

EAST-ADL

Concept Presentation

MAENAD Analysis Workbench

Outline, tooling with EAST-ADL

- MAENAD Modeling Workbench
 - EAST-ADL profile, implemented in Eclipse/Papyrus UML
- MAENAD Analysis Workbench
 - Eclipse “plugins” for use with MAENAD Modeling workbench
- Other EAST-ADL implementations
 - System Weaver
 - MetaEdit +
 - ARTOP

MAENAD Analysis Workbench (MAW)

- Primarily targeted at MAENAD Modeling Workbench
 - Using the EAXML exchange tool, other tools could be used as well
- Consist of:
 - Features and Variability in CVM (developed by TU-Berlin)
 - Simulink gateway (KTH)
 - Safety analysis with HiP-HOPS (KTH, University of Hull)
 - AUTOSAR gateway (CEA/Edona)
 - Functional Mock-up Unit (FMU) import (VTEC)
 - Timing Analysis with MAST (CEA/Edona)

Feature/Variability plugin

- Provide support for the management of feature-oriented modelling at vehicle level and vehicle configuration support on to the artefact level

Tool is twofold:

- Compositional Variability Management, CVM feature modelling editor
- Bridge with EAST-ADL Papyrus

Two Levels of Variability

Variability on the vehicle level:

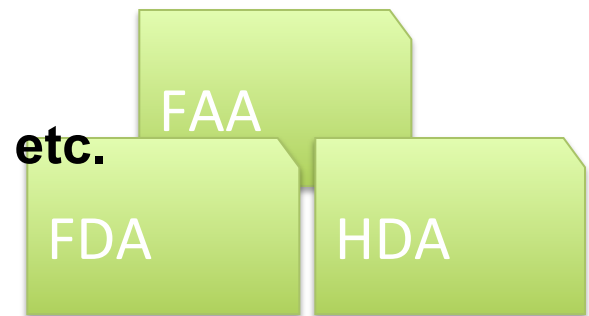
- **Very abstract; no design/implementation details.**
- **Distinction of customer vs. technical perspective.**
- **Modeling means: only Feature Modeling.**



Vehicle
Level

Variability on the „artifact level“:

- **Variability of the actual requirements, design, etc.**
- **Only technical perspective.**
- **Modeling means: Feature Modeling + Variation Points inside FAA/FDA**



Feature Models

Feature Model

Cardinality-based feature models

(cf. Czarnecki et al.)

with some modifications

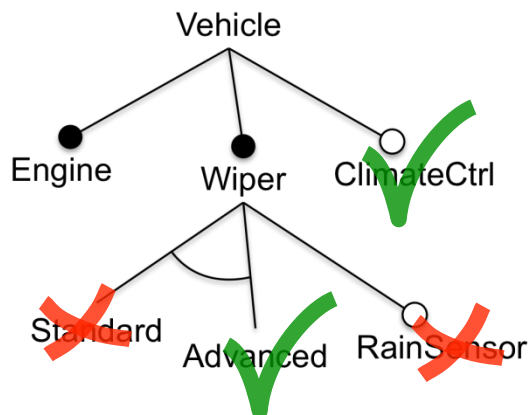
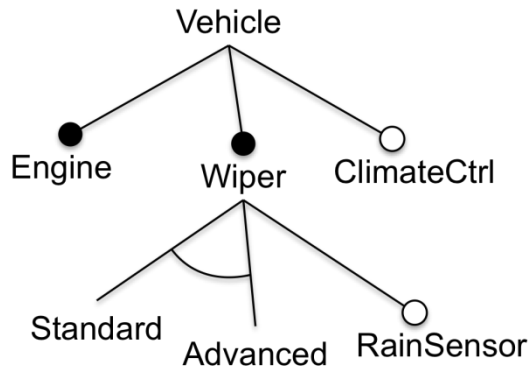
(e.g. 1+ root features per model)

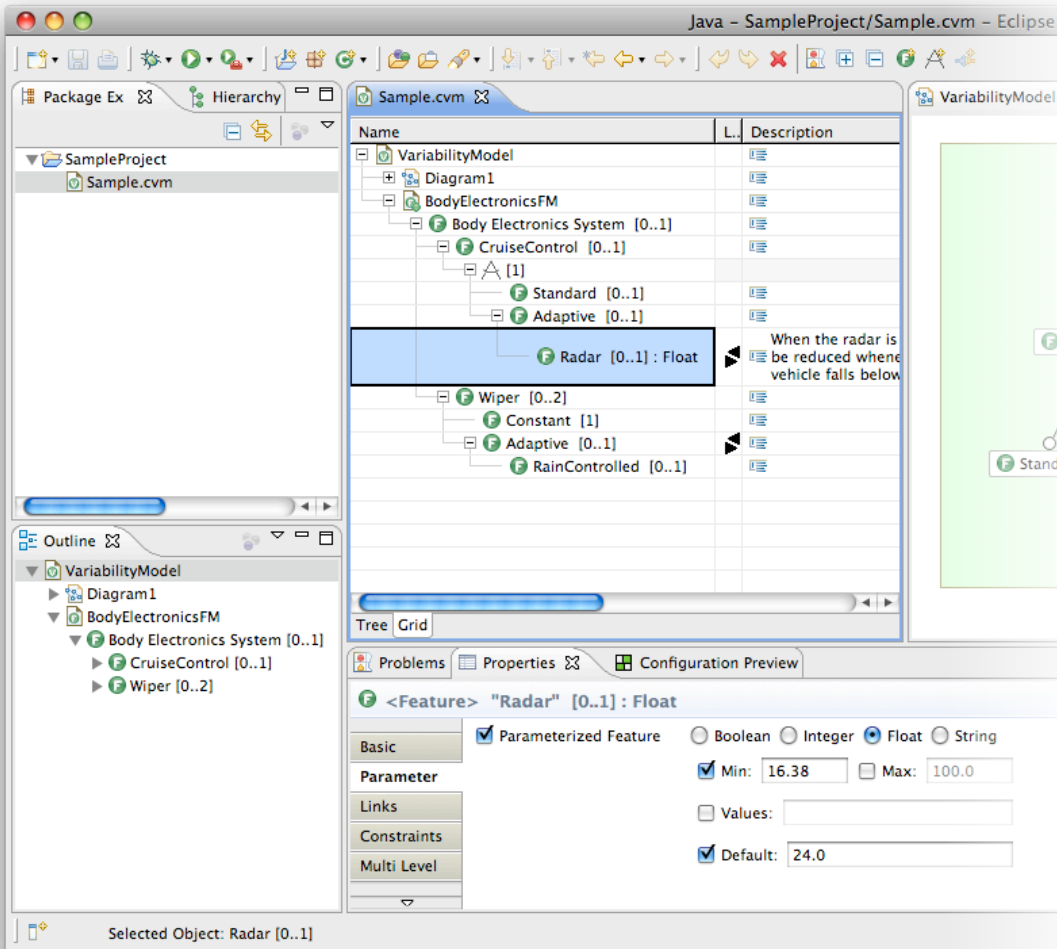
Configuration (of a Feature Model)

– parameterized features
(a.k.a. feature attributes)

– instances for cloned features

(subtrees of instances can be configured separately)





Java - SampleProject/Sample.cvm - Eclipse

Package Explorer: SampleProject, Sample.cvm

Hierarchy: VariabilityModel, Diagram1, BodyElectronicsFM, Body Electronics System [0..1], CruiseControl [0..1], Wiper [0..2], Constant [1], Adaptive [0..1], RainControlled [0..1]

Outline: VariabilityModel, Diagram1, BodyElectronicsFM, Body Electronics System [0..1], CruiseControl [0..1], Wiper [0..2]

Tree Grid: Name, Description

Name	Description
VariabilityModel	
Diagram1	
BodyElectronicsFM	
Body Electronics System [0..1]	
CruiseControl [0..1]	
Wiper [0..2]	
Constant [1]	
Adaptive [0..1]	
RainControlled [0..1]	
Radar [0..1] : Float	When the radar is selected, the car's speed will be reduced whenever the distance to the next vehicle falls below the specified threshold.

Properties: <Feature> "Radar" [0..1] : Float

Basic: ☒ Parameterized Feature, ☐ Boolean, ☐ Integer, ☒ Float, ☐ String

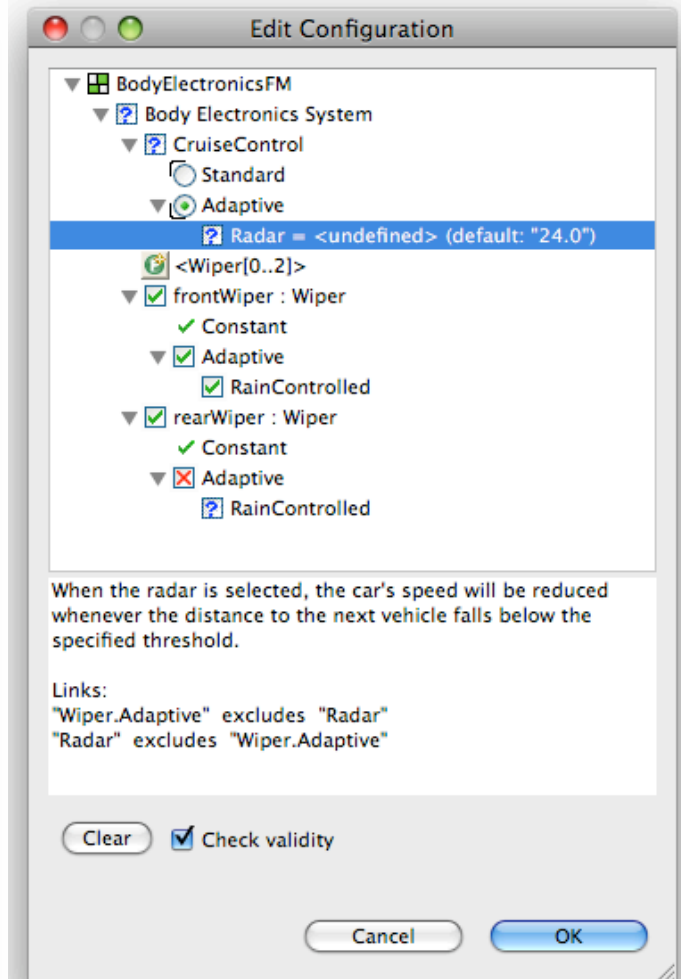
Parameter: ☒ Min: 16.38, ☐ Max: 100.0

Links: ☐ Values:

Constraints: ☒ Default: 24.0

Multi Level:

Selected Object: Radar [0..1]



Edit Configuration

BodyElectronicsFM

- Body Electronics System
 - CruiseControl
 - Standard
 - Adaptive

Radar = <undefined> (default: "24.0")

<Wiper[0..2]>

- frontWiper : Wiper
 - Constant
 - Adaptive
 - RainControlled
- rearWiper : Wiper
 - Constant
 - Adaptive
 - RainControlled

When the radar is selected, the car's speed will be reduced whenever the distance to the next vehicle falls below the specified threshold.

Links:
 "Wiper.Adaptive" excludes "Radar"
 "Radar" excludes "Wiper.Adaptive"

Clear ☒ Check validity

Cancel OK

Functional Mock-up Unit import

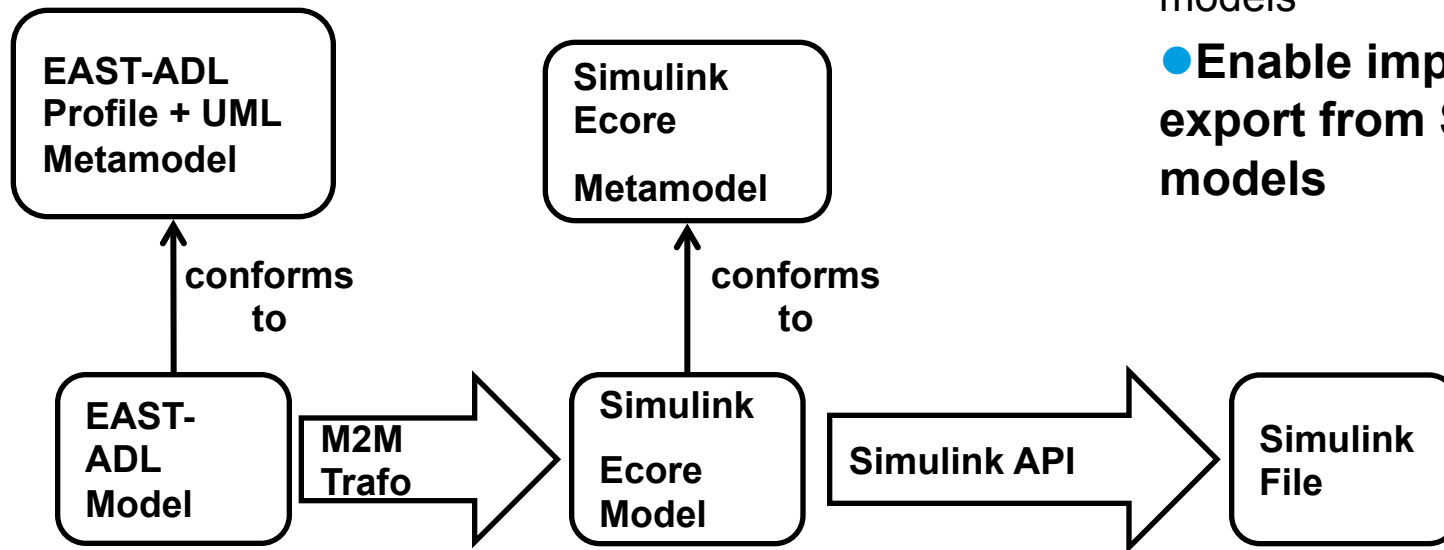
- The Modelisar project has defined Function Mockup Units to exchange and integrate simulation blocks from different modeling tools.
- Modelisar supports Vehicle Functional Mock-up, a next generation of methods, standards and tools to support collaborative design, simulation and test of systems and embedded software.

Functional Mock-up Unit import

- There is an Eclipse plugin that imports the Function Mockup Unit specification, called Function Mockup Interface (FMI). The Function Mockup Interface defines the input and output variables of each unit and also the data types of these variables.
- Based on this information, an AnalysisFunctionType with the corresponding interface is defined in EAXML.

Simulink gateway

- Goal: enable import/export from Simulink models
- **Enable import/export from Simulink models**



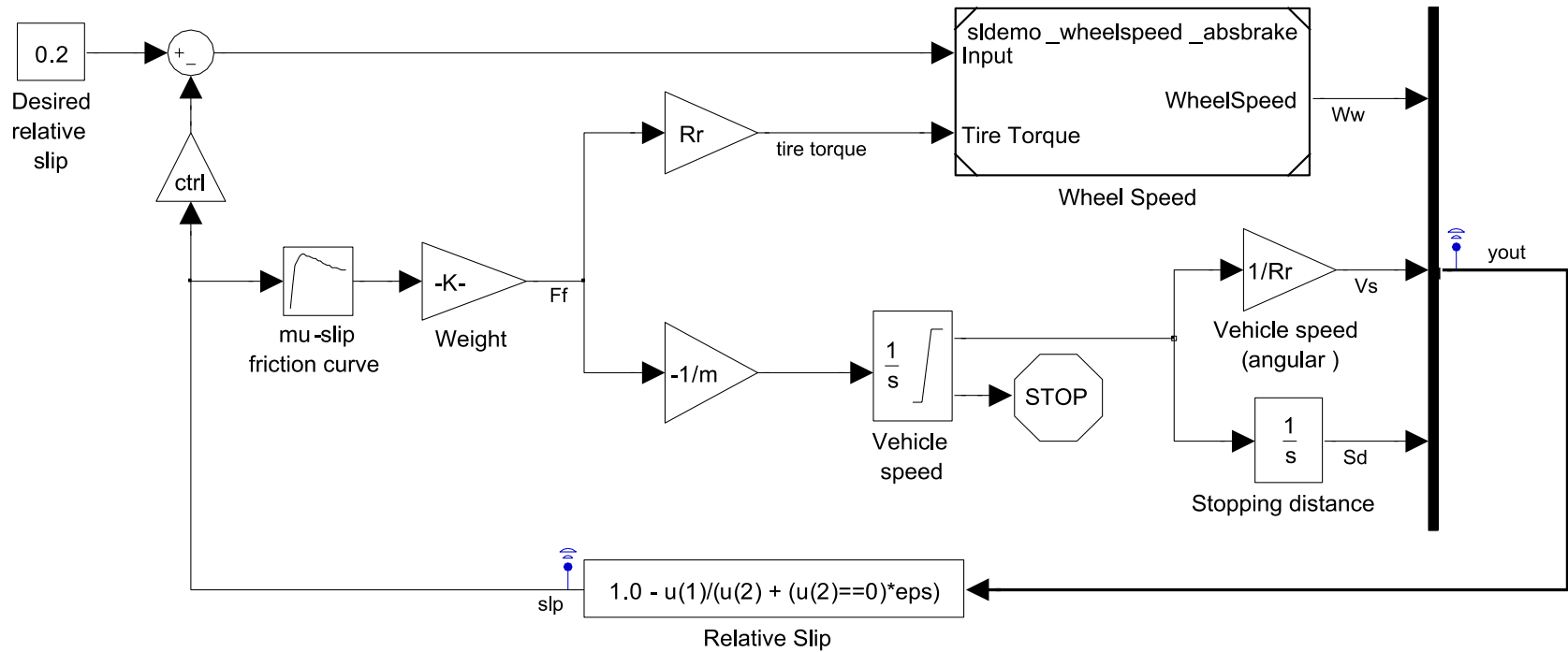
Step 1: Model to Model Transformation
Using ATL

Step 2: Using MATLAB/ Simulink API and JAVA to access Simulink models

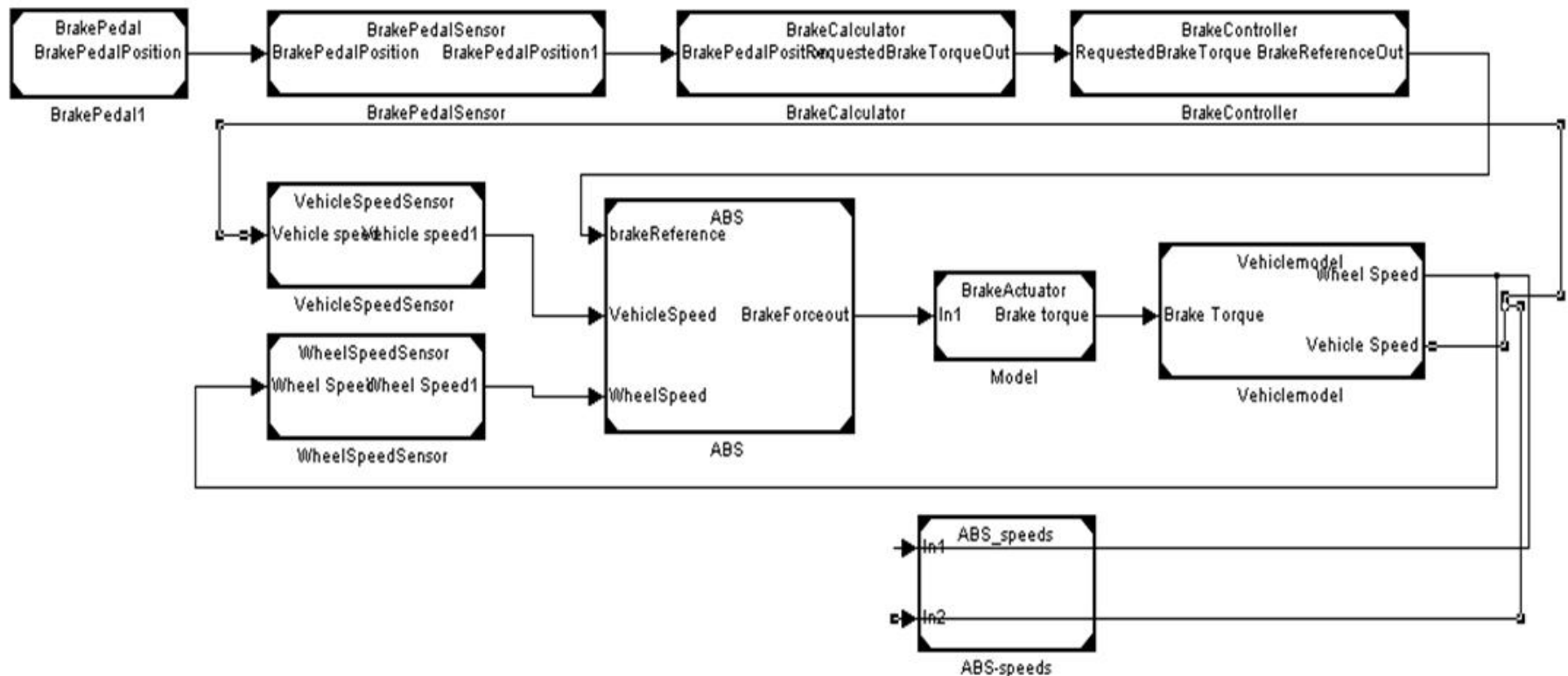
Content of Plant modelling

● Example case study: ABS modelled in Simulink

Modeling an Anti-Lock Braking System (ABS)



A Simulink model, structured using subsystems

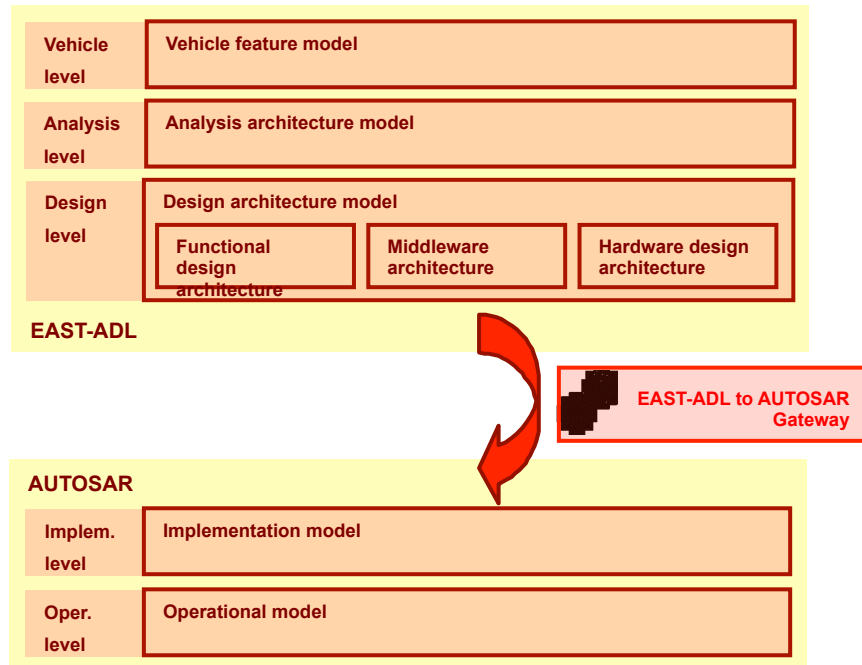


Basic mapping

EAST-ADL	Simulink
AnalysisFunctionType	Library block
AnalysisFunctionPrototype	Reference block
Port	Port
Connector	Line



AUTOSAR gateway

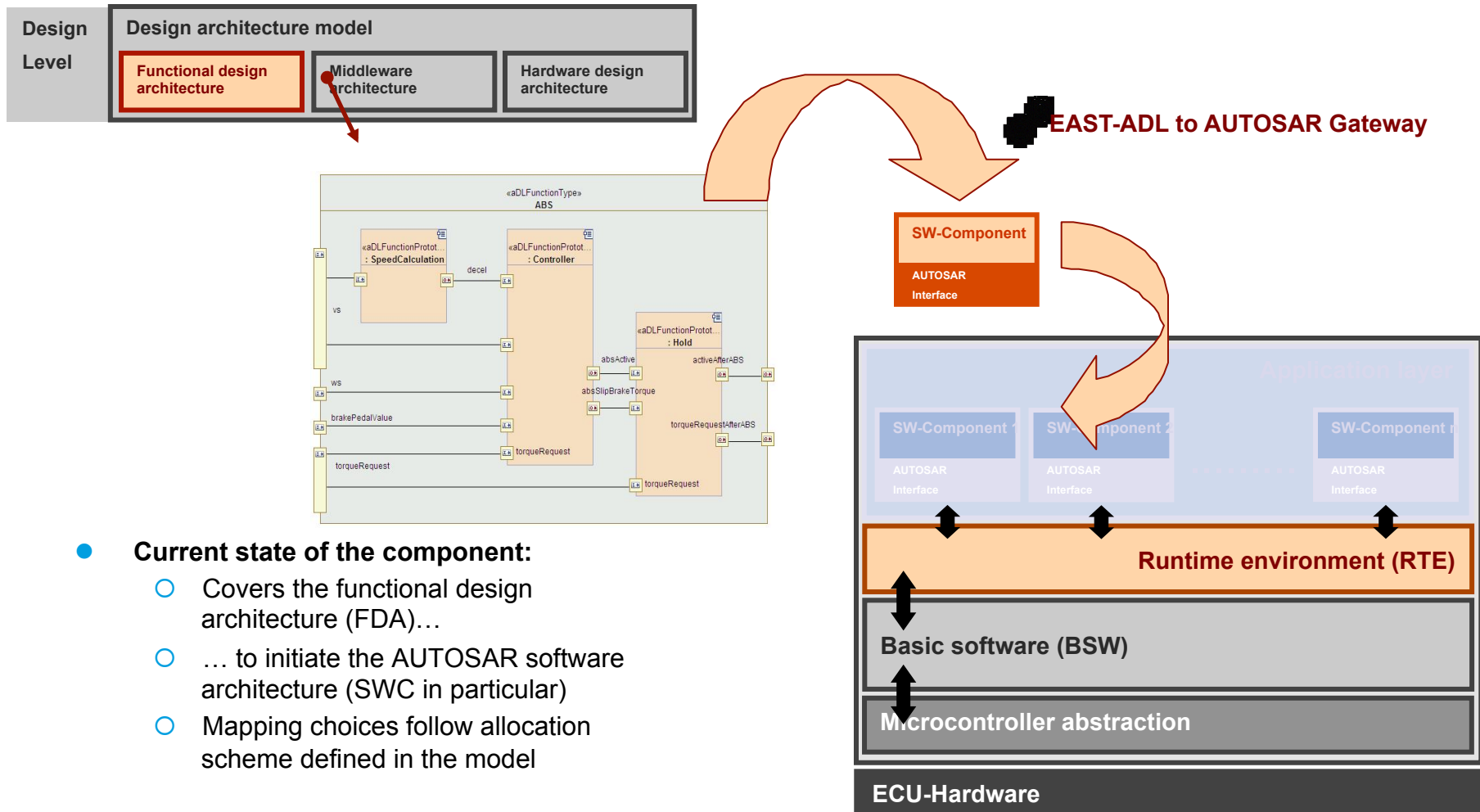


- Updated from ATESS2 and EDONA (French System@tic cluster) results

The refinement activity between EAST-ADL and AUTOSAR is:

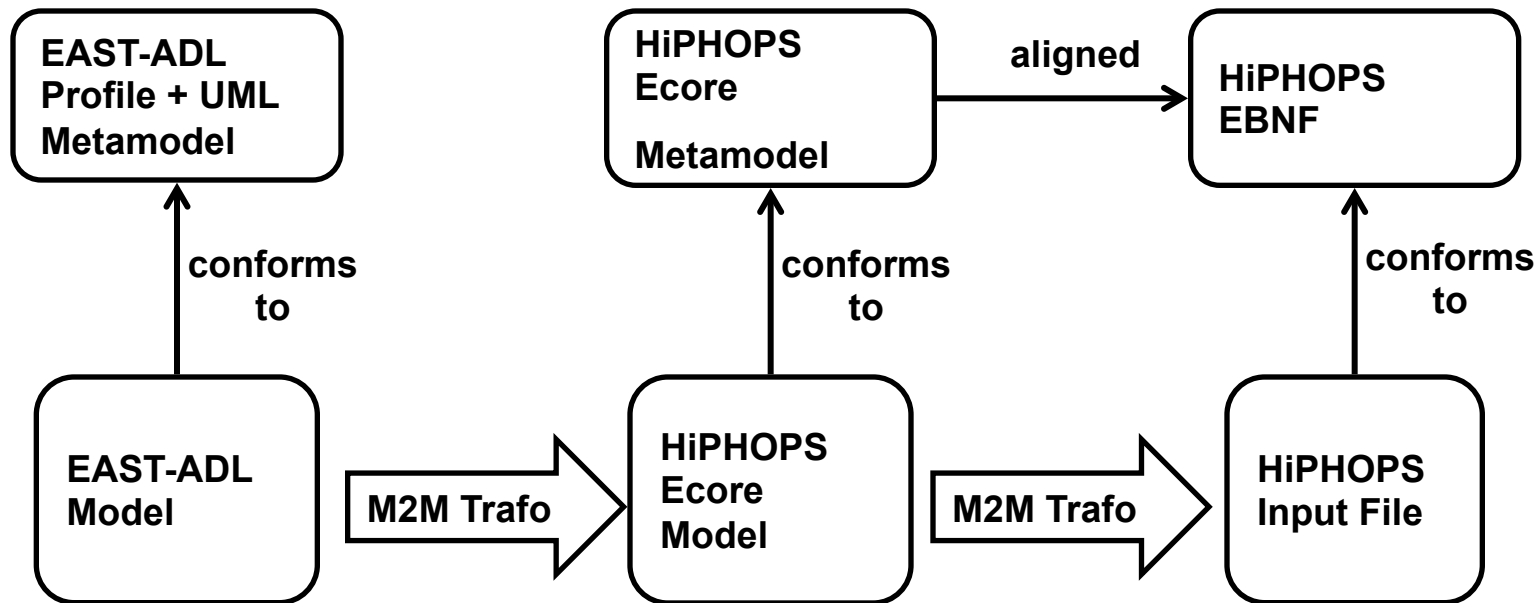
- Tedious and Repetitive
- Error prone
- Time consuming
- Uneasy as it is necessary to manage AUTOSAR consistency and to make mapping (function / software) choice at the same time
- Automated mapping taking into account allocation constraints and hardware architecture.

Transformation: overview



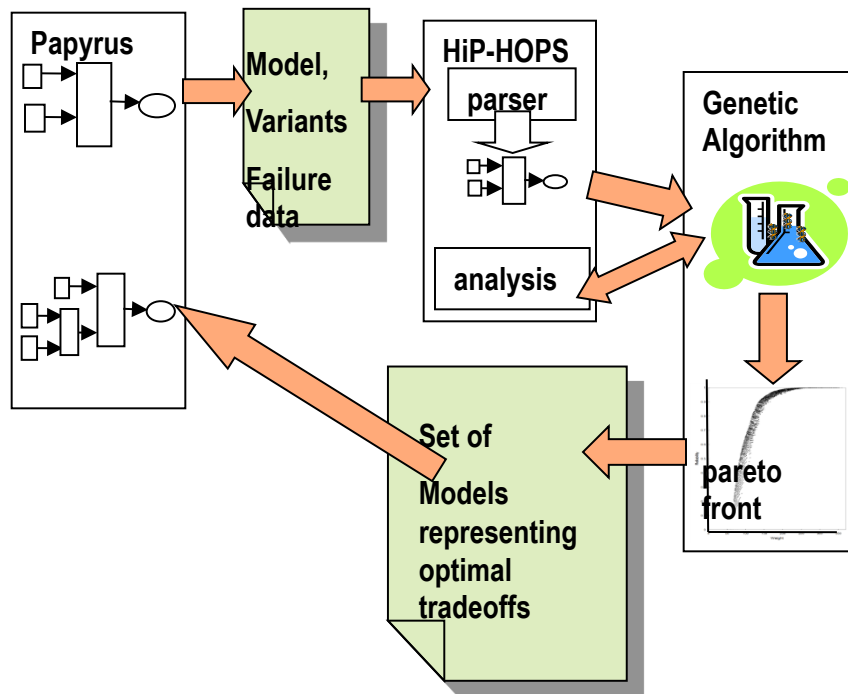
- **Current state of the component:**
 - Covers the functional design architecture (FDA)...
 - ... to initiate the AUTOSAR software architecture (SWC in particular)
 - Mapping choices follow allocation scheme defined in the model

Safety analysis plugin



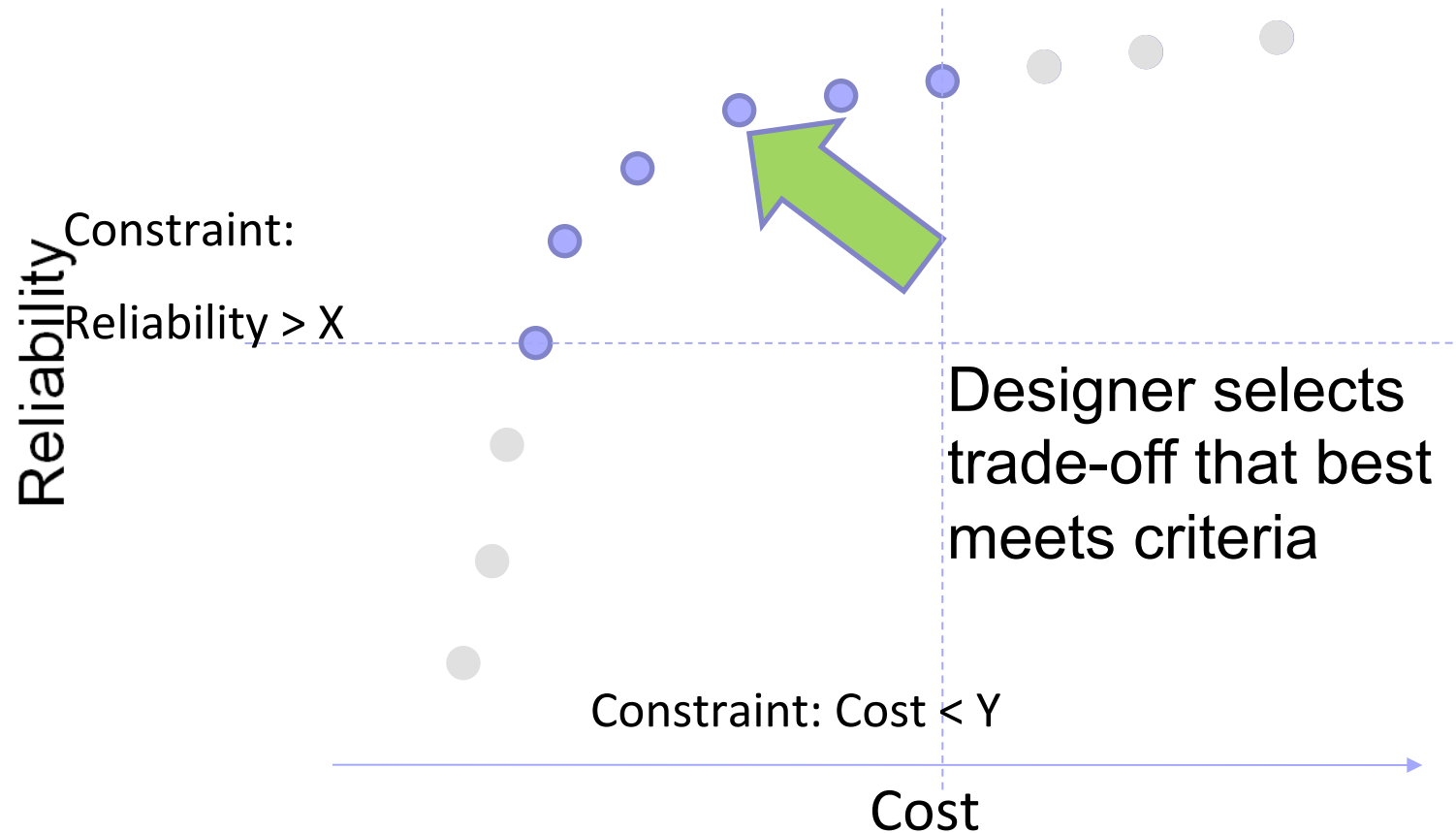
- Enable safety analysis with HipHOPS engine

Optimization in HiP-HOPS



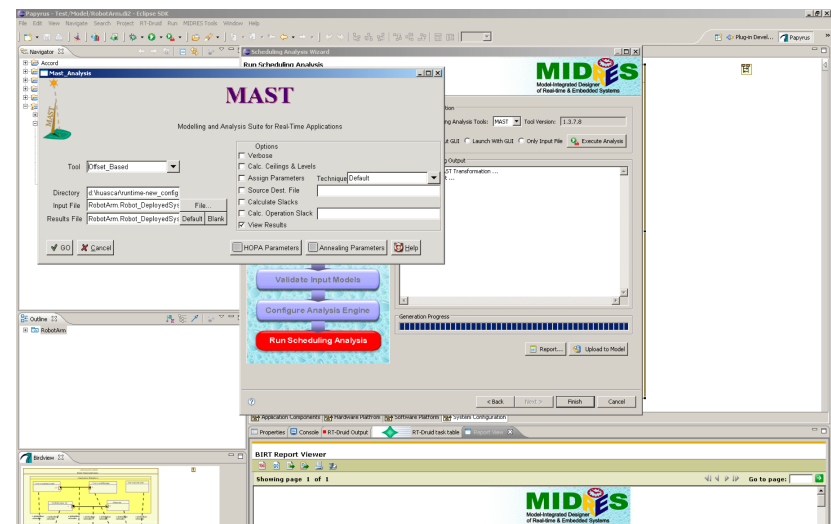
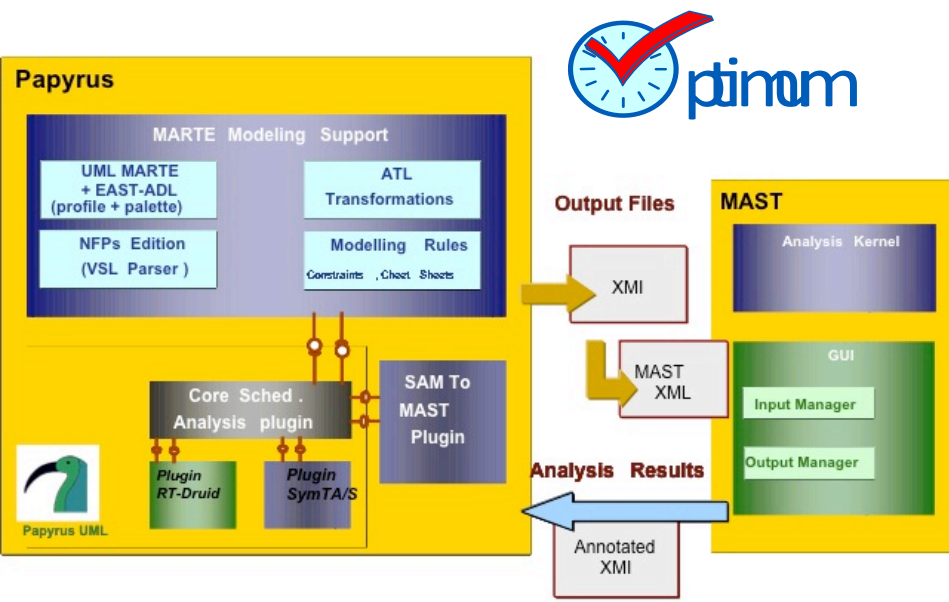
- Designer provides a system design in EAST-ADL, elements (functions/components) of which can be implemented by variants
- Error models and costs of these variants are given.
- Objectives of the optimization and constraints are defined
- HiP-HOPS performs automatic optimisation via selection and application of variants in the architecture and returns a set of “Pareto Optimal Designs”

Design solutions returned by HiP-HOPS



Timing analysis

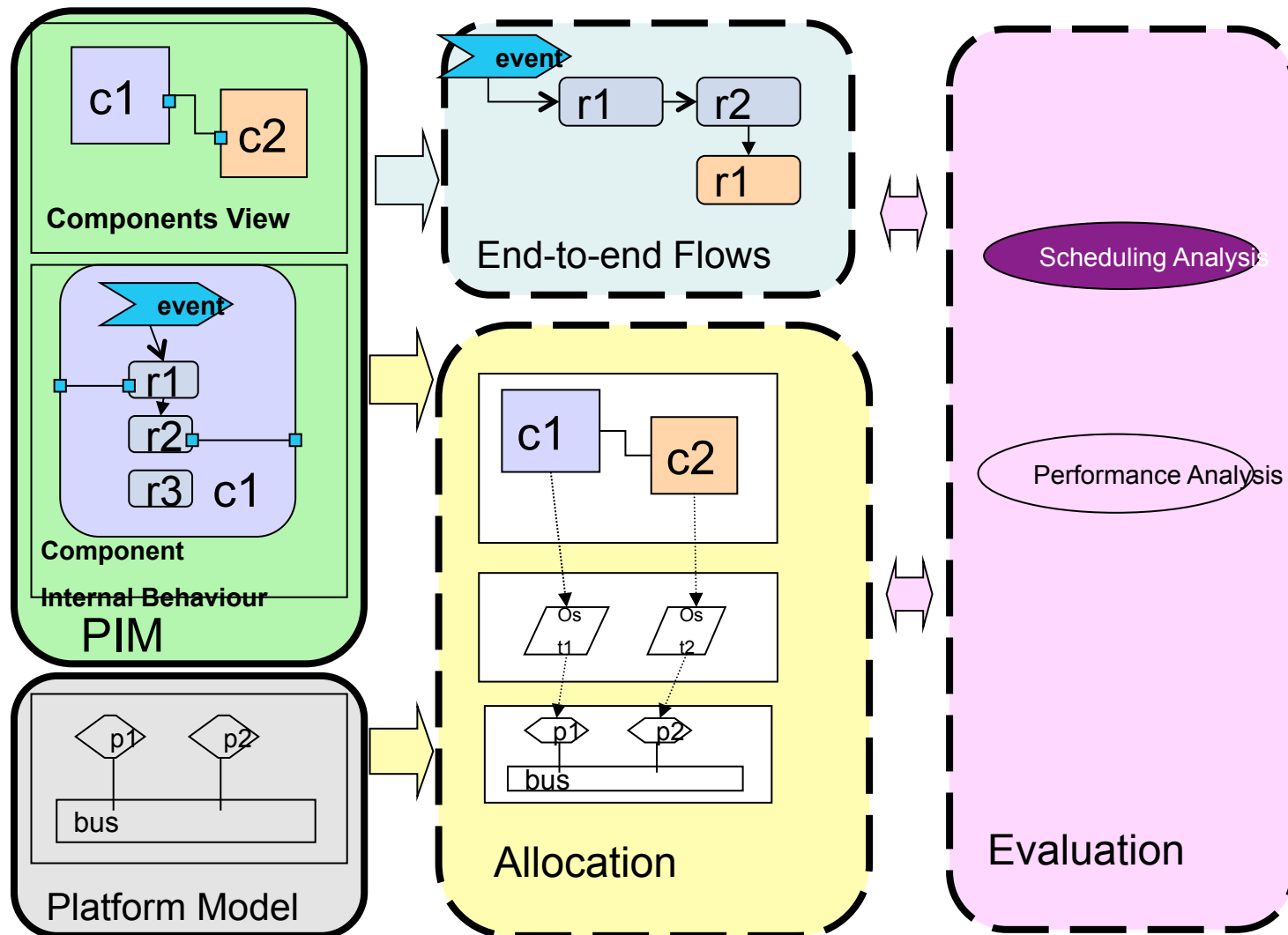
- Perform timing analysis at Design level based on timing assumptions captured through MARTE timing constructs annotations
 - OS abstraction (tasks, channels for communication)
 - End-to-end flows
- Currently being adapted for MAENAD from previous project results.



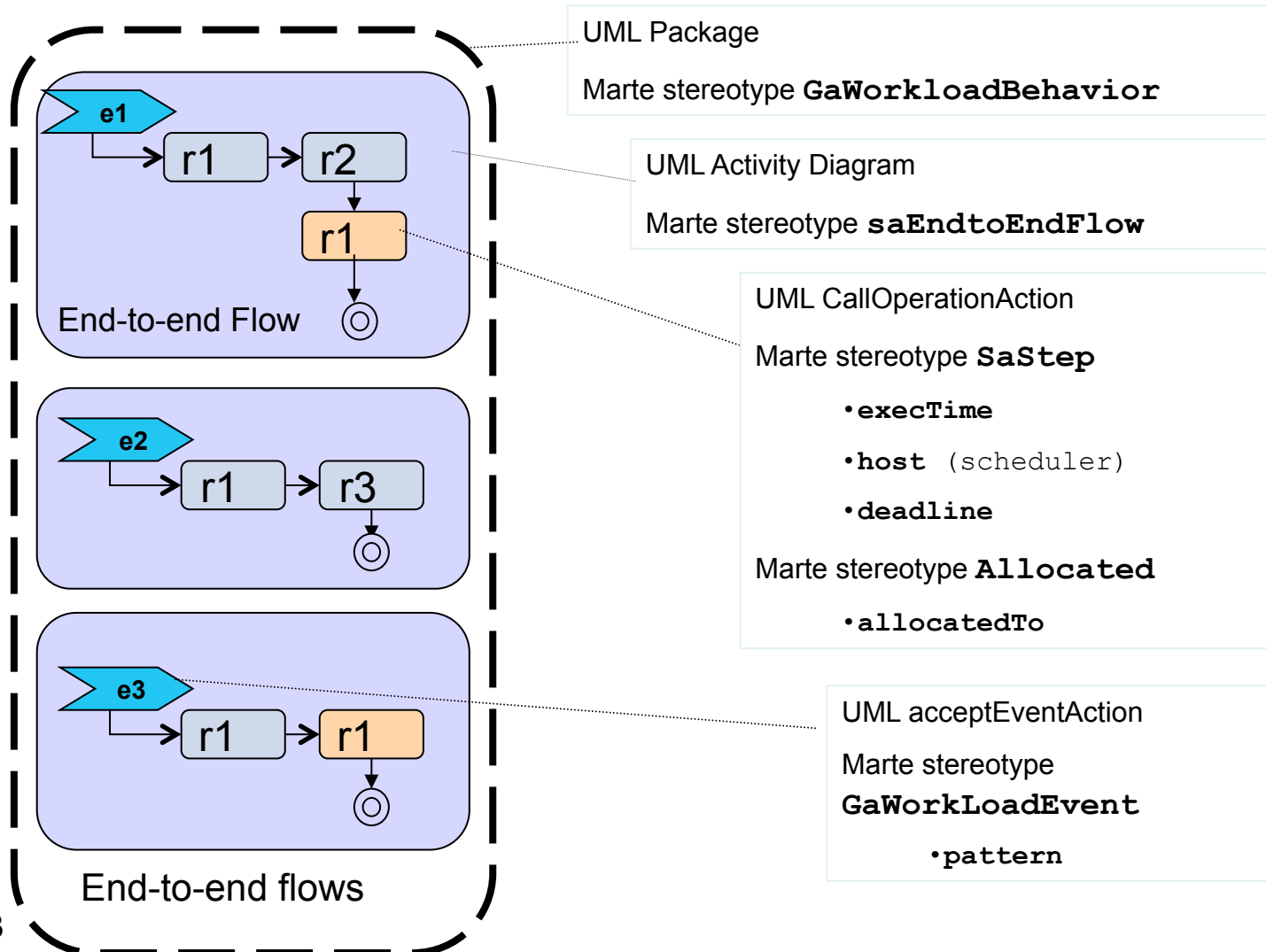
Summary of our Design Principles

- **Using Component Models** for modeling automotive applications
 - MARTE (EAST-ADL) Component Models
 - MARTE (EAST-ADL) Internal Behavior Models
- **Using Workload Behaviors of MARTE** to capture the behavior of that part of the system we need to analyze under possibly different workloads
- **Using the Allocation model of MARTE** to bind PMI elements on Platform elements. Many alternatives are possible and worth to be explored
- Selecting a small set of MARTE elements to feed **existing** scheduling (& simulation) analysis **tools**

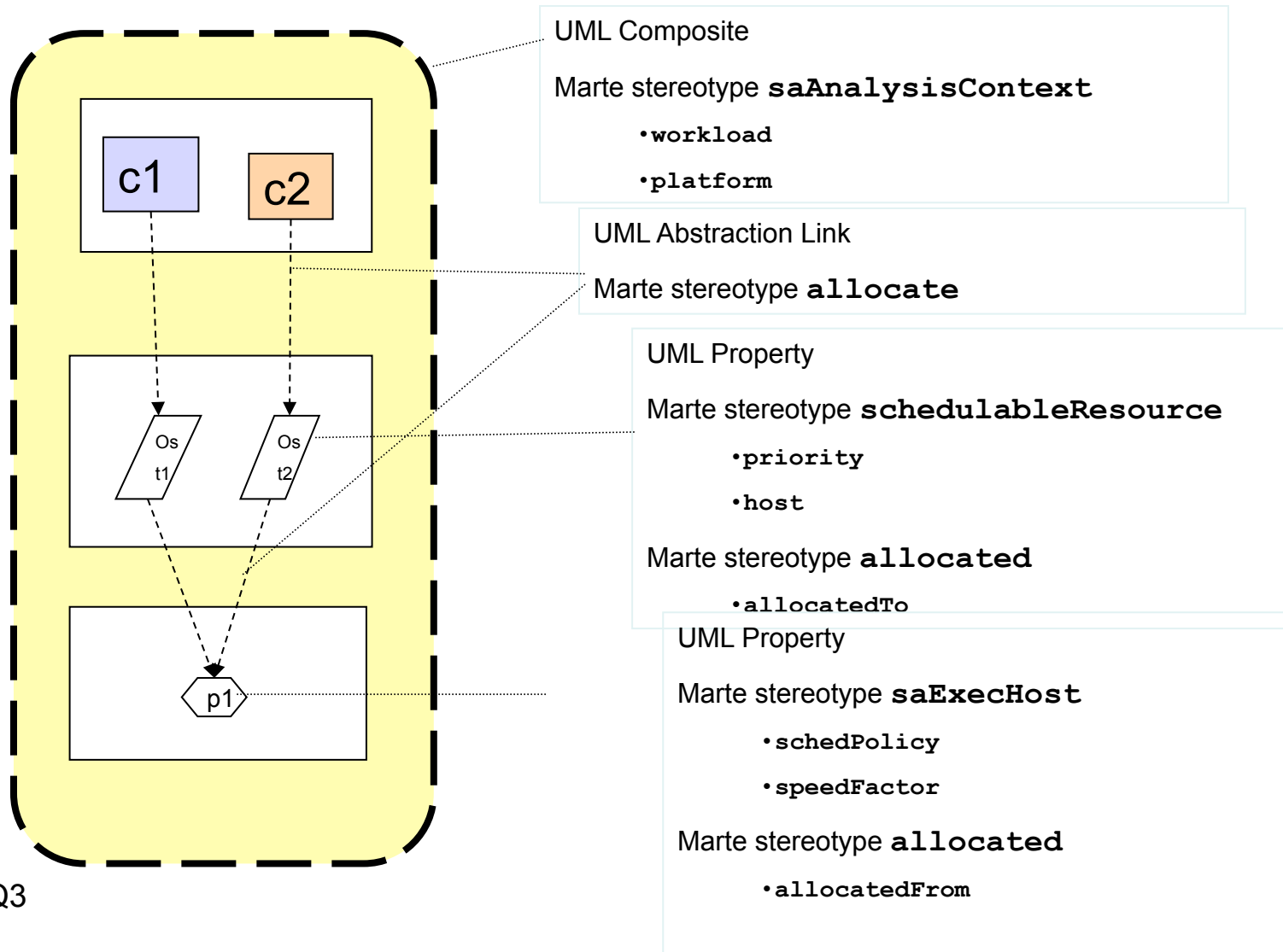
Design Flow for Model Evaluation



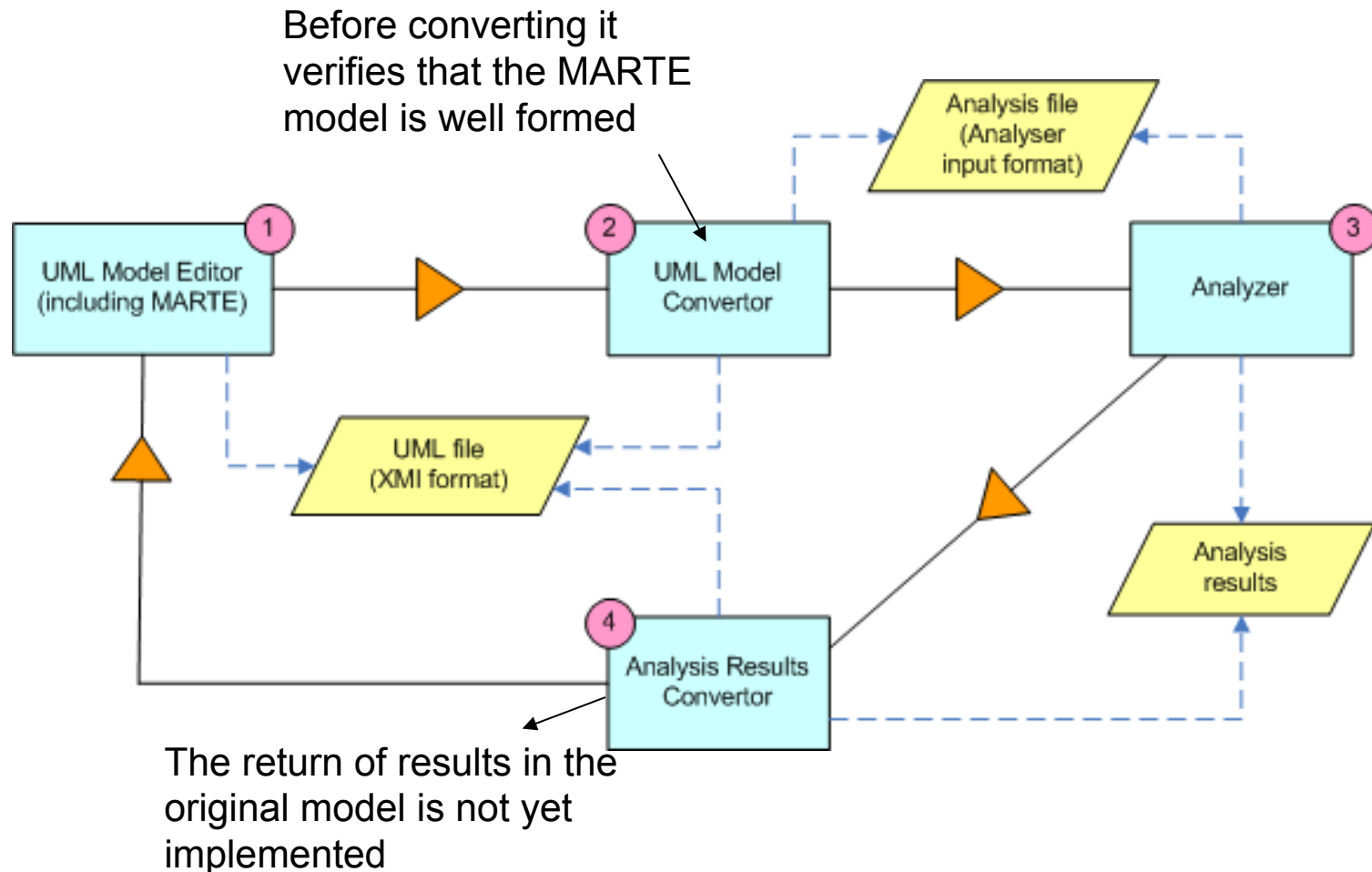
UML+MARTE end-to-end flows



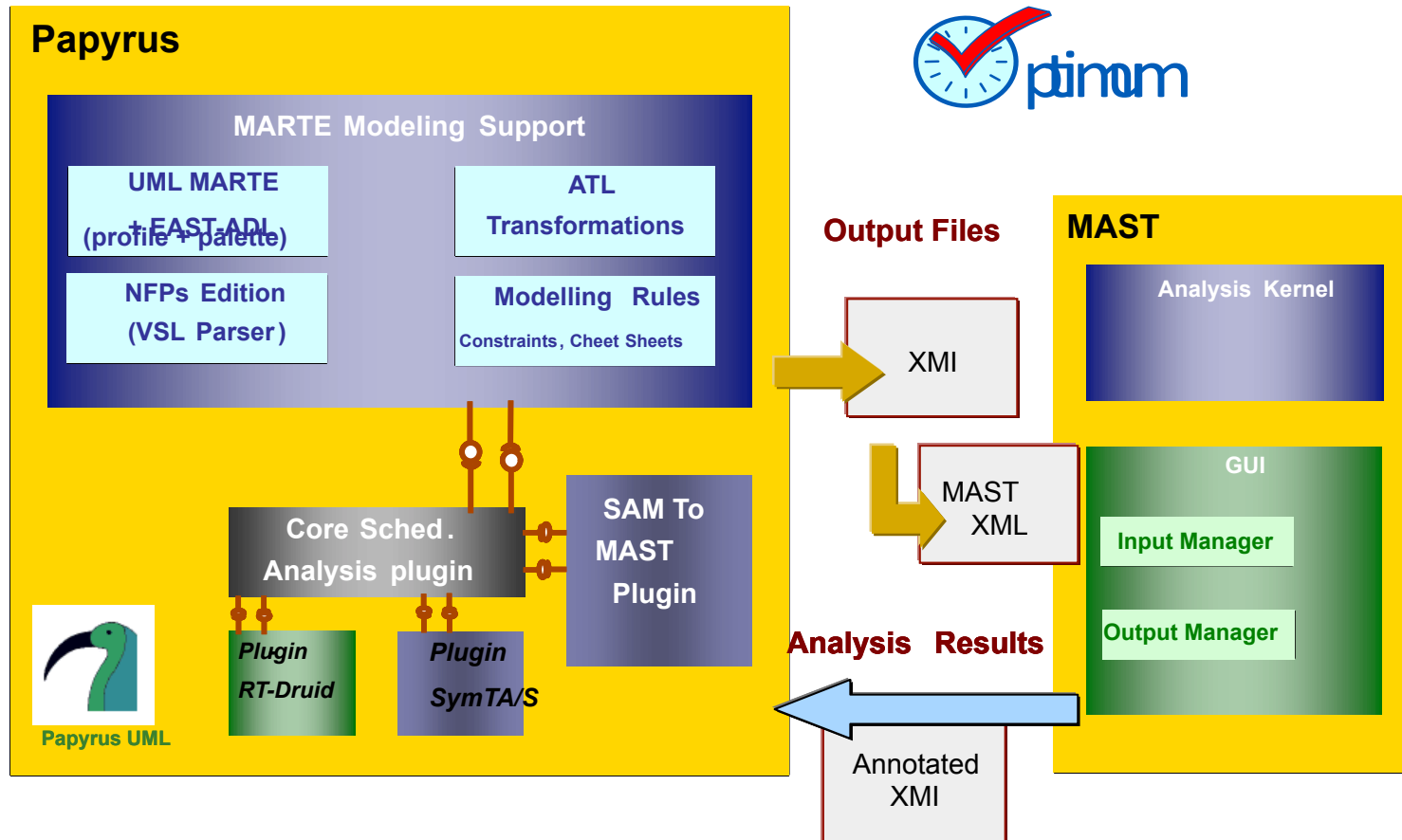
UML+MARTE Allocation



Tool Modules



Tool Technology Infrastructure



Conclusion on Timing analysis

- The chosen design method and tool is generic. Various entry languages could be used (here EAST-ADL, but others could also be used).
- The end-to-end flow modeling rests upon MARTE constructs. Identifying a clear mapping and possible lacks between MARTE/EAST-ADL has been undertaken. End goal is to have a MARTE-based implementation of EAST-ADL

Summary

- MAENAD Analysis Workbench consists of:
 - Features and Variability in CVM (developed by TU-Berlin)
 - Simulink gateway (KTH)
 - Safety analysis with HiP-HOPS (KTH, University of Hull)
 - AUTOSAR gateway (CEA/Edona)
 - Functional Mock-up Unit (FMU) import (VTEC)
 - Timing Analysis with MAST (CEA/Edona)
- More to be developed during the MAENAD project
 - Modelica/ModelicaML exchange
 - Architecture optimization and configuration
 - EAXML exchange