

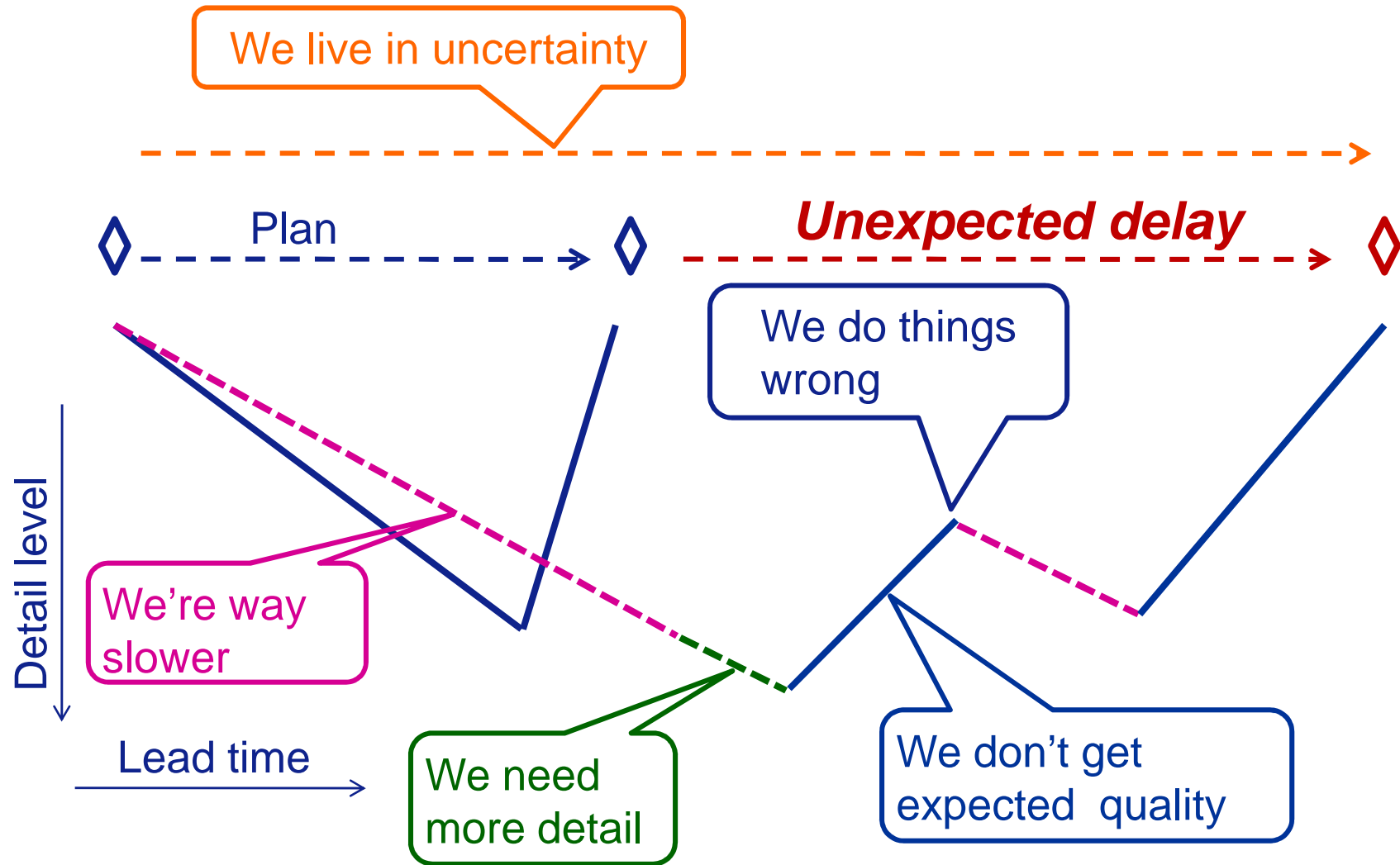


ASML

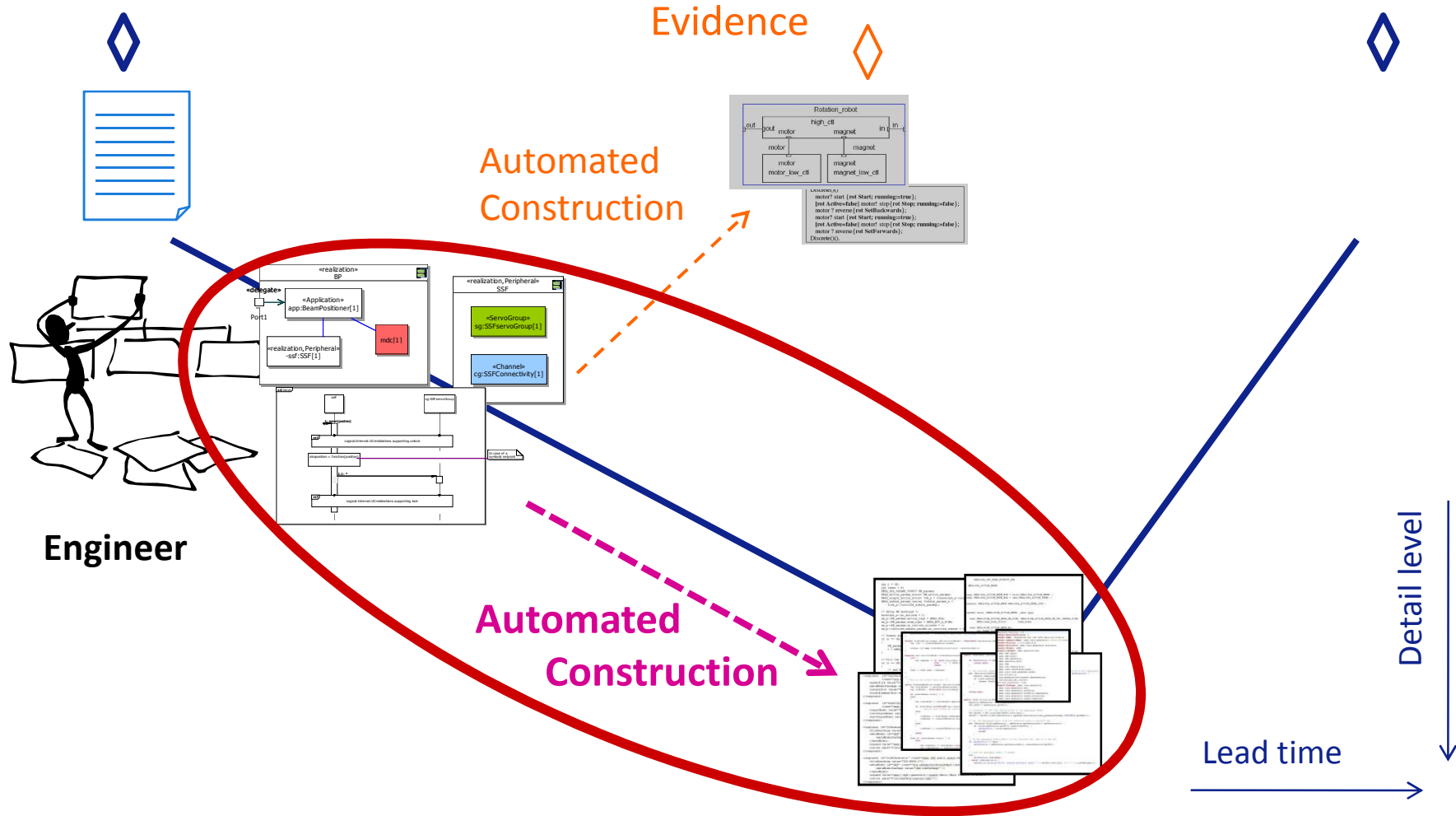
Using a DSL to boost construction efficiency

Practical experiences with a software constructor as
part of the MDE approach @ ASML
KWR09124

The V-model: Current practice



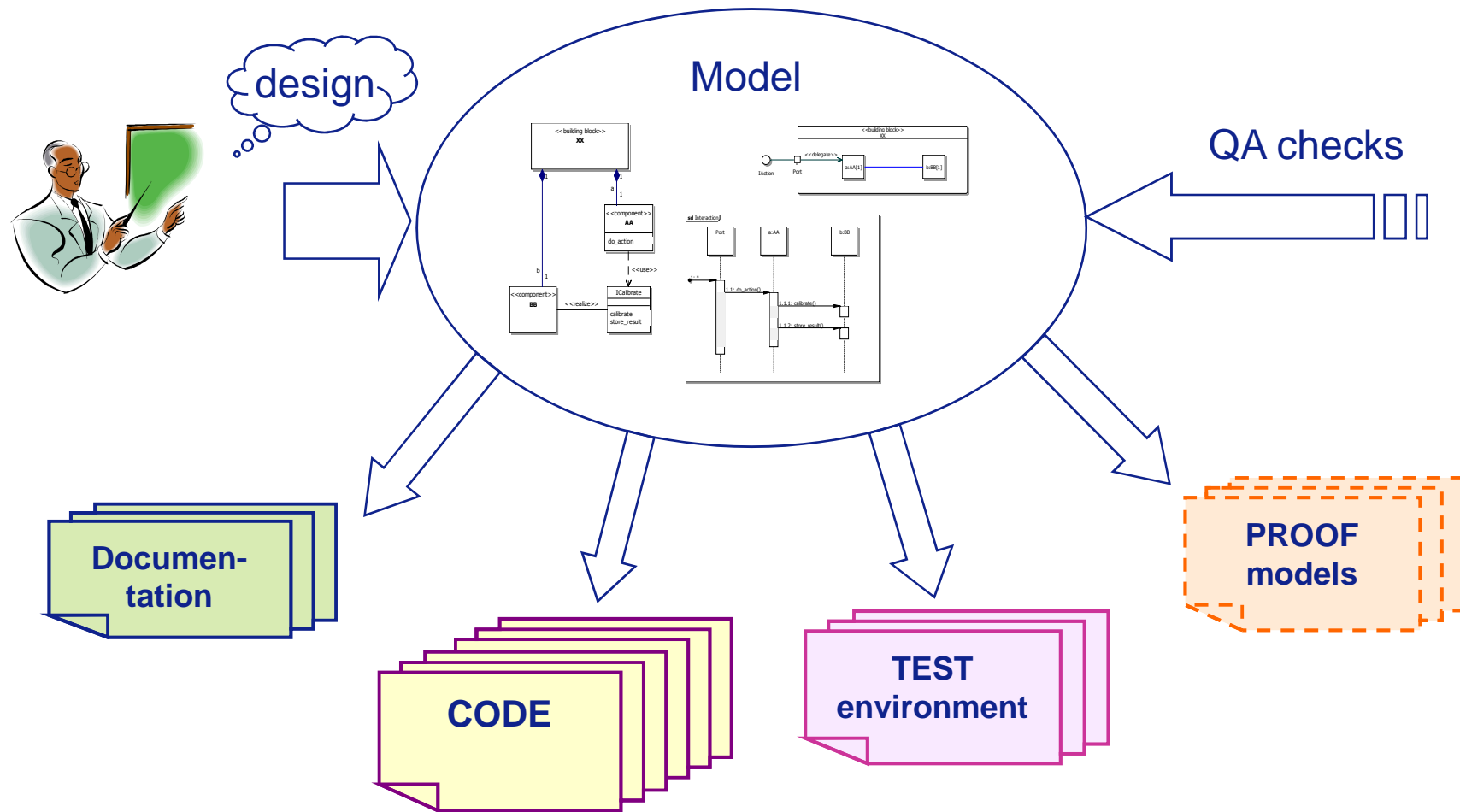
Context in the MDE approach



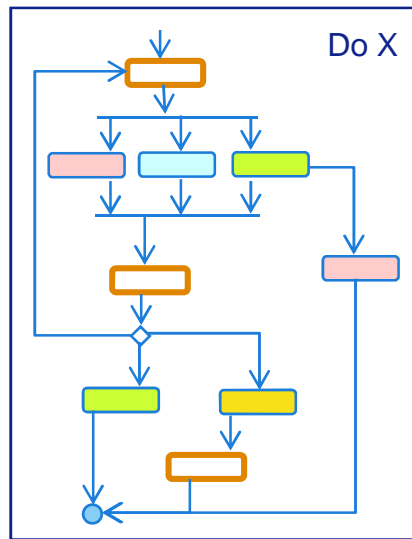
Constructor goals

- Reduce gap between requirements and implementation
 - Raise abstraction of design language
- Automate what can be defined by rules
 - Remove boring laborious work
 - Improve quality
 - Contain knowledge in transformations
 - Reduce “Accidental Complexity”
- Optimize environment for the engineer
 - Emphasis on construction
- Produce executable system a.s.a.p.
- Single source for all derived artifacts

Single source: Capture design, generate artifacts

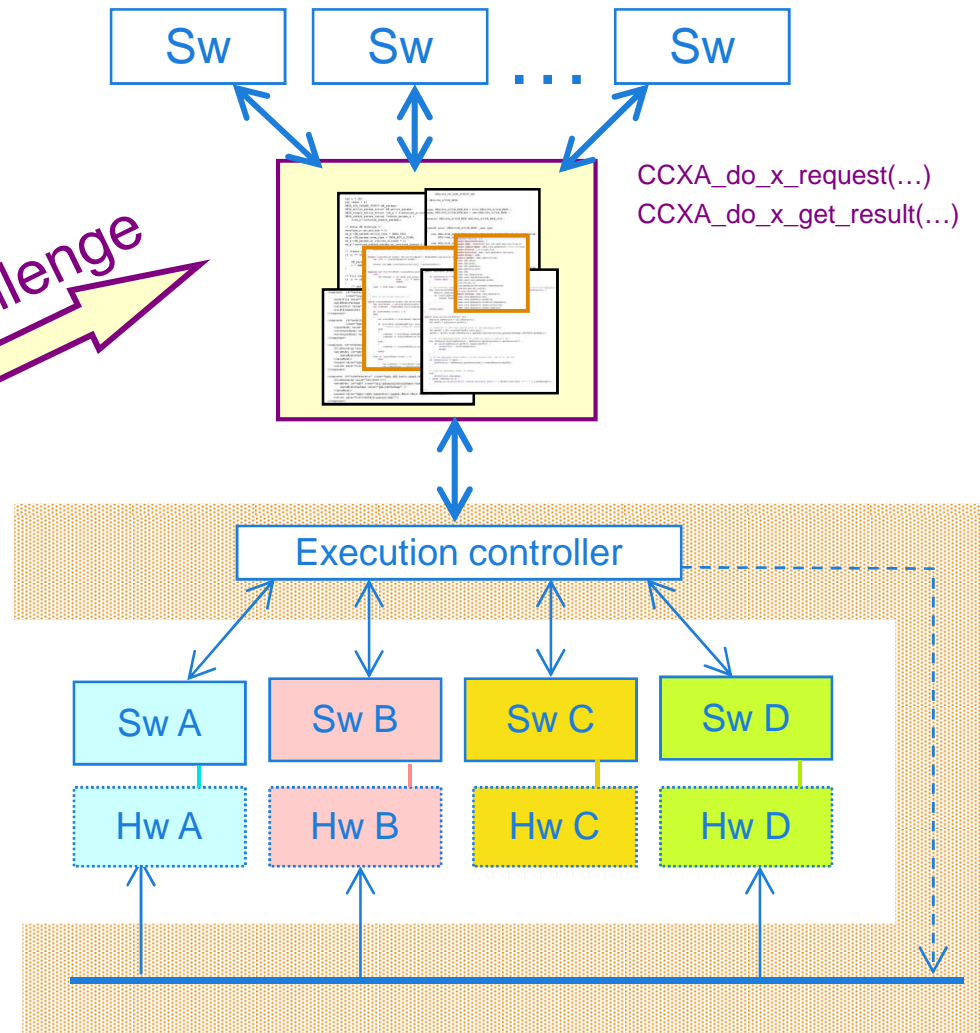


The LACE Case: Challenge



Requirement Specification

Design challenge



Logical Action: Code perspective

Requirements

Scan

Before/after

Parallel

...

Coding

Prepare/Calculate physical/logical

Counter, Wait for result, Passives

Envelope, Pack/Unpack

Set/get data, Cleanup

Action id, result available/requested

Queue, nr of entries, context

Events, Errors, Tracing

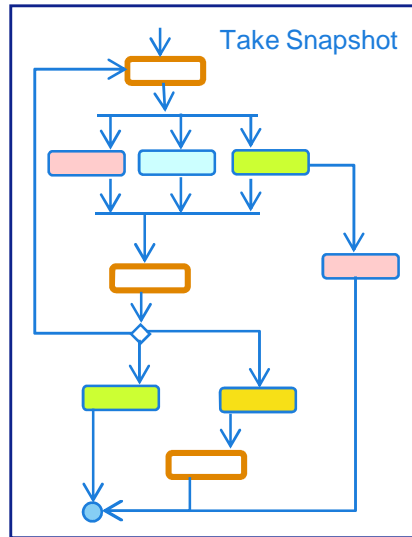
Initialise/Terminate/Status

...

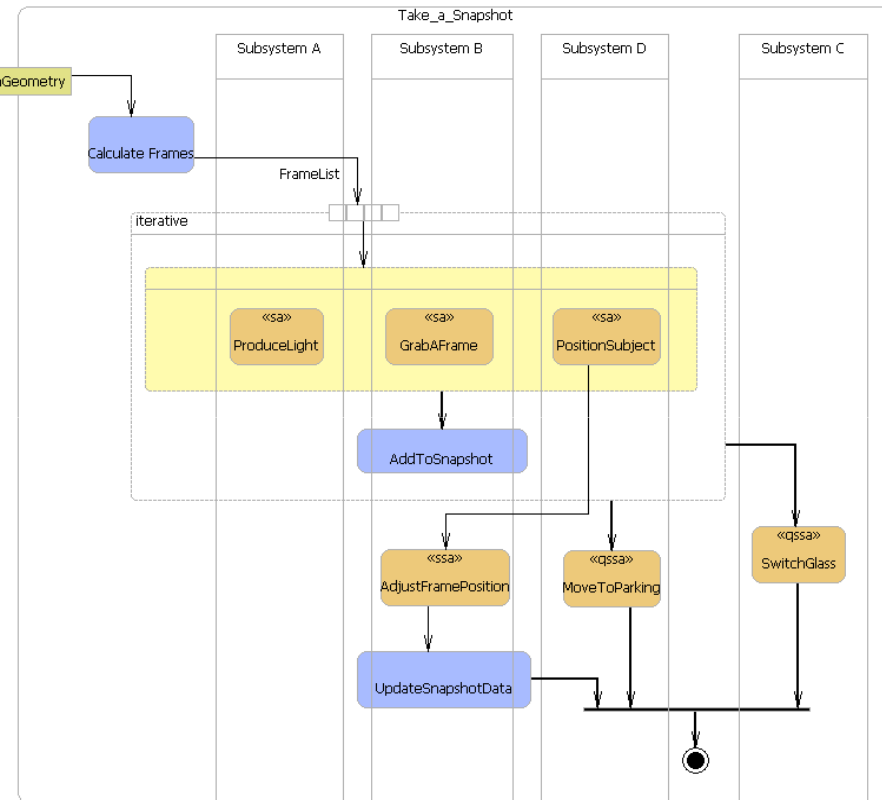
The LACE Case: Problems

- Translating concepts
 - Specification defines resulting system behavior
 - We construct a system that programs an execution controller such that it coordinates the software drivers such that....
- Error prone
 - Even when using a specialized wrapper
 - Unknown or misunderstood data
- Labor intensive
 - Typically, one behavior involves adjusting 24 code modules
- Testing is difficult
 - Requires specific knowledge to setup testing infrastructure

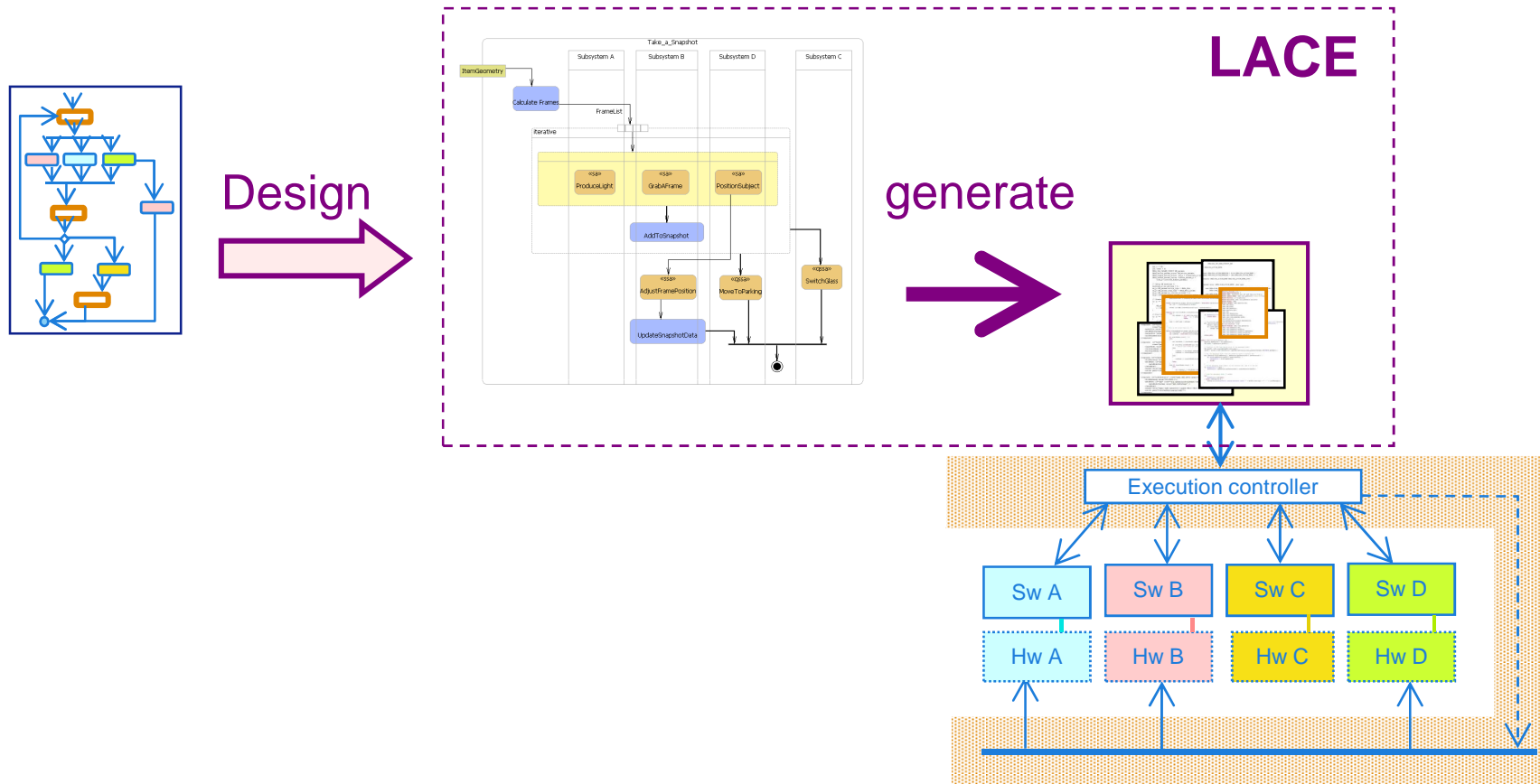
Reduce gap between requirements and design...



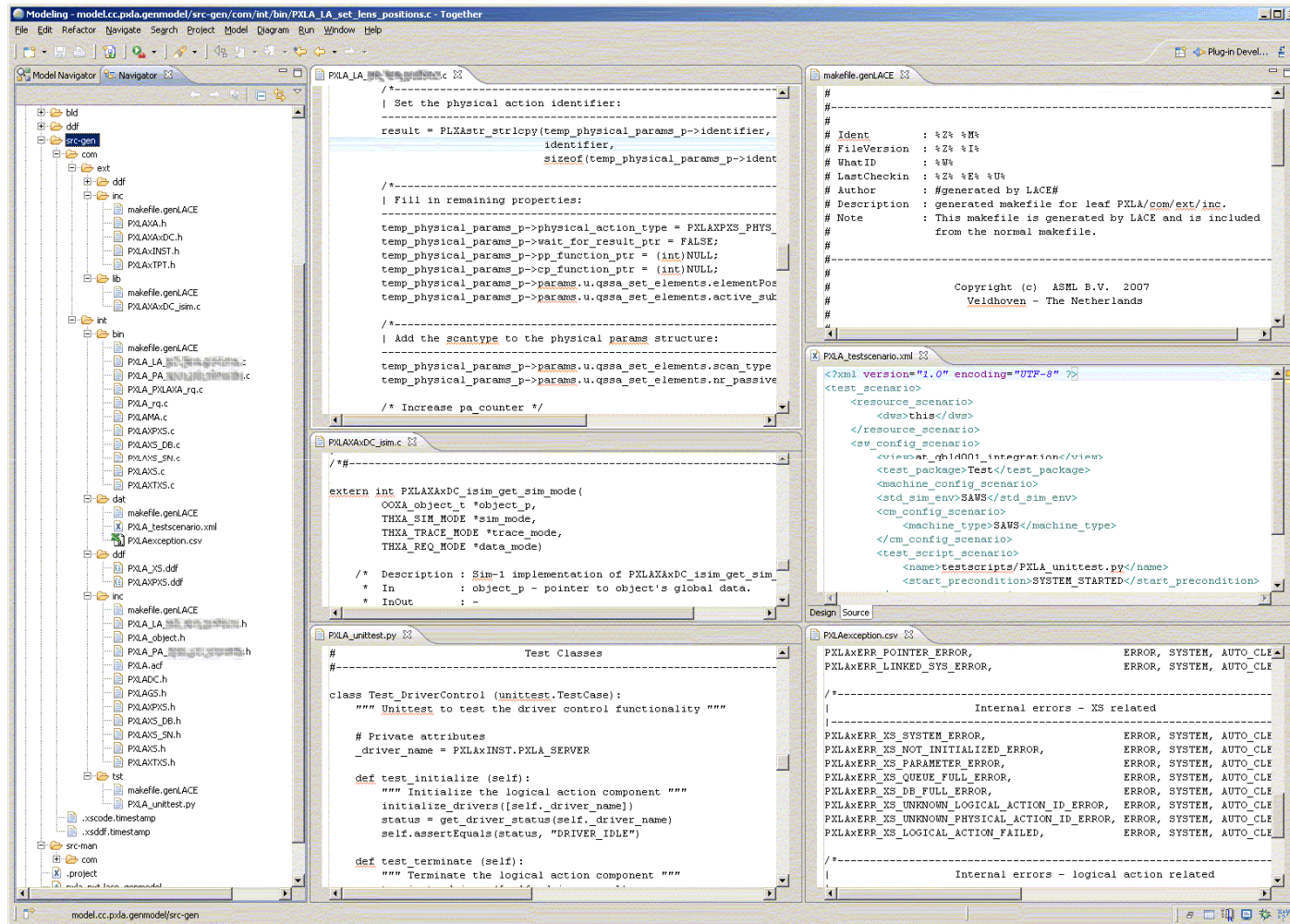
Design



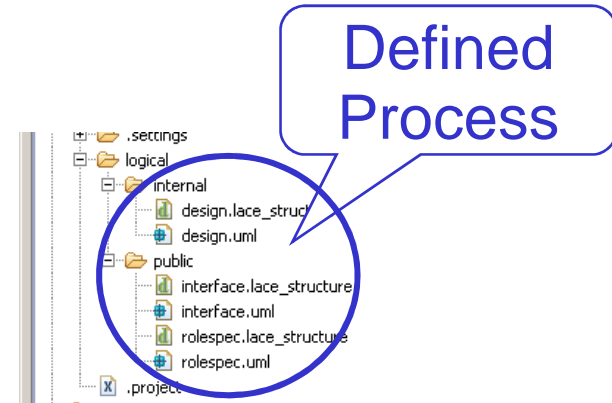
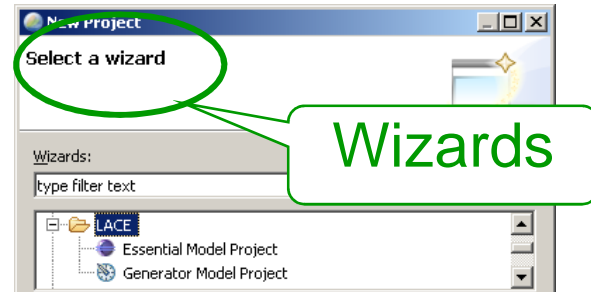
...and generate the realization.



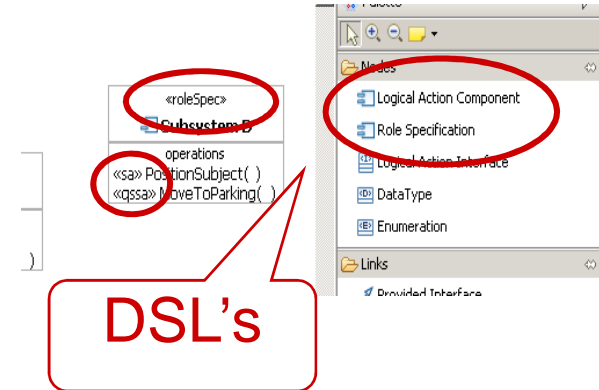
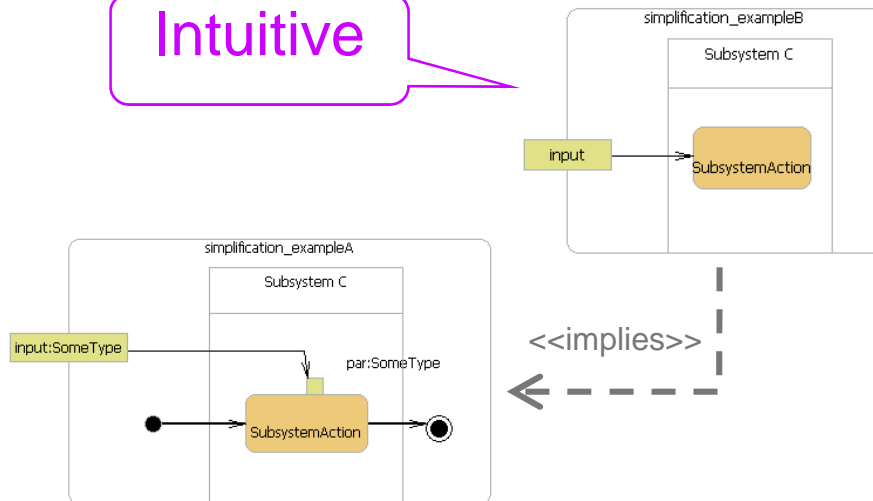
Let the computer do the work....



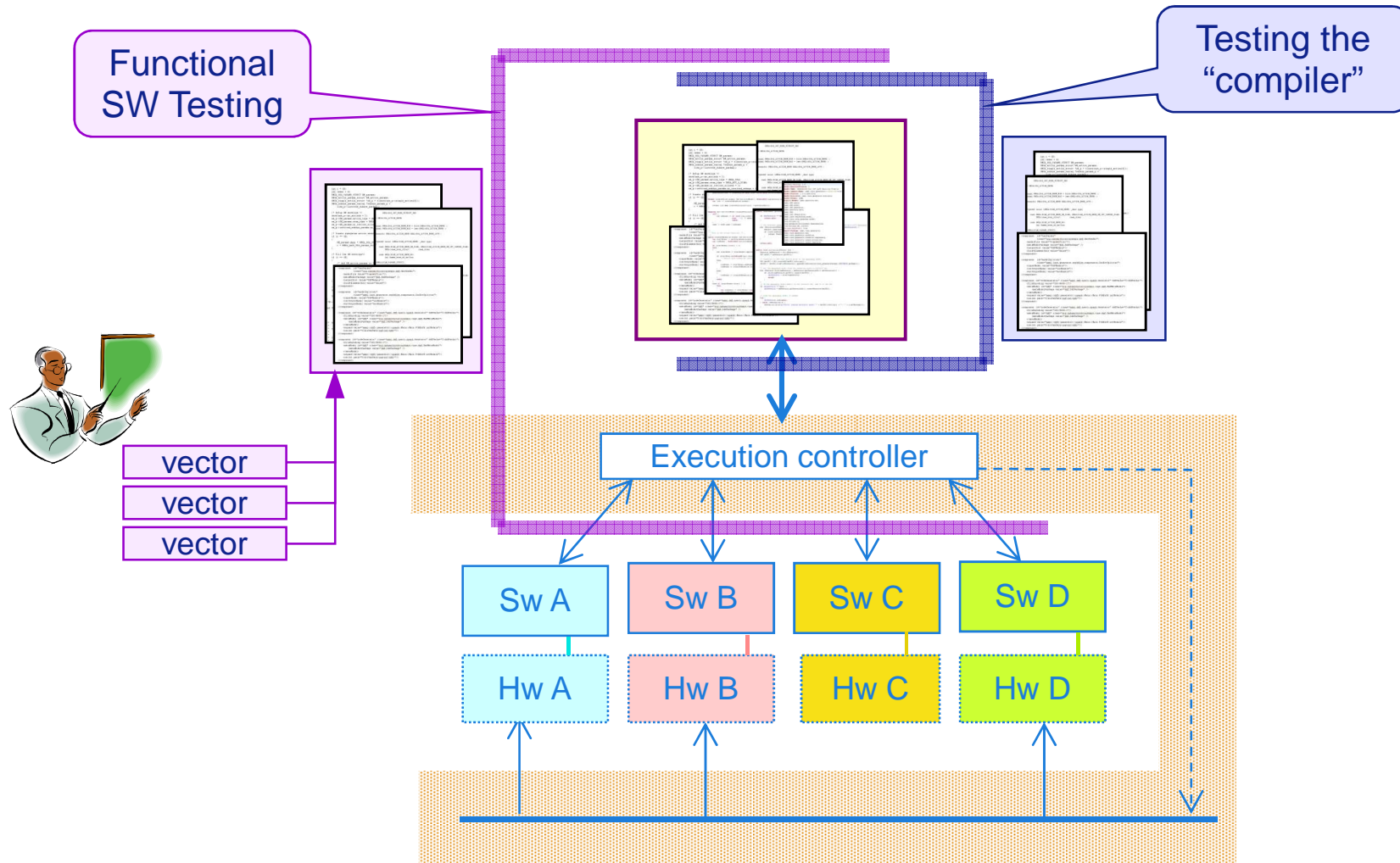
Optimized for engineer



Intuitive

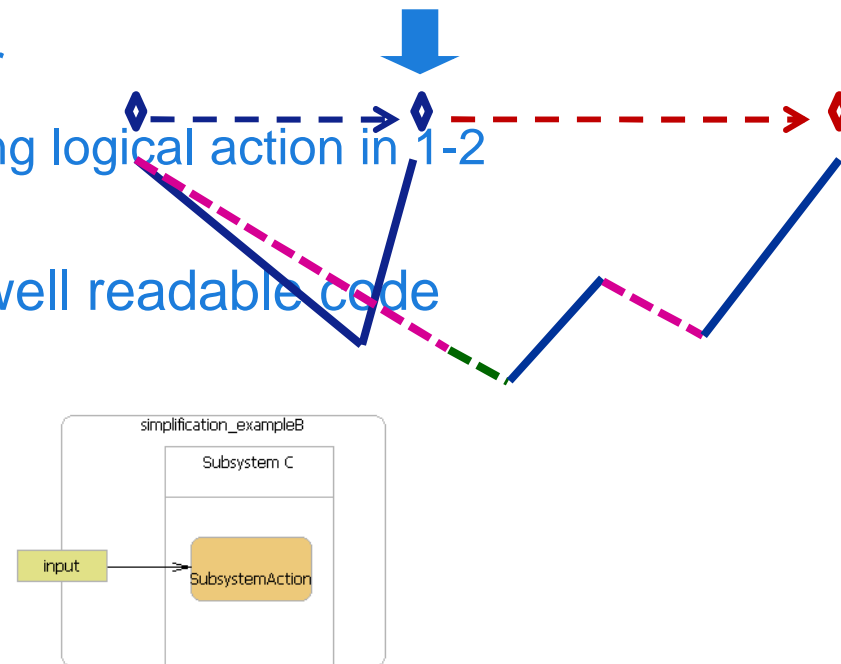


Test suite generated



LACE User Feedback

- Modeling is intuitive
- Learning LACE: 1-2 days
- Efficiency gain: 3x ... 7x faster
 - Also for maintenance: adding logical action in 1-2 days
- High quality, more reliable & well readable code
- > 6000 LOC generated from



Technology

- Eclipse Based IDE
- DSL's : EMF / ecore / GMF / XText
 - UML profile (essential model)
 - New metamodels (generator model, target model)
- Model 2 model (translate concepts, complex)
 - QVT(O)
- Model 2 text (straightforward, add detail)
 - XPand (template based generator)
- Generated artifacts
 - C, ddf, csv, makefile, xml, python, cnf, ...

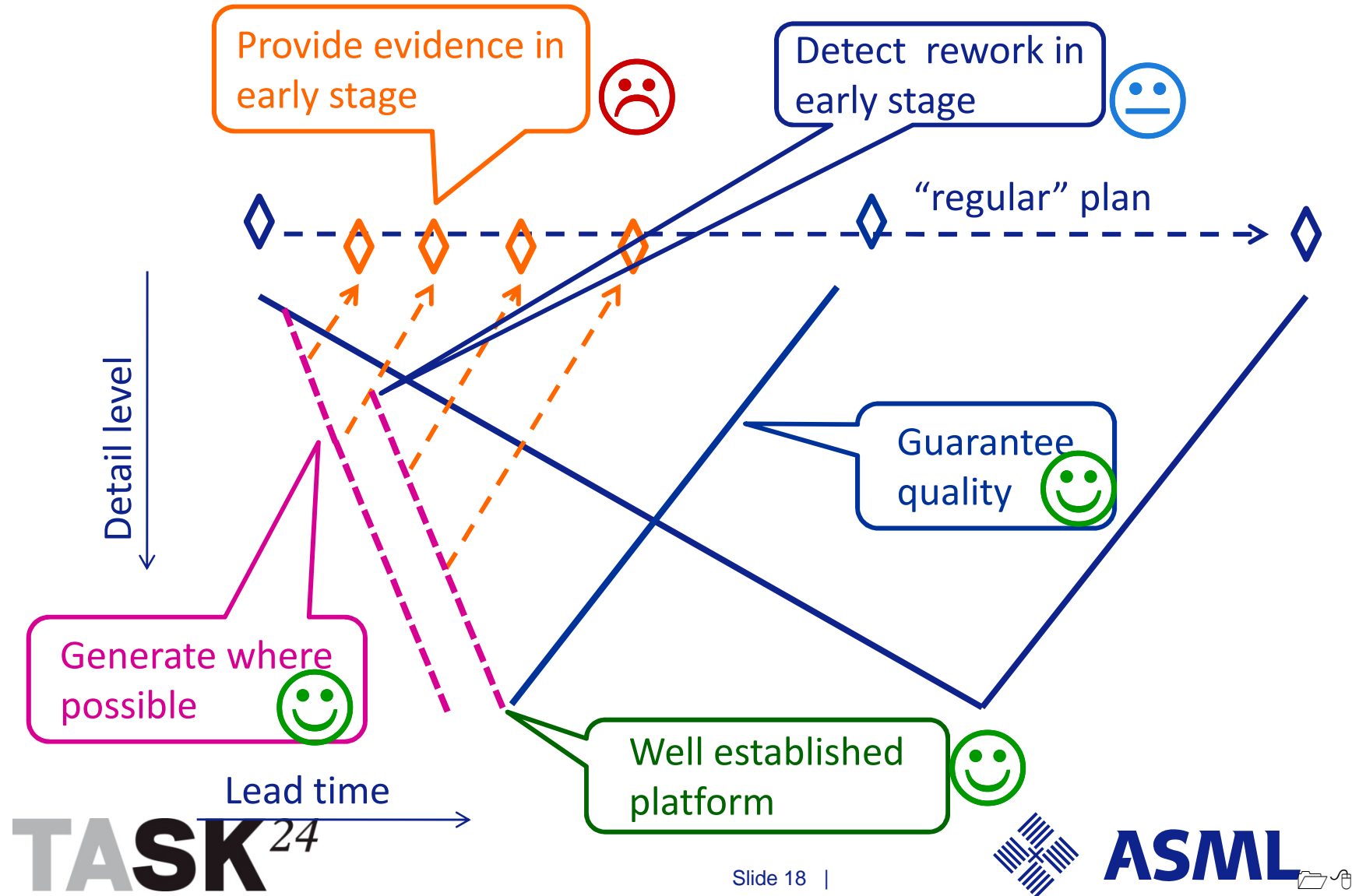
Lessons Learned (Technical)

- DSL definition: Meta-Modeling or Profiling?
 - We do both: make a trade-off
 - Re-use is beneficial, mind unforeseen extensions
- Divide transformation into levels
 - Decoupling during development
 - From abstract to realization level : M2M
 - Simplifies M2T
- Distinct “complete” DSL versus “partial” DSL at front-end
 - Currently implicit (pre-transformation)
 - Would ease verification, maintenance

Lessons Learned (Managerial)

- DON'T release too early
 - First impression is dominant
 - Pre-mature phase: develop for domain engineer
- Involve Domain Engineer in development
 - Inquiry to get feedback
- Invest in “pampering”
 - Complexity is a relative experience
 - Support, Wizards, Help, Integration....
- Reward users (Stimulation)
 - “Certified Engineer”

The V-model: The MDE way



Thank you!

...any questions?

TASK²⁴

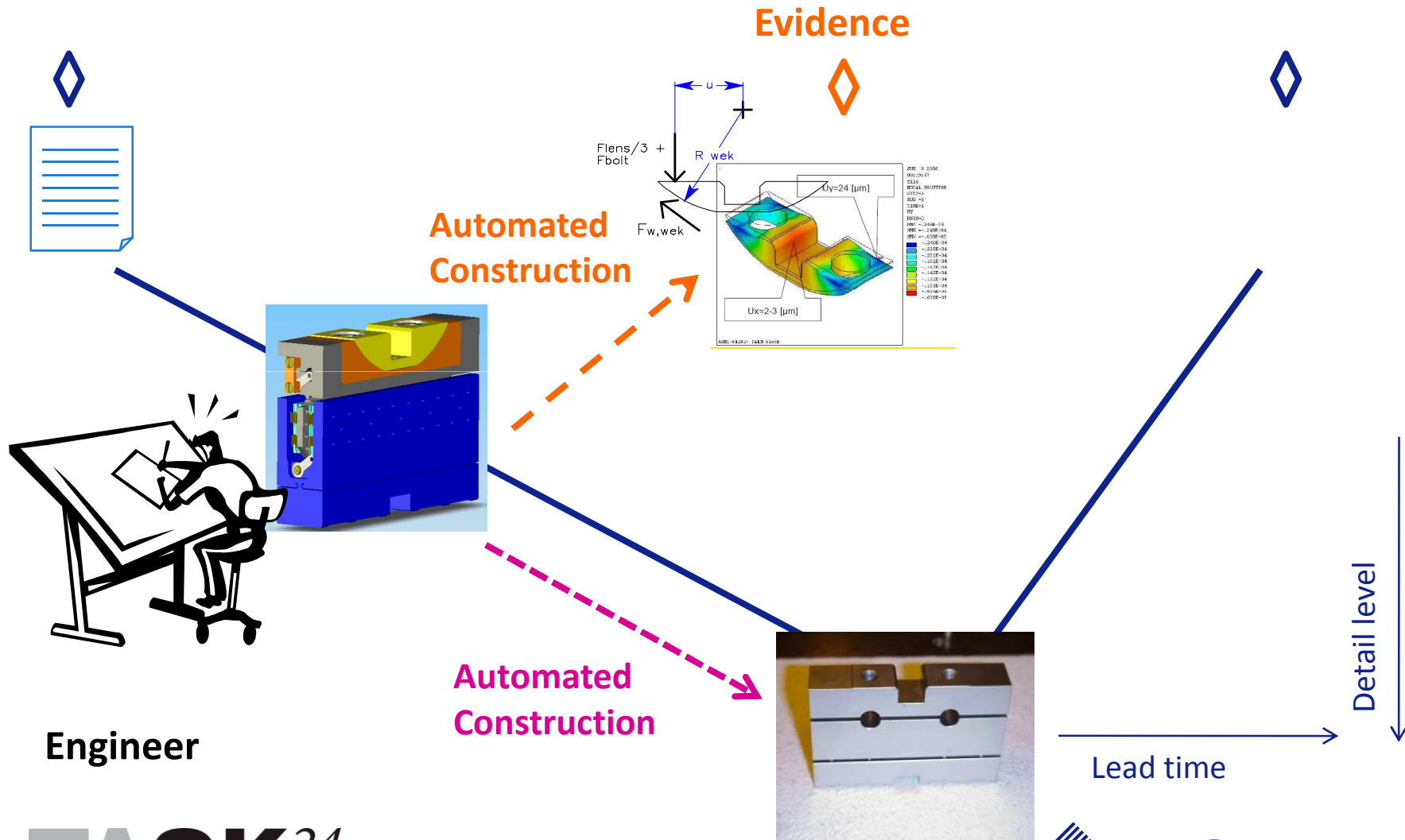


BACKUP

TASK²⁴

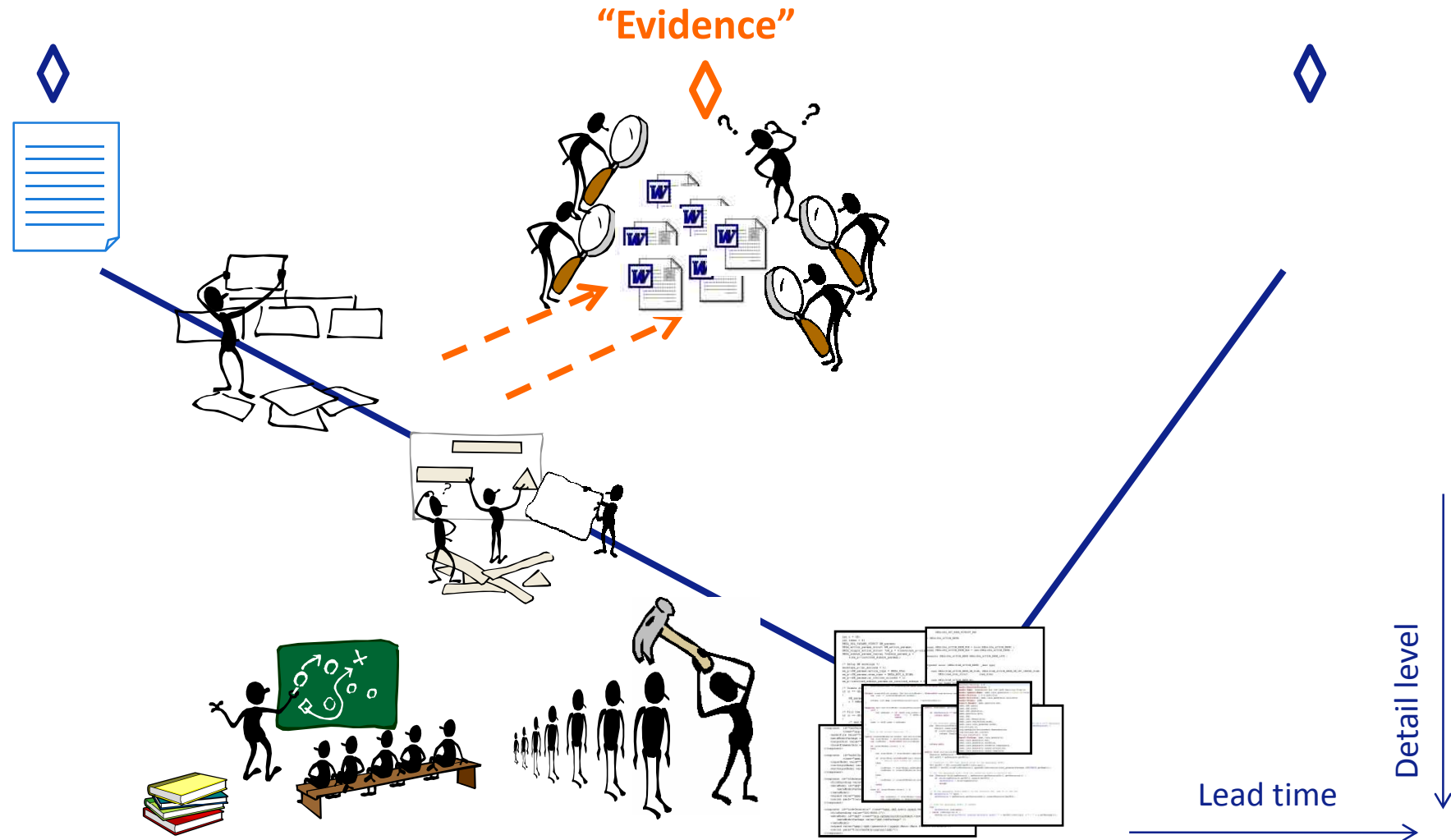


What do others do: Mechanical Engineering



TASK²⁴

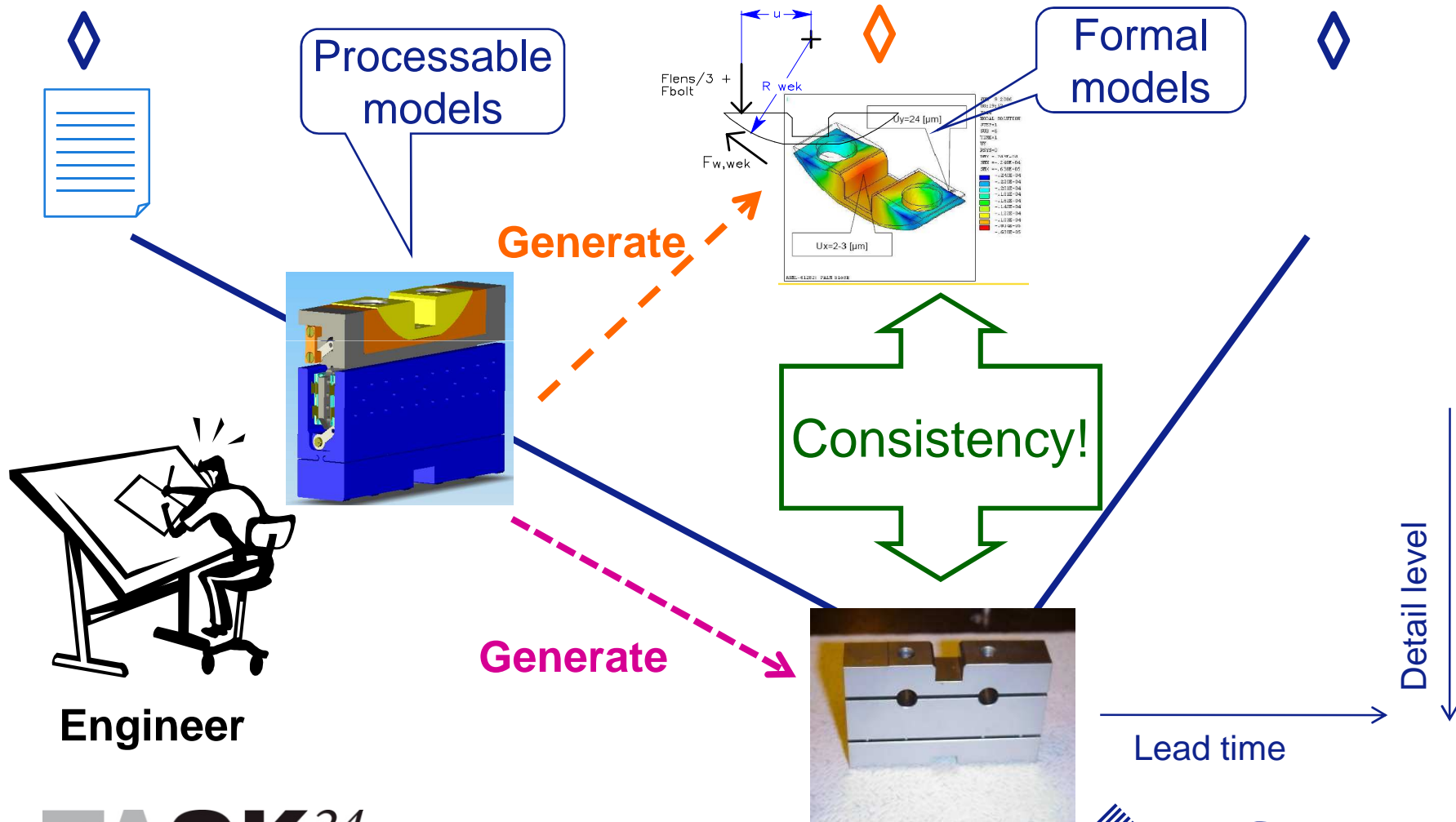
This is what we do: Software Engineering



TASK²⁴

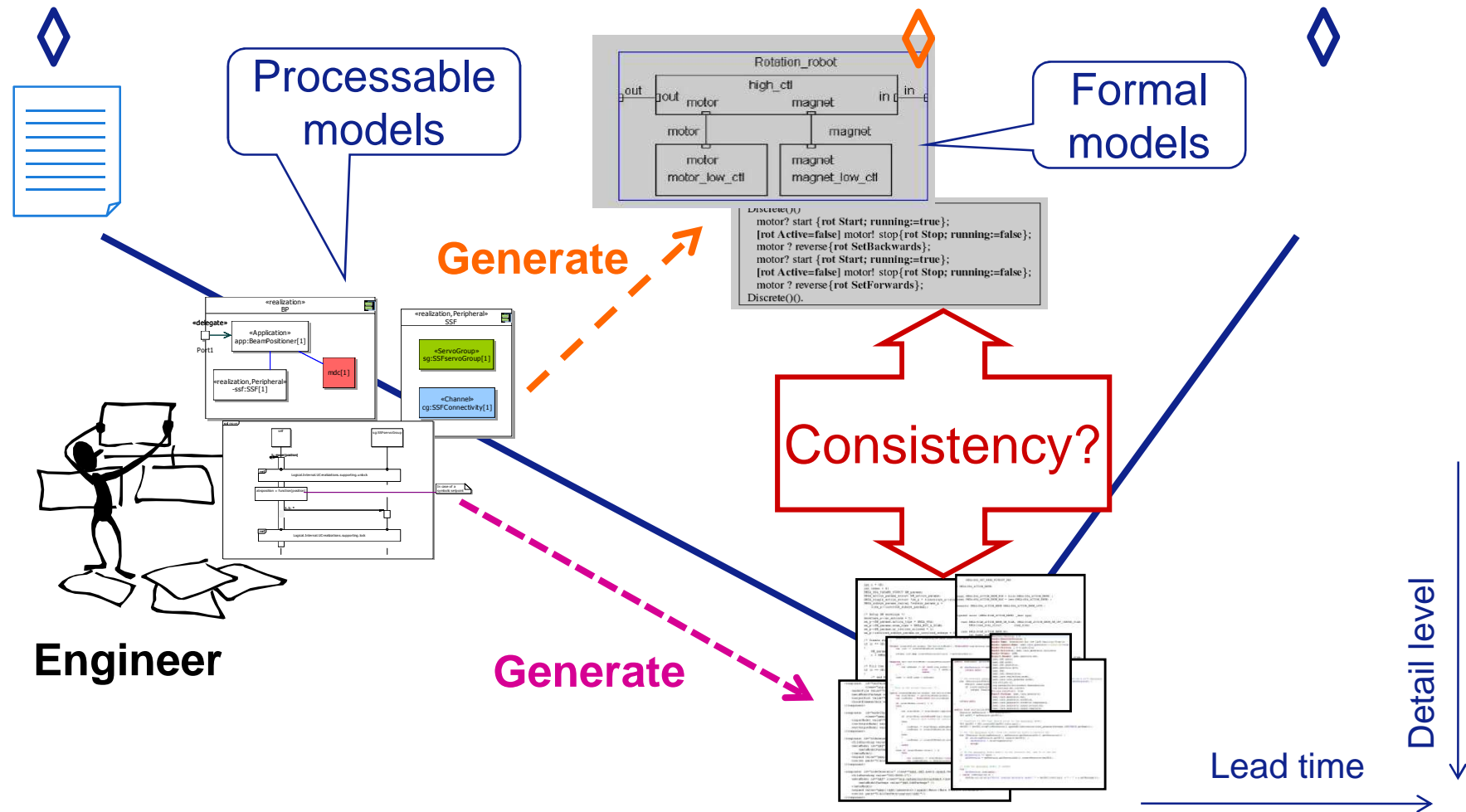


Engineering with models: Mechanical Engineering

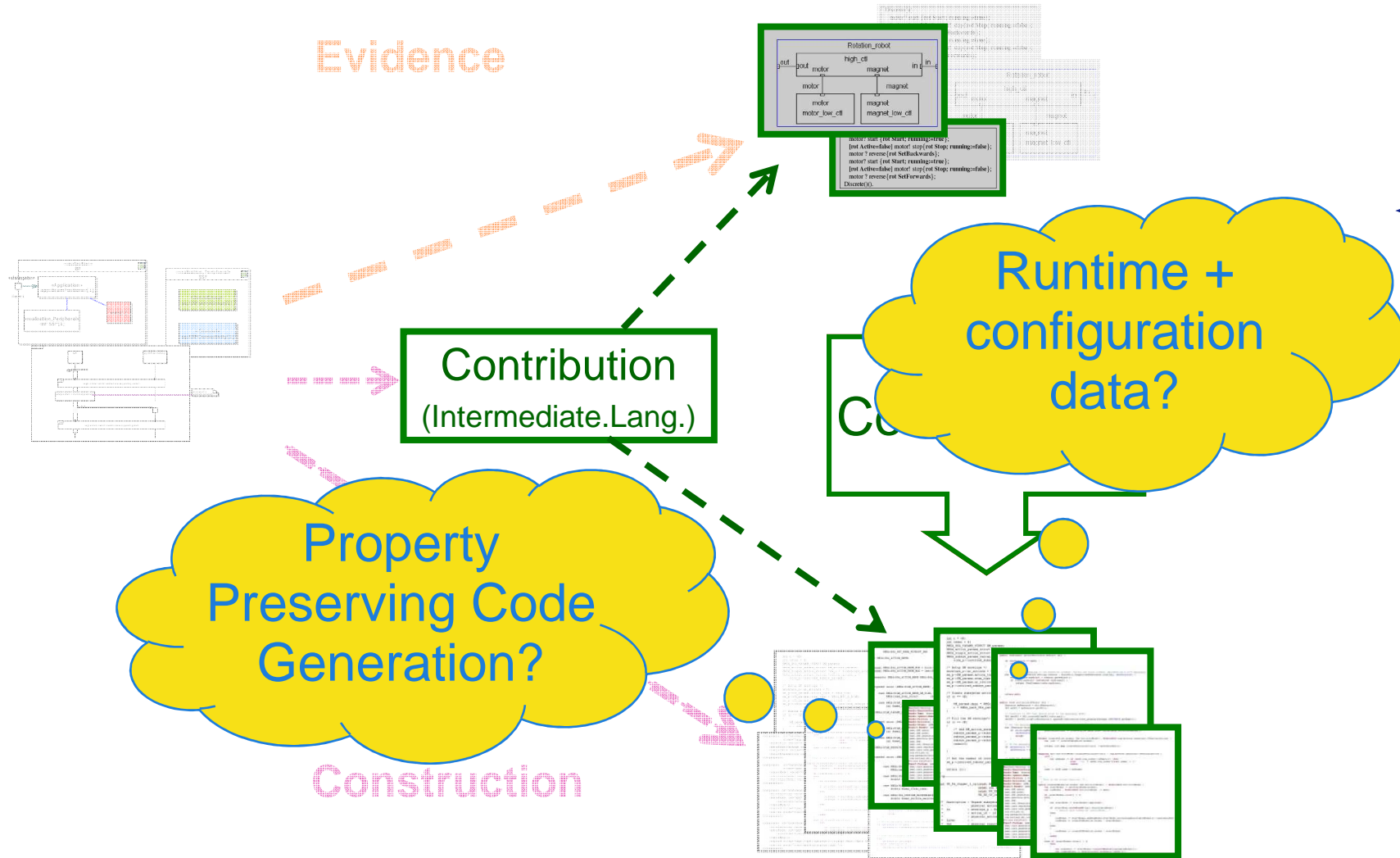


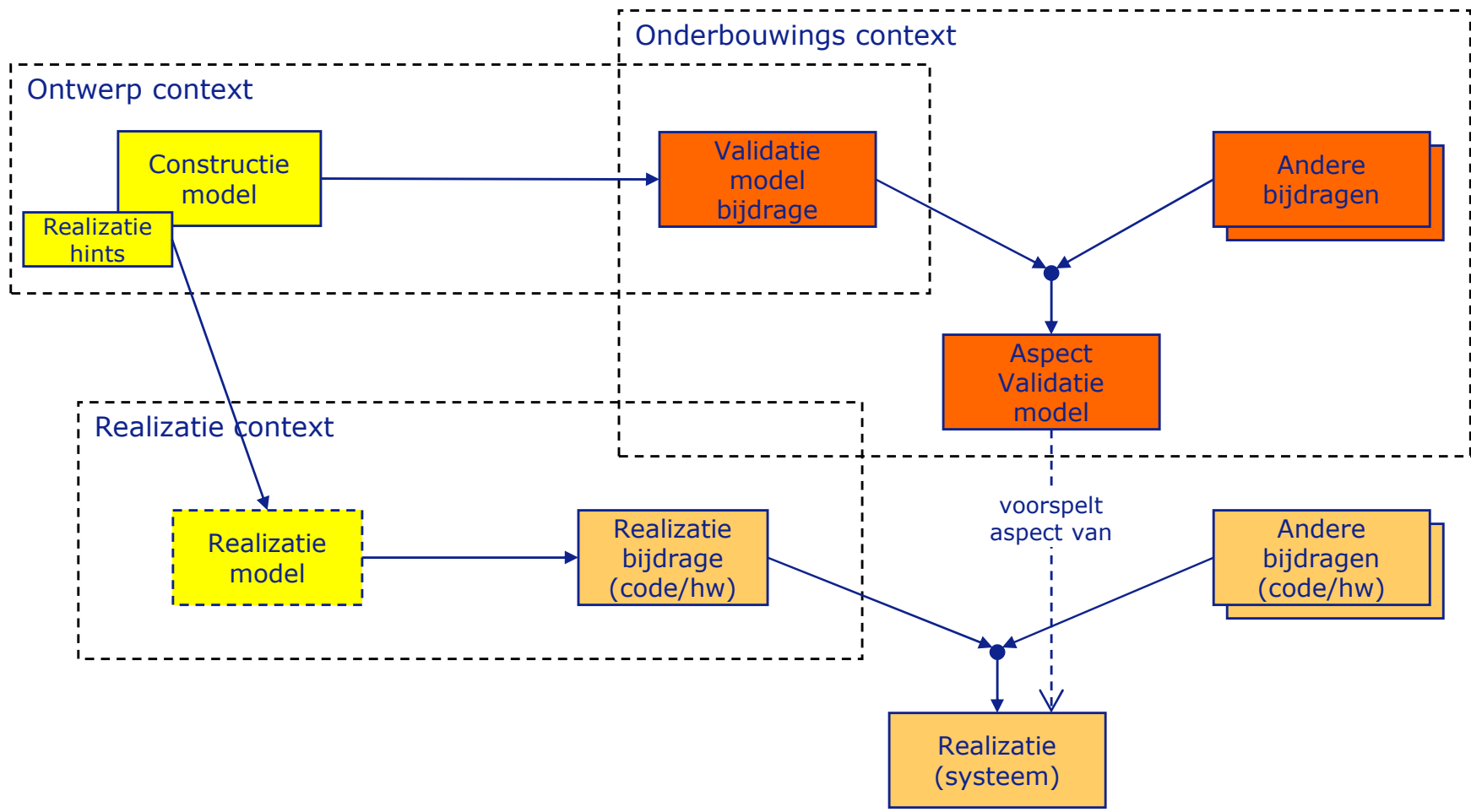
TASK²⁴

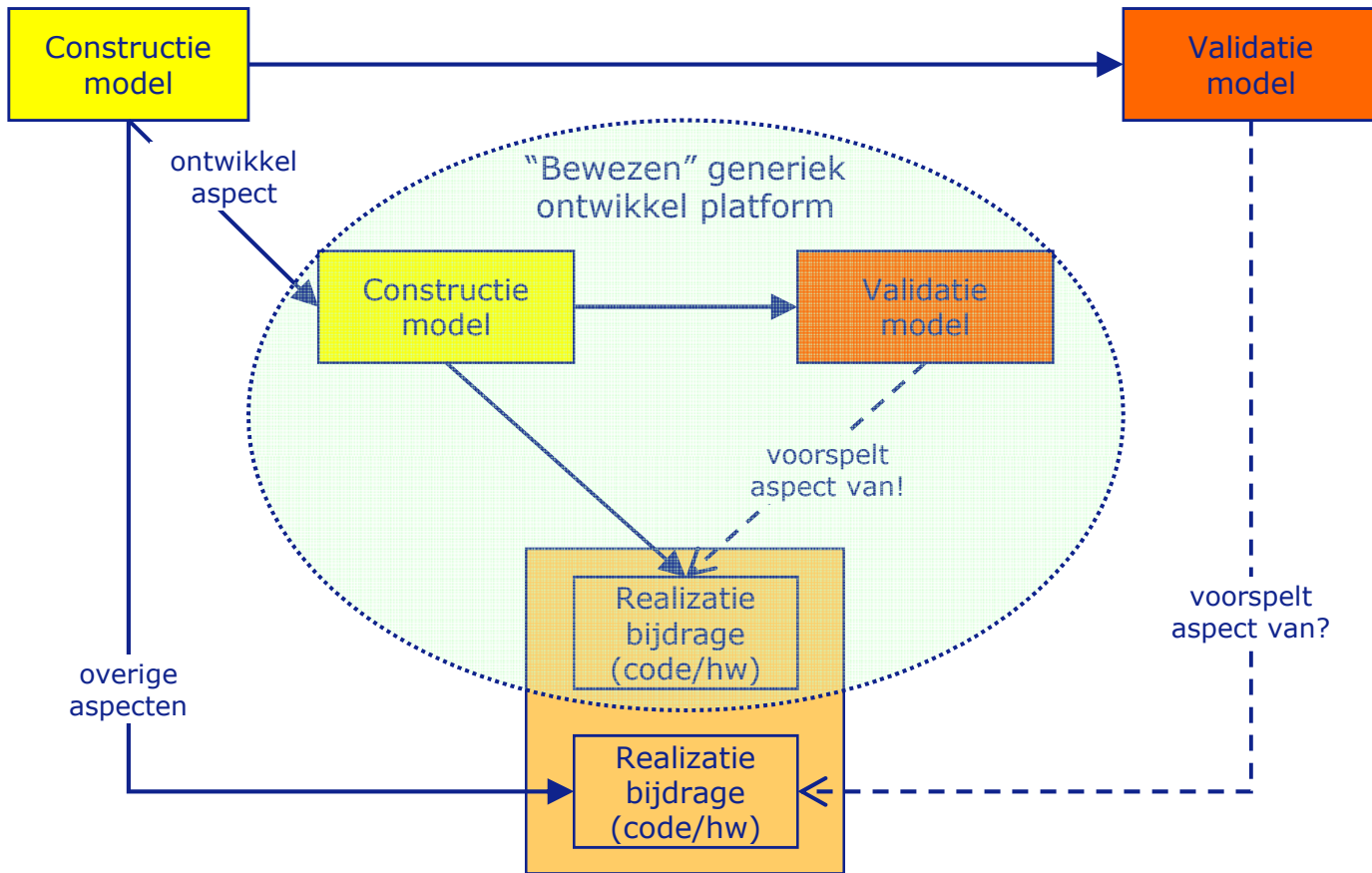
Engineering with models: Software MDE



Re-use of translation units







Generator Architecture: Overview

