Certification of Model Transformations

Dániel Varró

1st Workshop on the Analysis of Model Transformations (AMT 2012)

Sharing some challenges of the CERTIMOT project

Development Process for Critical Systems

Unique Development Process (Traditional V-Model)

Critical Systems Design

- requires a certification process
- to develop justified evidence
- that the system is free of flaws

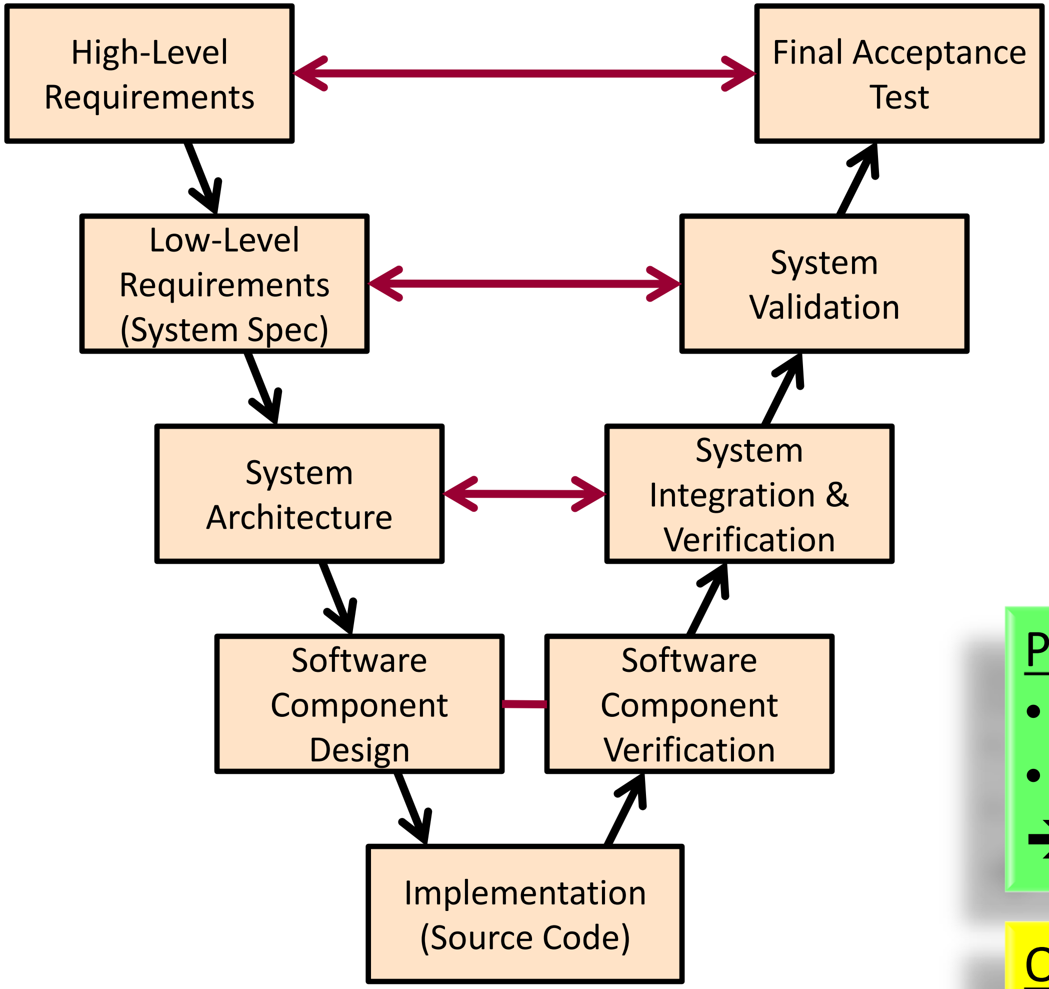
Software Tool Qualification

- obtain certification credit
- for a software tool
- used in critical system design

Innovative Tool → Better System

Qualified Tool → Certified Output

Qualification of Software Tools



High-Level Requirements

Final Acceptance Test

Low-Level Requirements (System Spec)

System Validation

System Architecture

System Integration & Verification

Software Component Design

Software Component Verification

Implementation (Source Code)

Development tools:
- input ➔ output deterministically
- introduce new errors

Verification tools:
- fail to detect errors

Promises of Tool Qualification
- reduce development + V&V cost
- increase quality and productivity

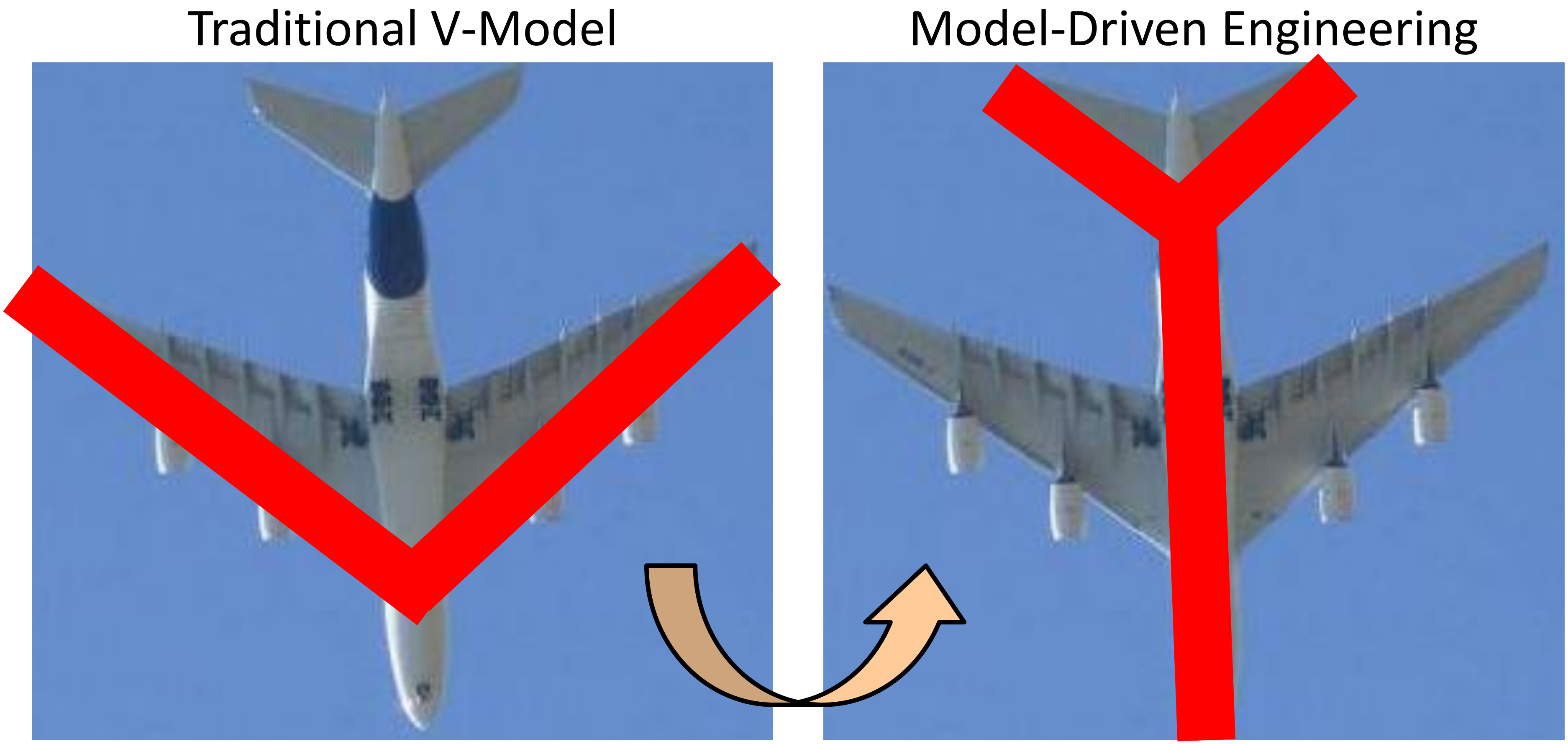
➔ reduce certification costs

Obstacles for Tool Qualification
- reusable features? tool chains?
- complex V&V tasks

➔ extreme qualification costs

A. J. Kornecki, J. Zalewski: The Qualification of Software Development Tools from the DO-178B Perspective, Journal of Defense Software Engineering, Apr, 2006

Model-Driven Engineering of Critical Systems



Traditional V-Model

Model-Driven Engineering

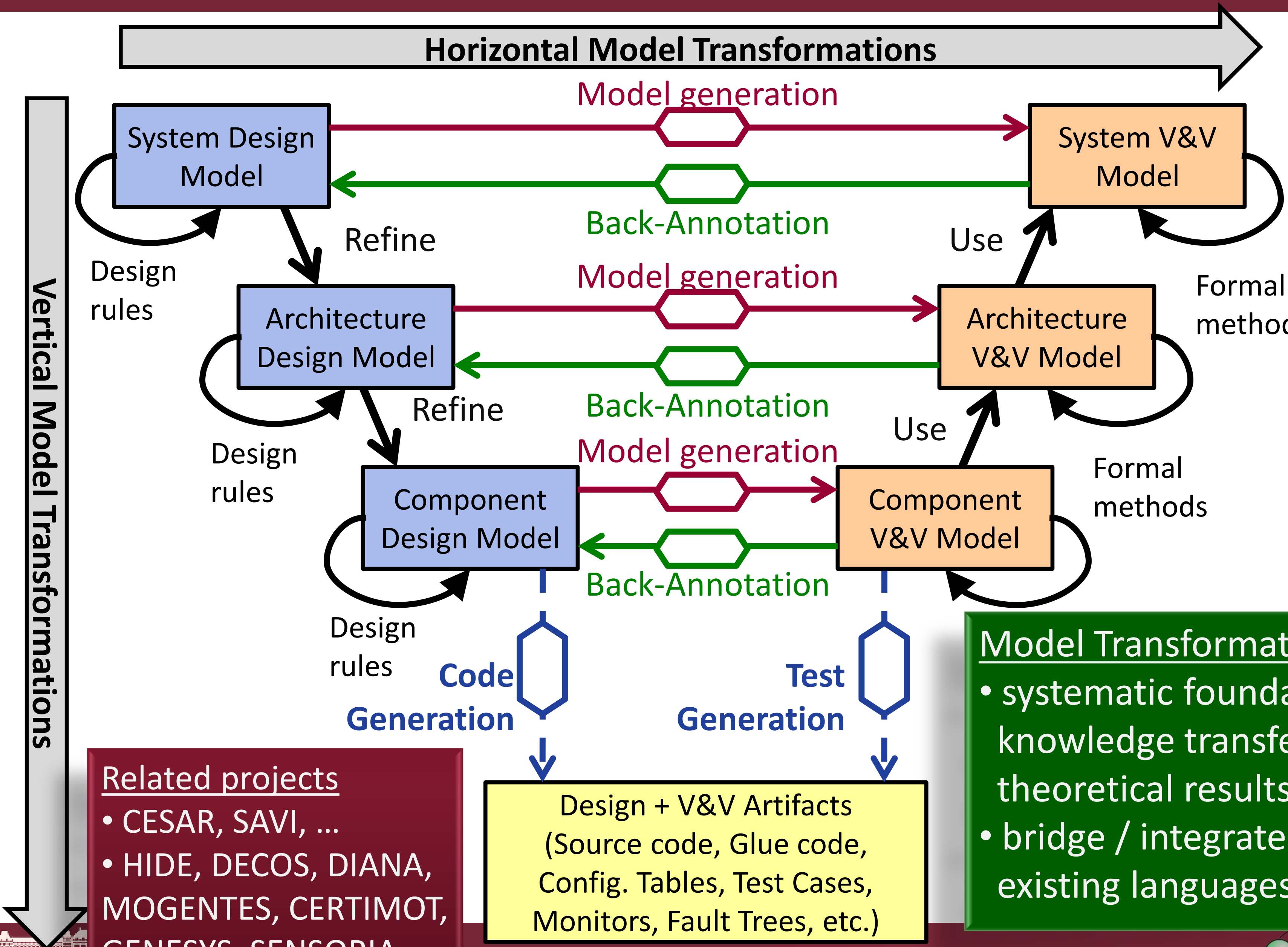
- DO-178B/C: Software Considerations in Airborne Systems and Equipment Certification (RTCA, EUROCAE)
- Steven P. Miller: Certification Issues in Model Based Development (Rockwell Collins)

<u>Main ideas of MDE</u>

- early validation of system models
- automatic source code generation

➔ quality++ tools ++ development cost--

Models and Transformations in Critical Systems



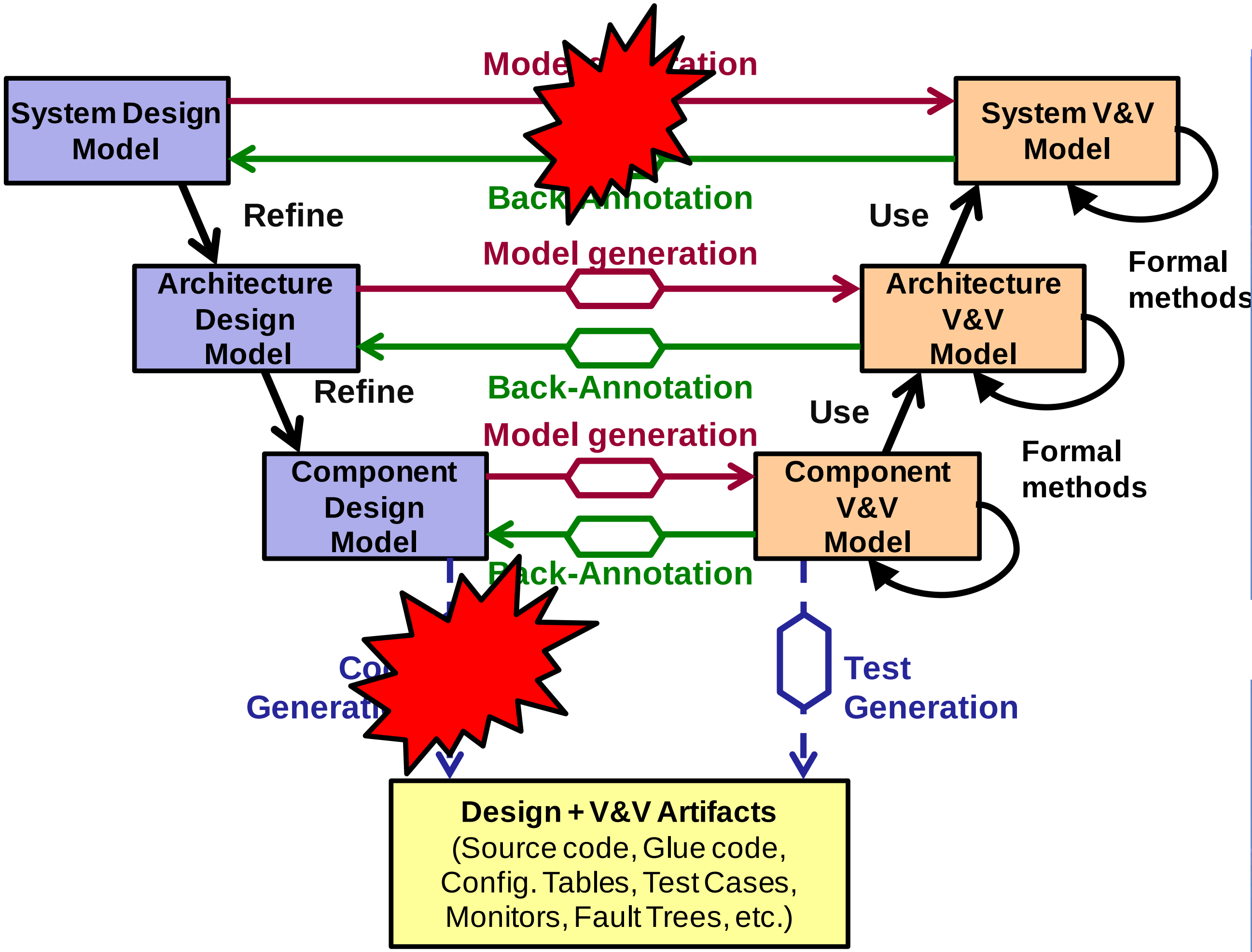
Related projects

- CESAR, SAVI, …
- HIDE, DECOS, DIANA, MOGENTES, CERTIMOT, GENESYS, SENSORIA

Model Transformations

- systematic foundation of knowledge transfer: theoretical results → tools
- bridge / integrate existing languages&tools

Problem: Transformation Errors

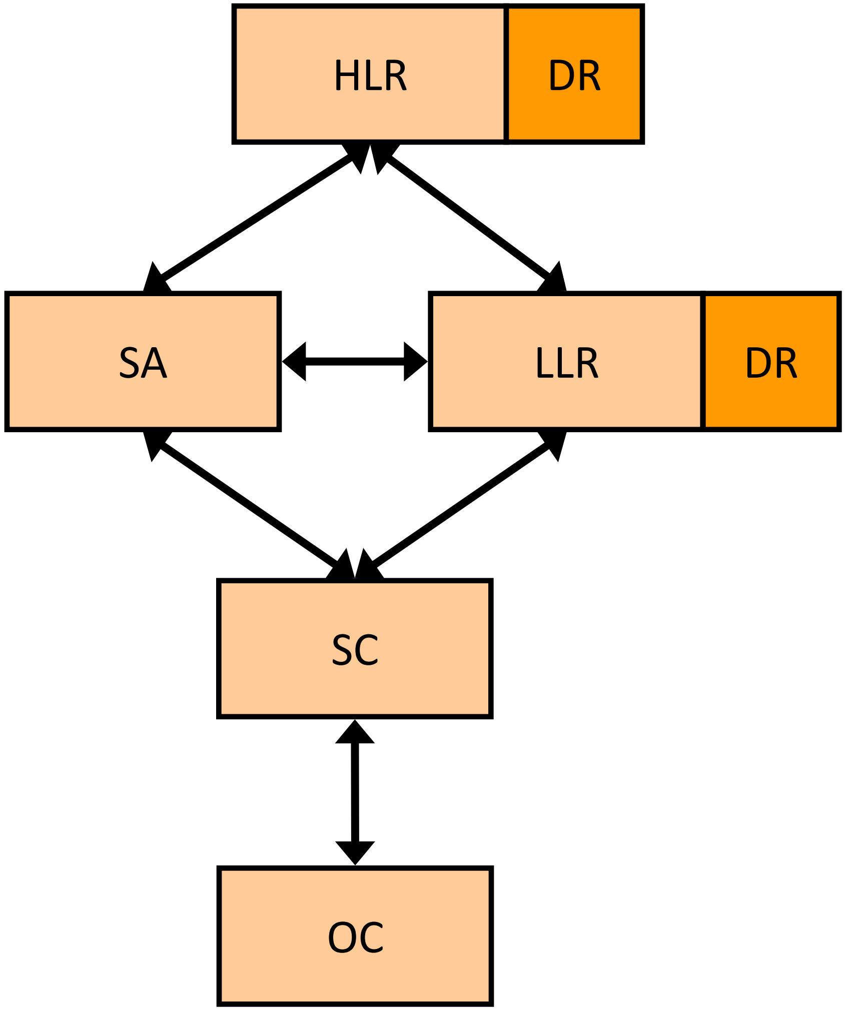


Design + V&V Artifacts
(Source code, Glue code, Config. Tables, Test Cases, Monitors, Fault Trees, etc.)

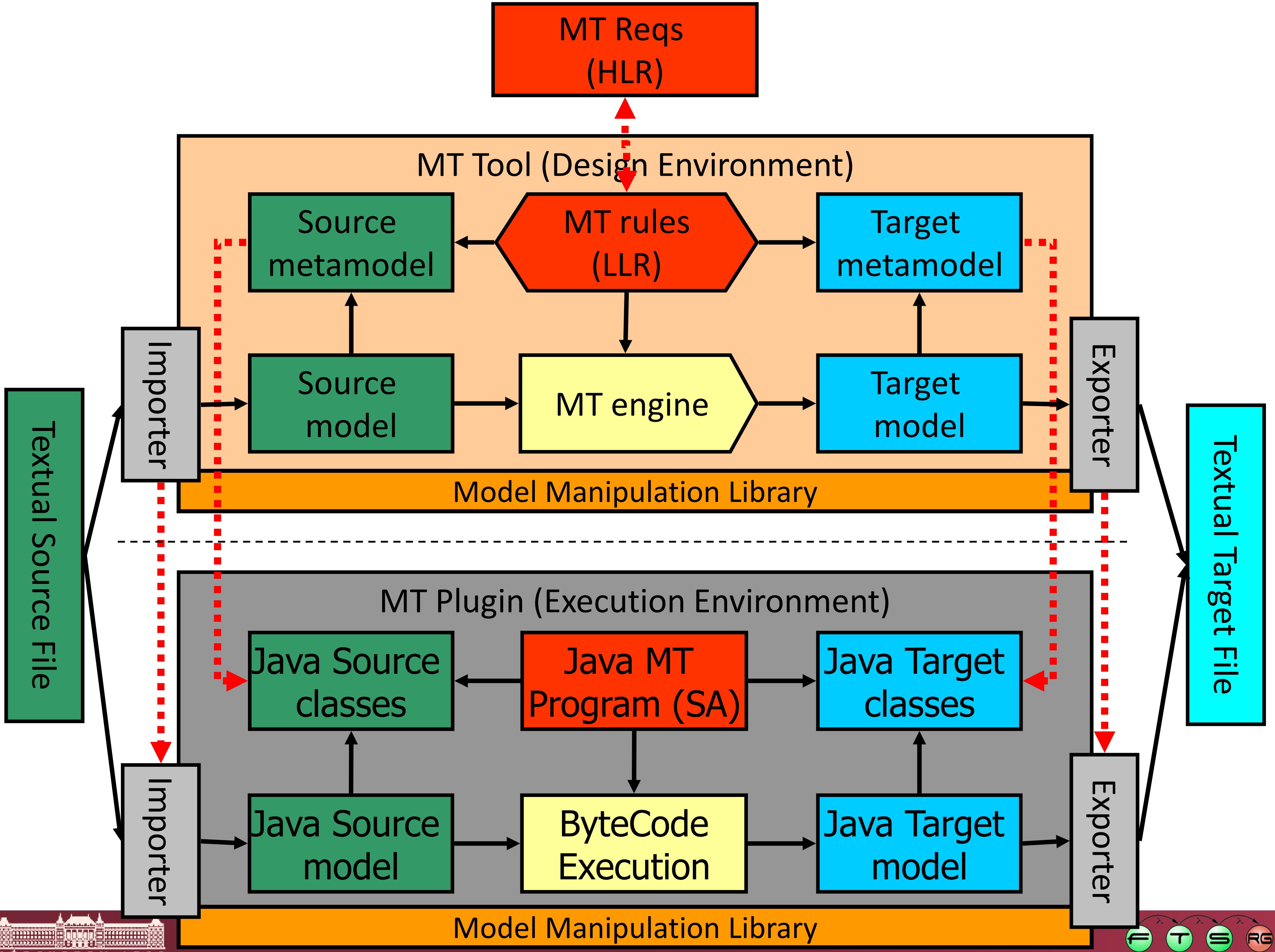
Code generator error
- model: OK, code: no

Model generator error
- model: OK, V&V: No
- model: No, V&V: OK

Main Certification Artifacts



- High Level Requirements (HLR):
  - black-box view of the software,
  - captured in a natural language (e.g. using shall statements)
- Derived Requirements (DR)
  - Capture design decisions
- Low Level Requirements (LLR):
  - SC can be implemented without further information
- Software Architecture (SA)
  - Interfaces, information flow of SW components
- Source Code (SC)
  - Code written in a source language
- Executable Object Code (EOC)
  - Obtained by traditional compilers


MT Reqs
(HLR)
MT Tool (Design Environment)
Source metamodel
MT rules
(LLR)
Target metamodel
Importer
Source model
MT engine
Target model
Exporter
Model Manipulation Library
Textual Source File
Textual Target File
MT Plugin (Execution Environment)
Java Source classes
Java MT Program (SA)
Java Target classes
Importer
Java Source model
ByteCode Execution
Java Target model
Exporter
Model Manipulation Library