27th annual INCOSE international symposium

Adelaide, Australia

July 15 - 20, 2017

Christian Webel, Rainer Steglich, Simon Darting

Modeling Legal Requirements

Motivation

- **Bring products to the market**
- **products are allowed on the market only if they are safe and if all regulatory requirements are met**
- **→ Products in public interest**

Legal Requirements

- Laws and regulations support societal goals by defining rules that represent the public interest
  → Legal Requirements

- Legal requirements
  - are requirements that are “mandatorily required by law”, are non-negotiable, and must be complied with
  - are constraints for the product/project, they
    (1) reduce the number of possible implementations by restricting the solution space
    (2) put restrictions on the development process.
  - sometimes refer to standards
    - compliance with the standard is often equivalent with adherence to the law
  - have to be refined into product/project requirements

- Traceability is crucial
  - compliance has to be shown through appropriate traces that all legal requirements have been elicited from the legal artifacts

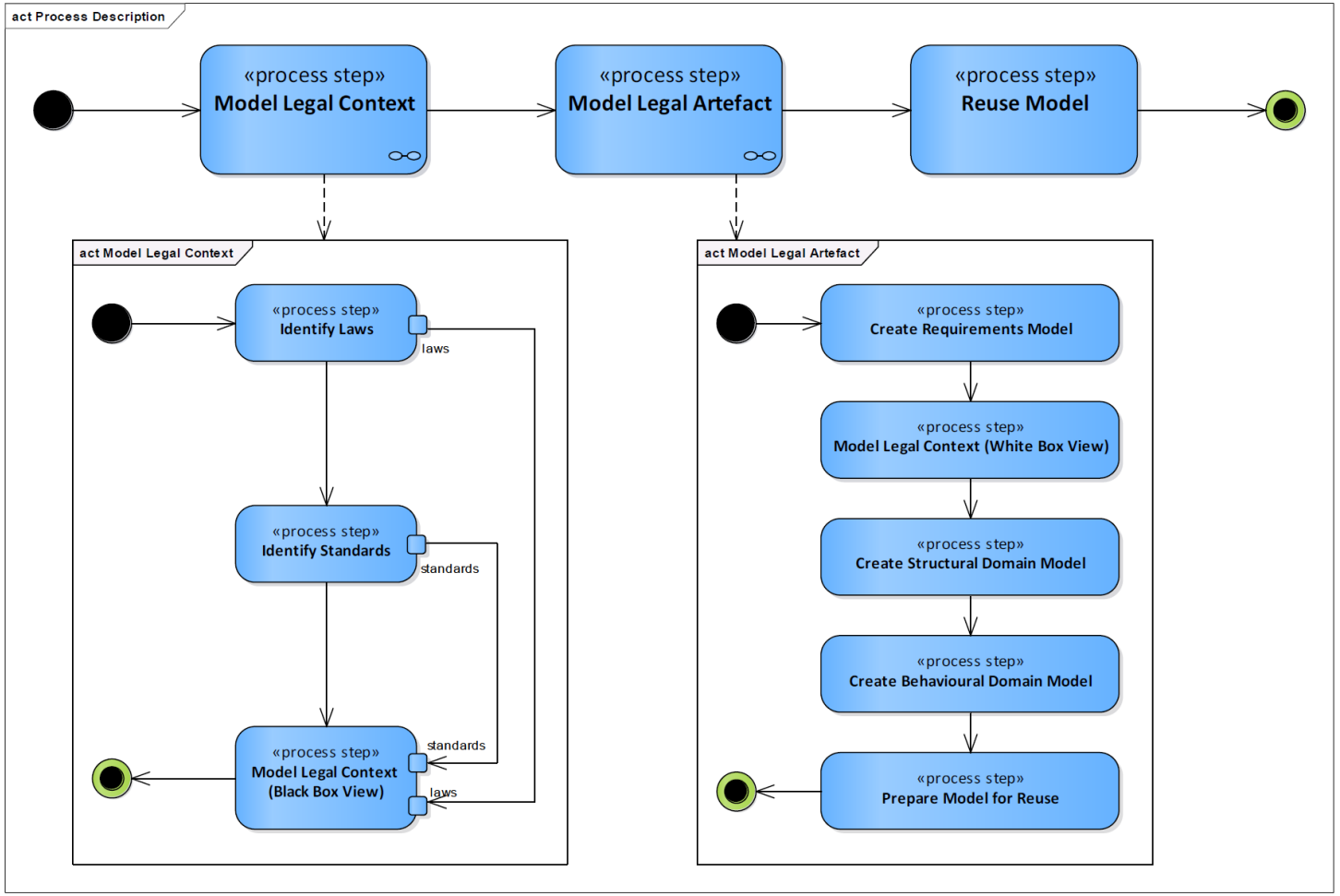
Goal

- Scope
  - Model-Based Requirements Engineering (MBRE) and Legal Requirements

- Develop a systematic model-based approach
  - Identify relevant legal artefacts
  - Elicit legal requirements (including traces)
  - Model requirements and legal artefacts
  - Ensure traceability during engineering lifecycle
  - Prepare for Reuse
- Use SysML (+ own profile)
- Assess in context of WLTP (GTR No. 15)

Overall Approach

base: GTR No. 15 (WLTP) – 2014-05-12 (ECE-TRANS-180a15e.pdf)



Step 1: Model Legal Context
#1.1 Identify Laws or Legal Constraints

- Legislation is one of the most important stakeholders
  - Especially in the domain of safety-critical systems

→ **regulatory approval is an important goal for a development project**

- Characteristics of laws
  - dominated by the domain (e.g. automotive or medical)
  - usually weak harmonization: different laws for different regions/countries
  - are often not a direct source of legal requirements
  - often reference standards that have to be adhered to

Step: Identify Laws

(1) Identify all laws that are relevant for the systems engineering context. Record the following properties: title, version, and publisher.
(2) Record the relationship between laws

Step 1: Model Legal Context
#1.2 Identify Standards

- compliance with a standard is regarded as an acceptable means of receiving regulatory approval
  - De-facto standards vs. required by law

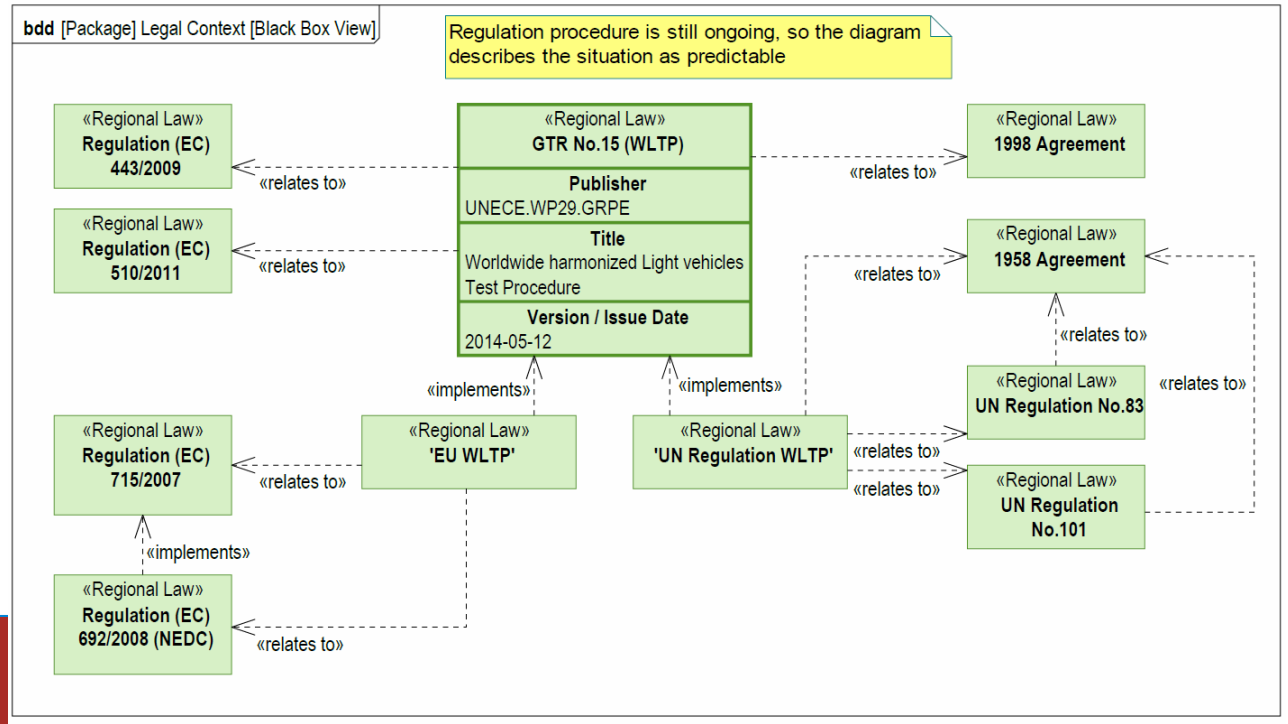
- Characteristics of standards
  - are dominated by domain but also by engineering discipline
  - compared to laws stronger harmonization
  - standards are legal artefacts that can be seen as a source for legal requirements
  - are manifold in terms of requirement types

Step: Identify Standards

(1) Identify the relevant standards by analysing the laws. Record the following properties: title, version, and publisher.
(2) Record the relationships between standards and between standards & laws

Step 1: Model Legal Context
#1.3 Model Legal Context (Black Box)

- Based on #1 and #2
  - Transfer the collected information to the model and visualizing the relationships between the model items
  - Modelled artefact is regarded as a black box
- Some important questions can be answered, e.g.
  - “What are the underlying conditions for the application of the legal artifact?”
  - “Which legal artefacts have to be monitored for changes?”

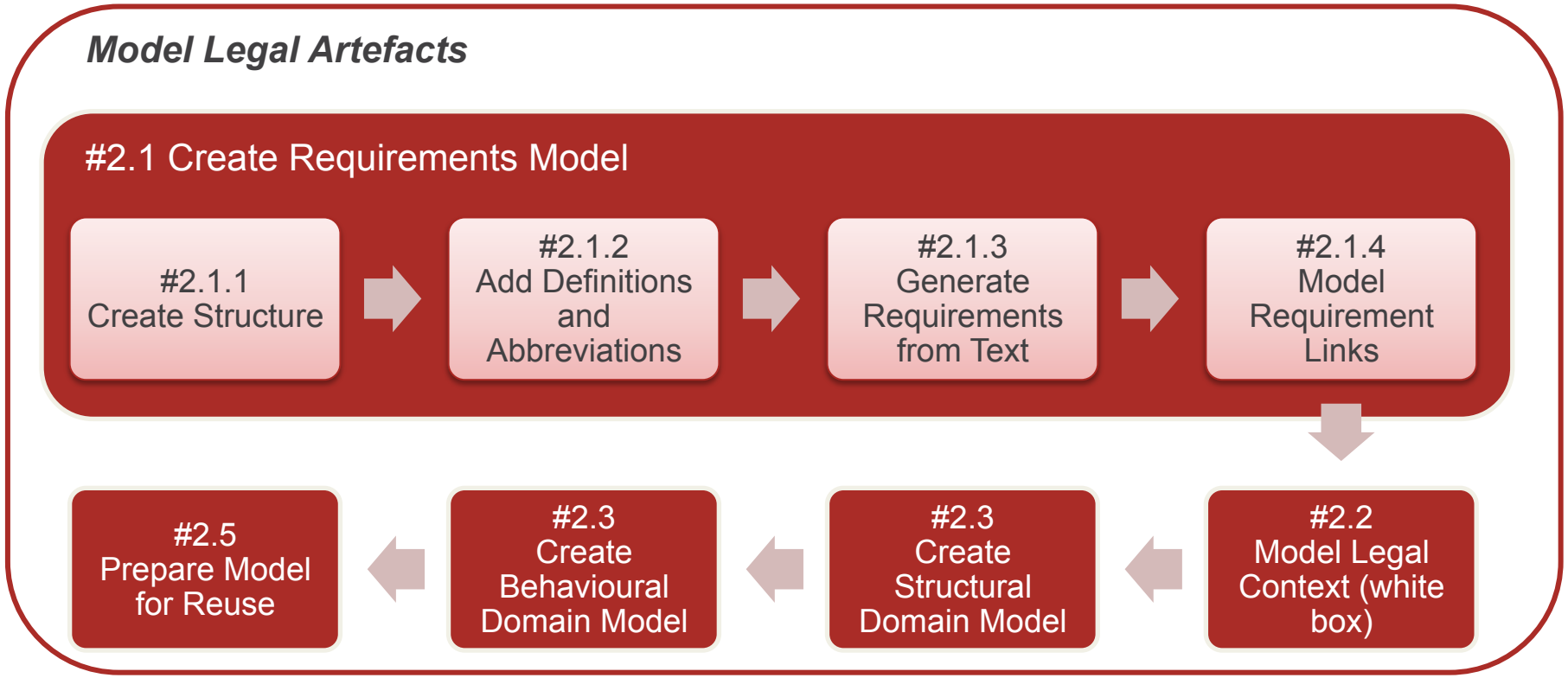


Step: Model Legal Context

(1) Create model items for the artefacts identified in the previous steps and apply appropriate stereotypes
(2) Add meta information (title, version, and publisher) to the elements and create links
(3) Create a diagram that visualizes the modelled relationships

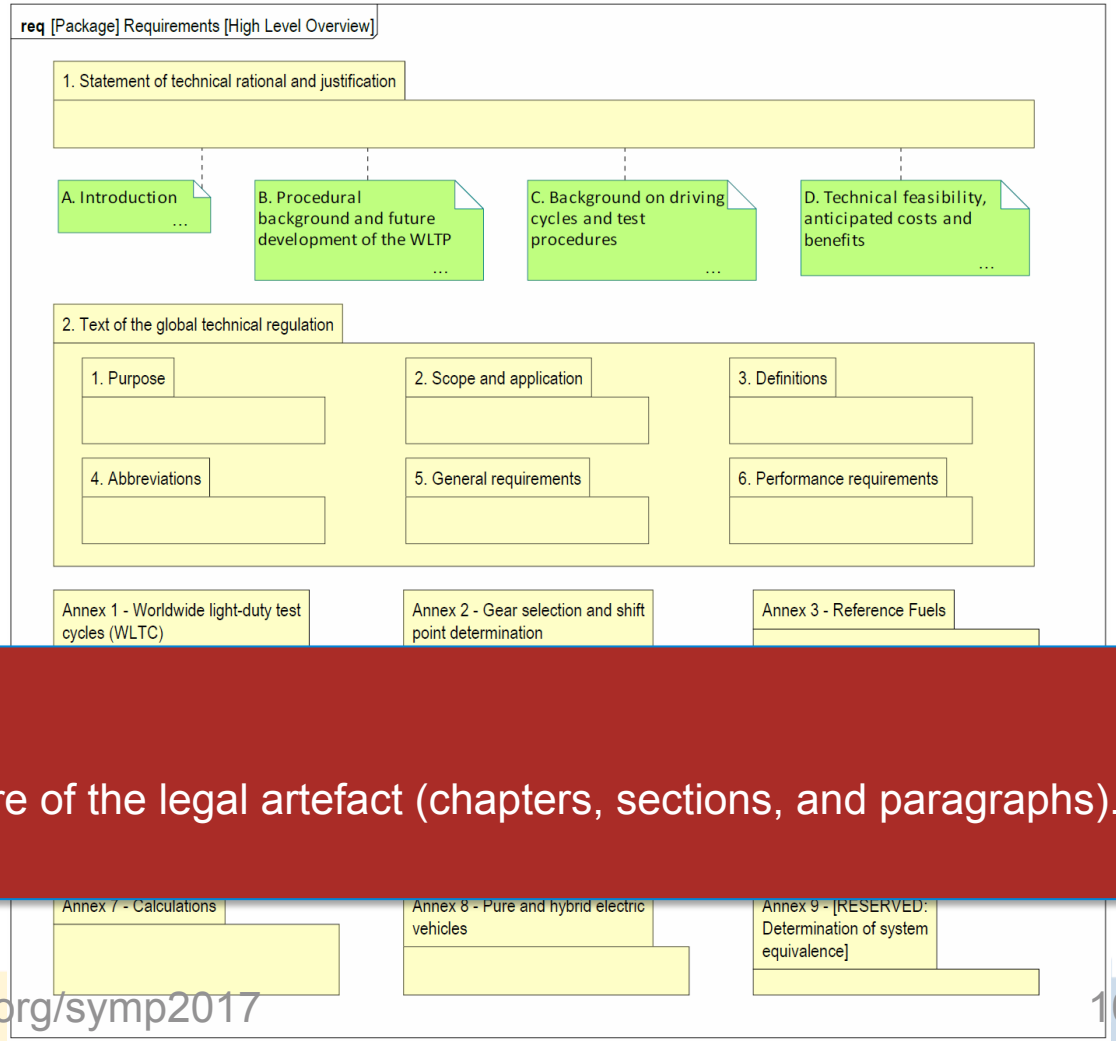
Step 2: Model Legal Artefacts

- covers all activities that need to be performed to derive a model from the legal artefact (standard or law)
  - text (natural language), figures, tables or formulas



Step 2.1: Create Requirements Model
#2.1.1 Create Initial Structure

- create the initial structure
  - to facilitate traceability (forward - implicit), it is recommended to structure the model in terms of packages similar to the structure of the legal document
  - Usually chapter and section headings are transferred to the model to create the hierarchy
  - legal requirements are added to these packages. If appropriate, further hierarchy levels can be added to structure requirements according to paragraphs.

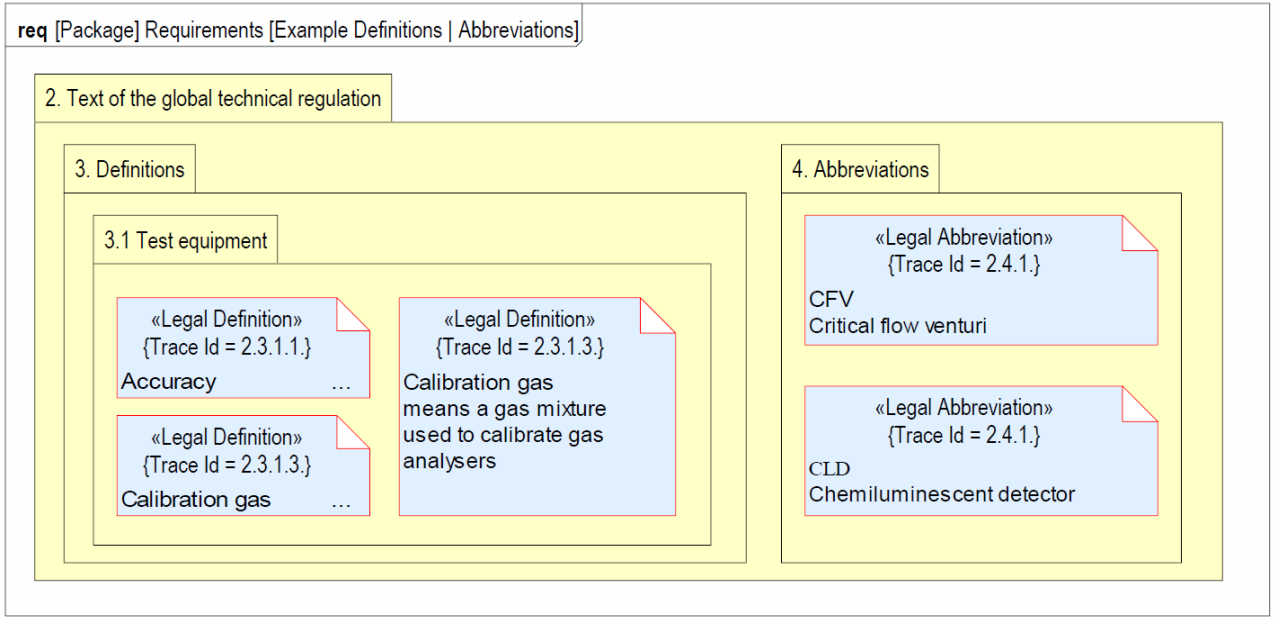


Step: Create Initial Structure

(1) Create a package hierarchy that represents the structure of the legal artefact (chapters, sections, and paragraphs). Facilitate traceability through equal naming.

Step 2.1: Create Requirements Model
#2.1.2 Add Definitions and Abbreviations

- definition and abbreviation are given as part of the glossary
  - together with the initial package hierarchy they build the basic infrastructure that is needed to elicit requirements from the legal artefact
  - definitions and abbreviations are modelled as individual (stereotyped) model elements to foster a better reuse → items of type comments
  - For traceability, the tagged value “Trace Id” is added



Step: Add definitions and abbreviation

(1) For each definition/abbreviation defined by the legal artefact:
  a. create a stereotyped model element
  b. add the traceability id (if available)

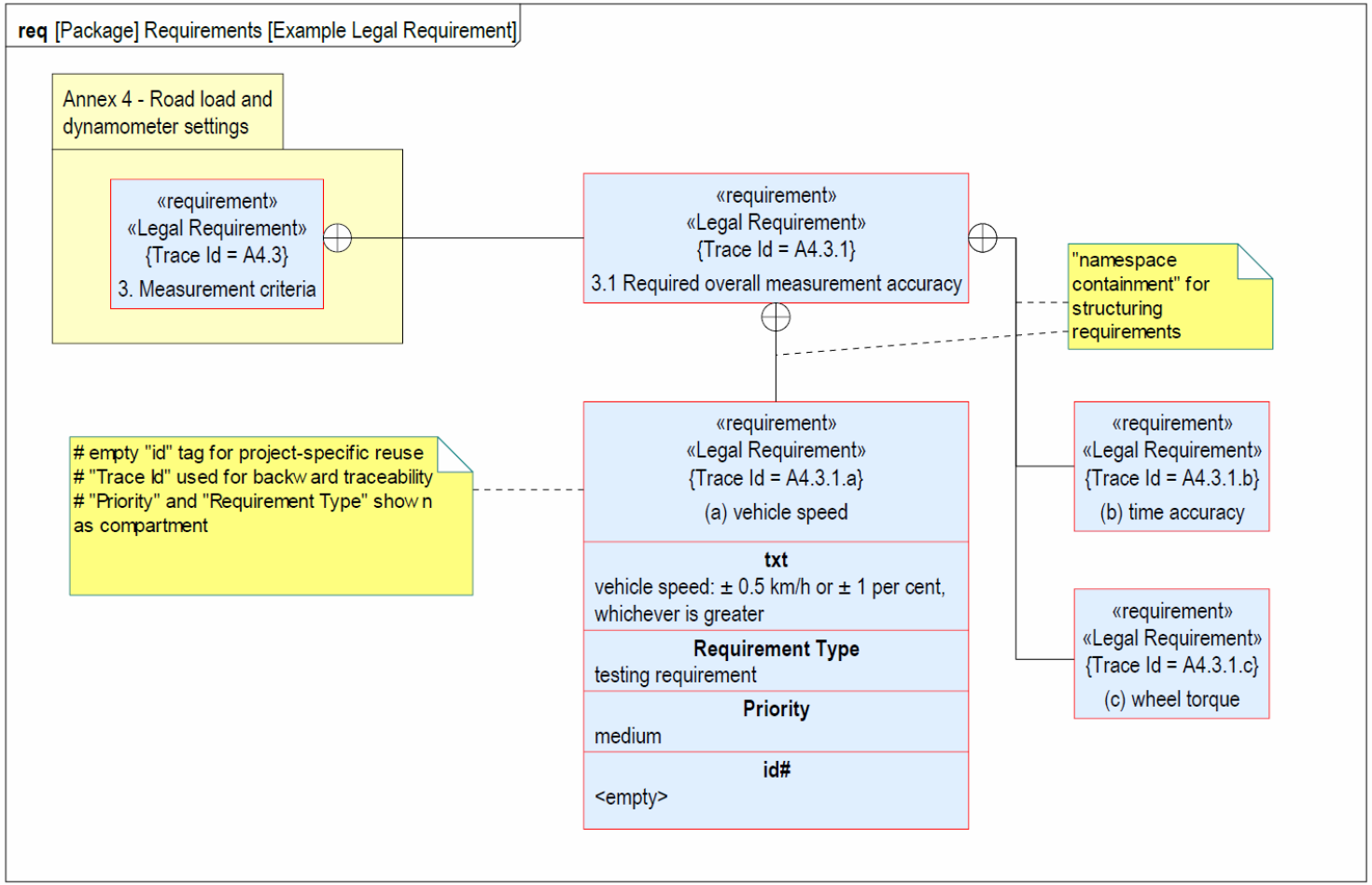
Step 2.1: Create Requirements Model
#2.1.3 Generate Requirements from Text

- legal requirements are identified and modelled
  - stereotyped variant of the SysML element “Requirement” → “Legal Requirement”
- text is directly transferred to the field that is provided by the SysML element
  - text of the legal artefact must not be changed → reformulating the original text involves the risk of faults
- Maintaining traceability is crucial when moving from “textual-world” to the model
  - backward traceability (model to legal artefact) is realized through an ID
  - forward traceability (document to legal artefact) is realized through using a similar structure in the model
- “natural language elements = text” can be directly transferred to legal requirement

Step: Generate requirements from text

(1) For each package that was created:
   a. Analyse the text and create a model element “legal requirement” for each identified requirement
   b. Transfer the text to the tagged value “text”
   c. Add the “Trace Id” and assign values for “Priority” and “Requirement Type”
   d. Analyse the requirement for references to figures or tables

Step 2.1: Create Requirements Model
Example Requirements from WLTP



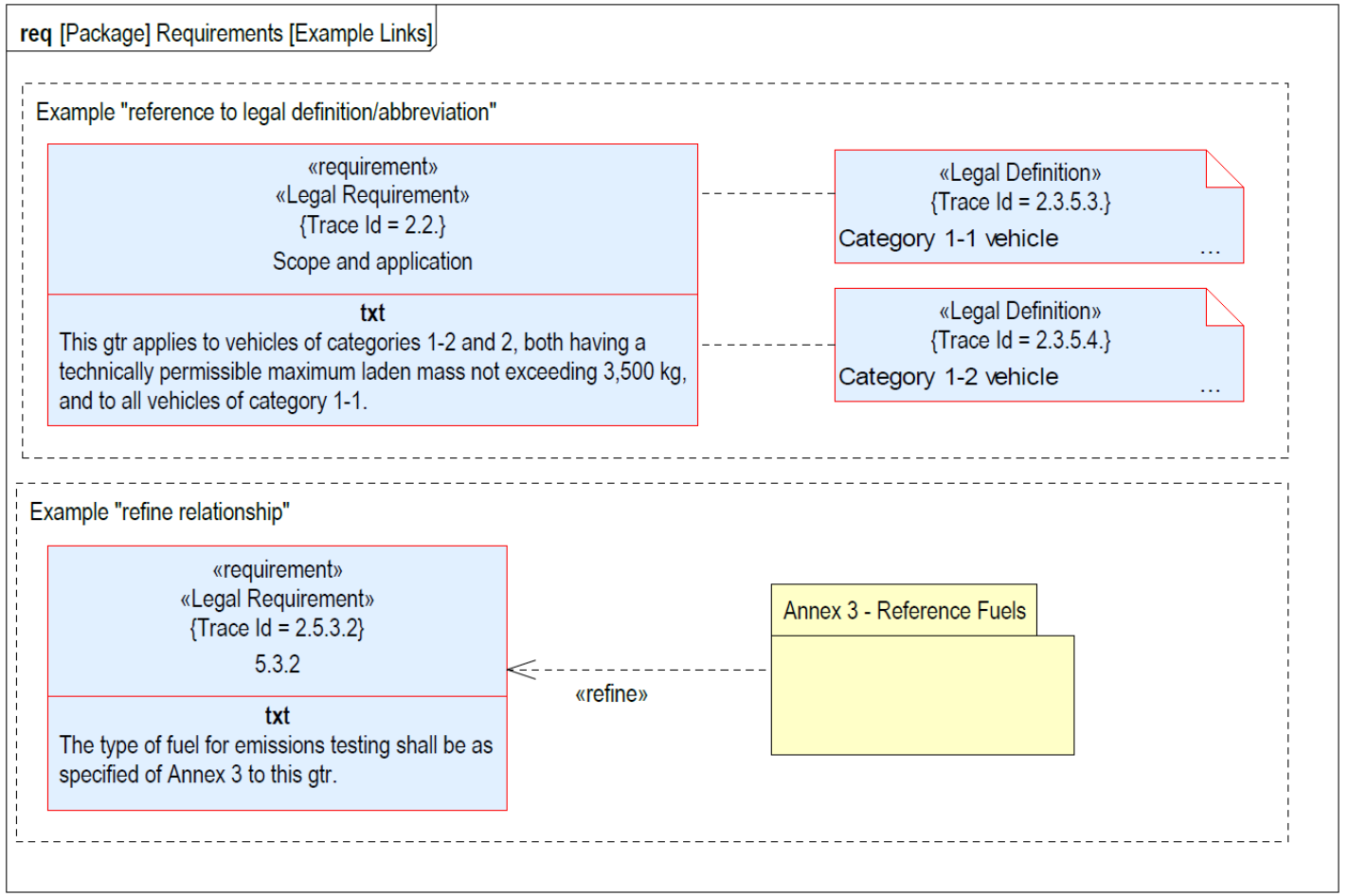
Step 2.1: Create Requirements Model
#2.1.4 Model Requirement Links

- Modelling all references in a standard document is a huge effort
  - should be tailored to an acceptable extent in case the expected benefit is low
  - working with a model element that has all relevant information is much more comfortable than searching the model
  - This is especially important in the context of “modelling for reuse”
- Different kinds of references and how they can be modelled with SysML relationships:
  - requirement refines another requirement →<<refine>>
  - requirement references another requirement → <<trace>>
  - requirement uses terms that are defined as “Legal Definition” or “abbreviation” → <<note link>>
  - other guidelines/standards are referenced → <<trace>>

Step: Model requirement links

(1) For each legal requirement:
   a. Analyze the text for references to definitions, abbreviations, chapters, or other requirements
   b. Establish an appropriate link

Step 2.1: Create Requirements Model
#2.1.4 Model Requirement Links



Step 2.2: Model Legal Context (white box)

- Extension of the model of the legal context is extended by a white box view
- Requirements of the legal artefact are analyzed for references to standards or guidelines
- Helps to answer questions like: “Are the relevant standards available (in the correct version)?”.

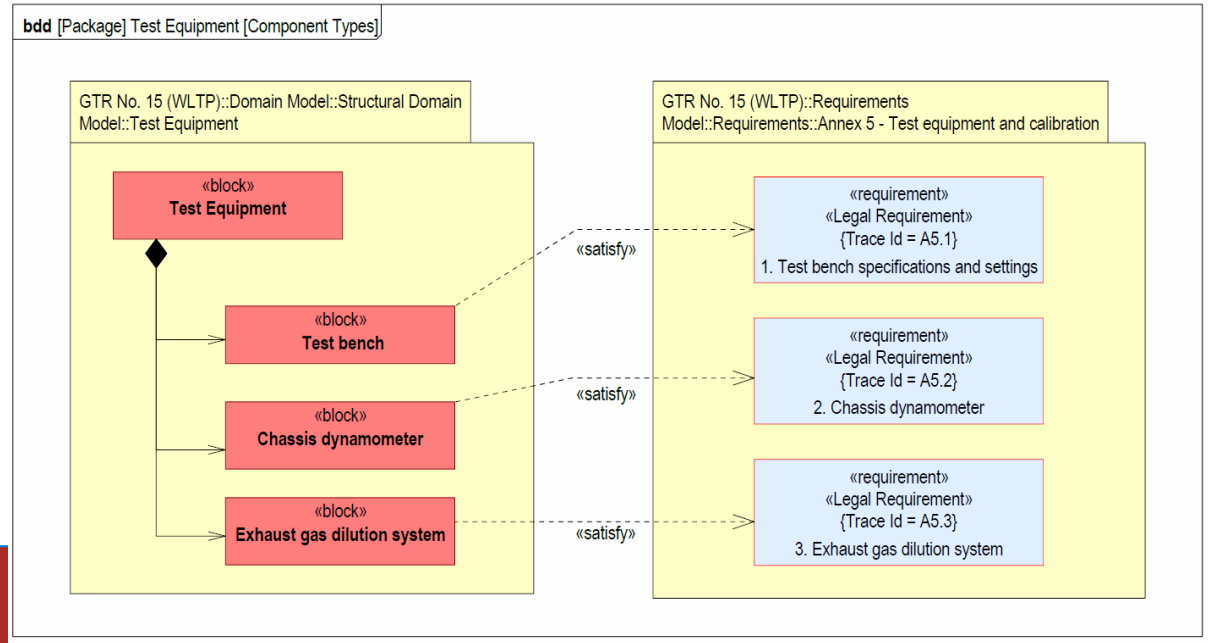
Step: Model legal context (white box)

(1) Analyze requirements for external references (standards, guidelines, …)
(2) For each external reference:
    a. Create a block for the external reference and apply an appropriate stereotype (e.g. <<standard>> or <<guideline>>)
    b. Add title, publisher, and version to the model item
    c. Establish a “trace to” relationship between the requirement it originates from and the model item
(3) Create a diagram that visualizes the legal context (white box)

Step 2.3: Create Structural Domain Model (1)

Model Domain Blocks

- The requirements are analyzed systematically for structural elements
- Tracing from requirements to domain blocks via “satisfy”
- domain blocks provide the basis for further modelling
- The structural domain model can answer questions like
  - “Which requirements are relevant for a specific domain block?”

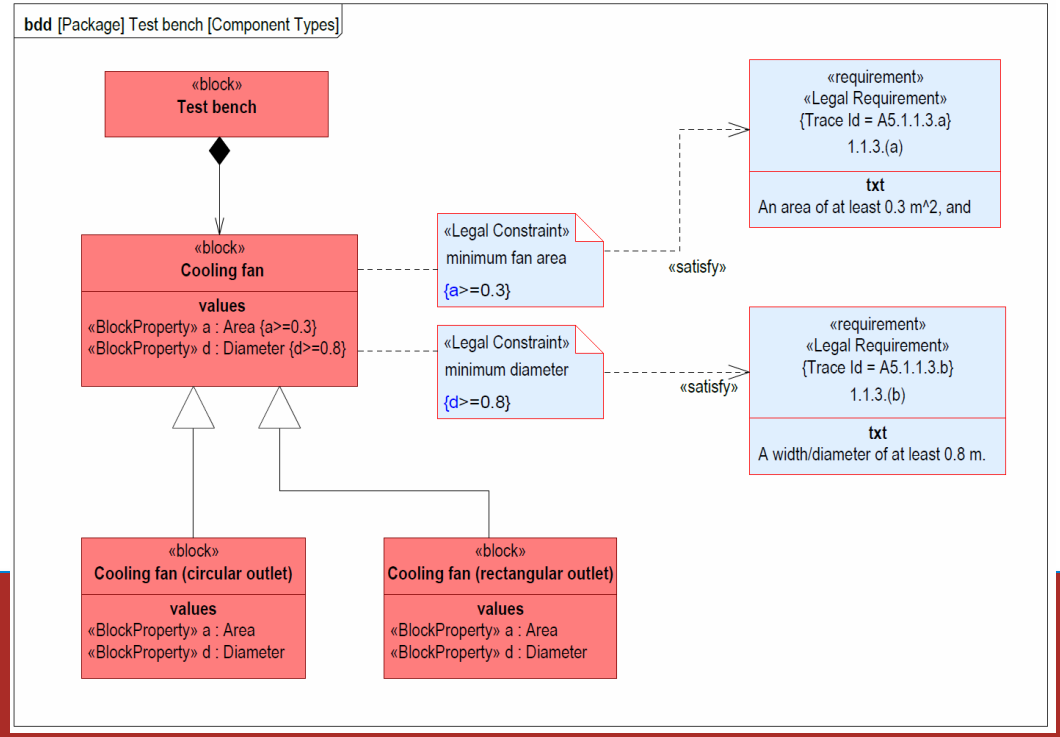


Step: Create structural domain model

(1) For each requirement (of a package/group of requirements):
  a. Identify structural elements (domain blocks)
  b. Define domain blocks in the model (SysML element “block”)
  c. Link the defined blocks to the requirement (“satisfy relationship”)
(2) Model the structure of the identified elements (composition + interfaces)

Step 2.3: Create Structural Domain Model (2)

- Model Constraints
  - The requirements are analyzed for constraints
  - Constraints are modelled and linked to domain blocks defined in the step before
  - The structural domain model with constraints can answer questions like
    - "What constraints are relevant for a domain block?" can be answered.



Step: Model Constraints

(1) For each requirement:

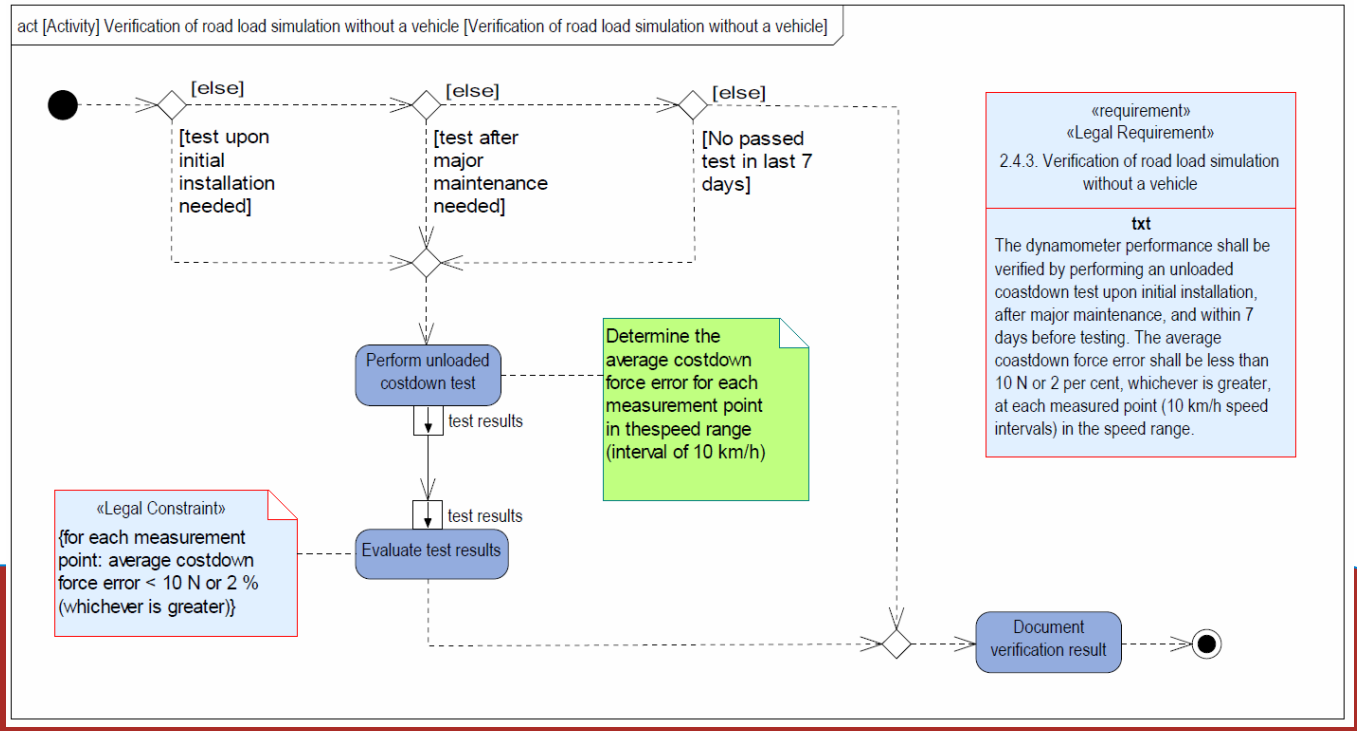
a. Identify constraints that belong to domain Model constraints and add them to domain blocks (use the stereotype "legal constraint")

b. Link the defined constraint to the requirement ("satisfy" relationship)

(2) If appropriate: model complex (mathematical) relations with the help of "parametric diagrams"

Step 2.4: Create Behavioral Domain Model

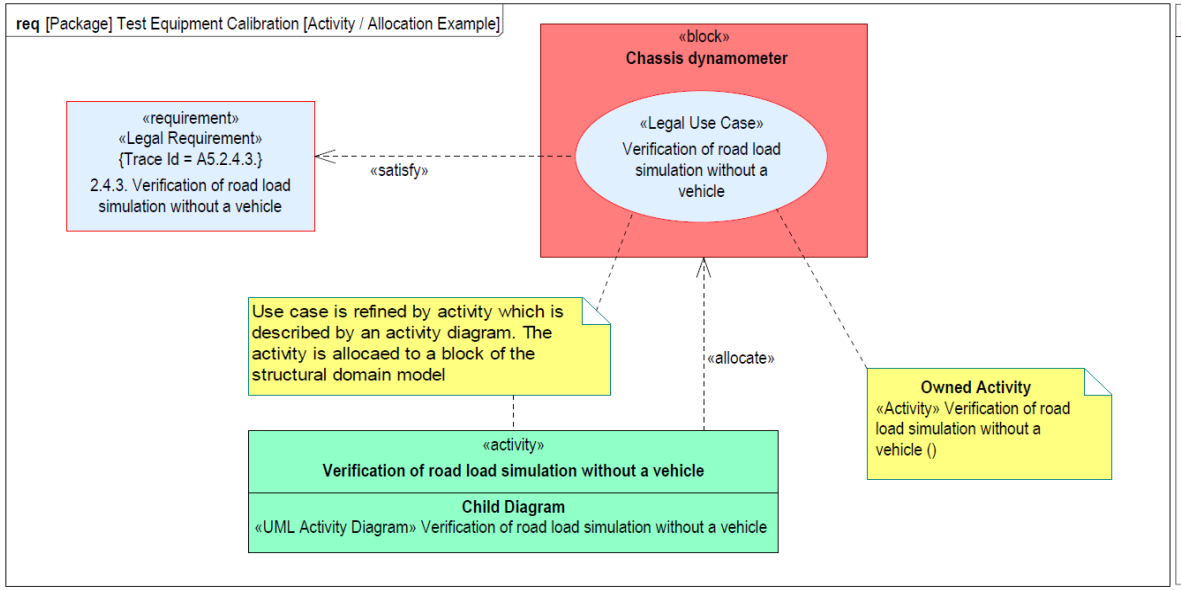
- Describes the behavioral aspects of a domain or system
  - Usable diagram types: activity, sequence, state machine, and use case diagram.



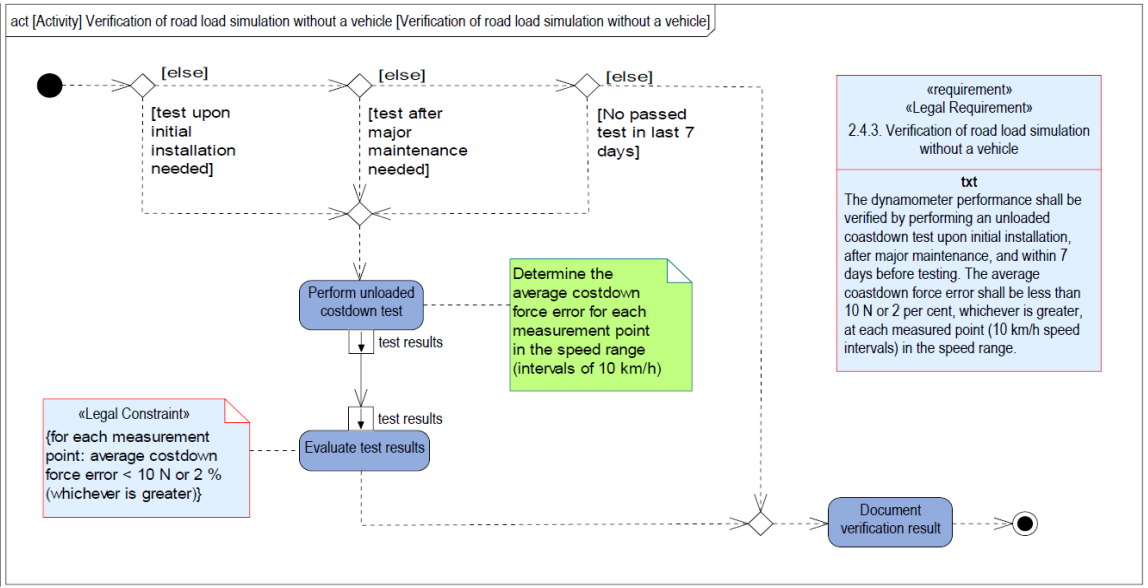
Step: Create Behavioral Domain Model

(1) For all requirements

a. identify behavioural aspects that are relevant for the domain model
b. choose an appropriate SysML element/diagram and model the behavioural aspect
c. assure traceability to legal requirements (<<satisfy>> or <<refine>> relationship)
d. If appropriate: link created elements to the structural domain model (use <<allocate>> relationship)



activity and allocation



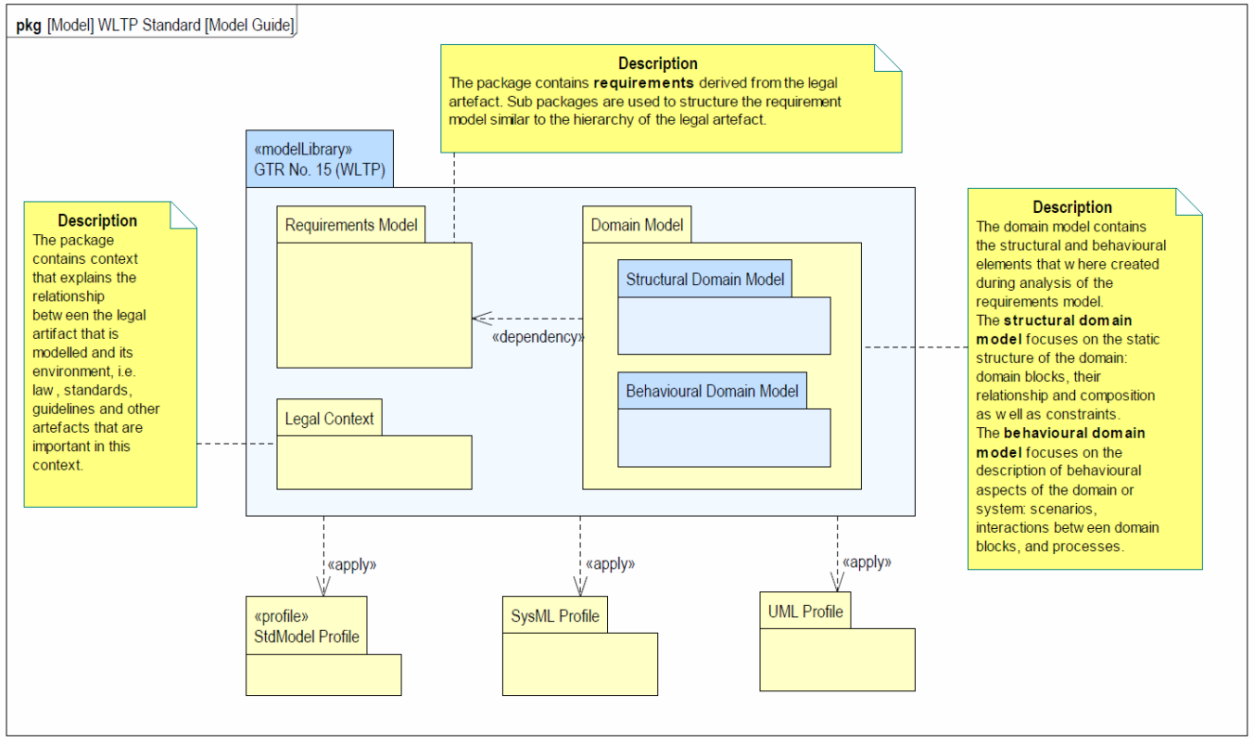
activity diagram and legal constraints

Step 2.5: Prepare Model for Reuse (1)

- Usually, the model of the legal artefact is developed for reuse in several projects.
- There are two major aspects that are relevant in this context:
  - (1) The model hast to contain some information on how to reuse it.
  - (2) Quality assurance is a major concern.

(1) Providing guidance for reuse

- Create a diagram that gives a rough overview of the model (structure, content categories, and dependencies)
- Notes can be used to explain the content structure.
- Dependency between packages and the application of profiles shall also be shown



Step 2.5: Prepare Model for Reuse (2)

- **Remarks**
  - For efficient reuse it is beneficial to explain how the model can and shall be used
  - An example project that is referenced by the model of the legal artefact shall explain the essential concepts
  - The following additional aspects have to be considered when “modelling for reuse”:
    (1) Changing the model shall be restricted to authorized staff only
    (2) Changes need to be tracked; noticeable to users based on unique version numbers
    (3) Establish a process that transfers knowledge gathered during reuse back to the model library

(2) Quality assurance

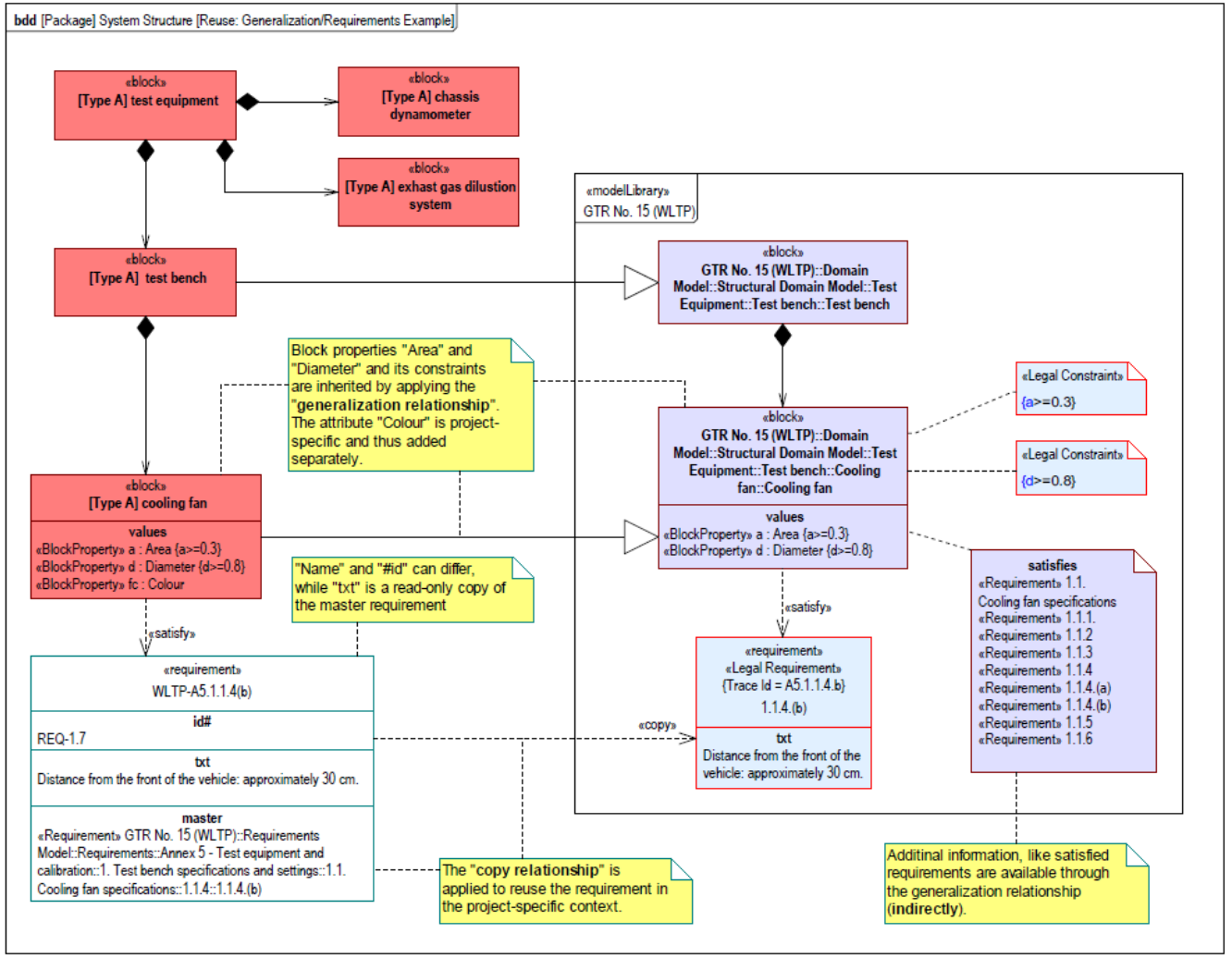
- In the worst case, an error or fault made during modelling is spread over all projects that reuse the model library
- Following two steps of quality assurance shall be performed:
  (1) Run checks on model
    - some modelling tools provide consistency, completeness, or correctness checks.
  (2) Carry out a model review

Step 3: Reuse Model in Project (1)

- A model of a standard or other legal document is a typical artefact that can be reused in several projects
  - SysML/UML provides a special type of package, stereotyped <<ModelLibrary>>
  - can be imported to a project-specific model
  - generalization mechanism can be used to inherit attributes, constraints, or requirements.
  - e.g., system element that is part of the standard model inherits all its requirements and constraints when the generalization relationship is defined in project-specific model.

- **Remarks**
  - knowledge gathered during a project must be transferred back to the model library
  - e.g. implementations hints for a requirement: “to comply with the standard you should do this and that …”)
  - Interpretations and remarks on ambiguous formulations shall be added in special comments (e.g. stereotyped with “interpretation”, “legal remark”, “recommendation”, or “best practice”).
  - Concrete example
    (1) Some requirement is formulated with ambiguity
    (2) The notified body is asked for clarification.
    (3) The results of the clarification are added as a stereotyped comment.
    (4) Knowledge can be added to the library when it is not context-specific, otherwise it is added to the project model.

Step 3: Reuse Model in Project (2)



Summary

- Systematic approach + step-by-step description
- SysML + StdModel Profile
- Traceability addressed in every step
- Modelling for and with reuse addressed
- Assessment in the context of the WLTP
- …

- Future Work
  - Reuse approach + process-related requirements
  - Tool collaboration
  - Views for regulatory approval
  - Regional variability and product line engineering

Questions?

Dr.-Ing. Christian Webel
invenio Systems Engineering GmbH
Harrlachweg 1, 68163 Mannheim, Germany
christian.webel@invenio.net

Rainer Steglich
Dorfstrasse 10
6353 Going am Wilden Kaiser, Austria
rainer.steglich@outlook.com

Simon Darting
Fraunhofer IESE
Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany
simon.darting@iese.fraunhofer.de

27th annual INCOSE international symposium

Adelaide, Australia

July 15 - 20, 2017

www.incose.org/symp2017