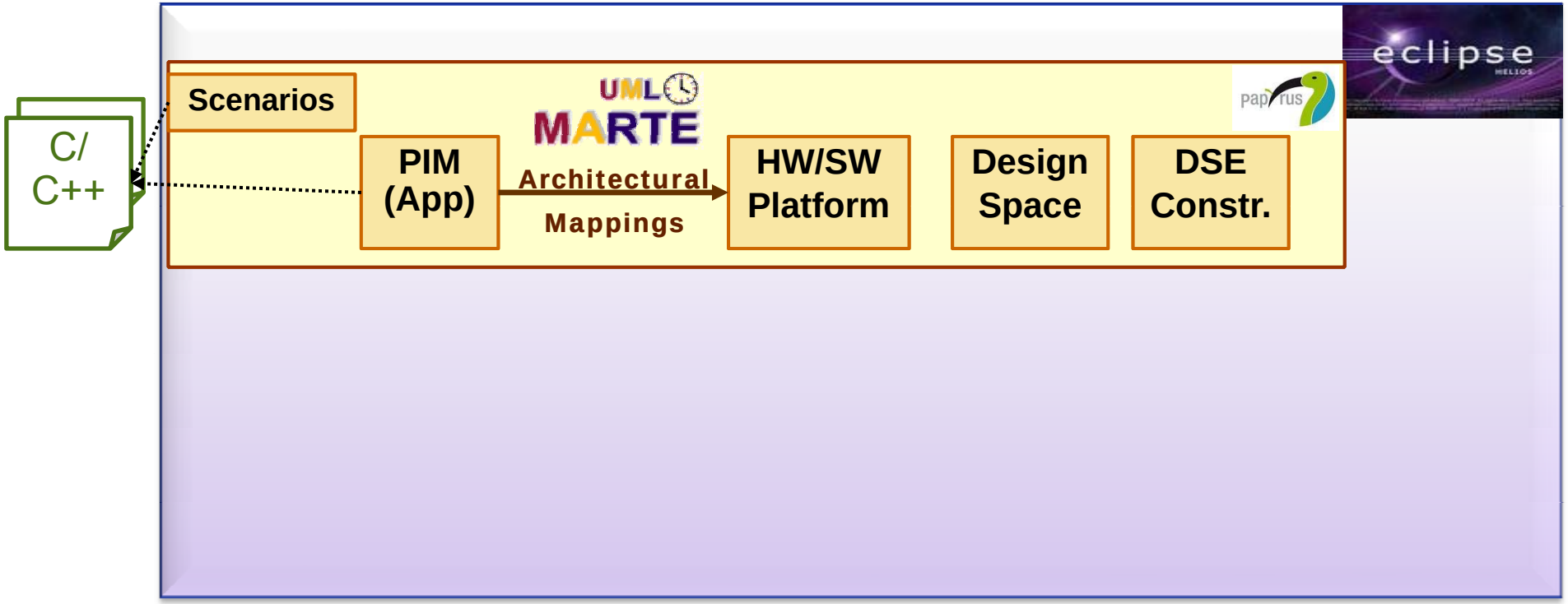
Verification View: Scenarios



- Scenario: A tuple of interactions covering the interaction of the system with the whole environment
  - A package within the Verification View
  - with the <<Scenario>> stereotype
- Several scenarios are possible



The COMPLEX Eclipse System-Level DSE Framework



Modeling the Design Space: General Features

- Capturing the Exploration Space in a single model
- Defining a set of Scenarios
  - allowing the selection of the scenarios to be explored
- Defining the Output metrics
  - used as input to the selection of the next experiment
  - determining the Pareto points

- The Design Space is composed of
  - A set of Architectural Mappings
  - A set of configurable attributes for Platform Components
  - A set of Platforms
  - A set of DSE Constrains and rules

Conclusions

- SW simulation and performance analysis
  - Essential Design Technology
  - HW/SW Embedded Systems
  - At different design steps
    - Different modeling and simulation technologies
    - Various performance*accuracy products

- UML/MARTE Modeling Methodology
  - MDD concepts
    - Separation of Concerns
  - CBE: Component-Based Engineering approach
  - SW centric
  - DSE oriented
  - Automatic Model Generation

Additional Information

- COMPLEX Website
  - http://complex.offis.de
- COMPLEX plug-in
  - https://complex.offis.de/eclipseupd
- Microelectronics Engineering Group
  - http://www.teisa.unican.es/gim/en/tema?id=4
- SCoPE
  - http://www.teisa.unican.es/scope

Thank You!

- Eugenio Villar
  - villar@teisa.unican.es