Hierarchical Megamodels for Model Management in Architecture-Centric Virtual Integration Development

Dominique Blouin

LTCI Lab, ACES Group

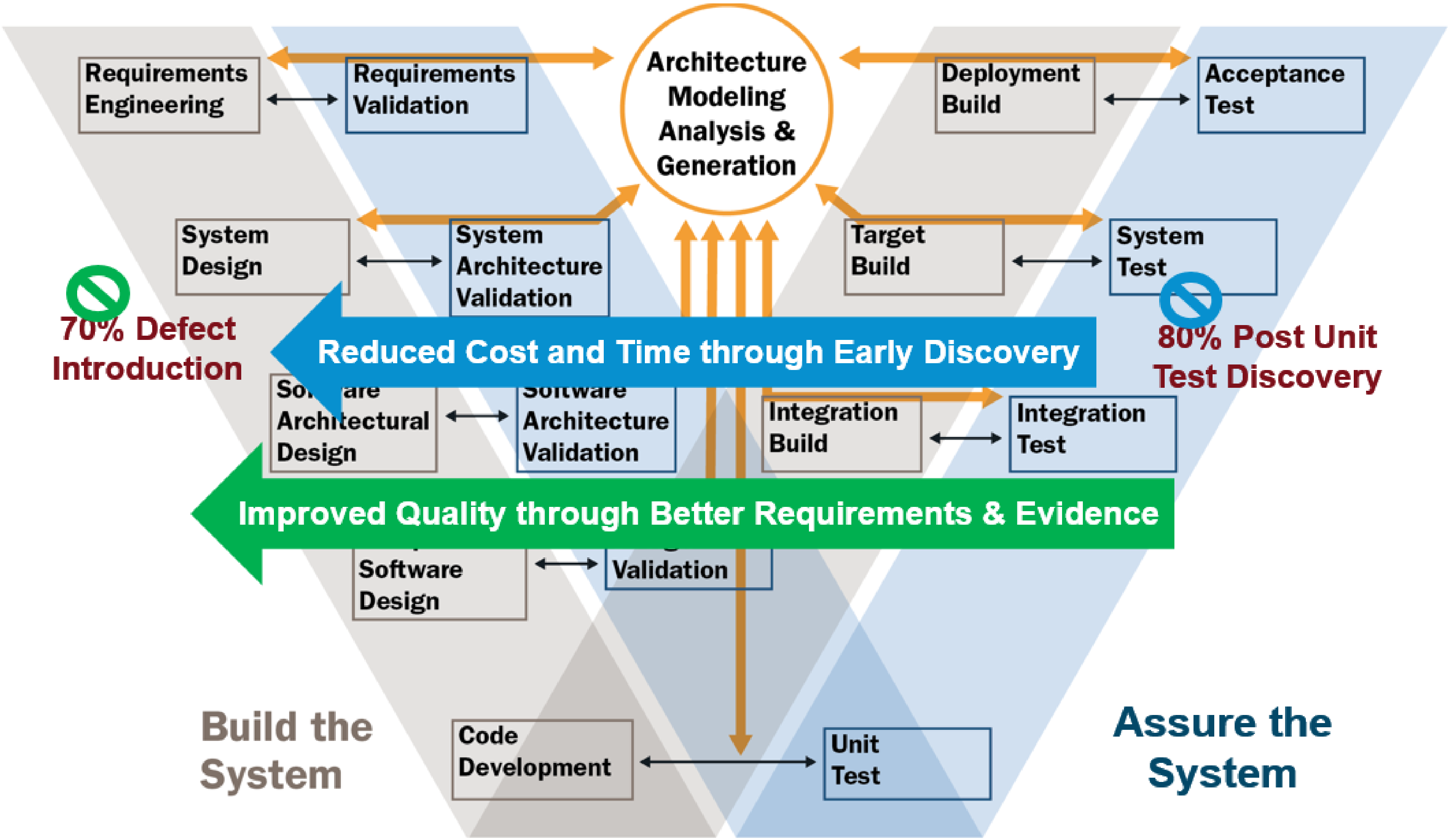
Telecom Paris, Institut Polytechnique de Paris, France

dominique.blouin@telecom-paris.fr

Outline

- Context
- Hierarchical Megamodels
- ACMoM Framework
- Conclusion

V-Cycle Model with Virtual Integration Activities (Architecture-Centric Virtual Integration Process)



Requirements Engineering

Requirements Validation

Architecture Modeling Analysis & Generation

Deployment Build

Acceptance Test

System Design

System Architecture Validation

Target Build

System Test

70% Defect Introduction

Reduced Cost and Time through Early Discovery

80% Post Unit Test Discovery

Software Architectural Design

Software Architecture Validation

Integration Build

Integration Test

Improved Quality through Better Requirements & Evidence

Software Design

Validation

Build the System

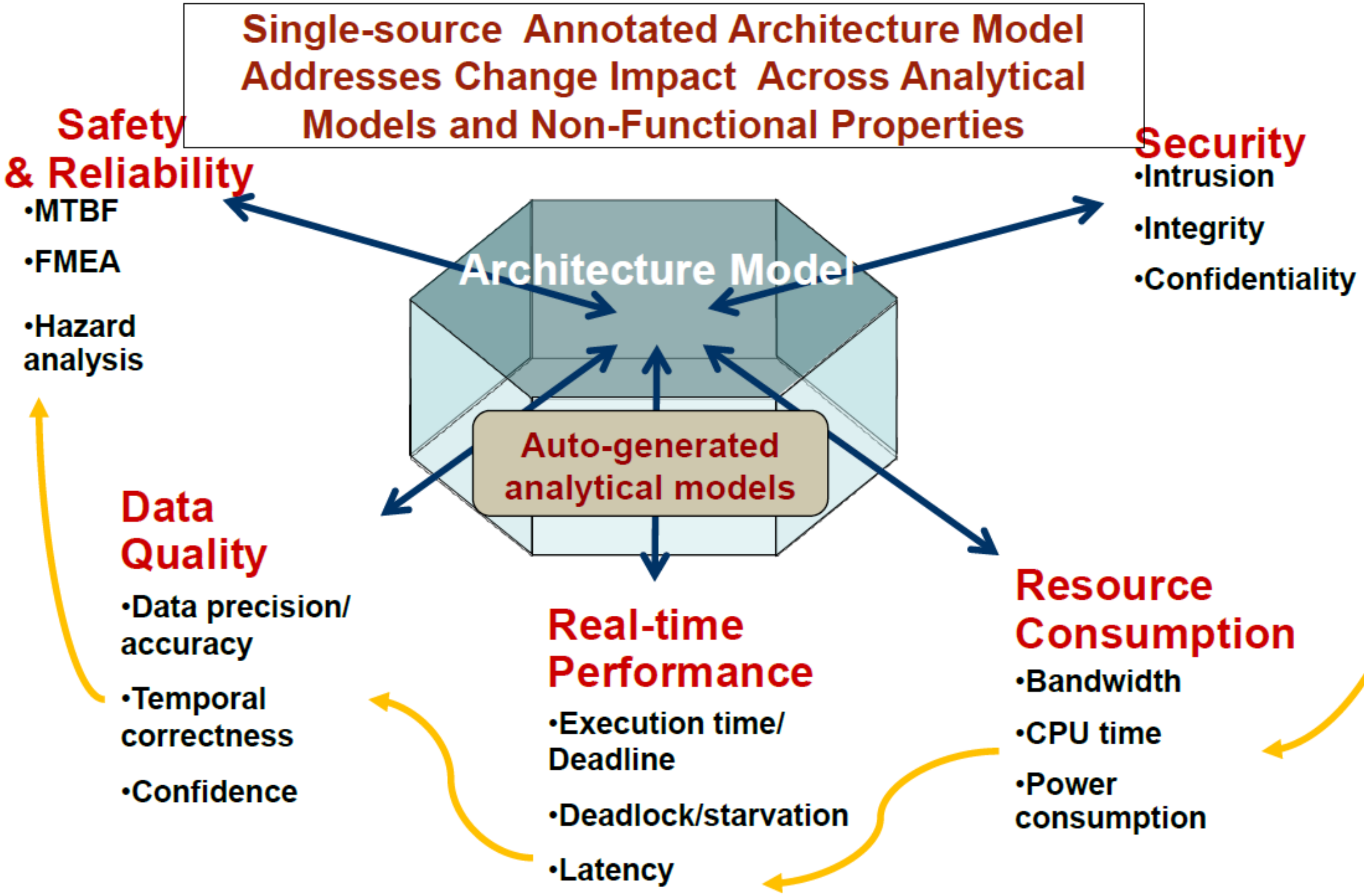
Code Development

Unit Test

Assure the System

From McGregor, Gluch, and Feiler, “Analysis and Design of Safety-critical, Cyber-physical Systems”, 2017.

AADL & SAVI (System Architecture Virtual Integration)



Single-source Annotated Architecture Model Addresses Change Impact Across Analytical Models and Non-Functional Properties

Architecture Model

Auto-generated analytical models

Safety & Reliability

- MTBF
- FMEA
- Hazard analysis

Security

- Intrusion
- Integrity
- Confidentiality

Data Quality

- Data precision/ accuracy
- Temporal correctness
- Confidence

Real-time Performance

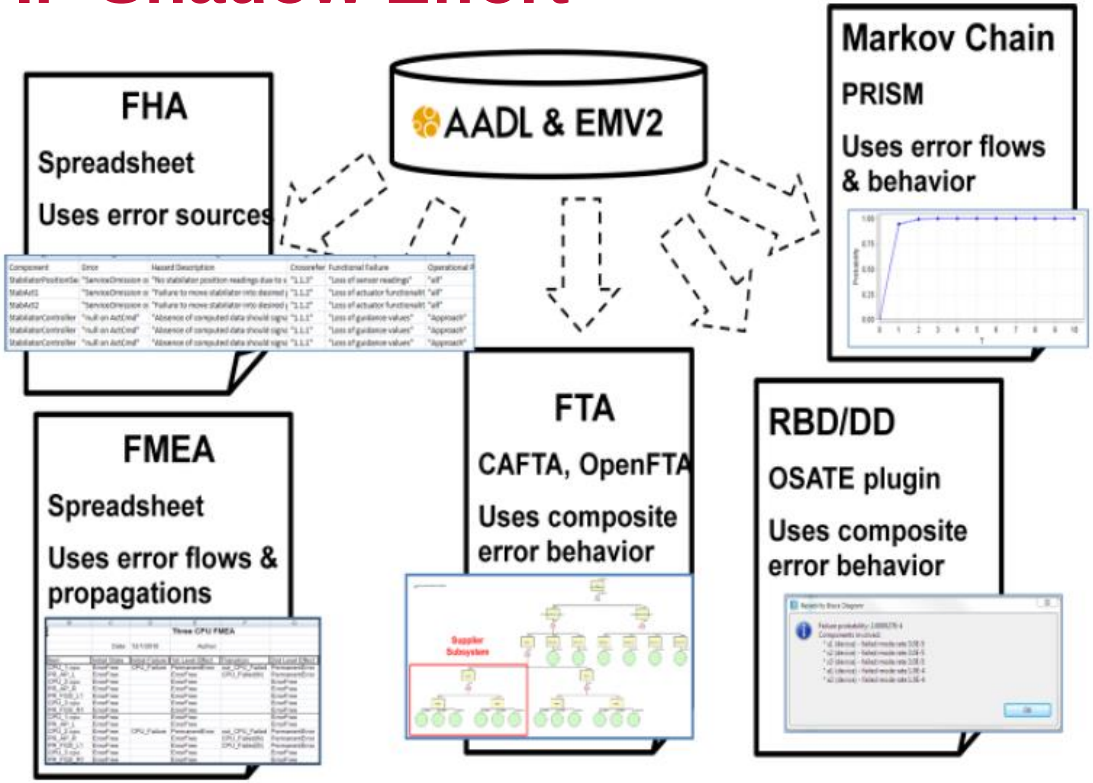
- Execution time/ Deadline
- Deadlock/starvation
- Latency

Resource Consumption

- Bandwidth
- CPU time
- Power consumption

From Feiler, Hansson, de Niz and Wrage. “System Architecture Virtual Integration: An Industrial Case Study”, 2009.

Joint Common Architecture Demonstration ACVIP Shadow Effort



AADL & EMV2

FHA

Spreadsheet

Uses error sources

Component	Error	Hazard Description	Crossrefer	Functional Failure	Operational P
StabilatorPositionSe	"ServiceOmission o	"No stabilator position readings due to s	"1.1.3"	"Loss of sensor readings"	"all"
StabAct1	"ServiceOmission o	"Failure to move stabilator into desired ;	"1.1.2"	"Loss of actuator functionalit	"all"
StabAct2	"ServiceOmission o	"Failure to move stabilator into desired ;	"1.1.2"	"Loss of actuator functionalit	"all"
StabilatorController	"null on ActCmd"	"Absence of computed data should signi	"1.1.1"	"Loss of guidance values"	"Approach"
StabilatorController	"null on ActCmd"	"Absence of computed data should signi	"1.1.1"	"Loss of guidance values"	"Approach"
StabilatorController	"null on ActCmd"	"Absence of computed data should signi	"1.1.1"	"Loss of guidance values"	"Approach"

Markov Chain

PRISM

Uses error flows & behavior

FMEA

Spreadsheet

Uses error flows & propagations

FTA

CAFTA, OpenFTA

Uses composite error behavior

RBD/DD

OSATE plugin

Uses composite error behavior

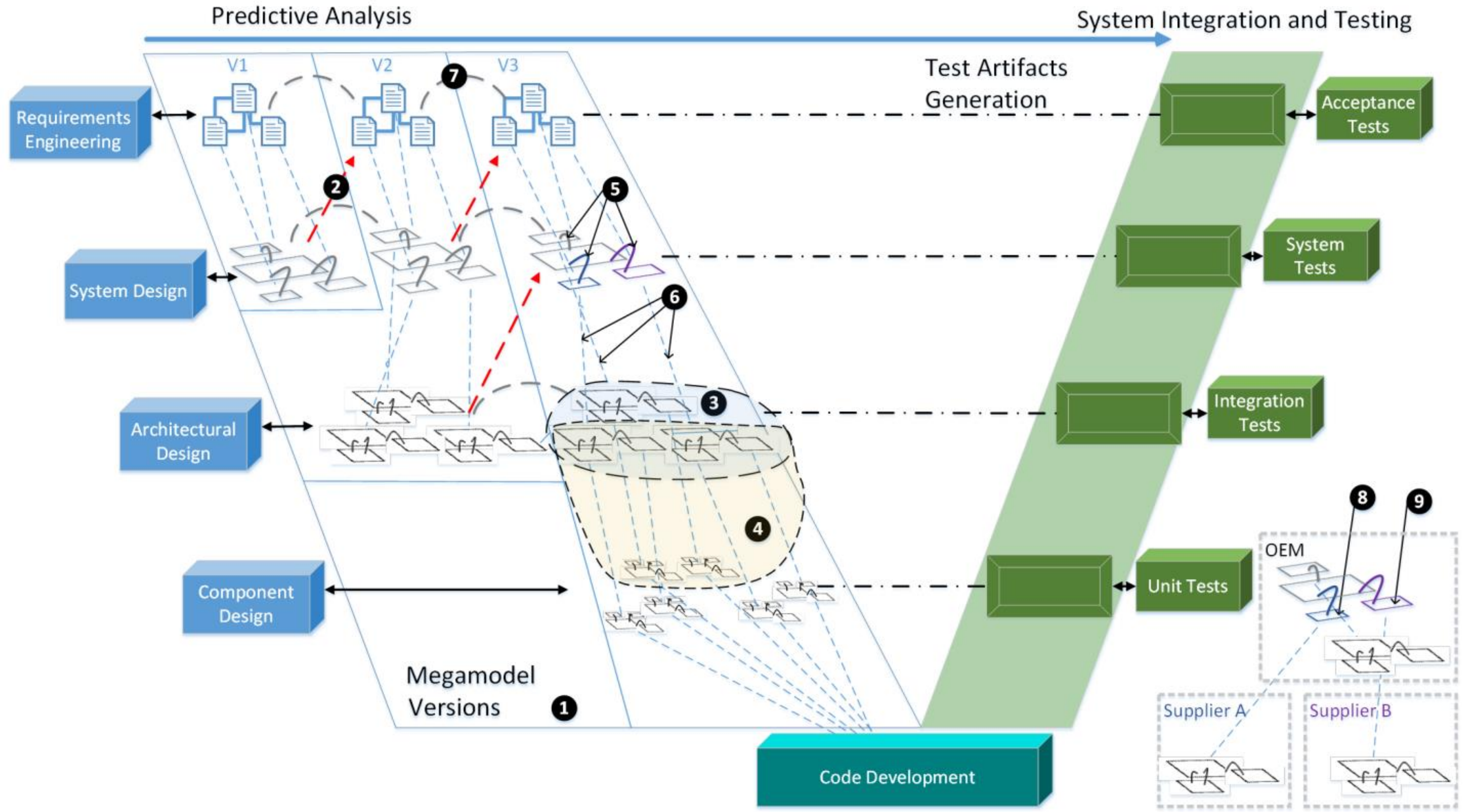
- *"Also, translation and exchange of models among different languages (e.g., UML, SysML, AADL, MatLab/Simulink and SCADE) and tools needs to be worked to allow government, integrators, and component suppliers to communicate seamlessly."*

From A. Boydston, P. Feiler, S. Vestal and B. Lewis, "Joint Common Architecture (JCA) Demonstration Architecture Centric Virtual Integration Process (ACVIP) Shadow Effort", 2015

Need for Model Management

- **Many models are employed:**
  - Joint Common Architecture Demonstration ACVIP Shadow Effort
  - PST project with ReqIF, SysML, AADL, etc.
- **Information overlap between models**
  - Consistency
  - Information preservation
- **Multiple teams manipulate models concurrently**
- **Different technical spaces (Ecore, XML, code, doc, etc.)**
- **Support continuous virtual integration (PST project)**

V-Cycle Model with Model Management Activities



From H. Giese and D. Blouin, miGMM DFG Project Proposal, 2016

Needs for Model Management Framework

- **What are the employed models, languages and tools?**
- **How are they related?**
  - Simple traceability?
  - Batch transformations?
  - Synchronization?
- **What is the development process**
  - Workflows
  - Modeling activities and **constraints**
- **Change management**
  - What model can be changed?
  - By who?
  - When?

Multi-Paradigm Modeling (MPM)

- **MPM main principles:**
  - Model every part and aspect of a system explicitly
  - At the most appropriate level(s) of abstraction
  - Using the most appropriate modeling formalism(s)

- **➔Model *model management***

Outline

- Context
- Hierarchical Megamodels
- ACMoM Project
- Conclusion

Megamodeling

- “A megamodel is a model with other models as elements”. “A megamodel contains relationships between models.” (Bézivin, 2003 / 2007)
- “... the idea behind a megamodel is to define the set of entities and relations that are necessary to model some aspect about MDE”. (Favre 2004 / 2005)
- Unified definition (Hebig et al.)

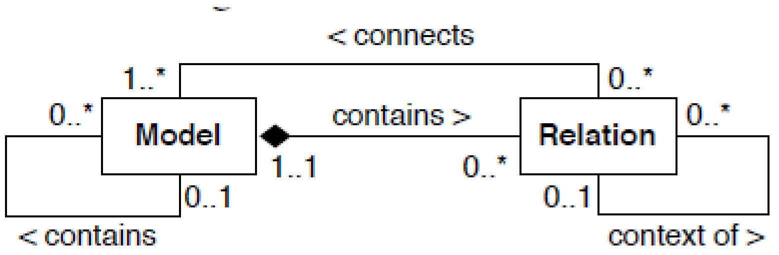
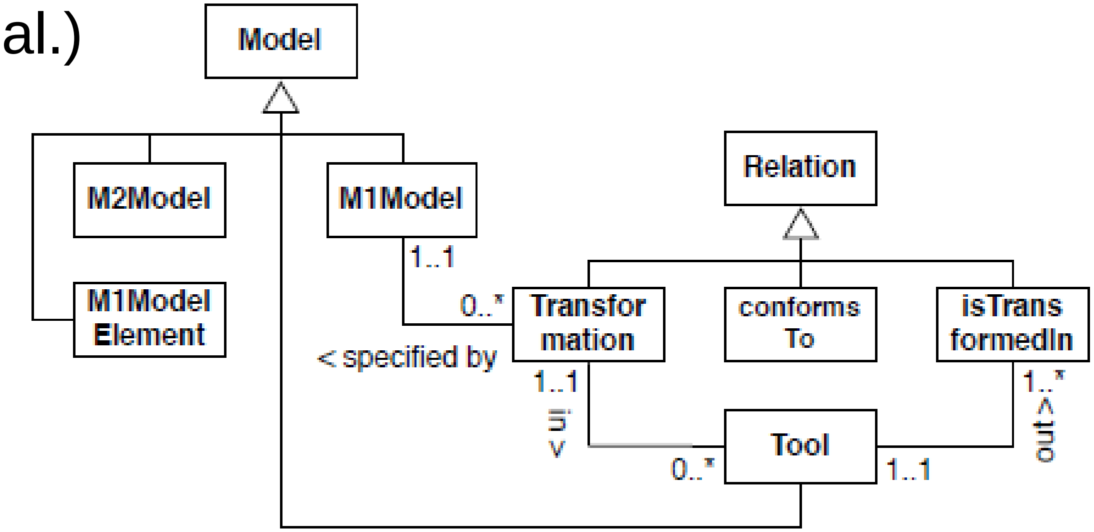
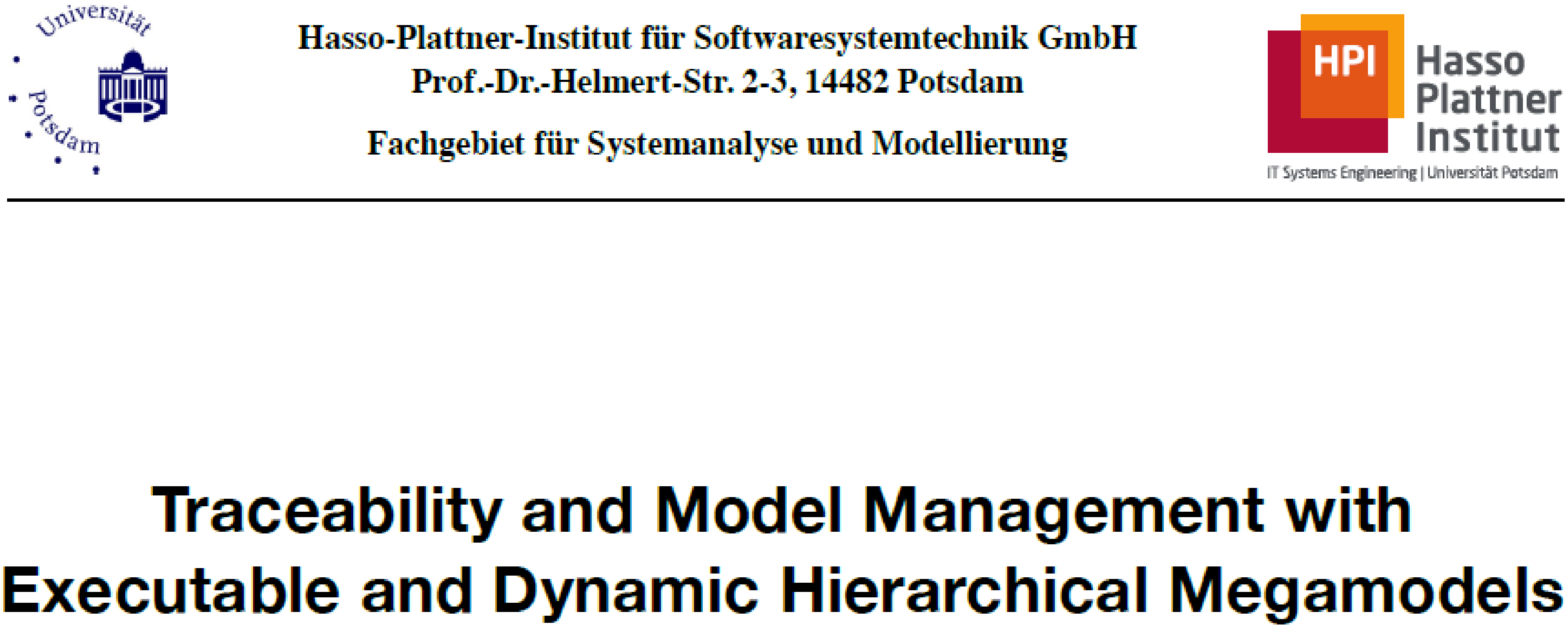


Figure 2: Core metamodel for megamodels



PhD Thesis of Andreas Siebel (2012)
System Analysis and Modeling Group



Universität Potsdam

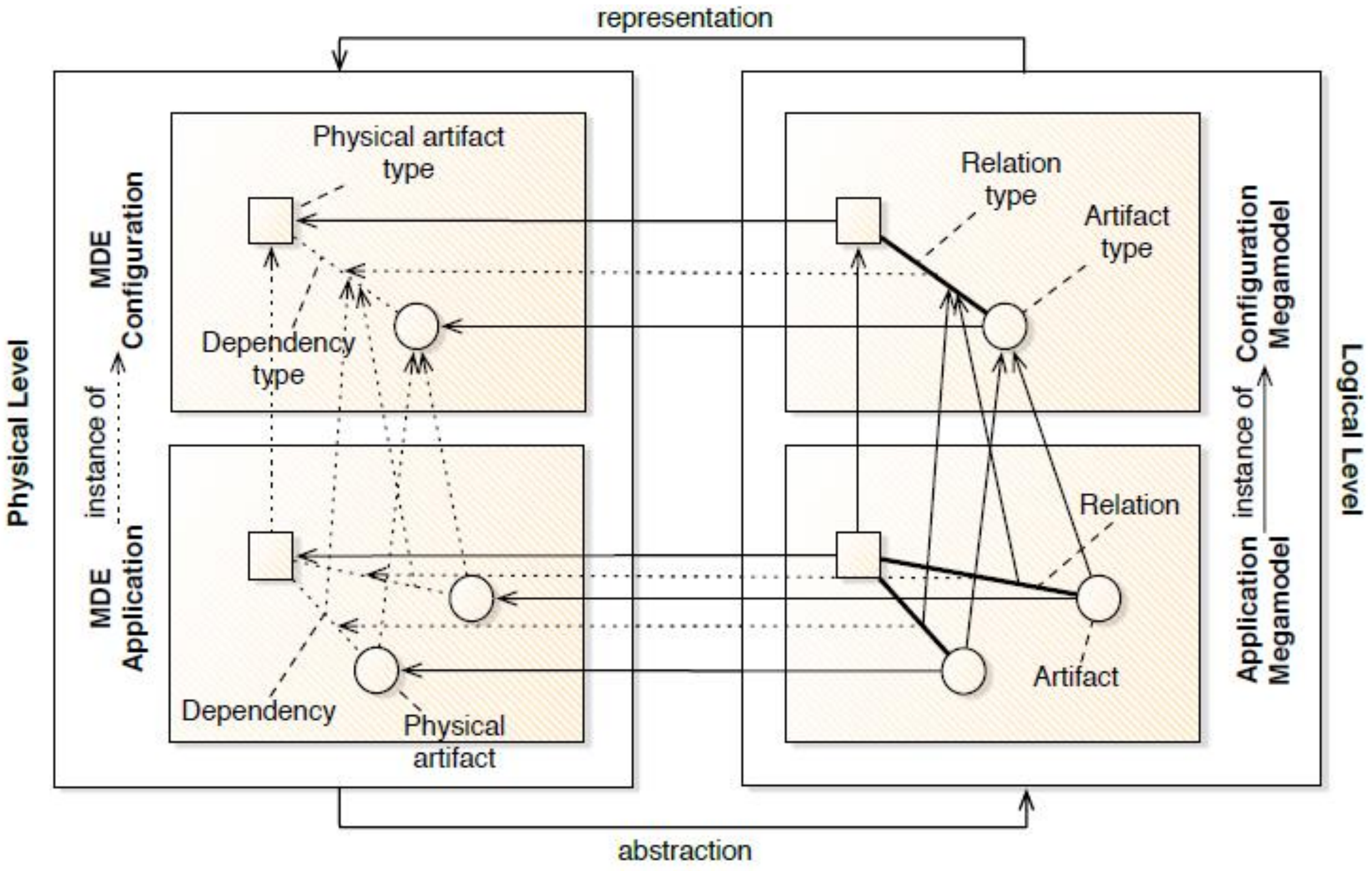
Hasso-Plattner-Institut für Softwaresystemtechnik GmbH
Prof.-Dr.-Helmert-Str. 2-3, 14482 Potsdam

Fachgebiet für Systemanalyse und Modellierung

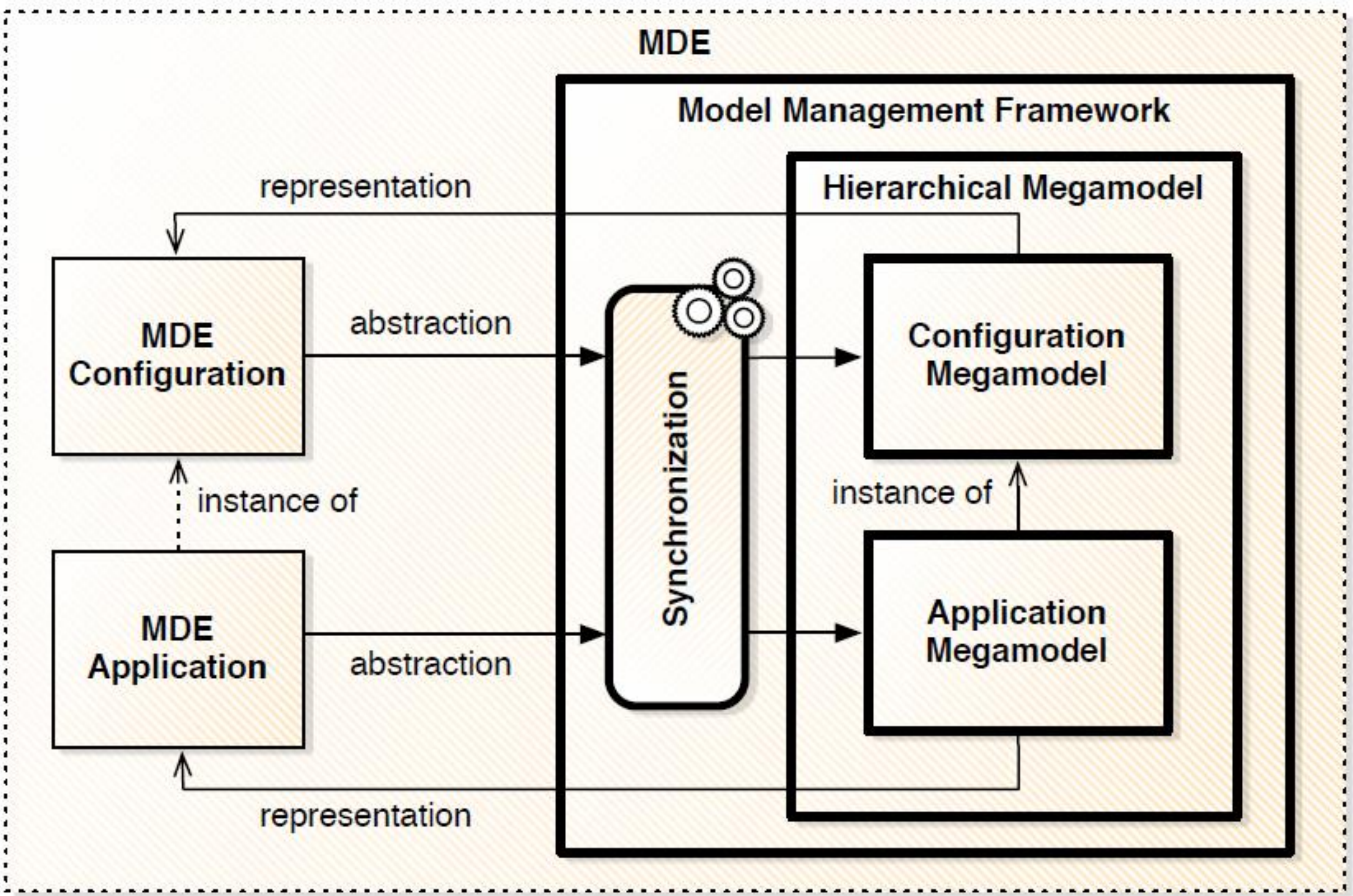
HPI Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

Traceability and Model Management with Executable and Dynamic Hierarchical Megamodels

Physical and Logical Levels in MDE Environment

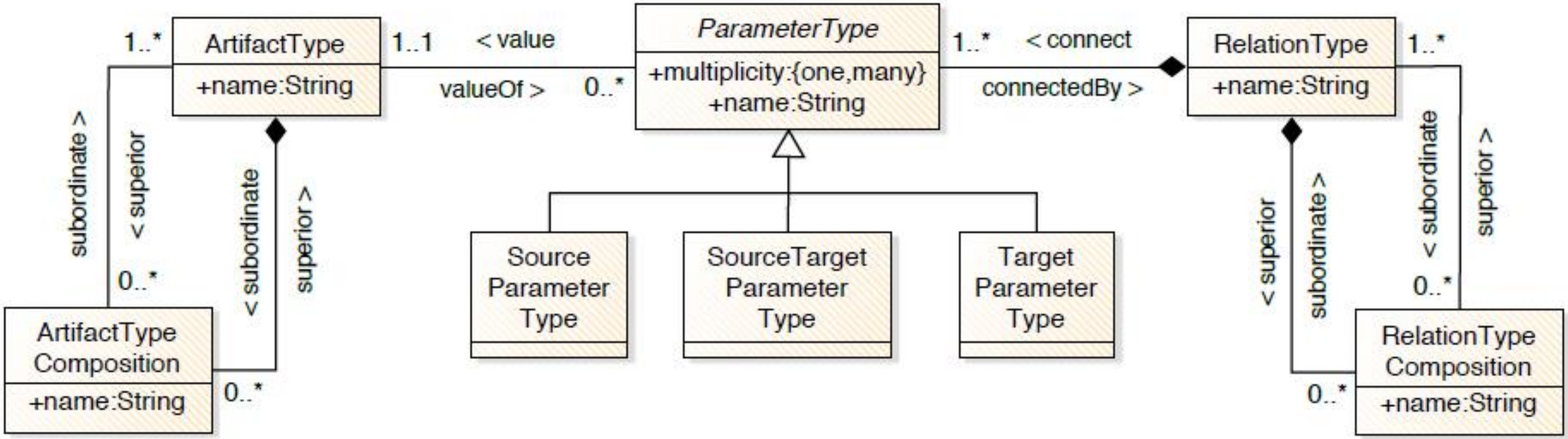


Configuration and Application Megamodels

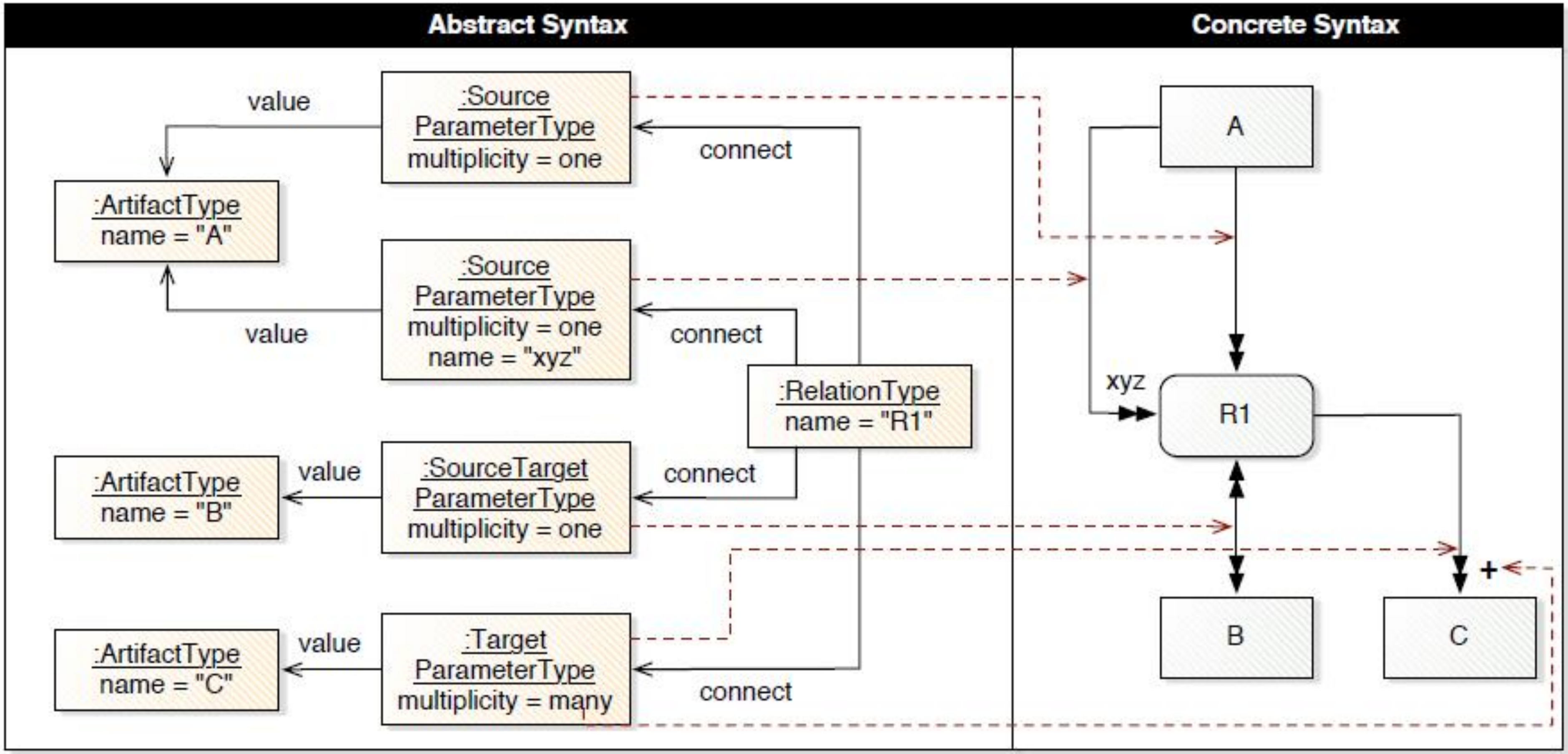


Configuration Megamodel Metamodel

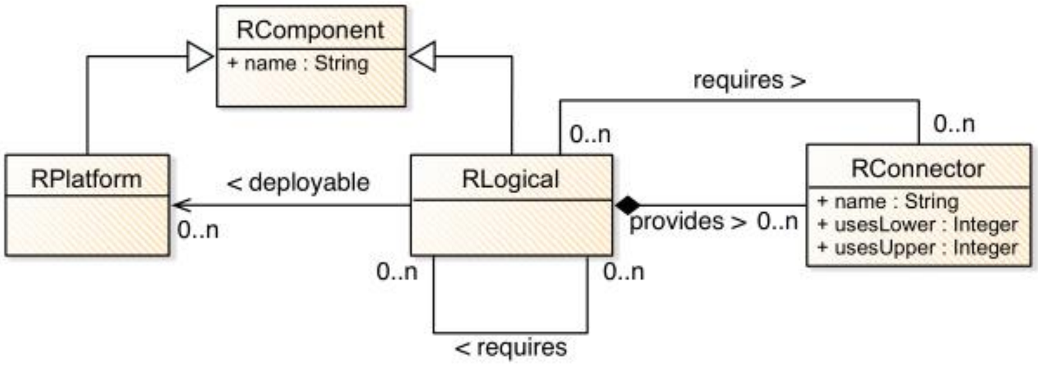
- ArtifactType: abstract representation of a physical artifact type
  - e.g., metamodel or metamodel element
- RelationType: captures and abstractly represents any physical dependency type between physical artifact types
  - n-ary connection between artifact types

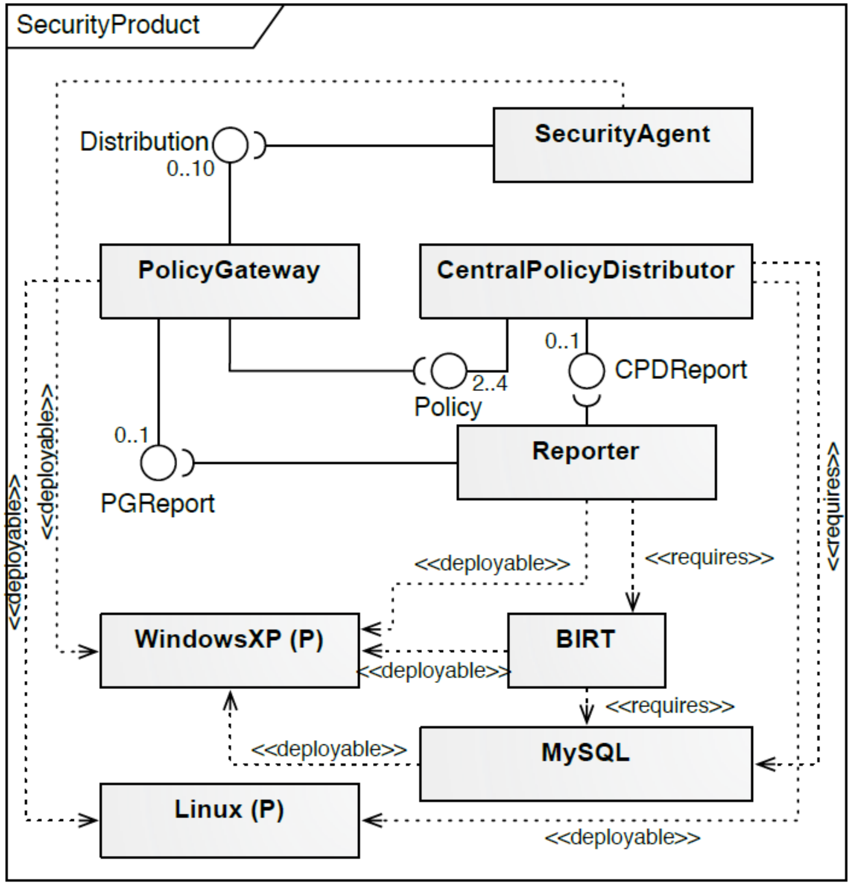


Example Relation Type between Artifact Types

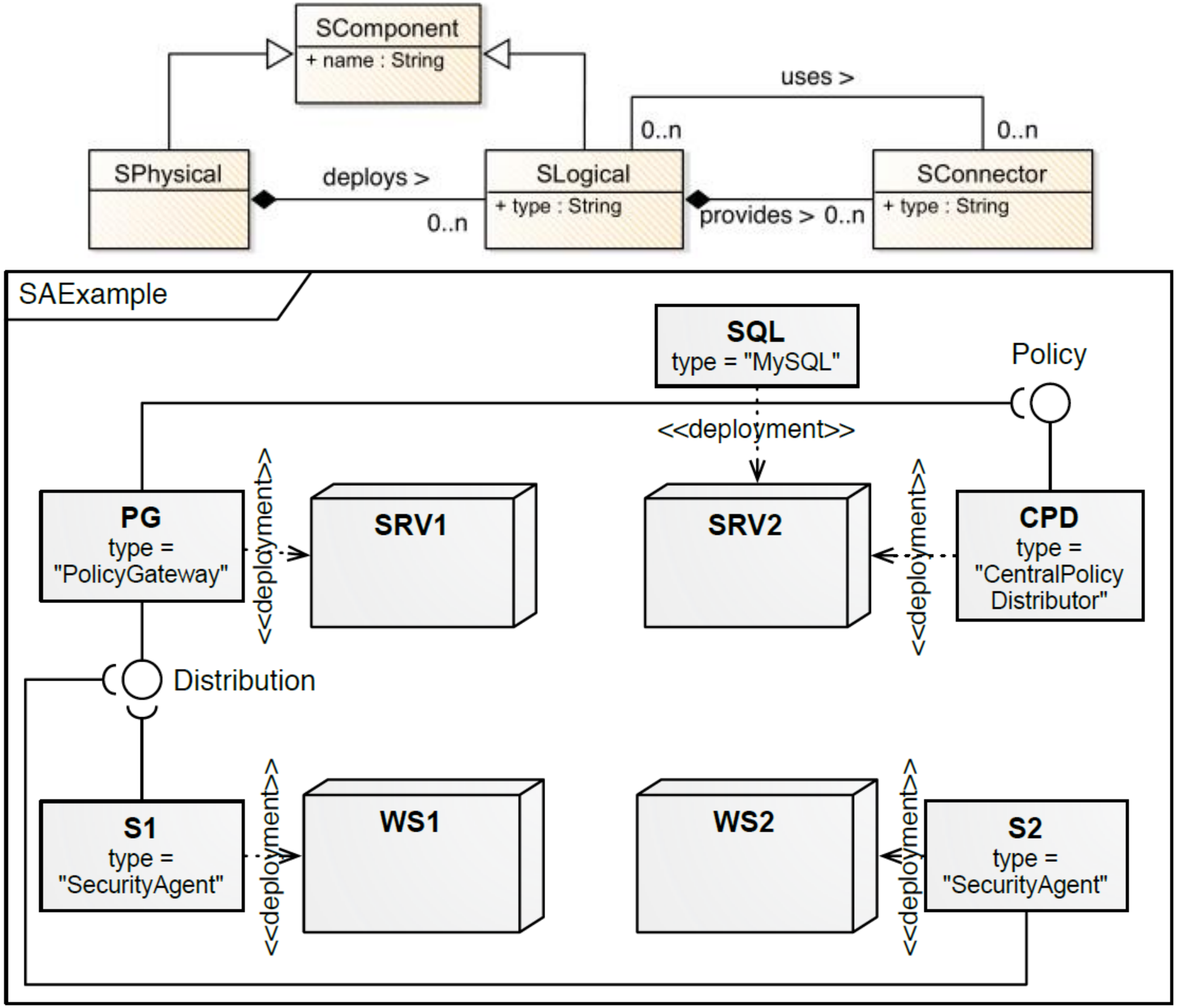


Deployment MDA (D-MDA) Case Study: Reference Architecture

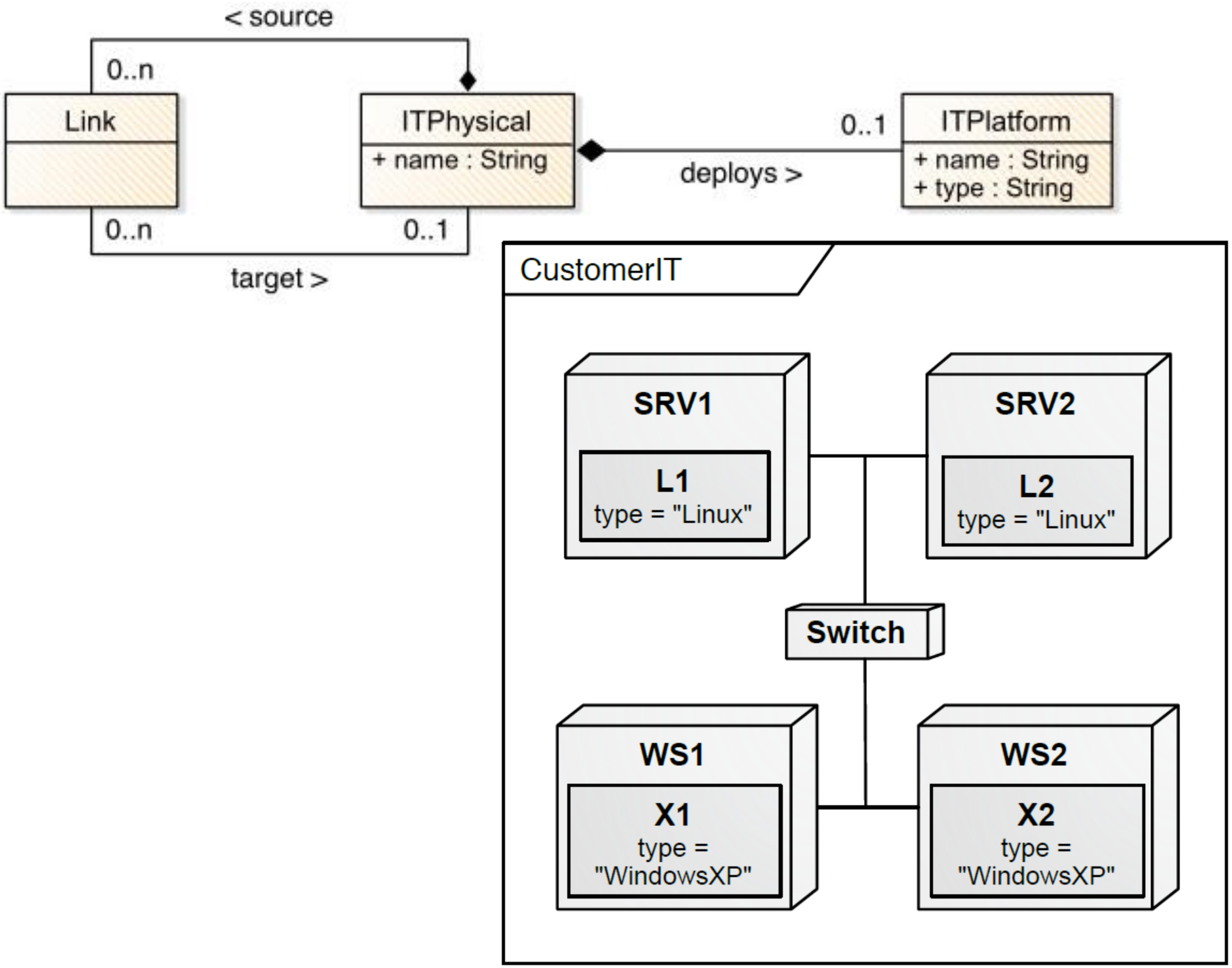




Deployment MDA (D-MDA) Case Study: Solution Architecture

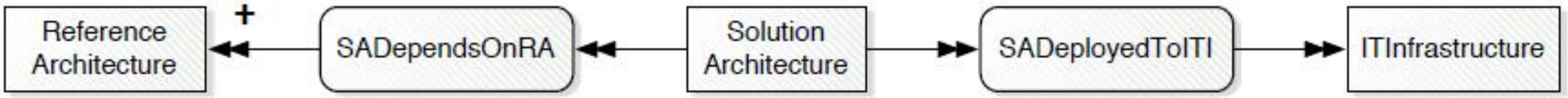


Deployment MDA (D-MDA) Case Study: IT Infrastructure

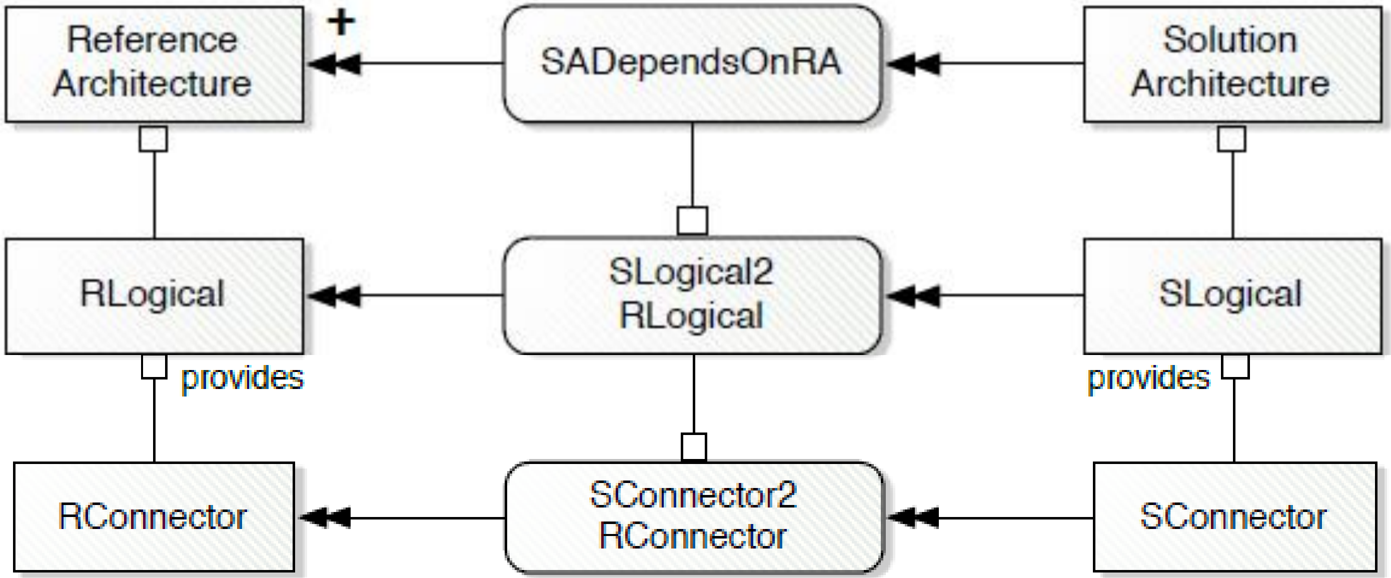


Hierarchical Configuration Megamodel for D-MDA Example

- Relation types between metamodels artifact types

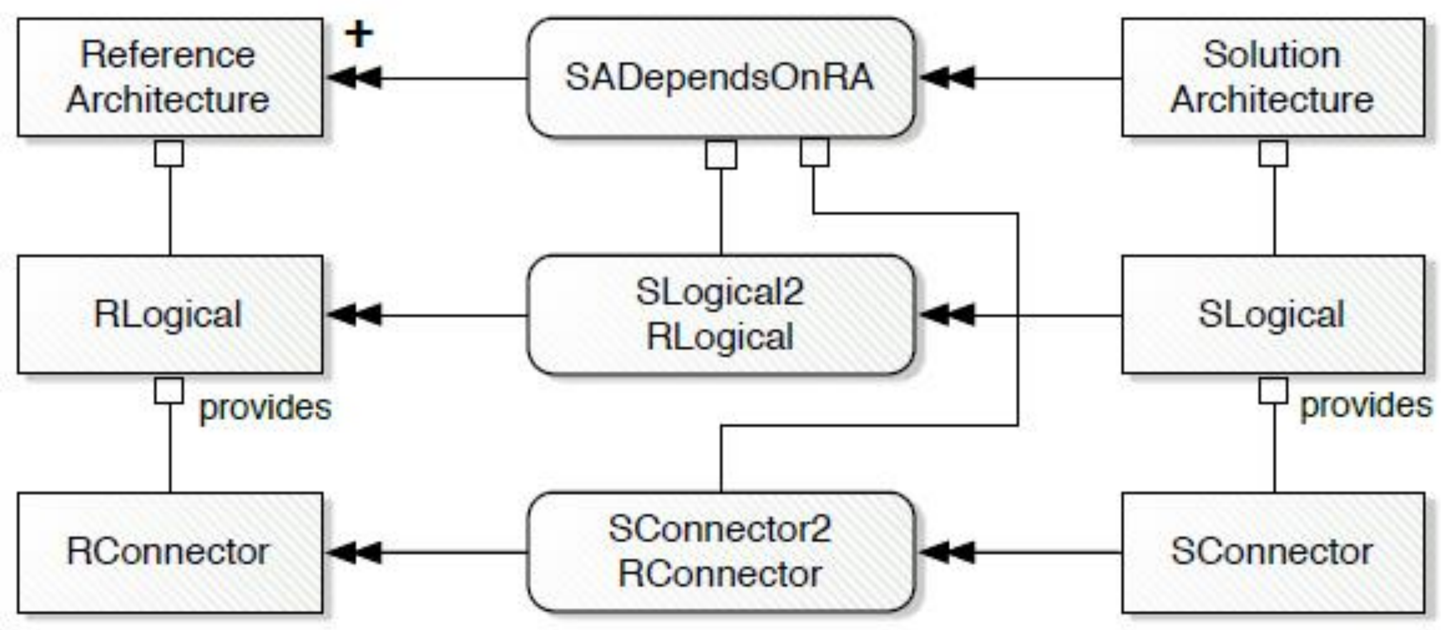


- **Bottom-up** context composition of relation types
  - Lower relation can only exist if upper one does



Hierarchical Configuration Megamodel for D-MDA Example

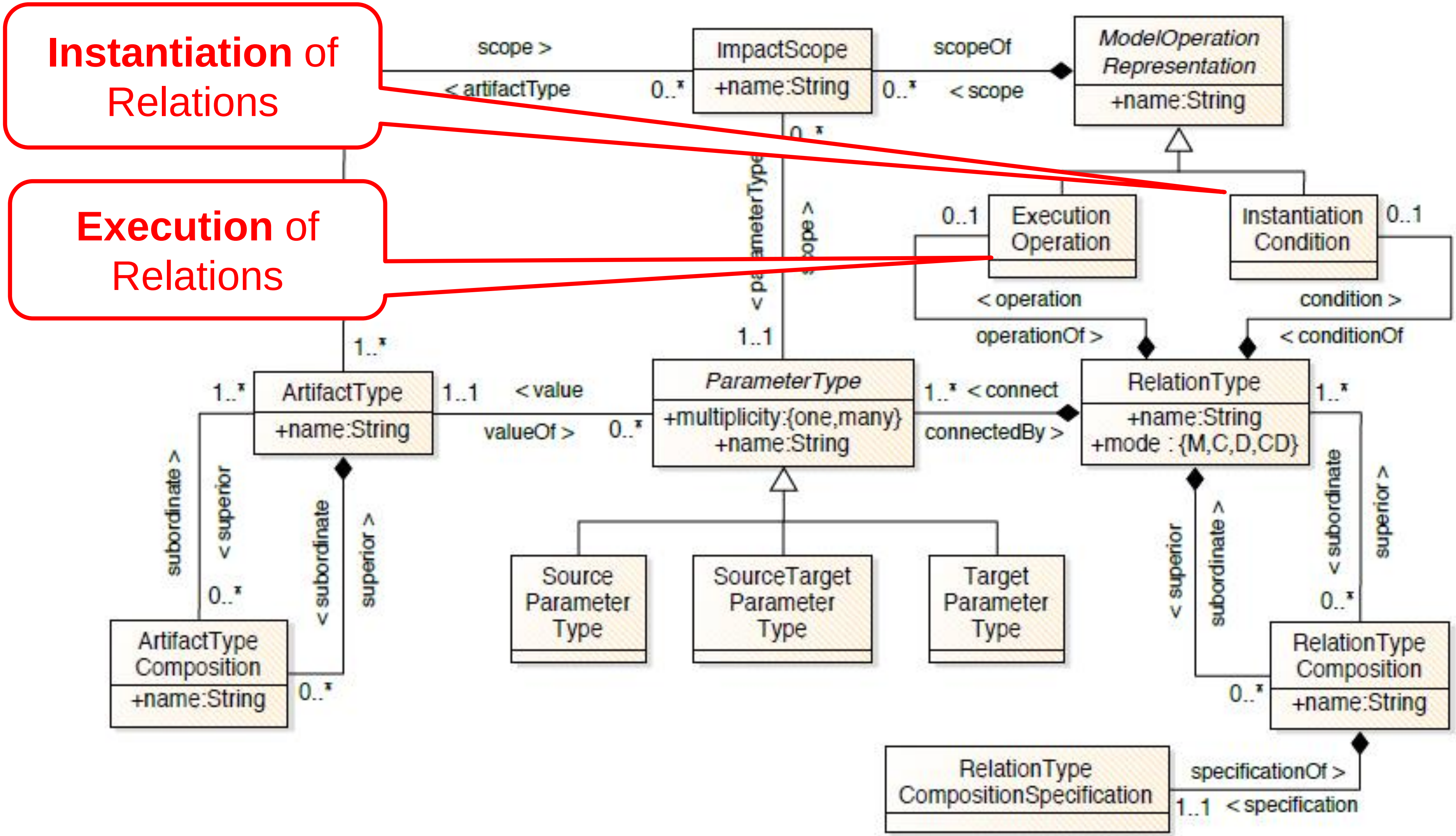
- **Top-down** context composition of relation types
  - Higher relation can only exist if one of the lower ones exist



Execution of Hierarchical Megamodels

- **Purposes:**
  - Maintain traceability
  - Perform model transformations
  - Synchronize models
- **Two execution strategies**
- **Batch:**
  - Relations of the entire megamodel are executed for every change event
- **Incremental:**
  - Only the relations concerned by the changes (and dependencies) are executed

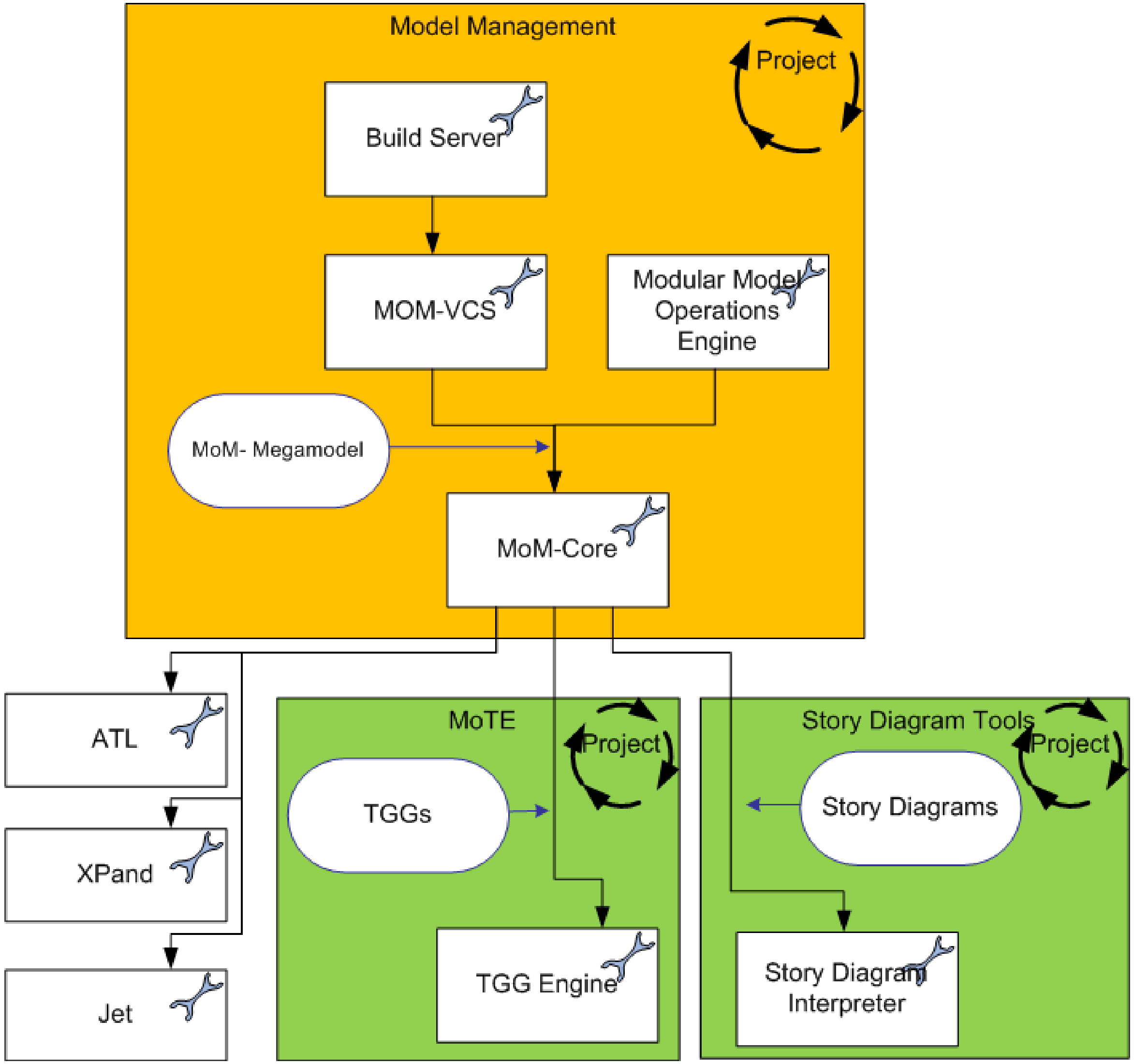
Execution of Hierarchical Megamodels



Instantiation of Relations

Execution of Relations

Tools Adapters



Outline

- Context
- Hierarchical Megamodels
- ACMoM Approach
- Conclusion

ACMoM (Architecture-Centric Model Management)

- **Support ACVIP (Architecture-Centric Virtual Integration Process)**
- **US Army funded project**
  - Ongoing, still a lot to do…
- **Reuse the best of each approaches**
  - Start from HPI approach
  - Add megamodel fragments
  - Workflow (from FTG+PM)

Prototyping and Case Studies

- **Eclipse Modeling Framework**
- **AADL and its tools**
  - OSATE
  - RAMSES
- **Mixed-Criticality Scheduling with the MC-DAG Framework**
- **Model Refinement and Code Generation with RAMSES**
- **AADL ←→ FACE Mapping**

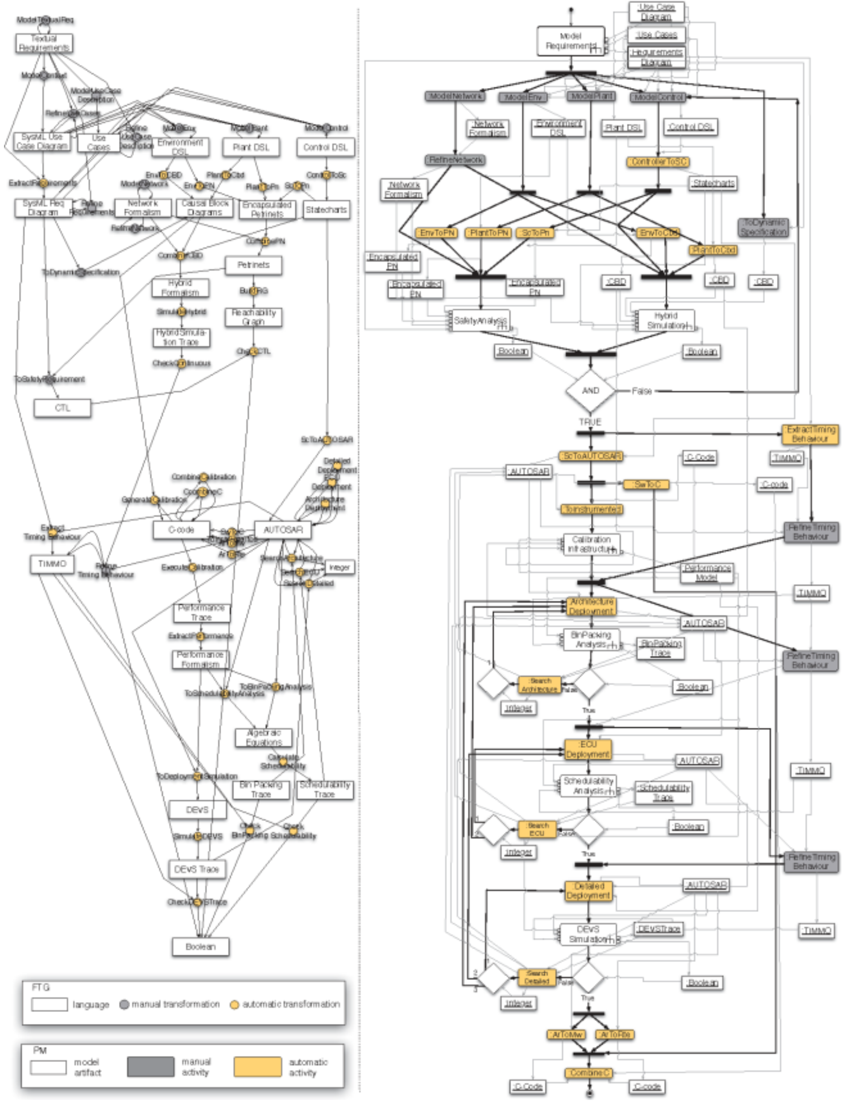
Example: FTG+PM for Power Window

Advantages:

- Includes process

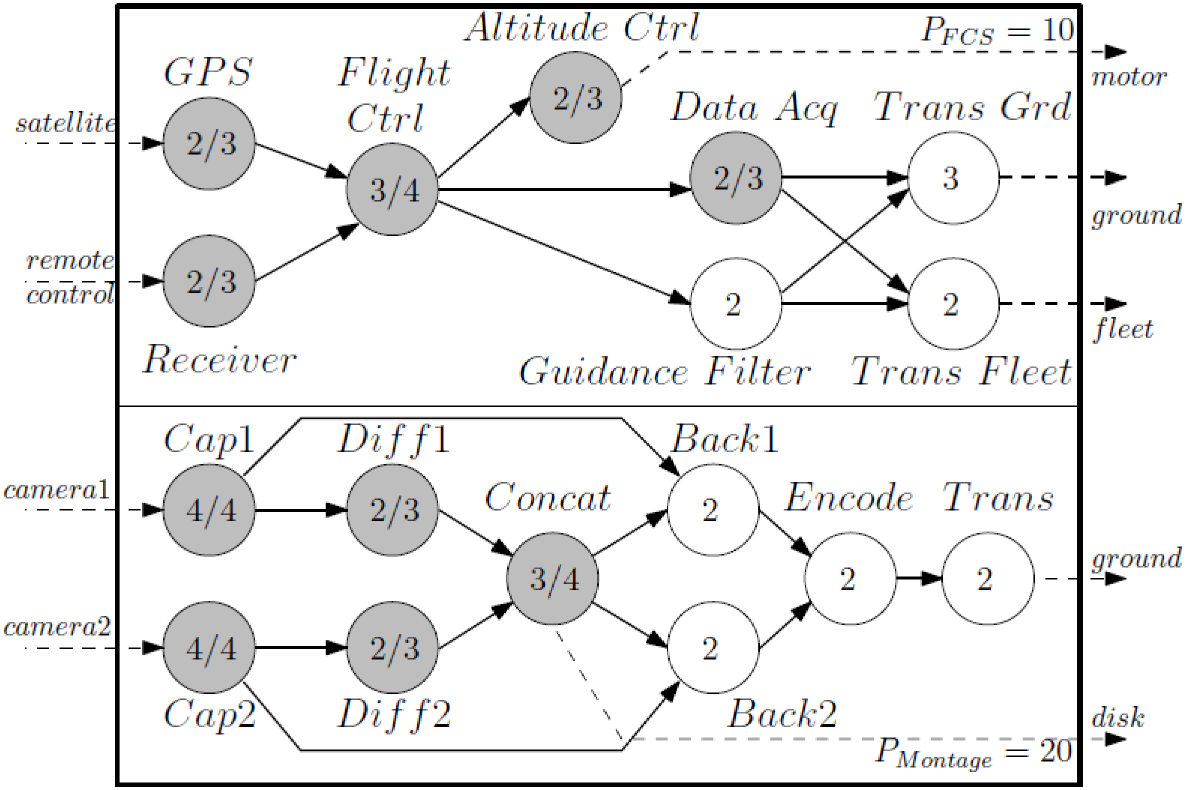
Disadvantages

- Only transformations are modeled
- No hierarchy
- Execution aspect not much developed



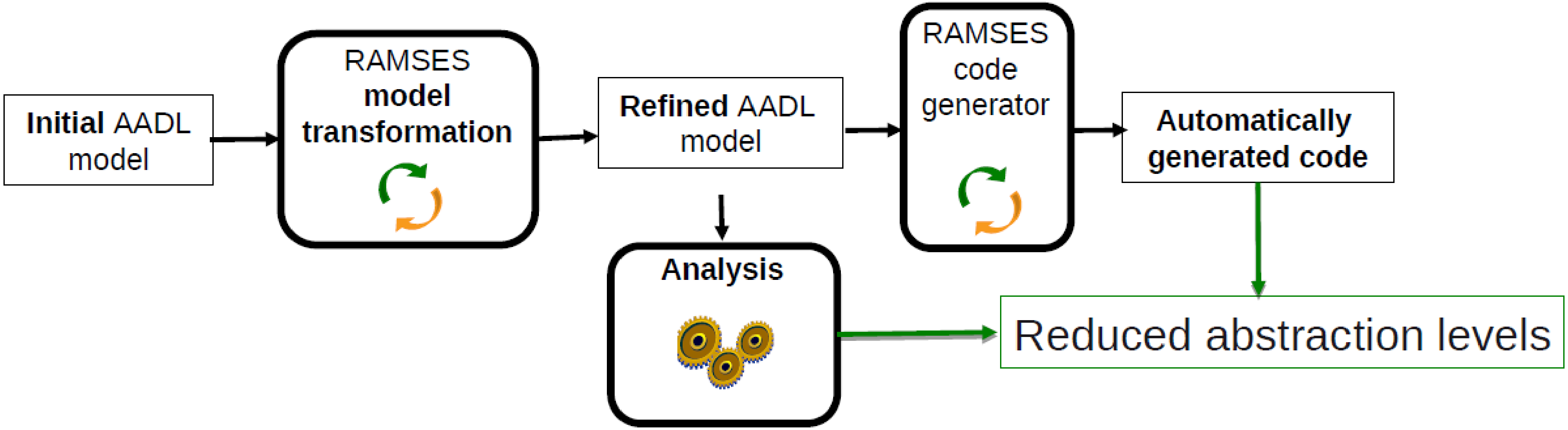
Mixed-Criticality Scheduling with the MC-DAG Framework

- **Horizontal transformation**
- **Bi-directional transformation**
  - Static scheduling properties valued in original model



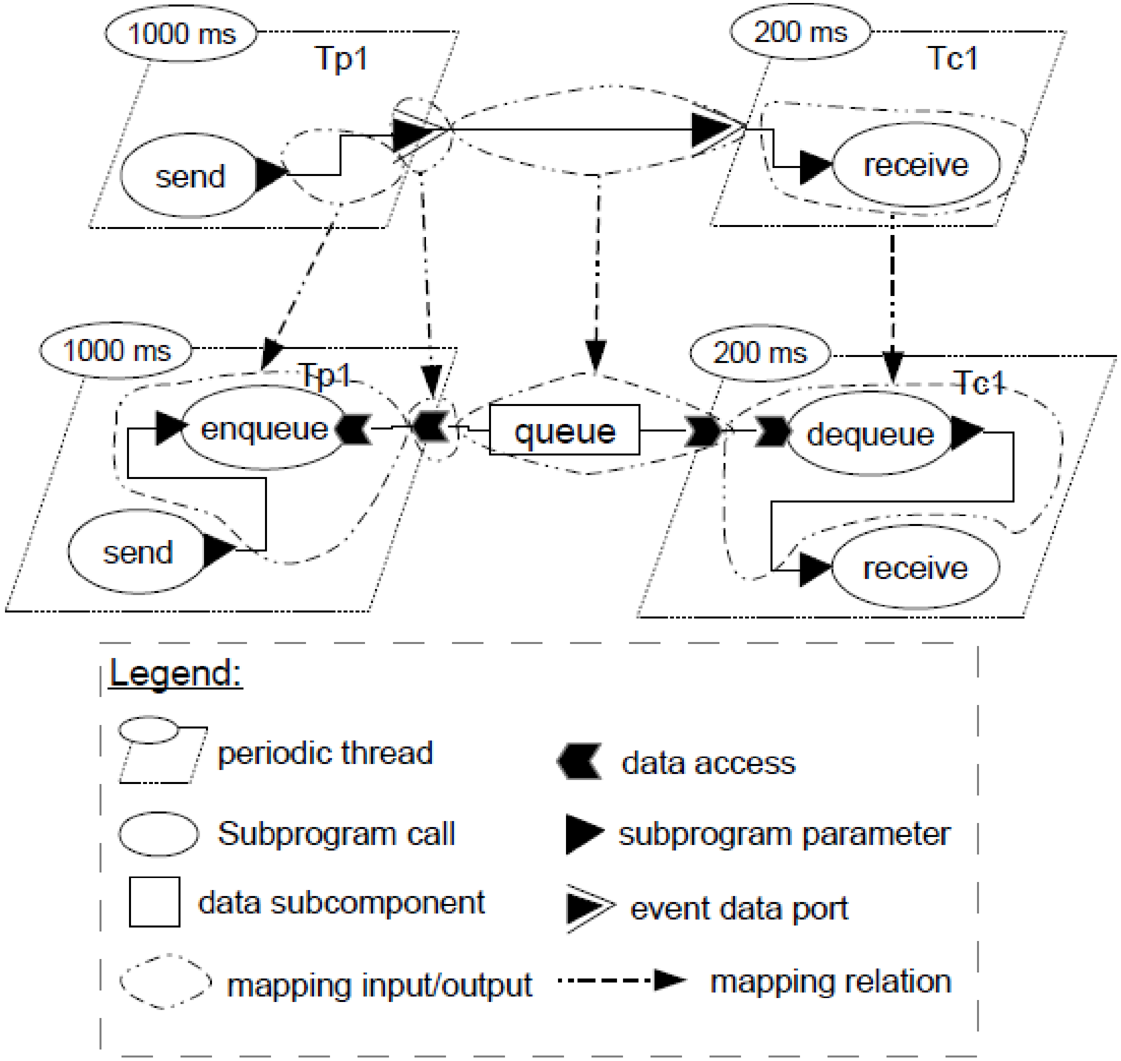
RAMSES: Refinement of AADL Models for the Synthesis of Embedded Systems

- **Vertical transformation**
  - Different levels of abstraction
- **Endogenous transformation**
- **Code generation**
- **Model workflows**



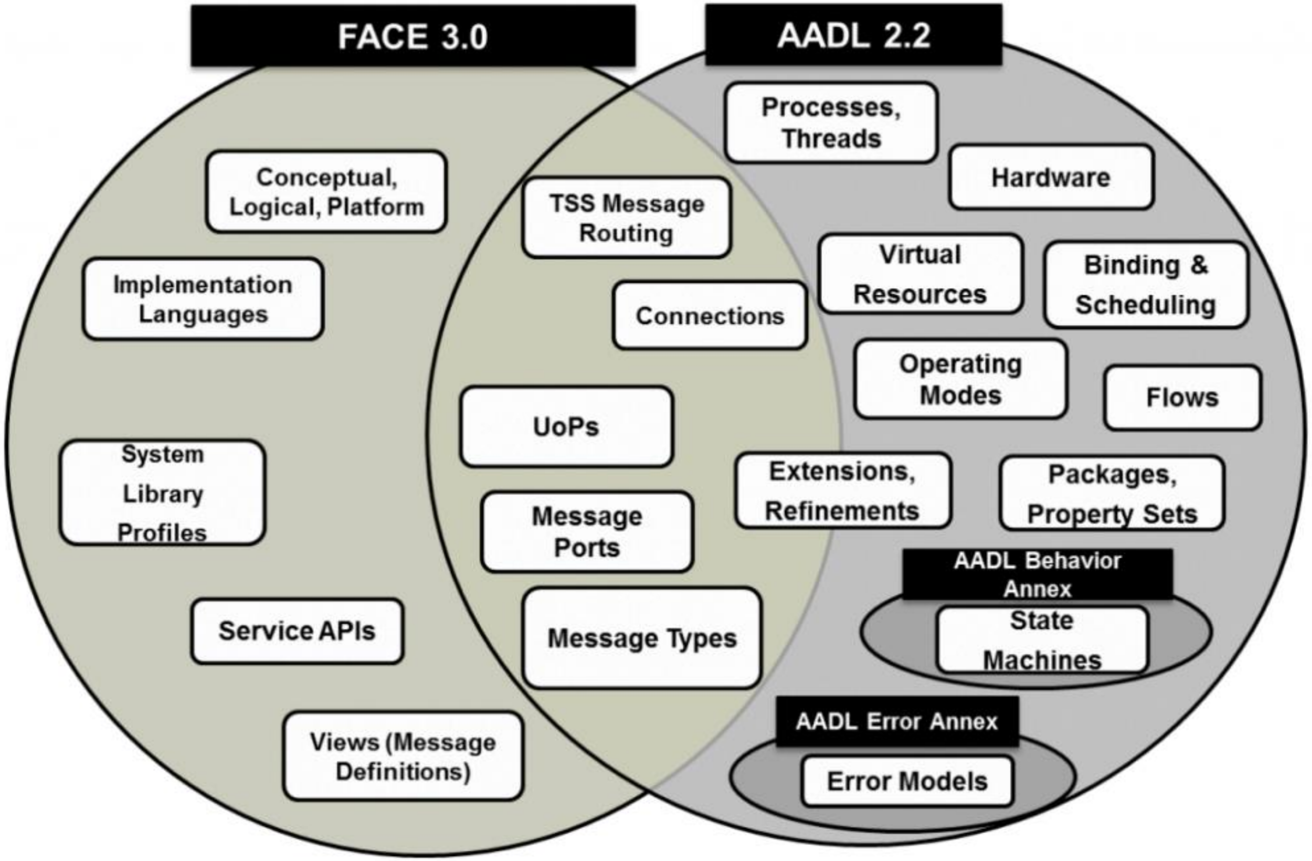
Initial AADL model
RAMSES **model transformation**
Refined AADL model
Analysis
RAMSES code generator
Automatically generated code
Reduced abstraction levels

Example of RAMSES Refinement Rule



AADL ←→ Face Mapping

- **Standardized mapping provided by Adventium Labs**
  - Bi-directional
- **Information overlap but does not coincide**



Conclusion

- **Model management is essential**
  - Several approaches already exist
- **ACMoM**
  - Based on best known approaches
  - Prototyped in Eclipse for the different case studies
  - Ongoing first implementation

- **Future work:**
  - Complete ACMoM prototype
    - Comparison / collaboration with Open Flexo?
  - Model change management for collaborative engineering
    - Authoritative Source of Truth (ASoT)
    - Model synchronization capabilities with respect to information preservation of tools
      - Benchmark started