



Grant Agreement 224442

Advancing Traffic Efficiency and Safety through Software Technology phase 2 (ATESST2)

Report type	Deliverable D3.1
Report name	Summary of EAST-ADL Update Suggestions
Dissemination level	PU
Status	Final
Version number	1.1

Authors**Editor**

Rolf Johansson

E-mail

rolf_johansson@mentor.com

Authors

Anders Sandberg

anders.sandberg@mecel.se

Andreas Abele

andreas.abele@continental-corporation.com

David Servat

david.servat@cea.fr

DeJiu Chen

chen@md.kth.se

Frank Hagl

frank.hagl@continental-corporation.com

Friedhelm Stappert

friedhelm.stappert@continental-corporation.com

Fulvio Tagliabò

fulvio.tagliabo@crf.it

Hans Blom

hans.blom@volvo.com

Henrik Lönn

henrik.lonn@volvo.com

Lars-Olof Berntsson

lars-olof.berntsson@volvo.com

Lei Feng

leifeng@md.kth.se

Magnus Persson

magnus.persson@md.kth.se

Mark-Oliver Reiser

mark-oliver.reiser@tu-berlin.de

Martin Törngren

martin@md.kth.se

Martin Walker

martin.walker@hull.ac.uk

Matthias Biehl

biehl@md.kth.se

Matthias Weber

matthias.weber@carmeq.com

Sandra Torchiaro

sandra.torchiaro@crf.it

Schierano Walter

walter.schierano@crf.it

Yiannis Papadopoulos

y.i.papadopoulos@hull.ac.uk

The Consortium

Volvo Technology Corporation (S)	VW/Carmeq (D)	Centro Ricerche Fiat (I)
Continental Automotive (D)	Delphi/Mecel (S)	
Mentor Graphics Hungary (H)	CEA LIST (F)	
Kungliga Tekniska Högskolan (S)	Technische Universität Berlin (D)	University of Hull (GB)

Revision chart and history log

Version	Date	Reason
1.0	2009-11-04	First release
1.1	2009-06-02	Update reflecting status at the end of ATESST2 project

Table of contents

Authors	2
Revision chart and history log	3
Table of contents	4
1 Introduction	6
2 Core System Description for Different Purposes.	7
2.1 Functionality/Features in Focus	8
2.2 Functional Building Blocks Realizing Features	8
2.3 Functional Building Blocks Allocated on Interconnected HW Resources	8
2.4 Software Components and Executable Runnables on a Resource Architecture	9
2.5 Relations Between Abstraction Levels	9
3 Requirements Engineering	10
3.1 Interfaces to Requirements Engineering Tool Infrastructure	10
3.2 Allocation of Requirements to System Model	10
3.3 Traceability of Requirements within System Model	10
4 Connecting Textual Requirements to Model Constraints	11
4.1 A General Patten: Non-functional Constraint – Model of the Constrained	11
4.2 Timing/Event Models – Timing Constraints	12
4.3 Fault/Failure Models – Safety Constraints	13
4.4 Other Non-Functional System Constraints	15
5 Dedicated Framework for Functional Safety According to ISO-DIS 26262	16
5.1 Identification of Item, Hazard and Hazardous Event	16
5.2 Risk Assessment and Safety Goal	18
5.3 Safety Requirements Allocated to Artefact Levels	18
5.4 Safety Case	19
6 Variant Management	21
6.1 Modelling Variants on Artefact Levels	21
6.2 Modelling Features in a Feature Tree on Vehicle Level	22
6.3 Rules for Binding/Configuration of Artefact Level Architectures by Feature Trees	23
7 Analysis/Optimization of Non-Functional System Properties	24
7.1 Finding Non-Functional System Properties of one Architecture Candidate	24
7.2 Architecture Evaluation (Assessment)	24
7.3 Architecture Optimization	24
8 Analysis of Functional (non-system) Properties	26
8.1 EAST-ADL Execution semantics	26

8.2 Transfer Function Definition26

9 References27

1 Introduction

This document is a summary of the EAST-ADL language. All concepts are briefly presented. For each concept there is a discussion of the planned and ongoing activities, indicating what the future state of the EAST-ADL will look like.

More detailed descriptions of the concepts can be found in the appendices. A complete documentation of all the details of the language can be found in D4.1.1.

The structure of this document is first a brief presentation of the four abstraction levels. Then follows a walk-through of concepts grouped in the following categories:

- Requirements Engineering
- Connecting Textual Requirements via Model Constraints to the nominal System Model
- Dedicated Framework for Functional Safety According to ISO-DIS 26262
- Variant Management
- Analysis/Optimization of Non-Functional System Properties
- Analysis of Functional (non-system) Properties

In the current status, the EAST-ADL Domain Model (DM) is compliant with the AUTOSAR Template UML Profile and Modeling Guide version 2.2.1 (part of the 3.1 release).

In coming versions of the EAST-ADL, there may also be a separate release compliant with AUTOSAR 4.0 and also even higher versions. This is however outside the scope of the ATESSST2 project.

2 Core System Description for Different Purposes.

EAST-ADL is an architecture description language (ADL). This means that the core concepts of the language are intended for describing the architectures of interest for the system under consideration. Around this core capability of describing the architectures in terms of components and their interconnections, there is a number of additional concepts, which are described in later sections. This section is about how to describe the architectures themselves.

In the EAST-ADL system model, there are four different abstraction levels. For each of these abstraction levels there is one dedicated architecture in which the architecture for the system under consideration is described.

In the domain model this is depicted as:

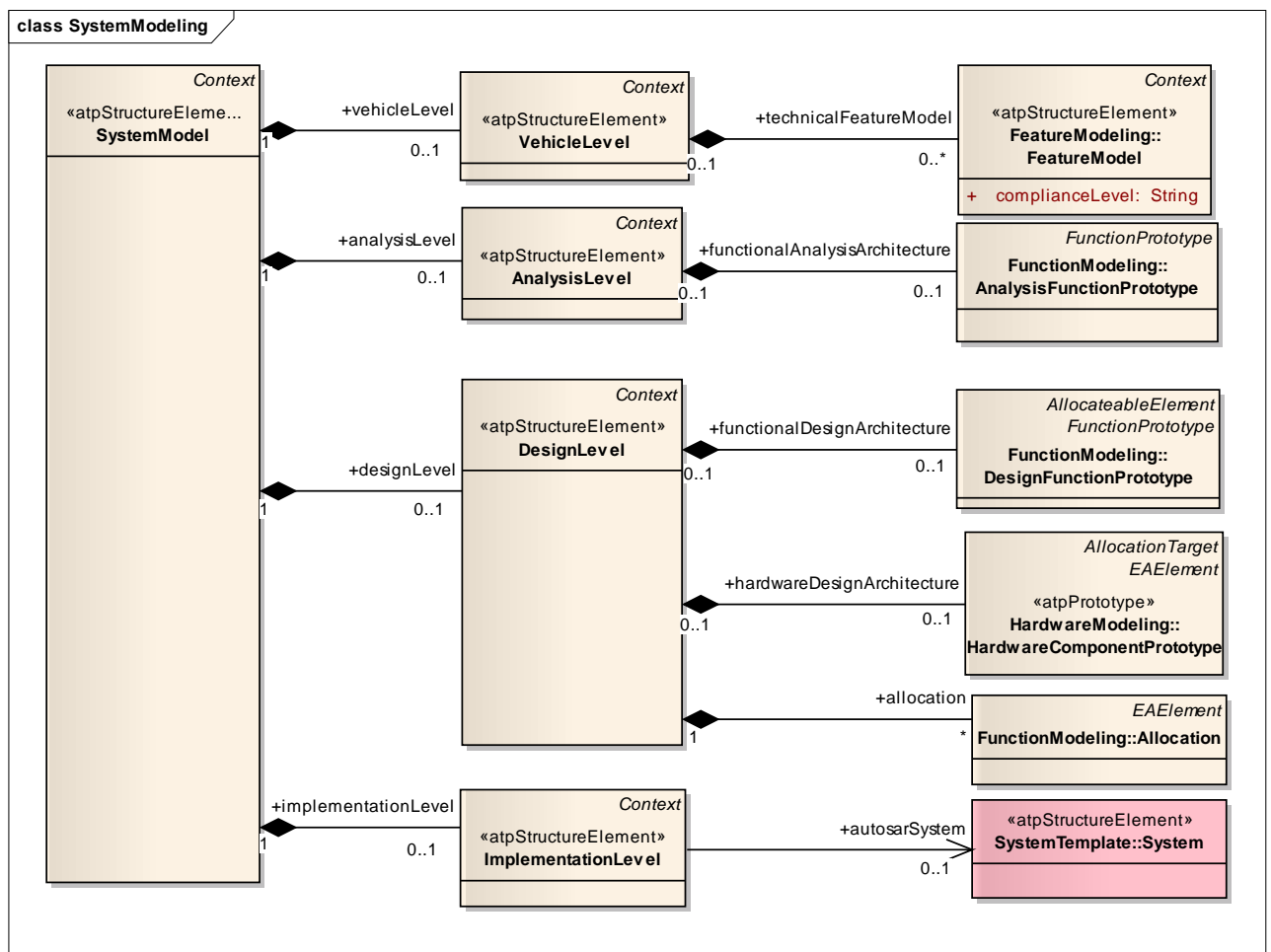


Figure 1. EAST-ADL System Model

Each of these four levels of abstraction is a complete system model of its own. The difference is that each of the four levels shows the system on a given level of abstraction. The following subsection describes the core concepts of each abstraction level.

Note that the system model then can be enriched with the concepts described in the following sections.

2.1 Functionality/Features in Focus

On the Vehicle level, the system is described in terms of one Technical Feature Model (TFM). The technical feature model is a structured hierarchical description of all functionality in the system. It is this technical feature model that can be said to be realized by the functional analysis architecture (FAA), the functional design architecture (FDA), and the AUTOSAR system (AS), respectively.

As is described in the following sections, the different nodes in the technical feature model might be the targets for concepts like allocation of requirements, item definition, specifying an error model, etc. It is also the target for the ADLRealize relation, i.e. when pointing out what part of an FAA realizes which nodes in the technical feature tree. However, there are no dependencies from this core concept to the other concepts. This means that the Technical Feature Model can be defined independently of the other concepts described in the following sections.

2.2 Functional Building Blocks Realizing Features

On the analysis level, the system is described by a functional analysis architecture (FAA). The FAA has the same scope as the technical feature tree.

The FAA is modelled as a number of interconnected FunctionPrototypes, where each FunctionPrototype itself can be (recursively) described as a number of interconnected FunctionPrototypes.

The architecture challenge of the FAA is to identify an appropriate set of such FunctionPrototypes, where a number of these are common for the realization of several different features in the technical feature model.

As is described in the following sections the different components in the FAA can be the target for concepts like allocation of requirements, specifying a behaviour, specifying a timing model, specifying an error model, etc. It is also the target for the Realization relation, i.e. when pointing out what part of an FAA realizes which nodes in the technical feature tree, and what parts of the FDA realize which parts of an FAA. However, there are no dependencies from this core concept to the other concepts. This means that the Analysis Architecture can be defined independently of the other concepts described in the following sections.

2.3 Functional Building Blocks Allocated on Interconnected HW Resources

On the design level, the system is described by three parts: one functional design architecture (FDA), one hardware design architecture (HDA) and one allocation. The FDA has the same scope as the technical feature tree and the FAA. Compared to higher levels of abstraction, this level also describes the resource architecture (HDA) on which the FDA is allocated. This allocation is modelled in the allocation.

The FDA is modelled as a number of interconnected FunctionPrototypes, where each FunctionPrototype itself can be (recursively) described as a number of interconnected FunctionPrototypes.

The architecture challenge of the design level is to

- identify an appropriate set of interconnected FunctionPrototypes
- identify an appropriate set of interconnected HW resources
- identify how to allocate the FunctionPrototypes on the HW architecture

As described in the following sections, the different components in the FDA can be the target for concepts like allocation of requirements, specifying a behaviour, specifying a timing model,

specifying an error model, etc. It is also the target for the Realization relation, i.e. when pointing out what part of an FAA realizes which parts in the FDA, and what parts of the AS realize which parts of an FDA. However, there are no dependencies from this core concept to the other concepts. This means that the Design Architecture can be defined independently of the other concepts described in the following sections.

2.4 Software Components and Executable Runnables on a Resource Architecture

On the implementation level, the system is described as an AUTOSAR system. This is done in terms of an association to an AUTOSAR model.

There is ongoing work to harmonize the AR and EAST-ADL concepts. Some of the concepts described in the following sections are already present in AUTOSAR and some are not. In either case there is no dependency from the core concepts of AUTOSAR to this additional concept. Each of these concepts can be applied for enriching a system model containing an AUTOSAR system. This is further elaborated for each concept in the following sections.

2.5 Relations Between Abstraction Levels

All the different abstraction levels of the system model are complete w.r.t the system to design. This means that it is possible to identify the relation between elements of the description on one level with elements of another abstraction level. The means in the EAST-ADL language for modelling this relation is by Realization. Realization is a relationship metaclass, which signifies the realization relationship between two sets of model elements, one (realized) representing a more abstract specification and the other (realizedBy) representing an implementation of it.

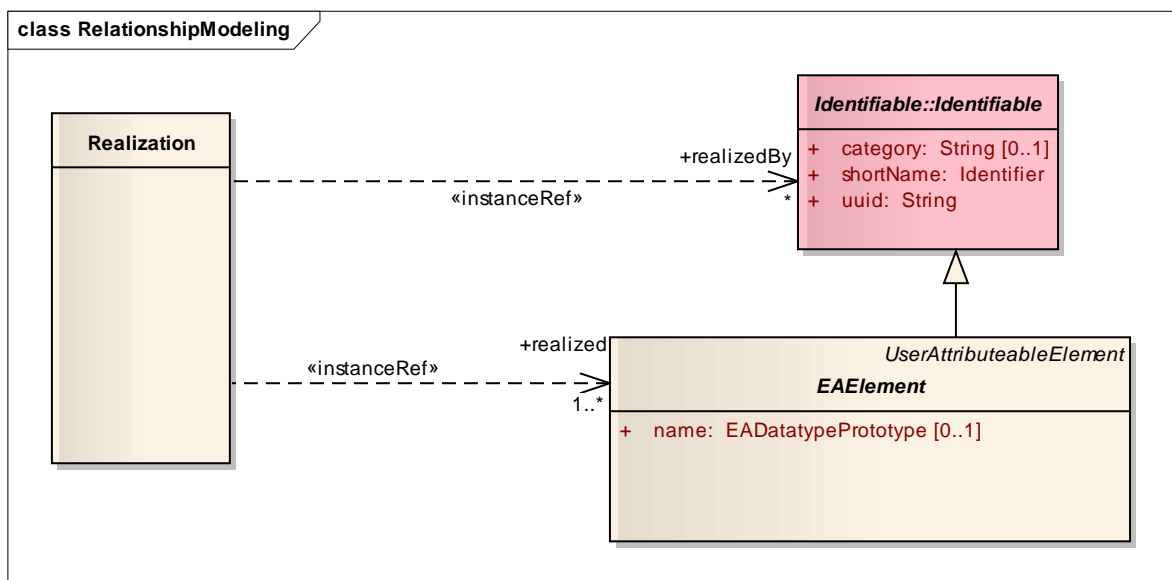


Figure 2. Relationship Modelling

3 Requirements Engineering

3.1 Interfaces to Requirements Engineering Tool Infrastructure

In general, it is assumed that all requirements engineering (RE) work at any automotive company, is done independently of EAST-ADL. However, given that there are requirements managed by an RE tool, the challenge for EAST-ADL is to be able to integrate these in a model-based framework.

The ability of EAST-ADL is to import a set of requirements on RIF (Requirement Interchange Format). This is covered in the Interchange package of the EAST-ADL domain model. For the interchange there is a dedicated RIF area in which all requirements are contained in tree hierarchy of requirement containers. The sub-nodes of a node are ordered, which means that the container tree may describe a requirement document preserving the order.

3.2 Allocation of Requirements to System Model

Each Requirement is allocated to exactly one abstraction level. This is an important part of a RE work flow (see also the methodology documentation). Requirements can be allocated to features or to artifact level architectures. In the domain model, Satisfy models the allocation. This generality in allocation is modeled by using the general meta class Identifiable as the target (role name satisfiedBy).

3.3 Traceability of Requirements within System Model

The DeriveRequirement is a relationship meta-class, which signifies a dependency relationship between two sets of Requirement, showing the relationship when a set of Requirement (derived) is derived from another set of Requirement (derivedFrom).

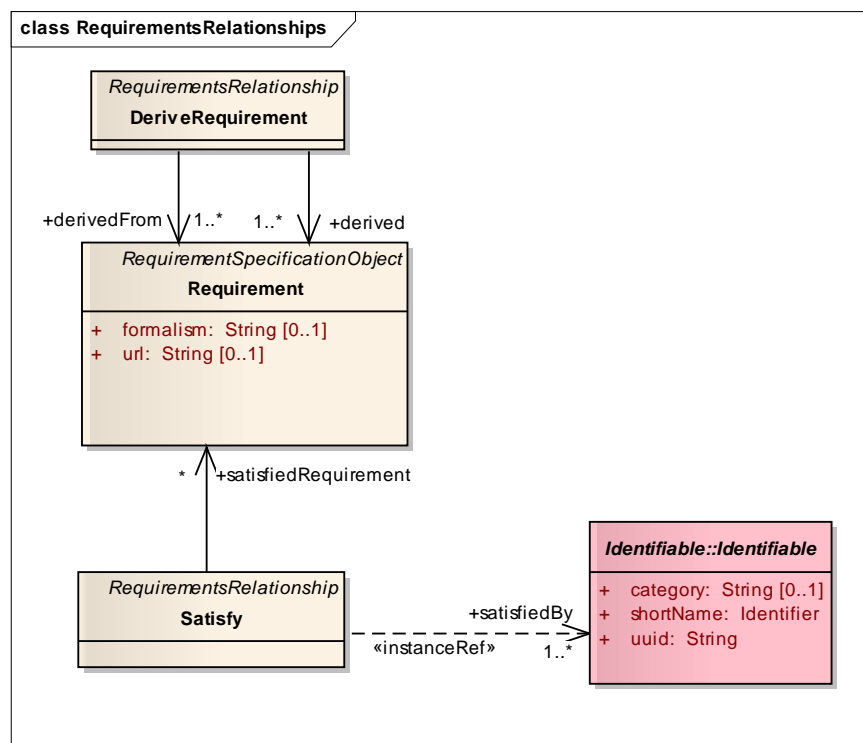


Figure 3. Requirements Relationships

4 Connecting Textual Requirements to Model Constraints

In order to use the system model for analysing the non-functional system properties, EAST-ADL has in its scope a number of constraint models. Such a constraint model should have a precise syntax in the domain modelling language to express a constraint for a non-functional system property. Examples of such non-functional system properties relevant for the scope of EAST-ADL are:

- Timing
- Safety
- Cost
- Weight
- ...

Such a constraint can be used for different purposes. Firstly it can be a way to more formally model a given textual requirement. Secondly, it can be a way of formalizing properties of a model (e.g. established by analysis or inspection). In this sense it captures what is promised by a supplier, i.e. a kind of data sheet purpose. Thirdly it can be a placeholder for a not yet received external requirement.

For each category of constraint model, there is a corresponding model giving the constraint a well-defined meaning in the system model context. For example a timing constraint is dependent on the timing/event modelling, and a safety constraint is dependent on fault/failure modelling. This is further described in the following sub-sections on the different kinds of constraints.

The timing model incorporates the results of the ITEA2 project TIMMO. The safety model could be further elaborated. A general model for “trivial” non-functional constraints like cost and weight is also part of this EAST-ADL release.

4.1 A General Patten: Non-functional Constraint – Model of the Constrained

The general pattern to express non-functional system constraints is by referencing a model of the non-functional system property to be constrained. In UML this could be described by the following picture:

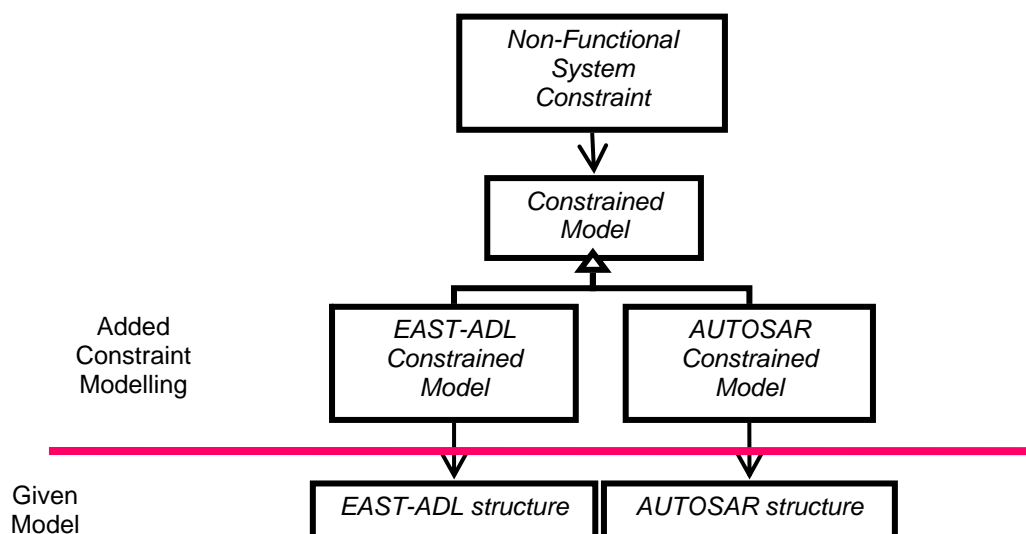


Figure 4. General constraint pattern

This means that the constraint could be the same for EAST-ADL and AUTOSAR. The model of what is constrained has a general part in common, and then there should be specializations for EAST-ADL and AUTOSAR respectively. The only non-functional system constraint existing for AUTOSAR so far is the timing constraint, which follows the above pattern. Hence, it should also be possible to define other non-functional system constraints in a similar way.

By having an association from the constrained model to the given system model in which the constraints are referencing, these constraint models can be added afterwards to system models developed without any constraints included. In other words the given model has no dependency on the constraint modelling, but the constraint modelling is dependent on the rest of the system model for its meaning.

4.2 Timing/Event Models – Timing Constraints

EAST-ADL fully integrates the domain model defined in the ITEA2 project TIMMO. TIMMO denotes this meta model Timing Augmenting Description Language, TADL.

In general the TADL can be described by the following UML diagram:

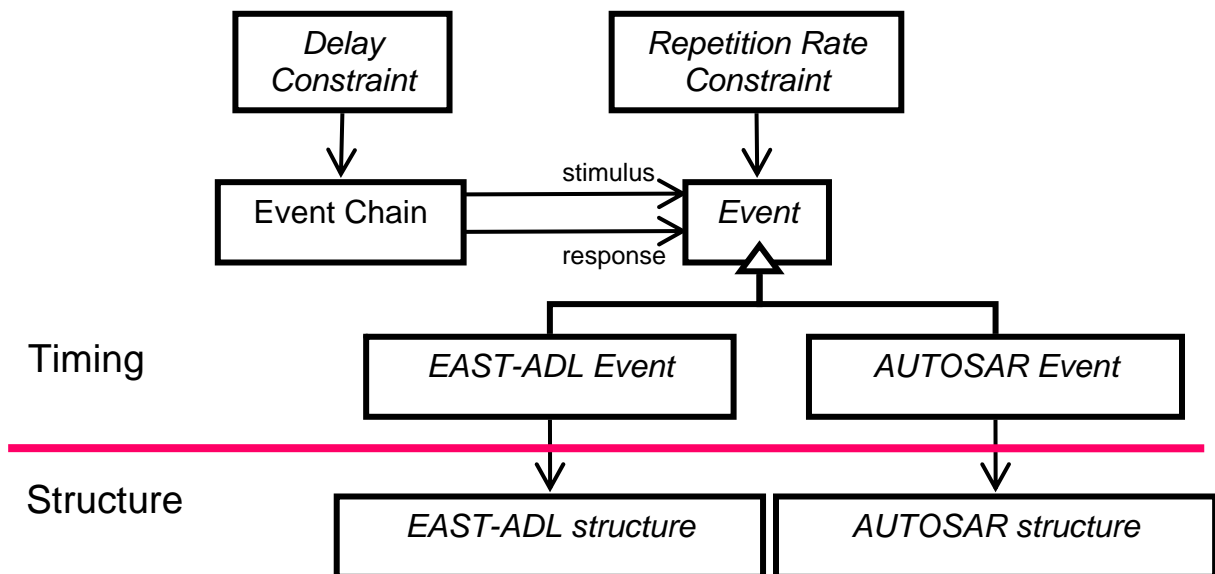


Figure 5. Timing Constraints in EAST-ADL

There are a number of predefined Events in the domain model. Each such event is a class of its own, specializing the Event class. The semantics of each event is defined in the domain model, and for each predefined event it is also given what to reference in the structural system model. For example there is a predefined event called EventFunctionFlowPort. This event is when data is available on the FunctionFlowPort of a FunctionPrototype. This event has an association to the actual FunctionFlowport in the system model. For each predefined event, it is prescribed what model element it should have an association to. Hence, there is one set of predefined Events having associations to AUTOSAR modelling elements, and another set of predefined event having associations to EAST-ADL modelling elements.

There are number of timing constraints defined in the domain model. Each of these depends on the existence of one or more predefined Events. However, they all have associations to the abstract class Event, i.e. they are all the same independently of EAST-ADL or AUTOSAR context.

The timing constraints are:

EventConstraints:

- * PeriodicEventConstraint
- * SporadicEventConstraint
- * PatternEventConstraint
- * ArbitraryEventConstraint

DealyConstraints:

- * ReactionConstraint
- * AgeTimingConstraint
- * InputSynchronizationConstraint
- * OutputSynchronizationConstraint

The EventConstraints constrain how often the same event may occur. They all have an association to the predefined event that is constrained. The different specializations of the EventConstraint have different attributes giving the full semantics of the constraints

The Delay constraints all have an association to an EventChain. An EventChain has one or more events in the role of stimulus and one or more events in the role of response. An AgeConstraint defines how long ago a stimulus could have happened for a given response. Similarly a ReactionConstraint defines how long time in the future a response may occur after a given stimulus. The difference between these two DelayConstraints is significant for multi-rate systems where under- and over-sampling may occur on the way between stimuli and responses. By separating the ReactionConstraint from the DelayConstraint, there is enough expressiveness in the language to deal with this problem. AgeConstraints can be composed and decomposed to AgeConstraints in an unambiguous way. Similarly, ReactionConstraint can be composed and decomposed to ReactionConstraint unambiguously. But composing AgeConstraint with ReactionConstraint would not be well-defined in a multi-rate system.

The InputSynchronizationConstraint defines how much the stimuli could differ in time from each other. Similarly the OutputSynchronizationConstraint defines how much the responses may differ from each other.

A more detailed documentation of the Timing Constraints and the Timing/Event models can be found in the public deliverables of the TIMMO project.

4.3 Fault/Failure Models – Safety Constraints

The Error Model of the EAST-ADL follows the general pattern of non-functional constraints. Without going into all the details, it could be described as in the following figure:

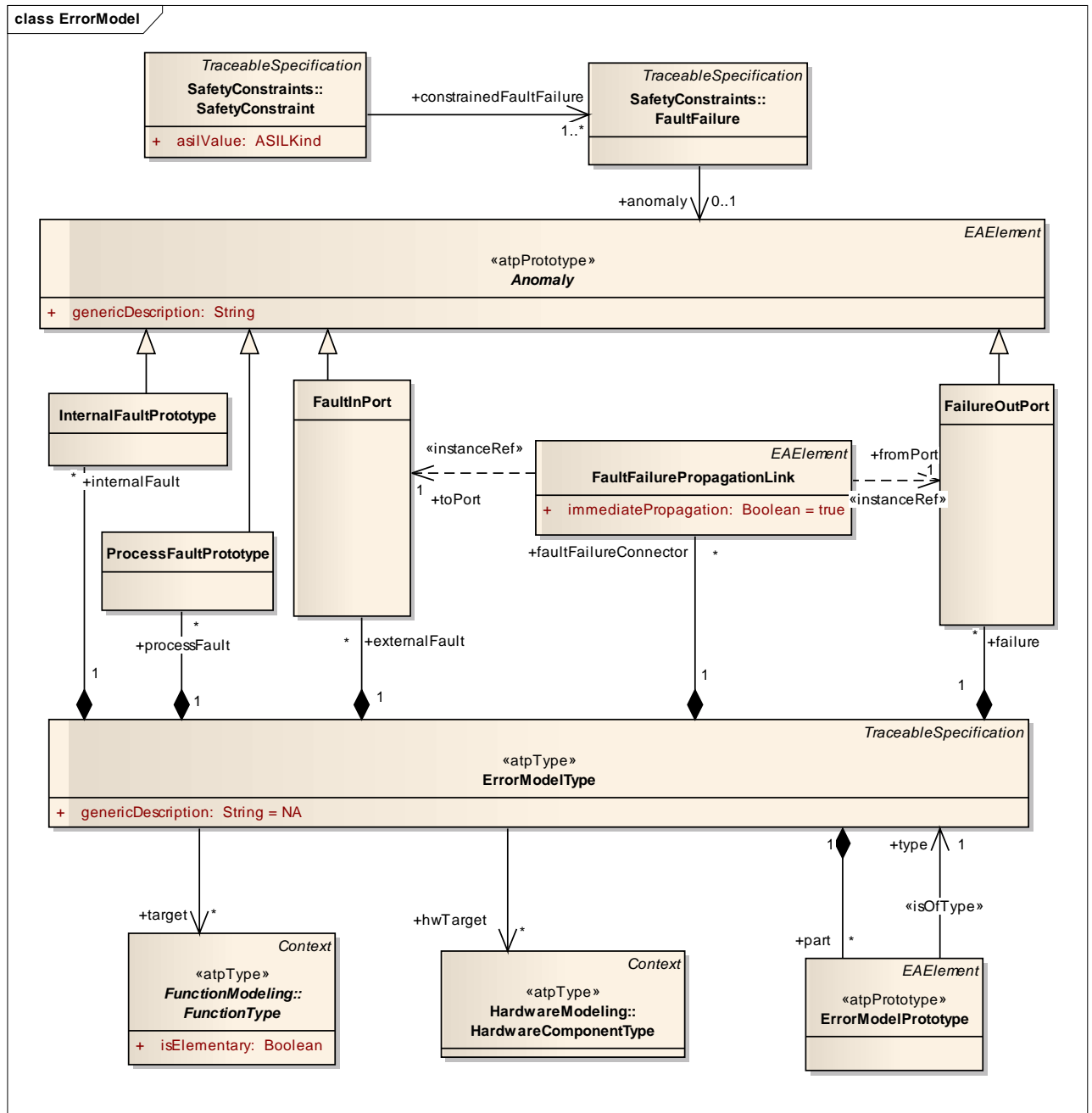


Figure 6. EAST-ADL Error Model

This means that each SafetyConstraint constrains a certain fault or failure value (FaultFailure) of either: a failure on the output (FailureOutPort), a fault on the input (FaultInPort), a process related fault (ProcessFaultPrototype) or a fault internal to the border of the actual error model (internalFaultPrototype). The constraint says how severe it is, i.e. what ASIL level it has. Furthermore it should specify how large the deviation from the nominal behaviour (toleratedDeviation) should be in order to be considered a safety-critical fault or the failure having the associated safety constraint with this ASIL.

The ErrorModel structure could be made independently of the structure in the nominal model. Each ErrorModel defines what part of the nominal model it describes by having an association to a core element either of EAST-ADL or of AUTOSAR. The ErrorModel follows the same type-prototype pattern for hierarchical models as the Function for nominal structure.

The ErrorModels can be connected to each other via their ports by means of fault-failure propagation links (FaultFailurePropagationLink) owned by the containing ErrorModel. Each ErrorModel has an internal behaviour describing the relation between the external and internal faults, and the failures of the ErrorModel. Hence the fault activation and error propagation inside one ErrorModel is described by the ErrorBehavior and the fault/failure propagation between ErrorModels is described by the structural connection of the fault/failure ports and propagation links. A more detailed documentation of the Fault/Failure Models and the Safety Constraints can be found in annex A3.2.

4.4 Other Non-Functional System Constraints

There are no other dedicated constraint models in EAST-ADL except those for timing and safety. The idea is that all other (trivial) non-functional constraints can be modelled by the same generic concept as depicted below.

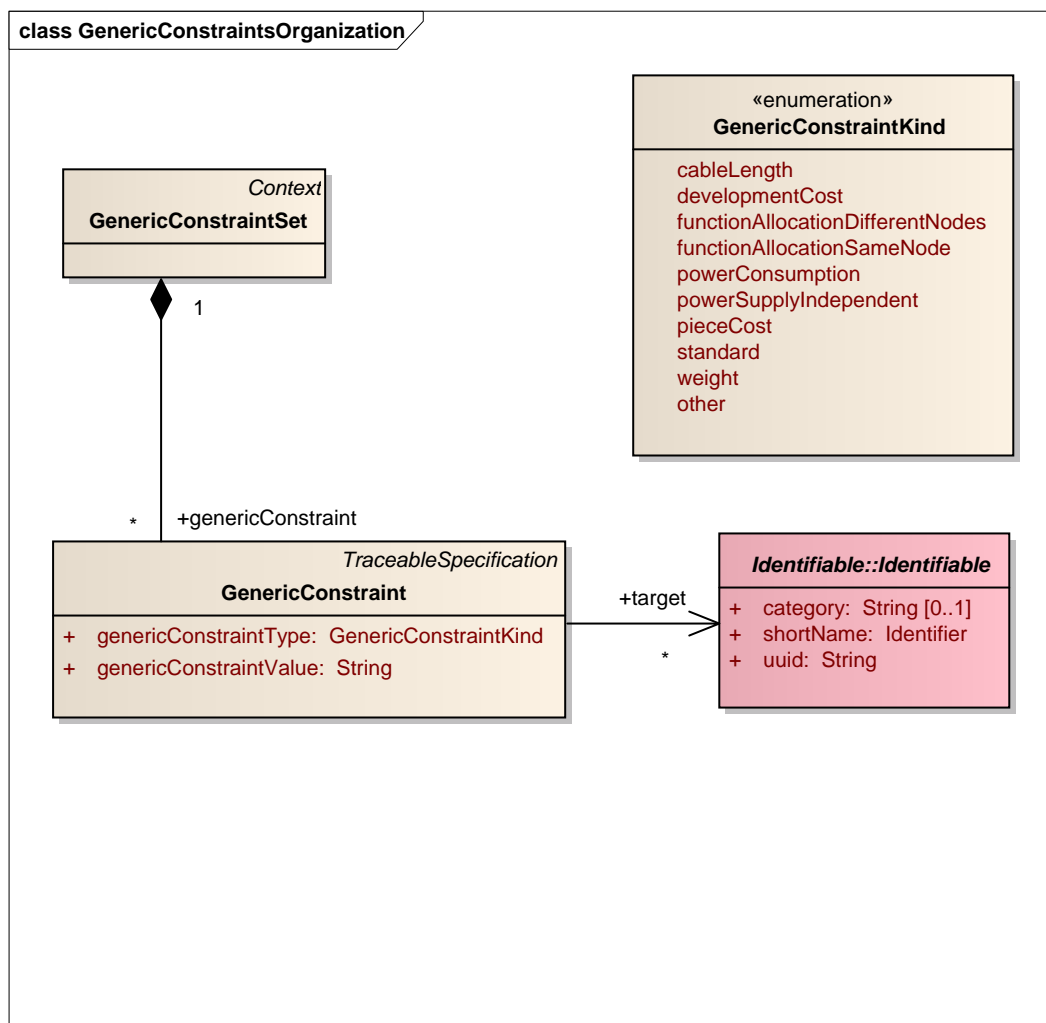


Figure 7. General constraint pattern

5 Dedicated Framework for Functional Safety According to ISO-DIS 26262

The EAST-ADL aims to provide a modelling context for the prescribed activities of the automotive safety standard ISO/DIS 26262. The ambition of EAST-ADL so far is to model what is needed for safety requirements and for safety cases. At a basic level, the safety requirements model what should be done and the safety case shows how those requirements are fulfilled.

For the identification and structuring of safety requirements, ISO/DIS 26262 has a general approach as depicted in the left part of the figure below. The complete picture illustrates that the different steps can be mapped to the different abstraction levels of EAST-ADL.

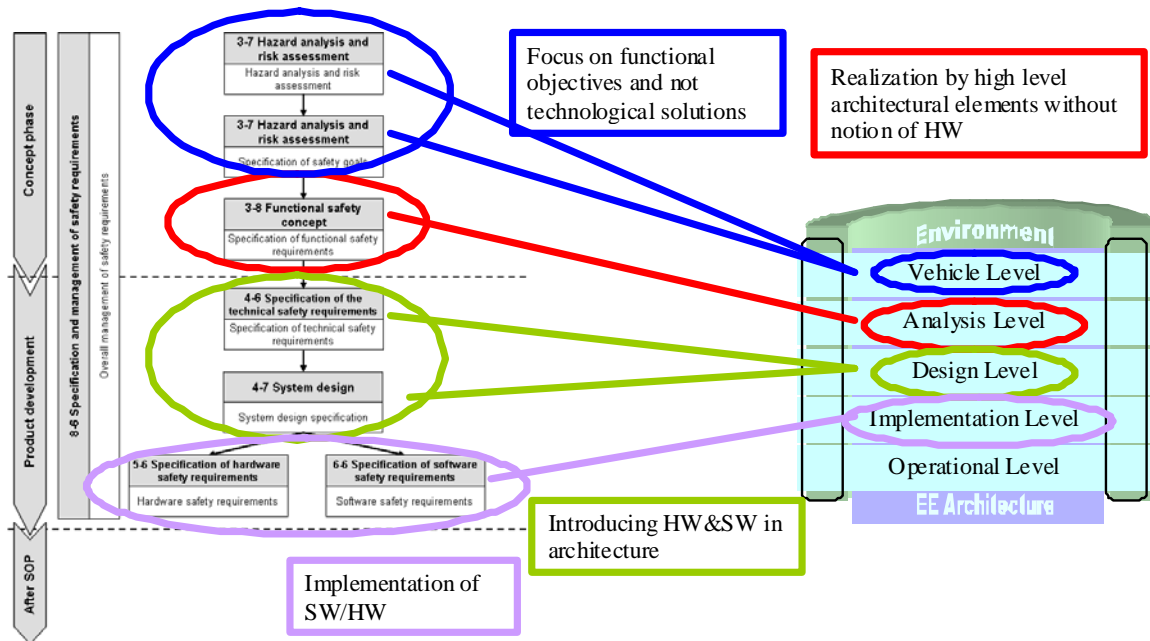


Figure 8. Relation between ISO phases and EAST-ADL abstraction levels

For the artefact levels (all but the vehicle level), this means that a number of safety requirements should be allocated to the system model of that abstraction level. In EAST-ADL this should be done by modelling each safety requirement as a safety constraint, which is then allocated to the architecture. This is further described in the section about Safety Constraints and Fault/Failure Models.

For the vehicle level, the modelling support for the activities is further described in the sub-sections below.

Besides the concepts for safety requirements, EAST-ADL also has a modelling framework for safety cases. This is further described as a sub-section at the end of this section.

A more detailed description of these concepts can be found in annex A3.2.

5.1 Identification of Item, Hazard and Hazardous Event

The "Item" in the terminology of ISO/DIS 26262 is identified in EAST-ADL as a node in the Technical Feature Model. The purpose of an Item is to confine the scope of the activities in the life-cycle, i.e. the "functional" aspect of Functional Safety. The complete E/E system is thus

categorized as a disjoint set of Items. The Technical Feature Model is also a complete description of the all the functionality of the E/E system. The Items of ISO/DIS 26262 can thus be identified as nodes in the Technical Feature Model, in such way that all the leaf nodes of this tree belong to exactly one Item.

The EAST-ADL domain model concepts for Hazards and Hazardous Events etc are depicted in the figure below. In general, the Hazards are identified by negating the functionality of the Item of consideration.

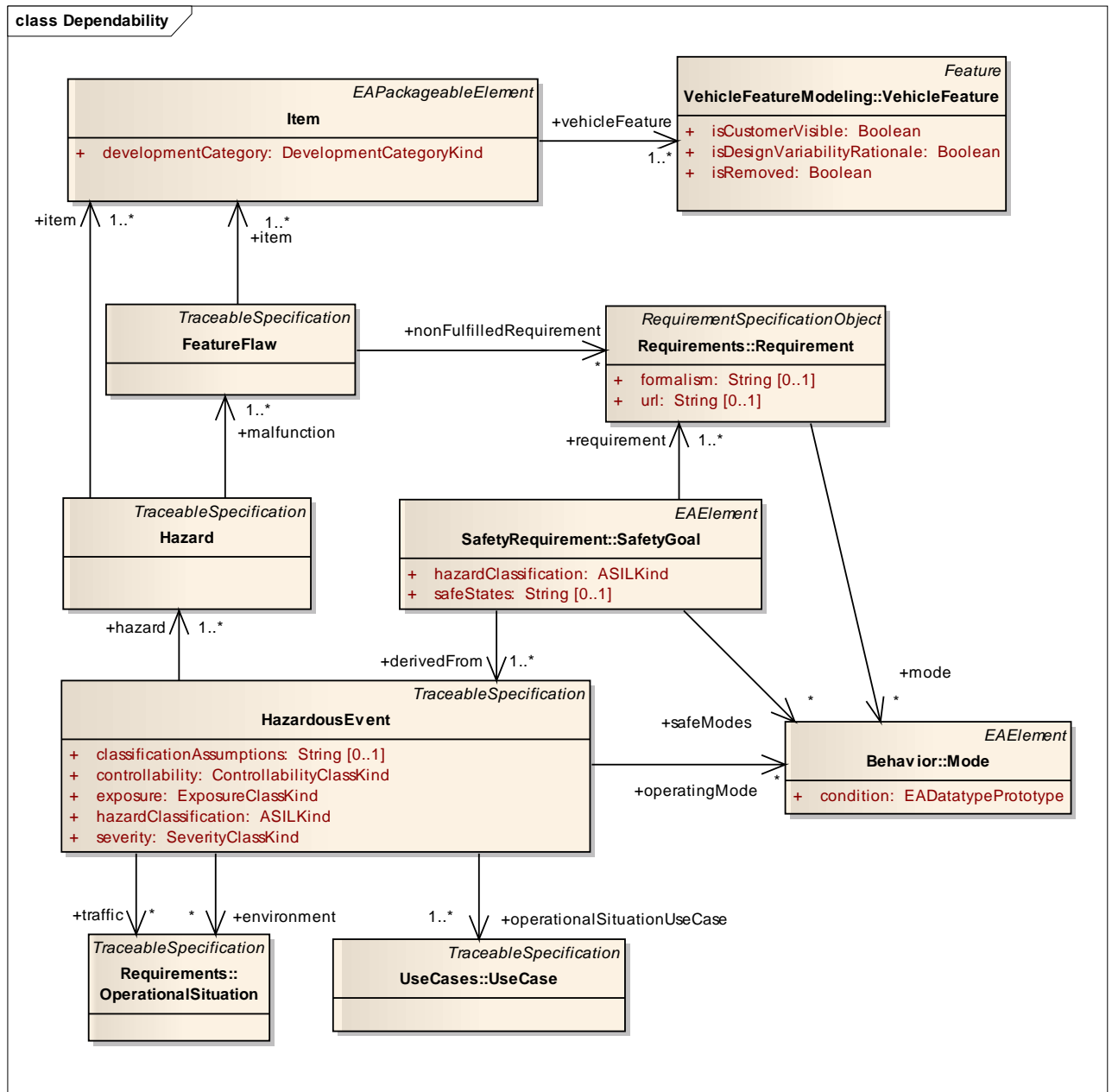


Figure 9. Modelling Support for ISO 26262

Risk Assessment is done on (a set of) Hazardous Events. The Hazardous Event is modelled as a Hazard in combination with a given operational situation. The operational situation is in EAST-ADL modelled as a combination of traffic and environment situations and a use case (specializations of Traceable Specification) for the Feature corresponding to the Item. This set could as a special

case include all Hazardous Events of a given Hazard, and in such cases the risk assessment is done with respect to a certain Hazard.

5.2 Risk Assessment and Safety Goal

A Safety Goal could be seen as a Safety Requirement on the vehicle level.

A Safety Goal is conceptually similar to a Safety Requirement, and it is modelled similarly in EAST-ADL. Then a text-based Safety Goal (modelled as Requirement) could also be associated to a syntactically and semantically more precise SafetyConstraint that is allocated to the system as described above in section 4.3.

5.3 Safety Requirements Allocated to Artefact Levels

A functional safety concept consists of a set of allocated functional safety requirements. In EAST-ADL terms this is modelled as a FunctionalSafetyConcept that references all Requirements it contains. The allocation is either modelled directly by the Satisfy Relation as described in the paragraph above on allocation of requirements, or it is done by expressing these textual requirements in terms of SafetyRequirements. Each such SafetyRequirement is then allocated directly by the reference via the ErrorModel to the element in the nominal architecture.

What is said above for the functional safety concept is also true for the technical safety concept. The only difference is that for the functional safety concept, the associated safety constraints are allocated in the model on design level instead of analysis level.

For the safety requirements for hardware and software respectively, associating these to safety constraints depends on this concept being available in AUTOSAR. So far, there is no such concept in AUTOSAR, but with the given overall concept it should be possible to add to the AUTOSAR domain model.

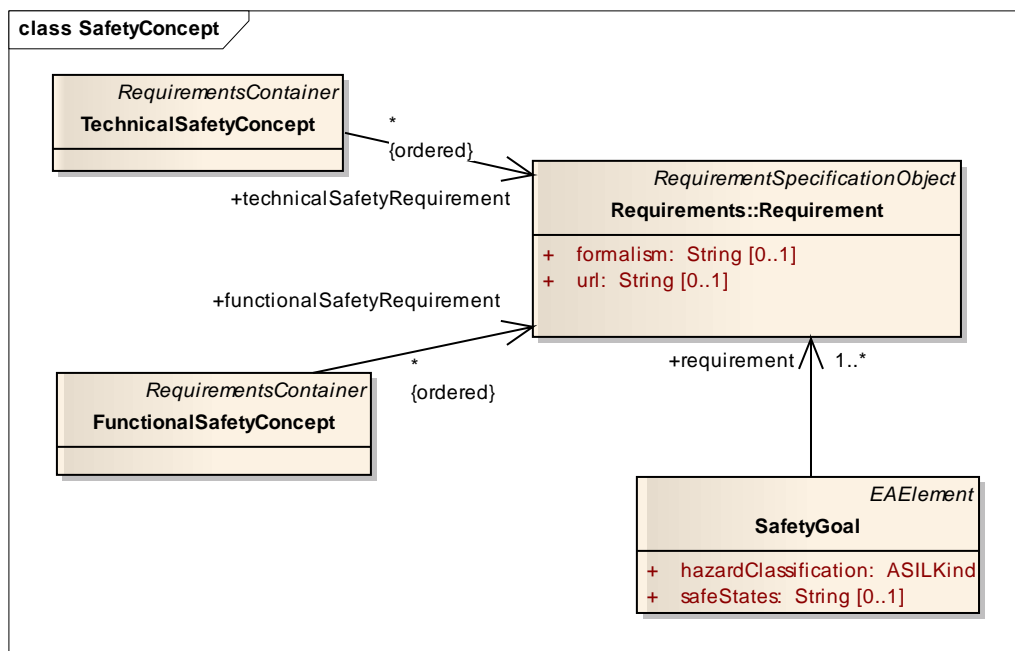


Figure 10. Safety Concepts

5.4 Safety Case

ISO/DIS 26262 requires the existence of a safety case, but there is no detailed prescription of how this should be done.

The domain model of EAST-ADL models the arguments (Warrant) for how the given evidences (Ground) can show (Claim) that system is safe in the given context. The SafetyCase itself is a specialization of the abstract class TraceableSpecification.

The following diagram shows how the safety case is made up of Warrants (arguments), Claims and Grounds (evidence).

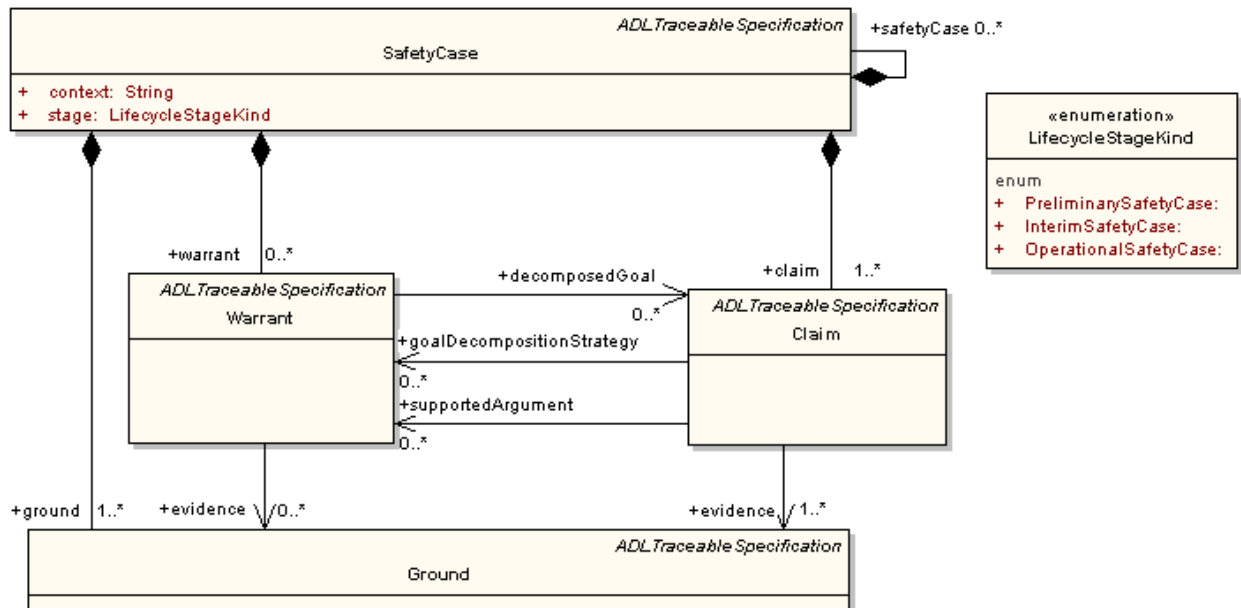


Figure 11. Safety Case

For each Claim there should be supporting arguments. This is defined by an association to the Warrants having the role of supportedArgument. The set of claims should cover the set of SafetyRequirements, i.e. for each applicable SafetyRequirement there should be a Claim saying that it is fulfilled. The claim is supported by arguments saying how the given evidence is enough in this context.

The following diagram of the domain model depicts how each Claim is associated with one or several safety requirements.

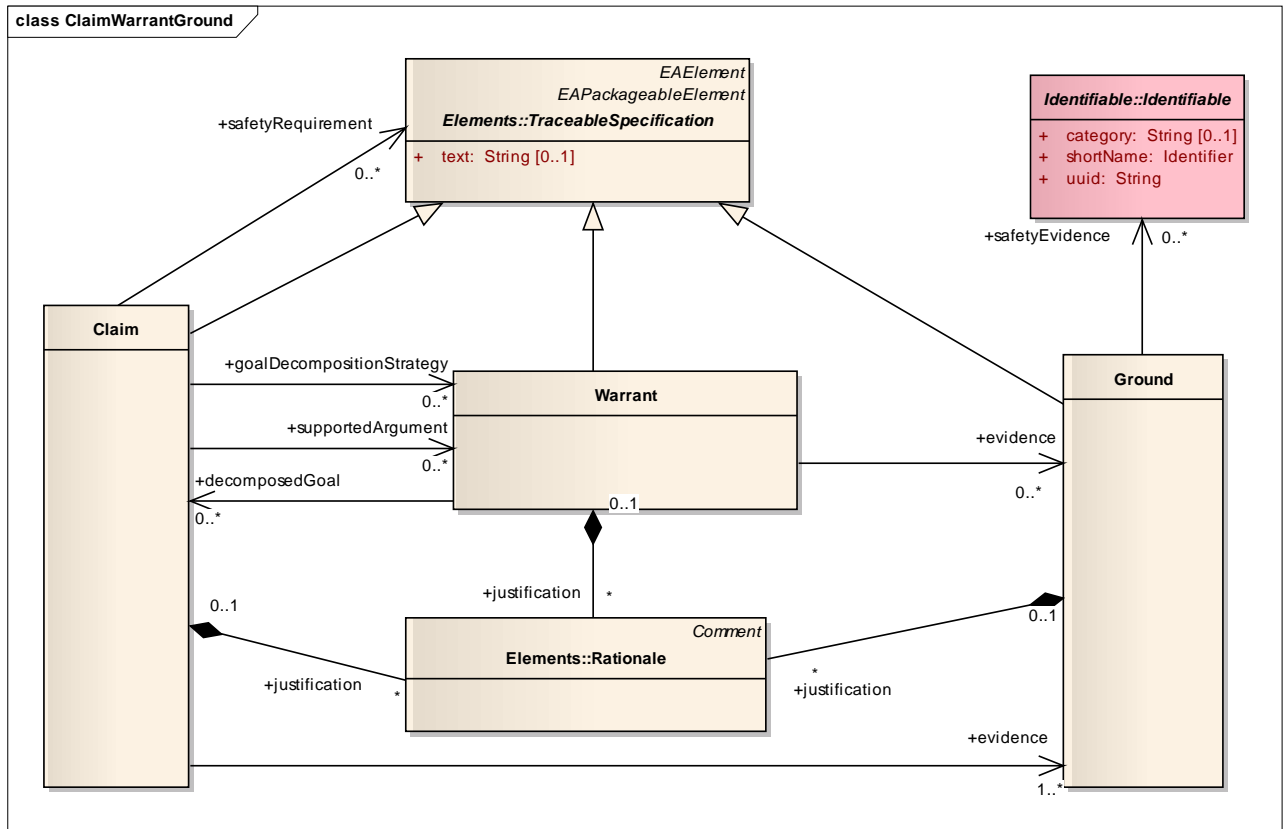


Figure 12. EAST-ADL Requirements (TraceableSpecifications) related to a Safety Case

6 Variant Management

In variant management the task is to model at least

- possible variants on artefact level
- possible features and their dependencies in a technical feature model on vehicle level
- configuration information showing how the configuration of features is binding the variants on artefact level

Furthermore, the scope of EAST-ADL also covers feature modelling on artefact levels, and product line engineering feature trees that can partly configure a technical feature tree.

The product feature trees on the vehicle level are there to give information about the product line structure, the markets and so on.

The following pictures illustrates how some of the above concepts are related:

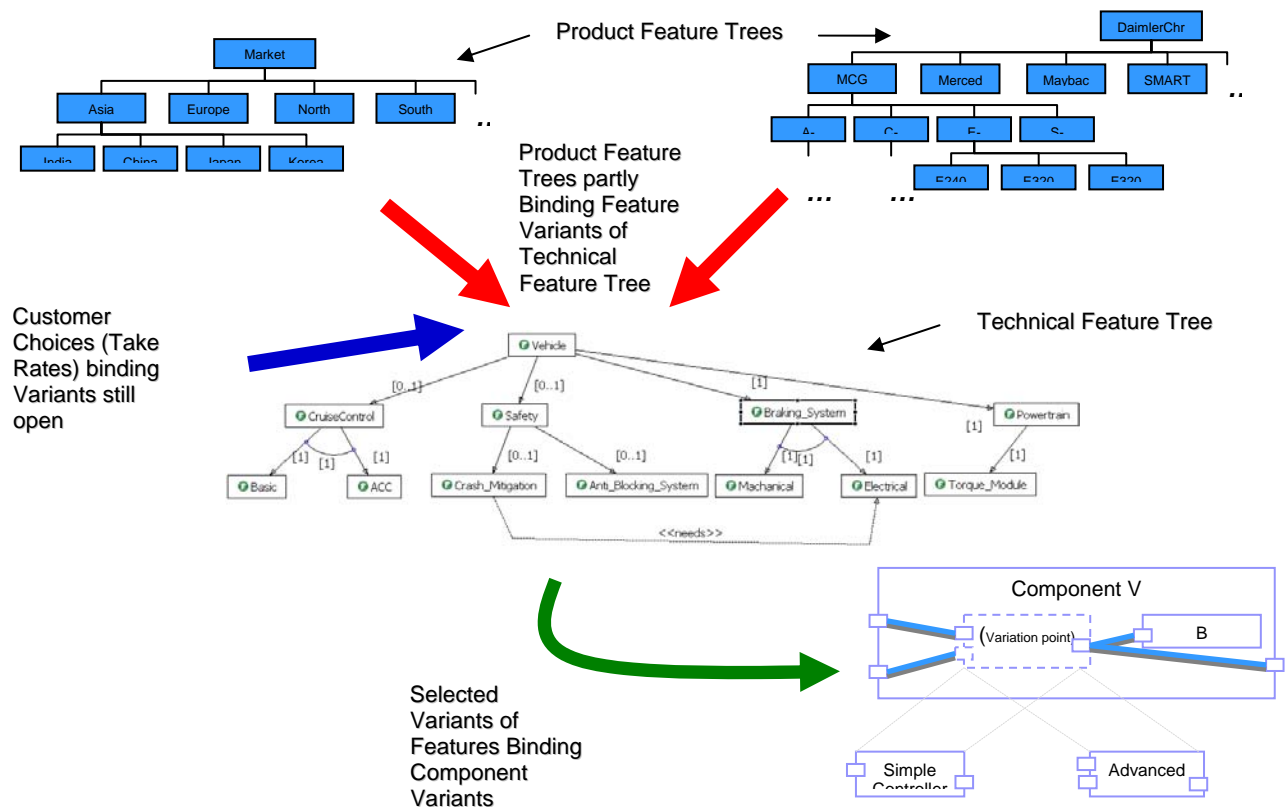


Figure 13. Variant Management Concepts

Among these, only modelling of variants on the implementation level is inside the scope of AUTOSAR. That means that the configuration of the AUTOSAR variants can be done by the configuration modelling of EAST-ADL.

6.1 Modelling Variants on Artefact Levels

This version of EAST-ADL is fully compliant with AUTOSAR version 3, in which there is no concept for variants at all. For AUTOSAR version 4 variant handling is supported. Even though the means for expressing variability differs between AUTOSAR version 4 and EAST-ADL, it could easily be

connected in a common system model where the feature level variability could be used to configure also an AUTOSAR architecture. The current scope of EAST-ADL is however confined to be complementary to AUTOSAR version 3, and this also gives the capability of introducing artefact variants on AUTOSAR level by means of a Variability extension package of EAST-ADL.

EAST-ADL defines an abstract class called VariableElement which may reference any element specializing Identifiable. The referenced Identifiable is thus declared as being optional. This means that elements on all artefact levels including AUTOSAR elements on the implementation level can be expressed as optional in this way.

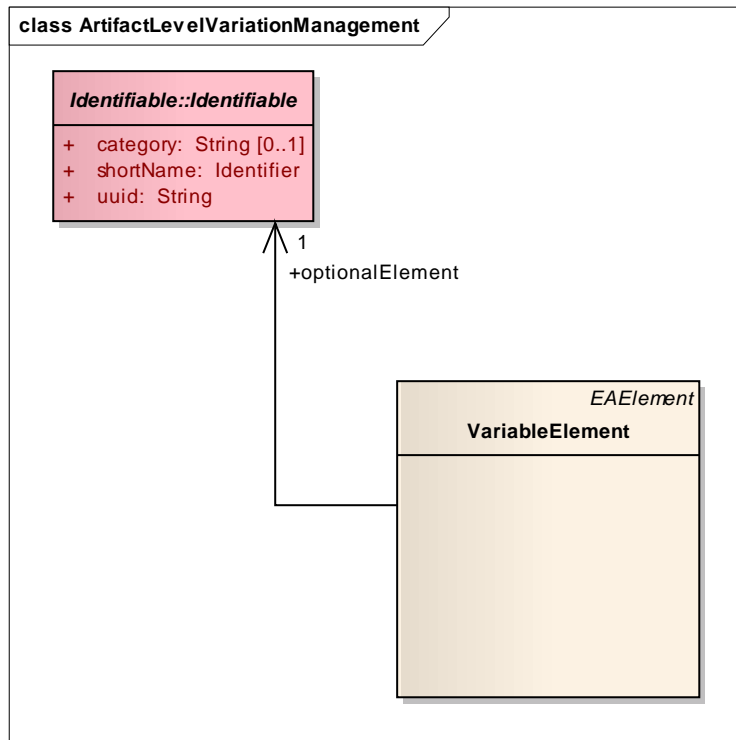


Figure 14. Artefact Level Variation Management

A more detailed description of the artefact level variability is given in Annex A3.3

6.2 Modelling Features in a Feature Tree on Vehicle Level

Variability management starts on the Vehicle Feature Level, where model range features and variability are viewed. At this point, the purpose of variability management is to provide a highly abstract overview of the variability in the system such as the complete system together with dependencies between these variabilities. A “variability” in this sense is a certain aspect of the complete system that changes from one variant of the complete system to another. “Abstract” here means that, for an individual variability, the idea is not to specify *how* the system varies with respect to this variability but only *that* the system shows such variability. For example, the front wiper may or may not have an automatic start. At vehicle level, the impact of this variability on the design is not defined; only the fact that such variability exists is defined by introducing an optional feature named "RainControlledWiping".

In addition to this technical variability on vehicle level, captured in the so-called technical feature model, there may be one or more feature models defined on vehicle level that provide orthogonal views on this technical variability, for example an end-customer view. These product feature models are linked to the core technical feature model by way of one or more configuration decision models, that define how to configure the technical variability in the core technical feature model

depending on a given configuration of the orthogonal view represented by the product feature models.

6.3 Rules for Binding/Configuration of Artefact Level Architectures by Feature Trees

The general concept "ConfigurationDecision" enables configuration of all variable elements including nodes in a product feature model, nodes in a technical feature model, and elements in any artefact architecture, including AUTOSAR. This is enabled by allowing the target of a ConfigurationDecision be any Identifiable. The source of a configuration decision could also be any of these possibilities, or just a constant expression without dependencies to any other part of the model. The selection criterion defining the logical expression is a specialization of the FormulaExpression also used in AUTOSAR 4.0.

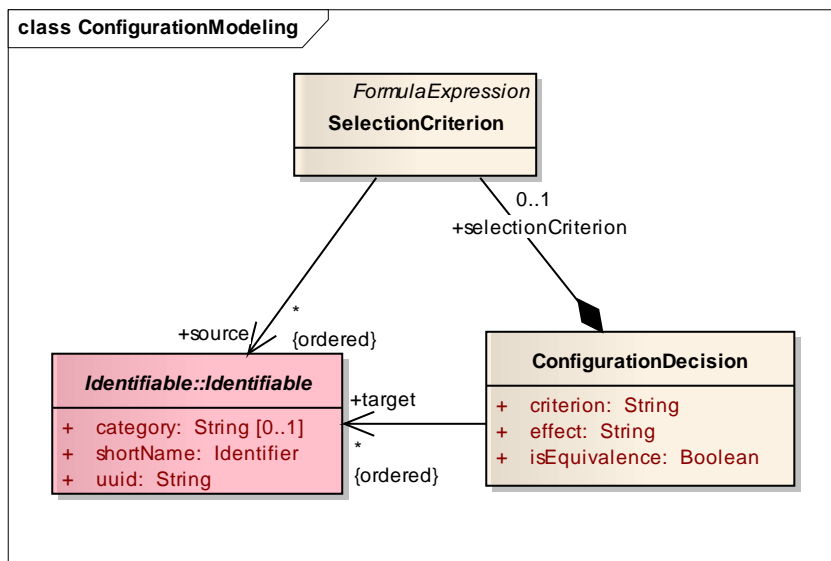


Figure 15. Configuration Modelling

A more detailed description can be found in appendix A3.3.

7 Analysis/Optimization of Non-Functional System Properties

So far, no dedicated language concepts are defined in the domain model for supporting analysis and/or optimization of non-functional system properties, except the capabilities in the area of constraints described in a previous section. However, the language provides a formal definition of the system which in itself is a basis for analysis.

This section describes the analysis/optimization process and discusses what this might lead to in terms of required language concepts.

7.1 Finding Non-Functional System Properties of one Architecture Candidate

In our context, architecture analysis is related to specific analysis reasoning frameworks used to predict different quality attributes. Quality attributes are specific criteria that can be used to judge the operation of a system (e.g., performance, safety, availability, resource usage). The terms “performance analysis” and “safety analysis” are commonly used to describe analysis of those quality attributes. Note that one architecture may include variability. Performing analysis of an architecture includes analysis of all its valid variants. Given assumptions on the distributions of binding decisions on the vehicle level, and the configuration decision information, one resulting analysis value reflecting all variants can be derived for a given attribute.

The concepts for constraints of non-functional system properties as described in a previous section together with the architecture model represent the language concepts needed to support architecture analysis.

7.2 Architecture Evaluation (Assessment)

Evaluation is the process of examining options and assessing their relative merits. Architecture evaluation is often used to describe assessment either before or after architecture analysis. It is also often used to describe the process of proving whether the quality attributes of an architecture option meet system requirements. Evaluation can also be used to select one out of many architecture options or candidates. Architectural options might be architecture patterns, scheduling policies, fault tolerant schemes, alternative subsystems, or anything else about which a decision is needed. When the space of possible solutions is large, evaluation can be done by “design space exploration”. Different trade-off strategies can be applied to make the “best” architectural selections. One strategy is optimization.

In addition to the concepts for constraints of non-functional system properties as described in a previous section, the notion of different architecture candidates need to be represented. Currently this is done using the variability mechanism. The binding time is set to SystemDesignTime to indicate that this variability is resolved by optimization.

7.3 Architecture Optimization

Optimization is the process of finding the best solution from all feasible solutions regarding certain objectives and subject to certain constraints. Maximizing performance and minimizing fuel consumption of a vehicle, and minimizing weight while maximizing the strength of a particular component are examples of optimization problems. A (multi-objective) optimization problem can be formalized as the searching of a solution x – i.e. an optimal design - (element of solution space X of possible designs), which optimizes a vector of objective functions $f(x) = [f_1(x), f_2(x), f_3(x), \dots, f_n(x)]$. For example, one optimization problem is to search for Pareto optimal (i.e. non-dominated) solutions. A solution x_1 dominates another solution x_2 if x_1 matches or exceeds x_2 in all objectives.

All these three activities can be either manual, semi-automatic (under guidance), or automatic (according to some algorithm). This means that, independently of tool support aspects, conceptual and methodological aspects should cover capabilities generic enough to be useful for different analysis, evaluation, and optimization concerns. In a stepwise refinement of the EAST-ADL approach, one can identify six levels or steps:

1. Means to model the system with enough information to do analysis
2. Means to automatically analyse EAST-ADL model
3. Back-annotation of analysis results in to EAST-ADL model
4. Means to define the design space
5. Assessment of architecture alternatives
6. Optimization, i.e. automatic modification of EAST-ADL model

For optimization to take place, it must be possible to define a design space. This may be achieved by means of the variability mechanisms in EAST-ADL (or an extension thereof) or may be achieved in another way. In some cases it may also be desirable to define or extend the design space by means of different architectural implementations, not just parametric information; this will likely be more complex but would allow a greater degree of flexibility during optimization, e.g. to consider different fault tolerant subsystem architectures or perhaps even different H/W-S/W allocation strategies.

These variants and/or alternative parameters are necessary for optimization to take place, as they provide the scope for different model possibilities to be explored and evaluated. However, for evaluation to be meaningful, it is important that these variants and parameters are substitutable, i.e. that one parameter or variant can be substituted by another.

The most important factor to take into account here is the notion of substitutability, i.e. for optimization to be valid it must be possible to replace set of parameters or variants with another set. In practice, this most often means that if a Function or other entity performs a certain function and possesses a certain interface, then substituting it for another variant or changing its parameters should not result in an incompatibility within the system being modelled. As a crude example, the steering wheel of an automobile is unlikely to be a valid substitute for one its tyres.

With EAST-ADL both substitutability and design space definition can be defined using the variability mechanisms.

8 Analysis of Functional (non-system) Properties

The previous section describes how to analyse an architecture from a system perspective. In this section the perspective is rather that of a single feature (functionality) at the time. On the vehicle level, the system information is organized purely from this perspective. Each node in the technical feature tree can be associated with information specifying the functionality of this feature. On the lower levels (artefact levels), EAST-ADL supports analysis of the system with the focus on such functional properties. This means that the behaviour of the structural elements in the architectures on the artefact levels is modelled. The means for this behavioural modelling are to give the architectural components execution semantics and to describe their transfer functions.

8.1 EAST-ADL Execution semantics

In EAST-ADL, behaviour modelling relies on the definition of a set of elementary functions that together represent the behaviour of the integrated or composed model. Causal functions have a trigger associated and are executed based on the assumption of synchronous run-to-completion execution (read inputs from ports, compute, and write outputs on ports). This was chosen to enable analysis and behavioral composition and make the function execution independent of behavioral notations: inside each function, the data transformation may be described with various languages and legacy tools including general UML tools and domain-specific tools (e.g. Simulink, ASCET, SCADE, etc).

The trigger defines the explicitly defined invocation pattern of time discrete functions. Functions may be time-triggered, in which case time alone causes execution to start. Event triggered functions may be invoked due to data arrival or calls on the input ports. This is defined by the triggerPolicy attribute. A textual/OCL TriggerCondition rule may be used to define the conditions for whether the function is invoked on the (time or event) trigger. For example, a port value has to exceed X or data must arrive on more than one port.

Behavioral information can be assigned to a composite function (isElementary = false). In this case the information is meant as a specification of the behaviour that will be realized by the set of elementary function prototypes instantiated inside the composite.

8.2 Transfer Function Definition

Each FunctionBehavior references the Function it models. It is characterized by two attributes, representation indicating the notation used, and path, defining where the behavioural definition is stored, for example in a mdl file for simulink.

9 **References**

- [1] “AUTOSAR Technical Overview” , v2.2.2, Release 3.1, www.autosar.org