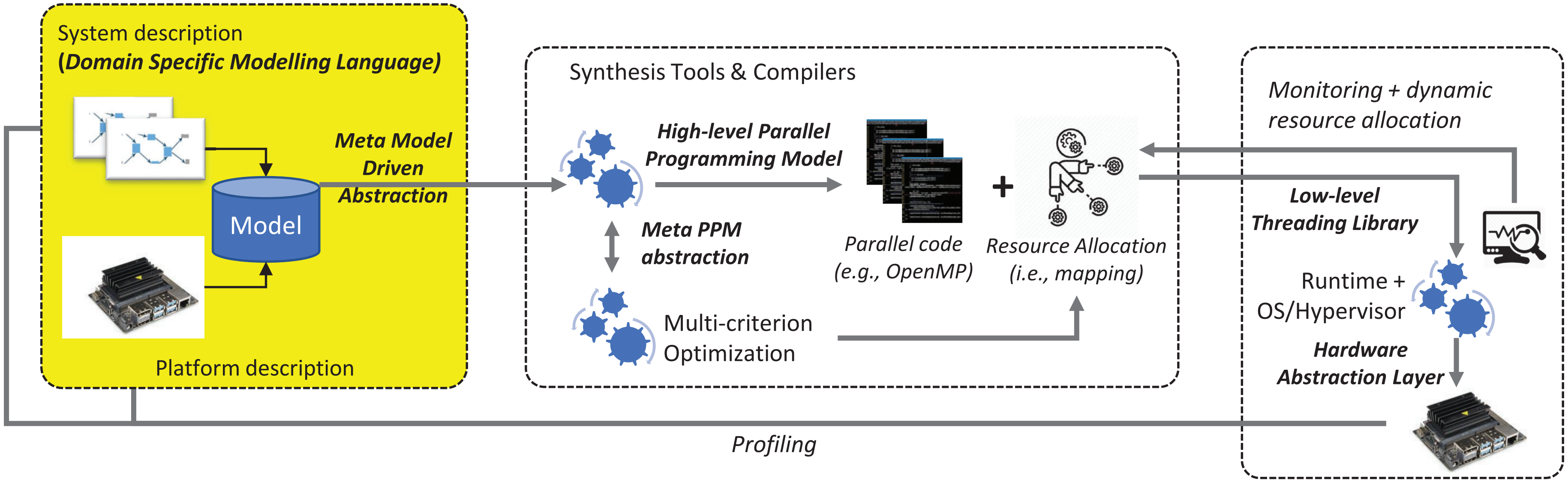
A Model-driven development framework for highly Parallel and EneRgy-Efficient computation supporting multi-criteria optimisation

Enhancing productivity through model driven engineering

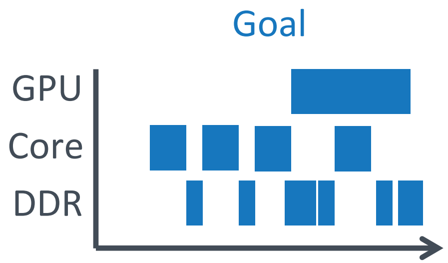
AMPERE Final Event Webinar

Michael Pressler — Robert Bosch GmbH
Olivier Constant — Thales
Thomas Vergnaud — Thales
27 June 2023

System Model Description and Use-cases



Benefits of model-driven engineering (MDE)



Improve **insight** in dynamic system behavior

- Systems in development
- Systems in field

Assess design choices & requirements

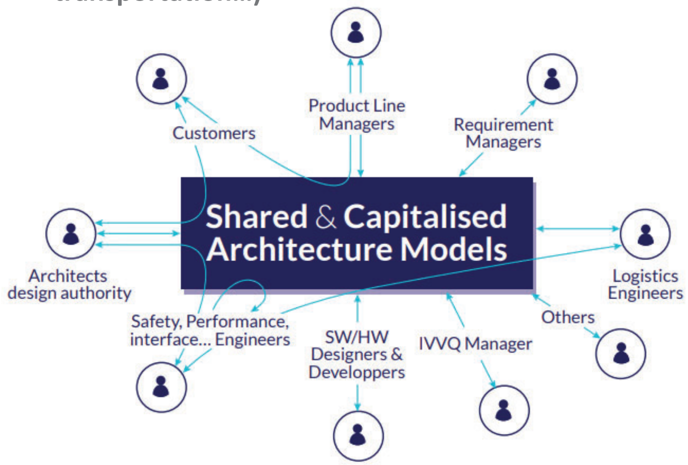
- Systems in acquisition
- Systems in planning or development

Identify opportunities

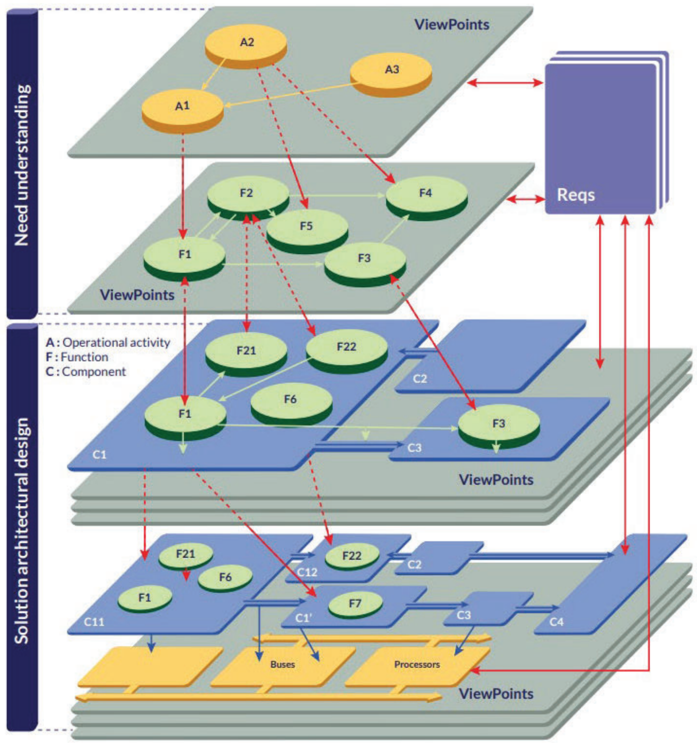
- Derive OS configurations
- Evaluate mapping of functions, data, and code to hardware platform
- Exploit parallelism of the hardware platform
- Prioritization of critical event-chains

System design with Capella: the ARCADIA method

- Initially developed & deployed at Thales
- Released in open source: eclipse.org/capella
- Used in many industrial sectors (aerospace, energy, transportation…)



Validating/Justifying
solution against Operational Need
Easing Impact Analysis



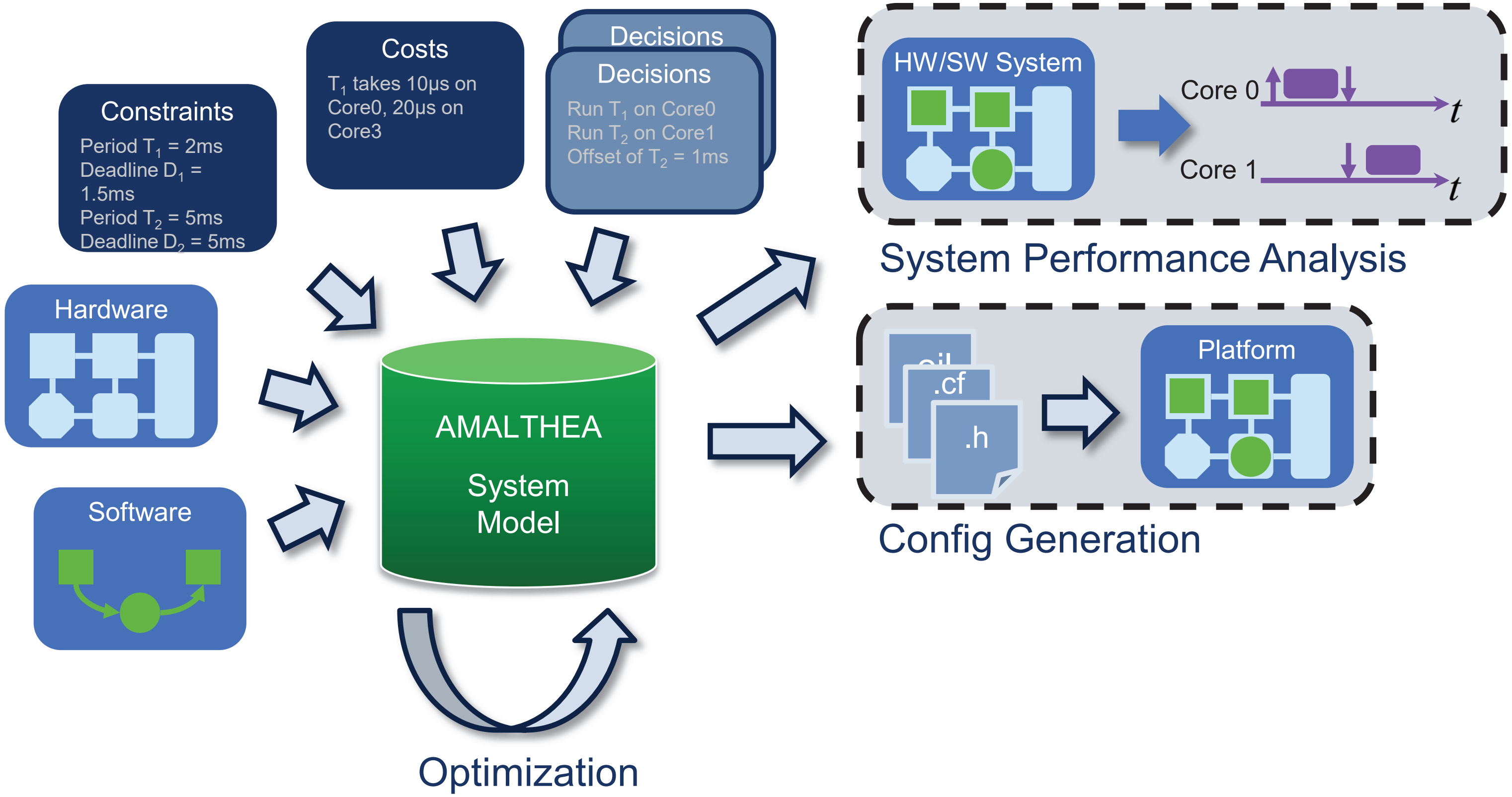
Operational Analysis
What the users of the system need to accomplish

Functional & Non Functional Need
What the system has to accomplish for the users

Logical Architecture
How the system will work to fulfill expectations

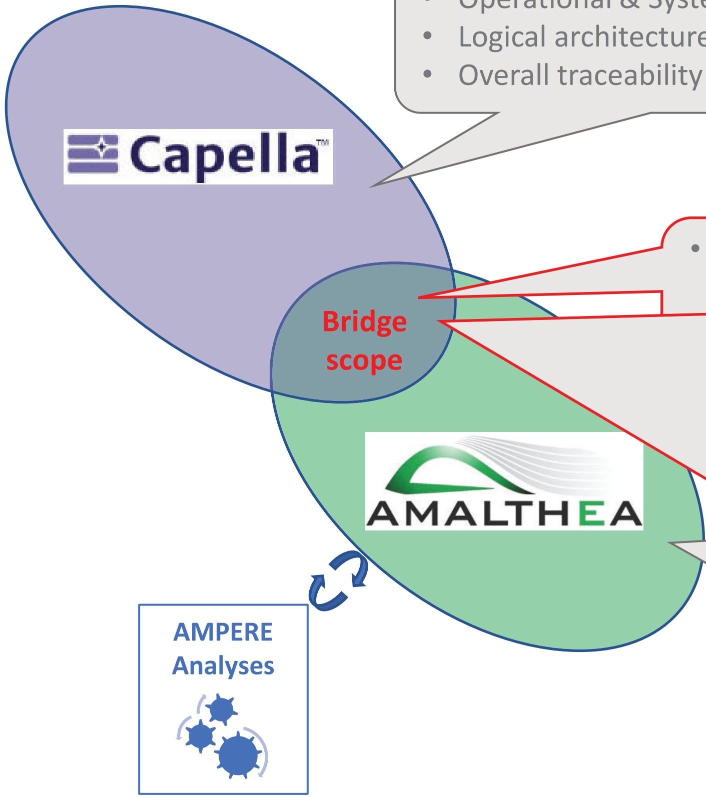
Physical Architecture
How the system will be developed and built

AMALTHEA – a common system model

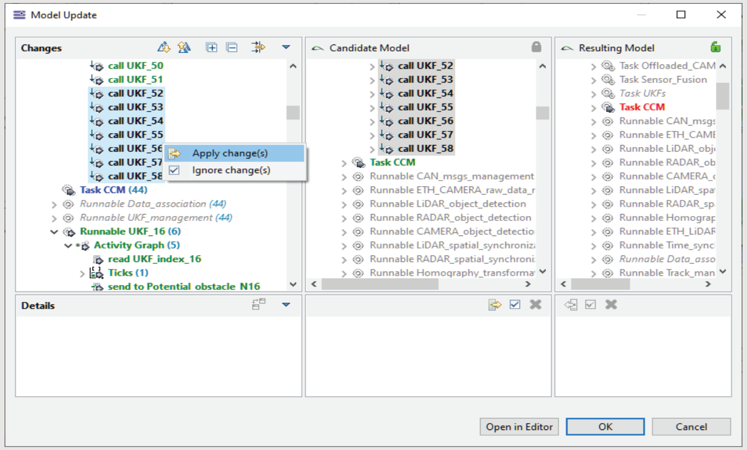


Capella vs. Amalthea concepts and Bridge

- Operational & System
- Logical architecture
- Overall traceability fo



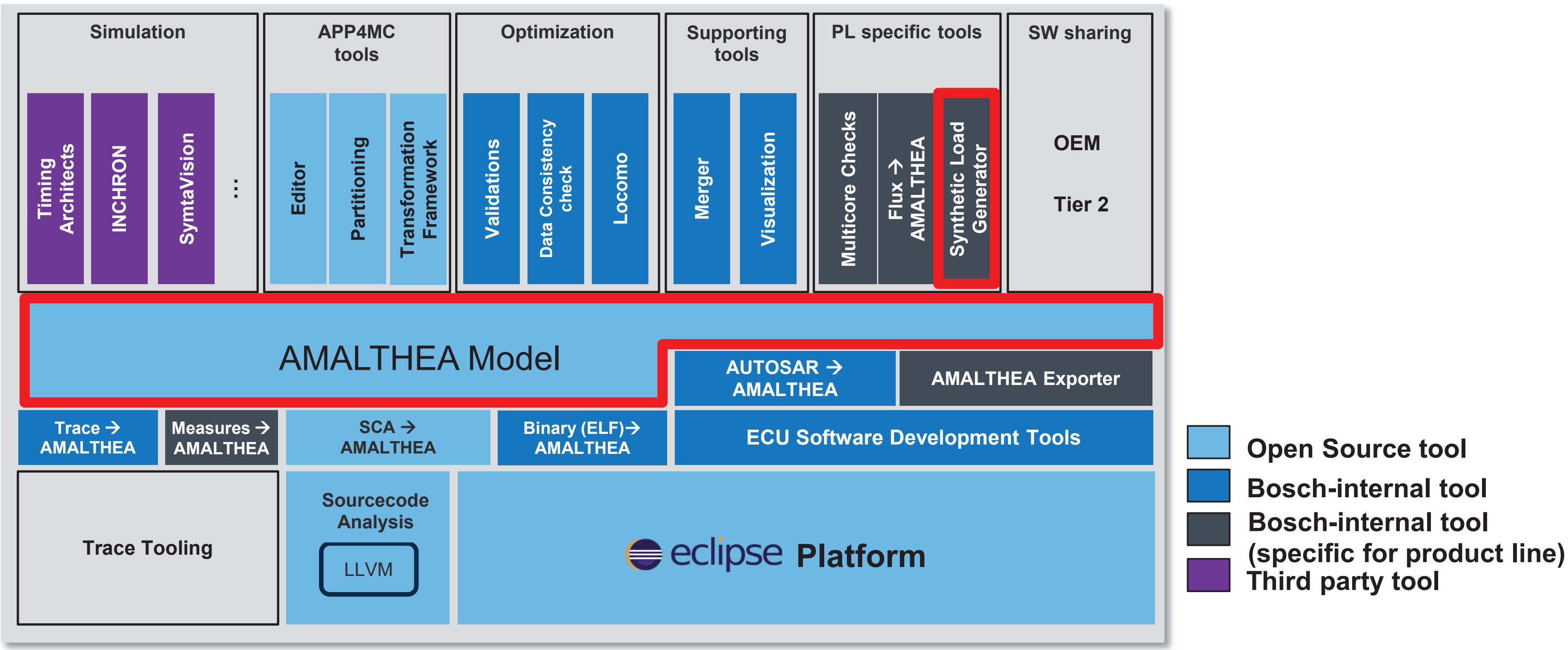
- The incremental Bridge allows synchronising the Amalthea model with the Capella model non-destructively.



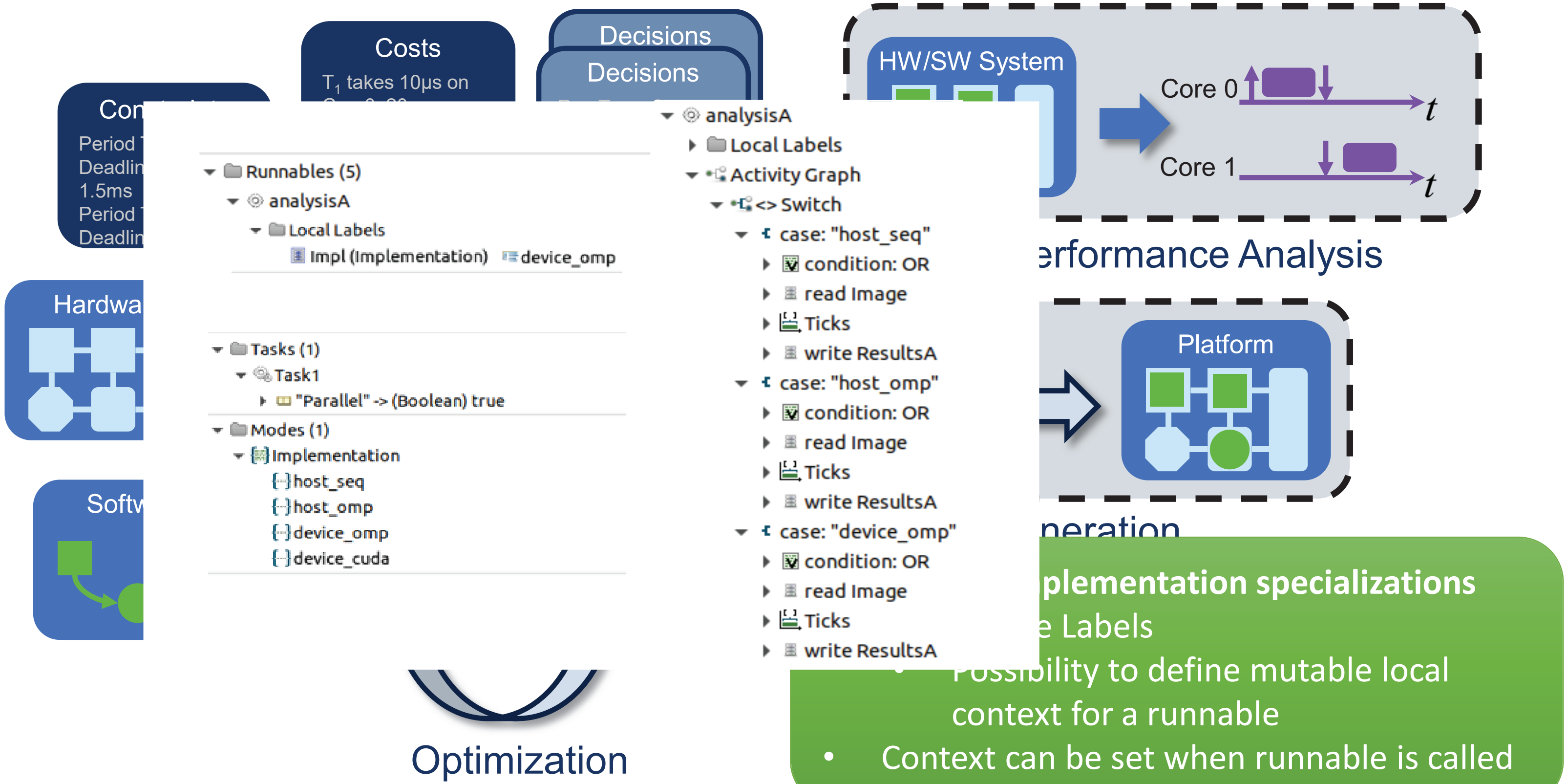
- It thus supports consistent concurrent modelling at different levels of abstraction, e.g., in Agile sprints.

  - Resource consumptions
  - Parallelism inc. SW activation, control flow, scheduling
  - Constraints on data / timing / parallelism

APP4MC Ecosystem



AMALTHEA – a common system model

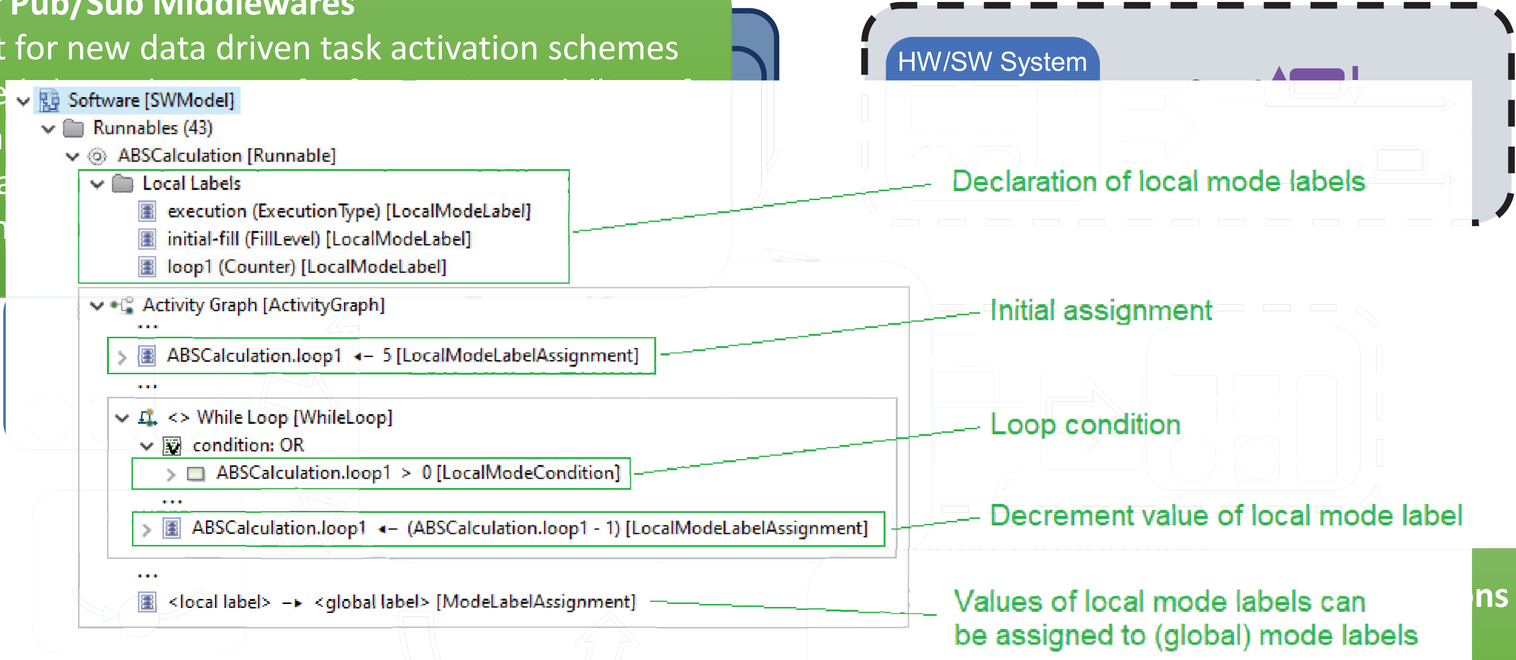


- Context can be set when runnable is called

AMALTHEA – a common system model

Support for Pub/Sub Middlewares

- Support for new data driven task activation schemes
- Extende
  buffer a
  - Rea
  - Wh



ns

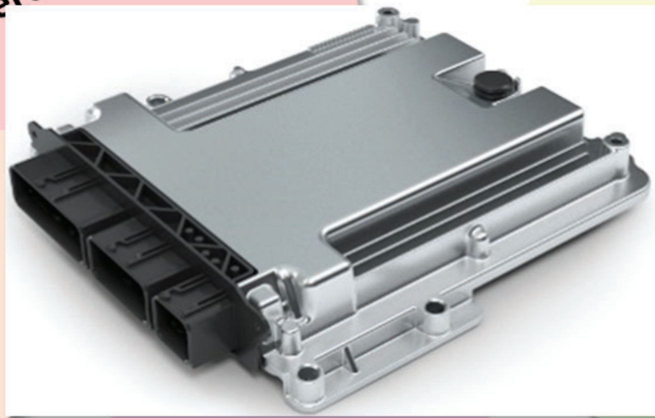
- Possibility to define mutable local context for a runnable
- Context can be set when runnable is called

Optimization

Synthetic Load Generator

How to estimate application sensitivity to interference?

How to estimate runtime/performance for customer requests?



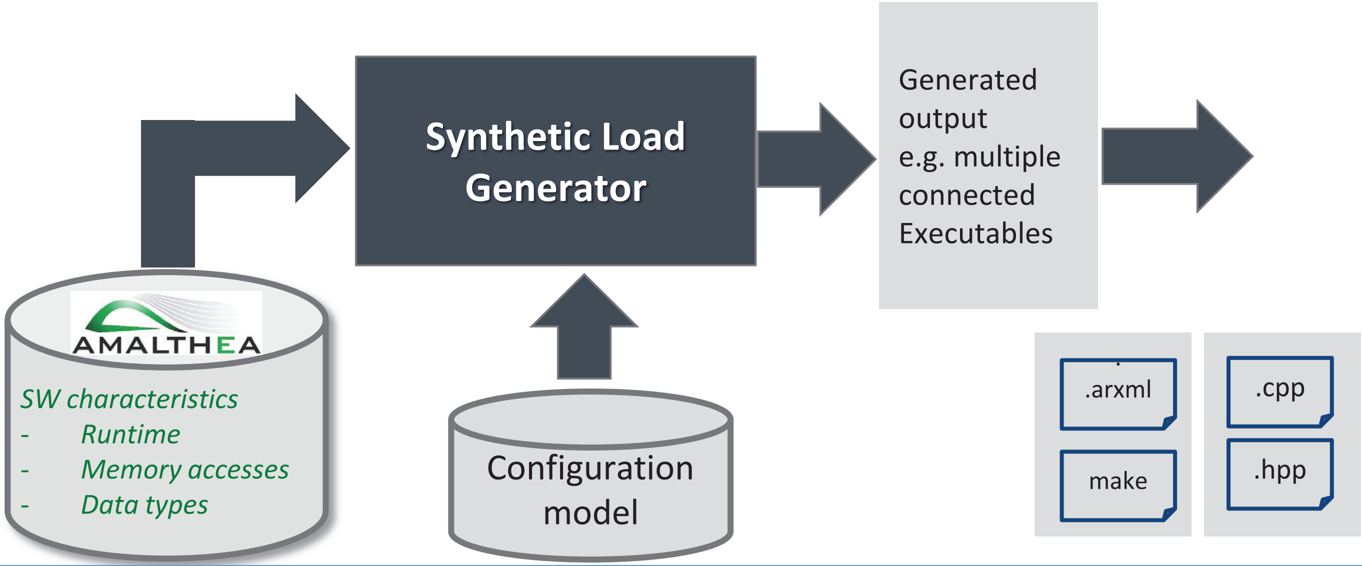
How to reason about future application budget (memory, runtime, energy)?

How to evaluate Middleware communication overhead/performance?

How to integrate customer applications?

Bosch: Synthetic Benchmark Generator (SLG)

Amalthea as single input source



TickSnippet

```
    int arraySizeWith10Multiples=arraysize-leftOverElements;
      int i = 0;
      int a = 0;
      for (i = 0; i < arraySizeWith10Multiples; i = i + 10) {         //ite
ration with 10 reads
```

LabelSnippet

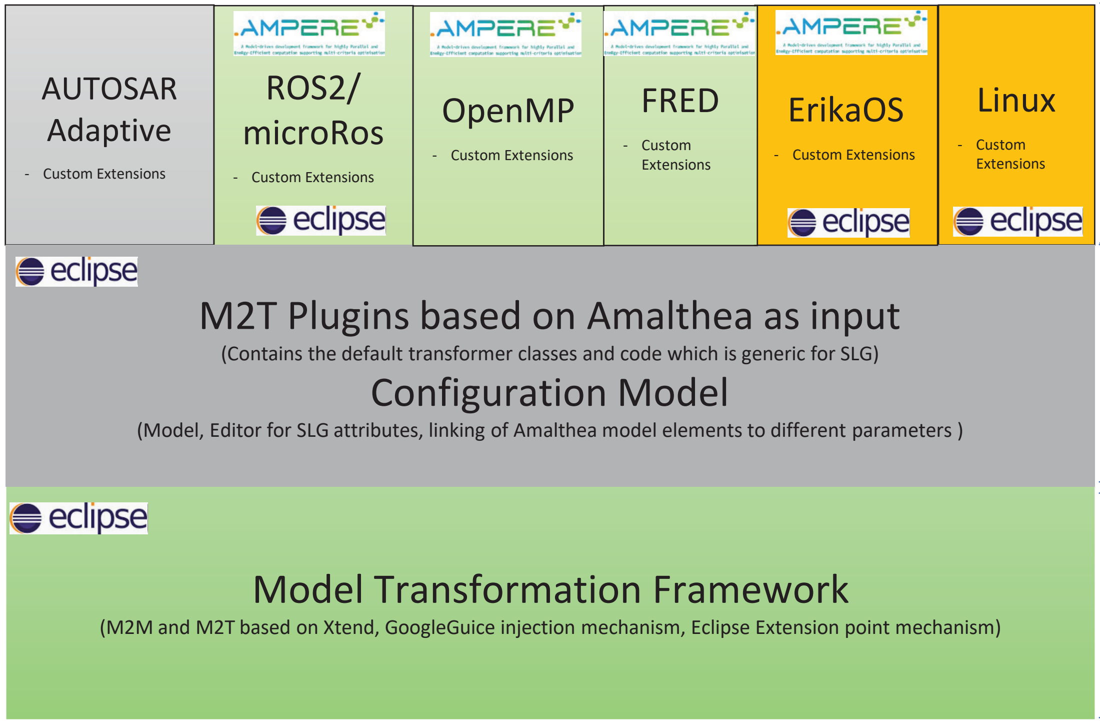
```
    for (int repeat = 0 ; repeat < labelAccessStatistics; repeat++){
      if(numberOfBytes < 4){
        numberOfBytes = 4;
      }
      int arraysize = sizeof(array)/4;

      //printf("number of bytes:%d\n",arraysize);
      int leftOverElements=arraysize%10;

      int arraySizeWith10Multiples=arraysize-leftOverElements;
      int i = 0;
      int a = 0;
      for (i = 0; i < arraySizeWith10Multiples; i = i + 10) {         //ite
ration with 10 reads
        a = DummyLabelRead[i];
        a = DummyLabelRead[i+1];
        a = DummyLabelRead[i+2];
        a = DummyLabelRead[i+3];
        a = DummyLabelRead[i+4];
        a = DummyLabelRead[i+5];
        a = DummyLabelRead[i+6];
        a = DummyLabelRead[i+7];
        a = DummyLabelRead[i+8];
        a = DummyLabelRead[i+9];
      }
```

Generating Executables which are directly deployable on the ECU

SLG: Software Architecture



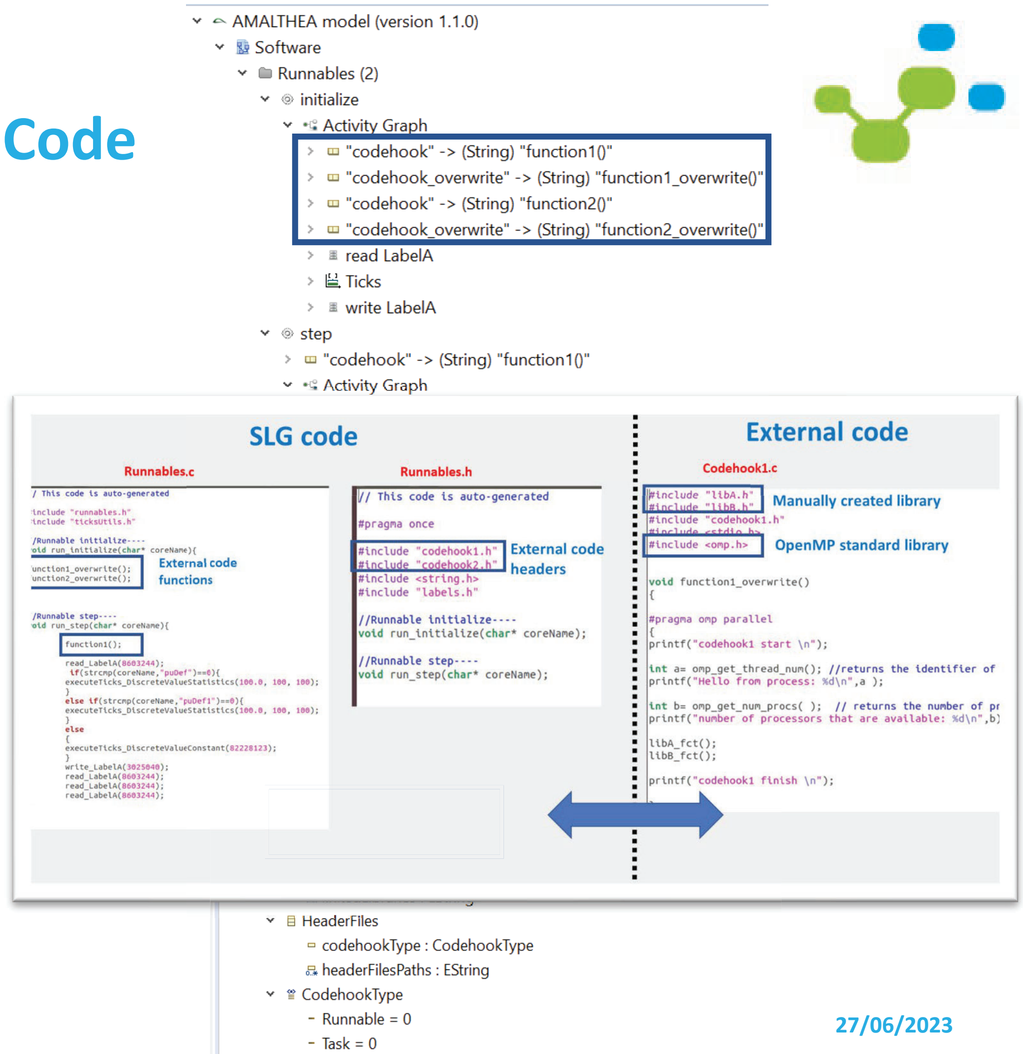
- Specific adaptions towards different middleware's and operation systems
  - Internal Autosar Adaptive code generation
  - ROS2, mircoRos, ErikoOS and Linux adapters are **open-sourced**

- **SLG.Commons:**
  - Contains central synthetic code elements common for all transformers, are **open-sourced**

- Generic transformation framework which provides infrastructure for building M2M transformations.

SLG: Extension for Custom Code

- Optional possibility to add **application specific code** to the SLG
- The user can provide code hooks for **custom code in the model at runnable or task level** to either override or contribute to the synthetic code
- Paths to **external libraries, code includes, and compiler keywords** can be specified in the configuration model to enable automatic generation of the make file



Thank you!

.AMPERE

www.ampere-euproject.eu

The AMPERE project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871669