 THE RESEARCH LEADING TO THESE RESULTS HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S SEVENTH FRAMEWORK PROGRAM (FP7/2007-2013) FOR CRYSTAL – CRITICAL SYSTEM ENGINEERING ACCELERATION JOINT UNDERTAKING UNDER GRANT AGREEMENT N° 332830 AND FROM SPECIFIC NATIONAL PROGRAMS AND / OR FUNDING AUTHORITIES.

CRitical S**YST**em Engineering **A**cce**L**eration

State of the art for automotive ontology

D308.010

DOCUMENT INFORMATION

Project	CRYSTAL
Grant Agreement No.	ARTEMIS-2012-1-332830
Deliverable Title	State of the art for automotive ontology
Deliverable No.	D308.010
Dissemination Level	PU
Nature	R
Document Version	V1.1
Date	2014-01-29
Contact	Thomas Karbe
Organization	TUB
Phone	+49 30 31478416
E-Mail	thomas.karbe@tu-berlin.de

AUTHORS TABLE

Name	Company	E-Mail
Thomas Karbe	TUB	thomas.karbe@tu-berlin.de
Kerstin Hartig	TUB	kerstin.hartig@tu-berlin.de
Andrea Leitner	AVL	andrea.leitner@avl.com
Alberto Melzi	CRF	alberto.melzi@crf.it
Cecilia Ekelin	Volvo	cecilia.ekelin@volvo.com
Ömer Can	TUB	tu-berlin.can@daimler.com
Christian Reuter	DAI	christian.c.reuter@daimler.com
Pierre du Pontavice	VALEO-F	pierre.du-pontavice@valeo.com
Jörg Settelmeier	AVL-R	joerg.settelmeier@avl.com

CHANGE HISTORY

Version	Date	Reason for Change	Pages Affected
0.1	2013-10-25	Initial document structure	all
0.2	2013-10-29	Added edit instructions	p4
0.3	2013-10-31	Added content on DODT and related references	Par. 3.5, Ch. 7
0.4	2013-11-07	Added content for paragraph 3.3 (and related abbrev./ref.)	Par.3.3, Ch.6,Ch.7
0.5	2013-11-13	Added content on EAST-ADL	Par. 3.2
0.6	2013-11-15	Added content on AUTOSAR	Par 3.3
0.7	2013-11-15	Added Introduction chapter and introduction of chapter 2	Ch. 1,2
0.8	2013-11-15	Added introduction of chapter 3 and content on other projects	Par. 1.3, Ch.3, Par. 3.6
0.9	2013-11-19	Added content on ontologies in general	Par 2.1
0.10	2013-11-21	Added content on technical realization of ontologies	Par 2.2

0.11	2013-12-11	Added content on OSLC	Par 3.1
0.12	2013-12-18	Added content on technical realization of ontologies	Par 2.2
0.13	2014-01-03	Added evaluation	Ch. 4
1.0	2014-01-03	Internal Review Version	all
1.1	2014-01-17	External Review Version	Appendix

CONTENT

Content of Figures

Content of Tables

Content of Appendix

1 Introduction

1.1 Role of deliverable

This document is the first deliverable in the CRYSTAL work package WP308, which is concerned with the specification of a domain-specific ontology for the automotive domain. The ontology can provide a common vocabulary for all deliverables in the automotive domain throughout the CRYSTAL project and will thus help to improve the quality of those documents. Furthermore, domain ontologies have the potential for application on the interoperability standard definition and data exchange.

The purpose of this document is the collection and evaluation of the state of the art regarding such an automotive domain ontology. To approach this objective, we will examine previous related projects as well as the standards, tools, and technologies that were part of those projects. The findings will be evaluated against their applicability. This document will, therefore, serve as a basis for building an automotive domain ontology in the remainder of the project.

1.2 Relationship to other CRYSTAL documents

The findings and the evaluation of this document will provide a basis for the subsequent deliverables D308.21 and D308.22. Those documents will describe the first and second version of the foreseen automotive domain ontology.

Besides the automotive domain, there are CRYSTAL subprojects for the aerospace domain, the rail domain and the health care domain. Each of those subprojects includes its own domain ontology work package (2.9, 4.7, 5.4), and provides the first deliverable on the state of the art for the respective domain ontology. There will be a strong connection between those documents.

The goals of the different domain ontology providers should be aligned in order to support the capabilities of the currently developed CRYSTAL Interoperability Specification platform. Therefore, there is a relationship to Subproject 6 and its deliverables.

Finally, the other deliverables in the automotive domain will be influenced by the findings on the automotive ontology.

1.3 Relationship to results from other projects

In order to collect the state of the art of automotive ontology aspects in system engineering, the outcome of several previous European funded projects are taken into consideration. This document includes results of projects such as CESAR, MBAT, EAST-EEA, ATESST, ATESST2, iFest, VeTeSS, and MAENAD.

1.4 Structure of this document

To reach the goal of this deliverable, we will follow the steps mentioned below through the document:

In **Section 2** we will summarize our general understanding of what an ontology is and fix the most basic concepts to have a common understanding of them throughout the document. Additionally, we will analyze typical technical realizations, languages and tools to work with ontologies.

In **Section 3** we will focus on domain specific ontologies and summarize our findings from different projects, technologies, tools and standards that are specific for the automotive domain.

In **Section 4** we will provide a first evaluation of our findings, to see whether there are conflicting findings, and whether all important aspects of the automotive domain are covered. We will also provide a first estimation of the usefulness of our different findings regarding the construction of a unified automotive domain ontology that can be used throughout the whole CRYSTAL project.

2 Ontology basics

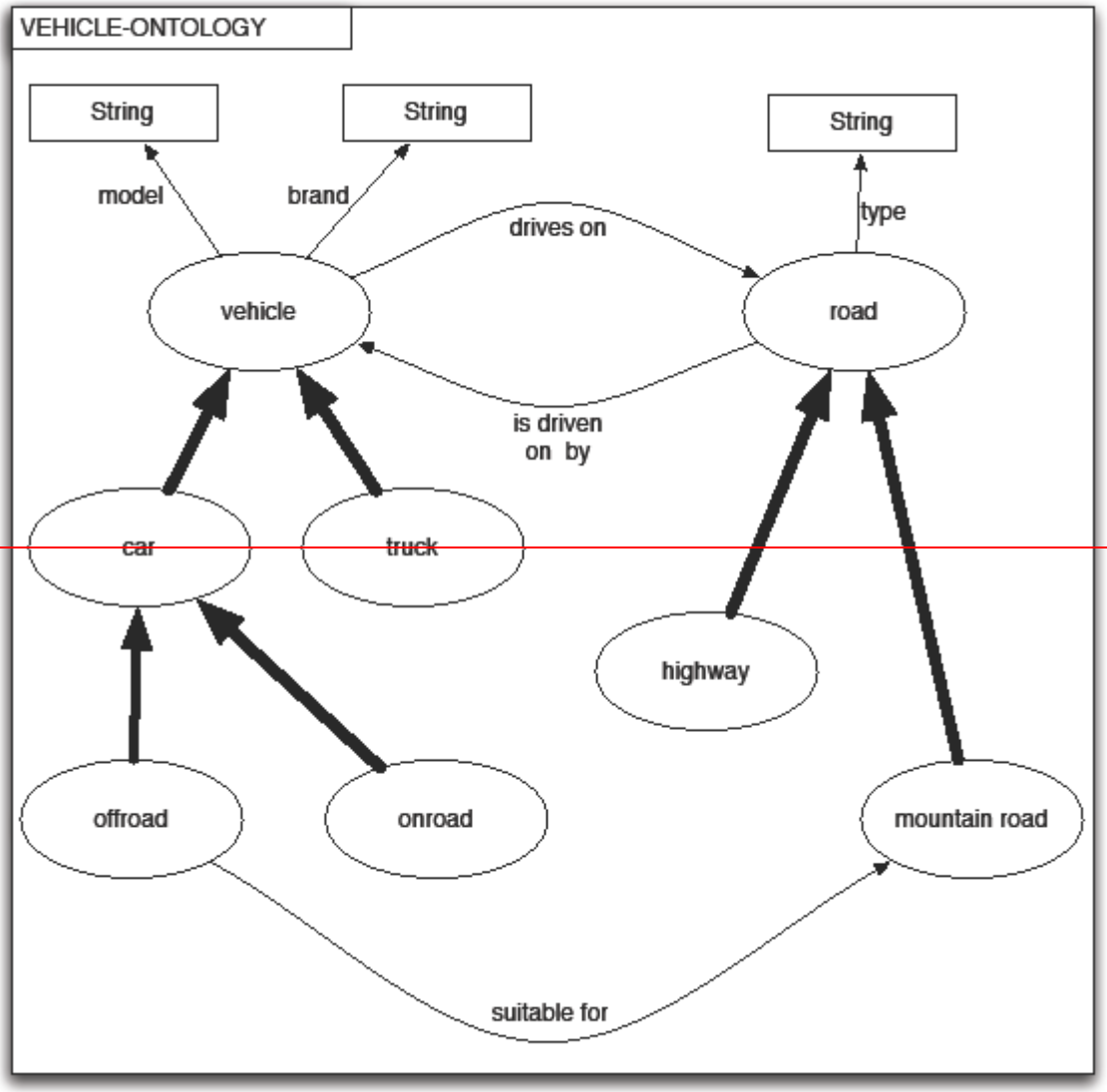
Since the main objective of this deliverable is the collection and evaluation of the state of the art regarding an automotive domain ontology, we need to establish a common understanding of the purpose and the most important ontology concepts. In this chapter we will start with a brief introduction of ontologies in general. In the second part of this chapter we will focus on the technical aspects, i.e., standard tools and formalized languages which are available to work with ontologies.

2.1 Ontologies in general

In philosophy ontology is “the study of being and existence”. However, there is no universal definition for ontologies because of a large spectrum of possible uses with partially conflicting meanings. Nowadays, in computer science we treat an ontology as a “formal, explicit specification of a conceptualization” [Gruber, 1993]. According to Gruber, “a *conceptualization* is an abstract, simplified view of the world”. *Formal* means that a system description should have a defined syntax and semantics. *Explicit* emphasizes the need for a clear definition of the elements. Ontologies extend the characteristics of glossaries and taxonomies. While a glossary is a simple set of definitions, taxonomies classify terms and concepts in a hierarchical way, e.g., a tree structure. In addition, ontologies are formal specifications of terms, their types, properties and relations.

Ontologies aim at explaining a discourse domain, also called the universe of discourse where a set of concepts can be represented by classes with assigned properties and relationships. Here, a domain stands for a collection of potentially discussible classes for the main terms within the area of interest. Properties are features that classes can have. Relations connect classes to each other and enhance the domain semantically. Axioms are logical formulas that define the rules that need to be considered when changing elements in the ontology.

Figure 2-1 depicts an example of a vehicle ontology which we will use to explain the most common ontology concepts. There are classes like *vehicle* and *road* with different attributes of type *String*. The vehicle has the attributes *model* and *brand*. The class road has the attribute *type* and two specializations, *highway* and *mountain road*. The class vehicle is related to road with the relations *drives on* and *is driven on by.* For vehicles there are again two specializations, *car* and *truck*, where *car* is again separated into *offroad* and *onroad.* Another relation is *suitable for* and connects mountain roads with offroad cars.


VEHICLE-ONTOLOGY
String
String
String
model
brand
drives on
type
vehicle
road
is driven
on by
car
truck
highway
offroad
onroad
mountain road
suitable for

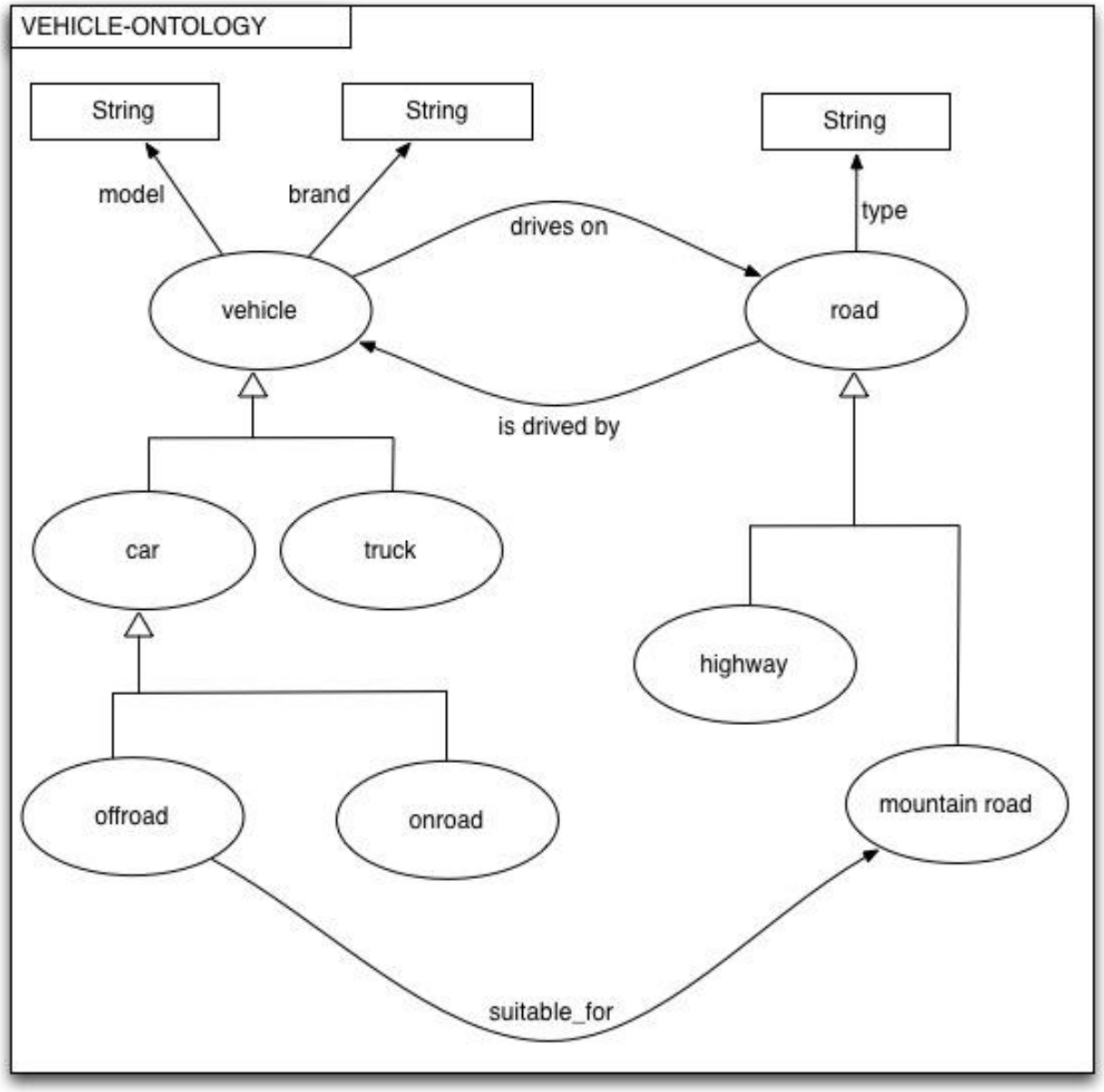


Figure 2-1: Example of a vehicle ontology

Ontologies can be classified into two major groups: upper ontologies and domain ontologies. An upper ontology represents very general concepts that are useful in many different domains without changing their meaning from domain to domain. In contrast, a domain ontology contains concepts that are specific for that domain or that has a specific meaning in that domain. Thus, terms in a domain ontology may or may not be part of other domain ontologies, but usually when they are part of other domain ontologies, their meaning could be slightly different.

The purposes for building ontologies can be manifold. They provide a common understanding of the discourse domain for human beings. Furthermore, building computational ontology models, that can be machine-processable, enables the use within software applications and support solving interoperability problems. Such ontologies are established and well-known in many fields like knowledge representation, semantic web, artificial intelligence, and software engineering. For the purpose of unified ontology representation standard ontology formats and respective tools have been developed. We will have a deeper look into some ontology formats, languages, and tools in the next section.

2.2 Technical realization

2.2.1 Languages

Over the past years, the necessity to give information a more formal structure rose especially as a result of the evolving World Wide Web. Search engines provide lots of data to specific user queries but often lacked the ability to understand the meaning and the relations for providing more useful information. The Semantic Web focuses on enabling machines with these capabilities using the encoded knowledge from ontologies.
There are several ontology languages for building ontologies. Formalizing and encoding knowledge with those formal languages enables machine processing and automated reasoning. Among others, the World Wide Web Consortium (W3C) specified several standards and languages. In the following some languages will be introduced.

- RDF – Resource Description Framework[1]
  Similar to concepts like UML class diagrams or entity relationship diagrams the RDF describes a schema to formulate triples containing subject, predicate and object, which are based on directed graphs *(e.g., libraries contain books, books are sold in bookshops)*. These resources are described as Uniform Resource Identifiers (URI). RDF is independent from a certain representation. However, the most common formats are XML and Notation 3 (N3).

  Figure 2-2 shows an example of an RDF graph. The corresponding XML code reads as follows [RDF, 2004].

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
             xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">
```

[1] Please read http://www.w3.org/RDF/ for more information.

```
  <contact:Person rdf:about="http://www.w3.org/People/EM/contact#me">
    <contact:fullName>Eric Miller</contact:fullName>
    <contact:mailbox rdf:resource="mailto:em@w3.org"/>
    <contact:personalTitle>Dr.</contact:personalTitle>
  </contact:Person>
</rdf:RDF>
```

Figure 2-2: An RDF Graph Describing Eric Miller [RDF, 2004]

- RDFS – Resource Description Framework Schema[2]

  RDFS supports describing and linking terms and, therefore, supports the creation of RDF vocabularies. RDFS construction elements are classes and their properties. Classes can be arranged hierarchically. Properties describe relations between resources.

  The following XML code shows an RDFS example [RDF, 2004].

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
```

[2] Please read http://www.w3.org/TR/rdf-schema/ for more information.

```
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://example.org/schemas/vehicles">
<rdf:Description rdf:ID="MotorVehicle">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>
<rdf:Description rdf:ID="PassengerVehicle">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>
<rdf:Description rdf:ID="Van">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>
<rdf:Description rdf:ID="MiniVan">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#Van"/>
  <rdfs:subClassOf rdf:resource="#PassengerVehicle"/>
</rdf:Description>
</rdf:RDF>
```

- OWL – Web Ontology Language[3]

  OWL extends existing languages in order to build more complex ontologies. Additional language constructs allow expressions similar to first-order logic. OWL differentiates between classes, properties and instances. There are variants of OWL represented by different stages of extension – OWL Lite, OWL DL, and OWL Full.

- Topic Maps

  Besides RDF and OWL, Topic Maps is another possibility to represent knowledge. It uses topics (RDF: classes, OWL: classes), associations (RDF: predicate), and occurrences (RDF: object, OWL: properties).

- SPARQL – SPARQL Protocol and RDF Query Language[4]

[3] Please read http://www.w3.org/OWL/ for more information.

[4] Please read http://www.w3.org/TR/rdf-sparql-query/ for more information.

SPARQL is a graph based query language for RDF. It allows reading and editing data from databases.

2.2.2 Tools

The most common tool for ontologies is the Protégé Ontology Editor[5] with the Stanford University OWL-Plugin. Besides that, there is also an Eclipse Plugin and other general tools and frameworks. Some tools support validation, others focus on interference or translate, e.g., UML into OWL.

Protégé facilitates the creation of knowledge bases. Knowledge can be added and edited. Furthermore, queries are supported. To get an impression of the functionality some screenshots of use cases based on Protégé 3.1 are shown in Annex I.

Query Languages like SPARQL provide means to ask simple questions about an ontology. For more complex questions there are reasoners, which allow to combine simple questions to more complex ones. A very prominent tool that allows for more complex requests is the RDF-Store “Jena”, which supports different evaluation strategies and forward- as well as backward rules.

Similarly, for OWL the most famous reasoner is “Pellet”, which uses a subset of SPARQL as query language.

[5] Please read http://protege.stanford.edu/ for more information.

3 Approaches in automotive system engineering and their ontology aspects

This chapter presents state of the art which is related to automotive domain ontologies such as technologies, tools, or standards dealing with embedded system development. Some of them are results from previous projects, especially other European funded projects from ARTEMIS JU. Others are the outcome of community efforts.

OSLC (Open Services Lifecycle Collaboration) is an open community aiming at an easier integration of software development by improving collaboration of different tools. In order to reach this goal, general vocabularies and several specifications are defined. Even though no specific automotive specifications exist, in Section 3.1 the ontological potentials of OSLC will be presented, since it could provide a good basis for the IOS and is related to ontologies.

As the *ISO 26262* Standard for Automotive Functional Safety encompasses standard definitions for specific terms related to the deployment of the functional safety process along the entire engineering environment chain, Section 3.2 will describe ontological aspects of this standard.

AUTOSAR (AUTomotive Open System ARchitecture) is an open standard for automotive E/E (Electrics/Electronics) architectures, jointly developed by a consortium of automotive manufacturers and suppliers. AUTOSAR contains an automotive-specific glossary described in Section 3.3.

EAST-ADL (Electronics Architecture and Software Technology - Architecture Description Language) is an ADL for automotive embedded systems developed in various projects such as EAST-EEA, ATESST, ATESST2 and MAENAD. It aims at adding higher levels of abstraction to AUTOSAR and will be described in Section 3.4.

The ARTEMIS project *CESAR* (Cost-efficient methods and processes for safety relevant embedded systems) aimed at improving efficiency of the embedded system development process in a multi-domain manner. In the project the Domain Ontology Design Tool (DODT) was developed in order to create domain ontologies and formalize requirements. DODT will be introduced in Section 3.5.

3.1 Technology: OSLC

OSLC is not directly relevant for the construction of an automotive ontology, since there is no special automotive part in it. Nevertheless, OSLC could provide a solid basis for the CRYSTAL IOS and its vocabularies are close to ontologies, which makes OSLC relevant for the ontology to be built.

3.1.1 Short description

OSLC[6] is the shorthand for *Open Services for Lifecycle Collaboration*. The name stands for an open community that aims at the integration of independent tools from the software and product development lifecycle, such as requirements management tools or test management tools. More concrete, the community is organized in different workgroups, that create a family of web services specifications for products, services, and other tools that support all phases of the software and product lifecycle. Most workgroups are dedicated to specific integration scenarios belonging to a topic from Product Lifecycle Management (PLM) and Application Lifecycle Management (ALM).

The different OSLC workgroups can roughly be categorized into three groups: Specification Writing, User Group, and Organizational:

- **Specification Writing Workgroups**
  Most of the OSLC workgroups write specifications for integration scenarios from a specific ALM or PLM topic. Currently, those groups are:
  - ALM-PLM Interoperability
  - Architecture Management
  - Asset Management
  - Automation
  - Change Management
  - Configuration Management
  - Core
  - Estimation and Measurement
  - Performance Monitoring
  - Quality Management
  - Reconciliation
  - Requirements Management
- **User Groups**
  User Groups are specific OSLC workgroups that are also dedicated to a specific topic, but that do not write specifications for them. Instead they aim at reusing existing specifications for their topic. Currently, those groups are:

[6] http://www.oasis-oslc.org/

- Communications
- Embedded Systems
- Mobile

- **Organizational**

  Currently, there is one workgroup that is dedicated only to organizational tasks: the *Steering Committee.* The Steering Committee is responsible for approving new workgroups, approving new specifications, and for all changes in the general OSLC community.

The goal of OSLC is to create specifications for interactions between tools, and thus, each OSLC-compliant tool offers OSLC protocols that are described by the OSLC core specification and at least one other OSLC domain specification. However, OSLC does not try to limit or to standardize the behavior or the capabilities of tools. Instead, the core idea is to specify a minimum amount of protocol and a small number of resource types, just enough to enable the integration of tools.

Technically, OSLC is based on the W3C concept of *Linked Data* authored by Tim Berners Lee. This means, that each lifecycle artefact is a HTTP resource, identifiable by a URI and that each artefact has an RDF representation (see Chapter 2.2 for details).

For the integration of two tools there are basically two techniques available:

- **Linking data via HTTP**

  OSLC specifies a common tool protocol for creating, retrieving, updating, and deleting (CRUD) lifecycle data based on internet standards like HTTP and RDF using the Linked Data model. This protocol can be used by any tool or other programmatic client to talk to any other tool that implements the specifications. Linking is achieved by embedding the HTTP URL of one resource in the representation of another.

- **Invocation of HTML Web User Interface**

  OSLC specifies a protocol that allows a tool or other client to cause a fragment of the web user interface of another tool to be displayed, allowing a human user to link to a new or existing resource in the other tool or see a preview of information about a resource in another tool. This enables a tool or other client to exploit existing user interface and business logic in other tools when integrating information and process steps. In some circumstances this is more efficient and offers more user function than implementing a new user interface and then integrating via an HTTP CRUD protocol.

In the Cesar project, which is a predecessor of Crystal regarding the development of the IOS, the steering board decided in 2011, that OSLC will be the basis for the IOS.

3.1.2 Ontology aspects

In OSLC there is no automotive-specific workgroup, specification, or vocabulary. Nevertheless, OSLC offers very basic and general vocabulary, on which other domain ontologies can be based. Each of the specification writing workgroups is responsible for a domain-specific specification document. The specifications describe the minimum requirements for OSLC service providers and are given as HTML wiki pages. Most of the specifications are available in a final version 2.0. Some of them already exist as 3.0 draft versions. In each specification a namespace is defined. Associated with the namespace there is an RDF file, in which the ontological details of the specification can be found.

3.2 Standard: ISO 26262

3.2.1 Short description

The ISO 26262 [ISO, 2011-2012] standard describes the requirement of functional safety applied to the development of on board electronic/electric systems in road vehicles.

This standard extends the IEC 61508 [IEC, 2010] applied to functional safety of electrical / electronic / programmable electronic safety-related general systems. But, ISO 26262 is not only the adaptation of IEC 61508 to the specific automotive domain. The new standard, actually, applies to all activities during the entire safety lifecycle of the safety-related systems comprised of electric, electronic, and software components in road vehicles. Therefore, the ISO 26262 standard provides appropriate requirements and processes in a more general, complete, articulated, and self-consistent framework than the IEC 61508. ISO 26262 supports the entire automotive safety lifecycle (management, development, production, operation, service, decommissioning) and contains an automotive scheme for hazards classification. The key issue of the ISO 26262 is the definition of the objectives that a component/system, until to the integration on a vehicle, must fulfil for assuring its compliance with the established safety requirements, depending on the application scenario (the vehicle integration, the vehicle characteristics and intended behavior and performances, with related environmental conditions and operational situations).

Moreover, system safety is achieved through a number of safety measures, which are implemented in a variety of technologies (for example: mechanical, hydraulic, pneumatic, electrical, electronic, programmable

electronic, etc.) and applied at the various levels of the development process, and, although ISO 26262 is concerned with functional safety of electric and electronic systems, its framework could be applied to the safety-related systems based on other technologies.

Additionally, its formal and structural completeness, from design until to the decommissioning of a vehicle, assures the level of responsibility (liability) of the car maker with respect to the safety compliance of the vehicles produced. The application of ISO 26262 in the automotive domain aims to deliver the necessary documentation that confirms the effectiveness of the safety behavior of the vehicle and its components. This result could be recognized at law level, as the best practice to which refer, but (until now) is not yet certified in an official way by organizational bodies external to the automotive stakeholders.

ISO 26262 standard is constituted of ten parts linked to each other:

- Part 1: Vocabulary
- Part 2: Management of functional safety
- Part 3: Concept phase
- Part 4: Product development at the system level
- Part 5: Product development at the hardware level
- Part 6: Product development at the software level
- Part 7: Production and operation
- Part 8: Supporting processes
- Part 9: Automotive Safety Integrity Level (ASIL)-oriented and safety-oriented analyses
- Part 10: Guideline on ISO 26262

3.2.2 Ontology aspects

The terminology described in Part 1 (Vocabulary) of the ISO 26262 standard encompasses (by definition) the main ontology topics about the automotive functional safety. Some of the standard definitions are specific in the workflow, while others are common to the HW and SW application fields related to the general automotive environment.

The framework of the CRYSTAL platform has to deal with logical structures, terms and definitions, in relation to systems, architectures, tools and interoperability of them, while in ISO 26262 [ISO, 2011-2012, Part 1] there are:

- elements related to the HW and SW structures and elements in the system under analysis (e.g. Hardware part, Component, Element, System, Item, Embedded software, Software tool, Software unit, Software component, Architecture, Allocation, etc.).

- specific terms related to the deployment of the automotive functional safety assessment process along the entire product engineering chain (e.g. Safety plan, Item, Impact Analysis, Hazard analysis and risk assessment, Verification, Validation, Confirmation Review, Safety Analyses, etc.).

There is a bidirectional relationship between ISO 26262 and the general design process as shown in Figure 3-1. Both will have an impact on the automotive domain ontology, but the ISO influences the automotive design process as well as items from the design process become relevant in the ISO, and therefore there might be overlapping vocabularies, that we have to consider during the construction of the automotive domain ontology.

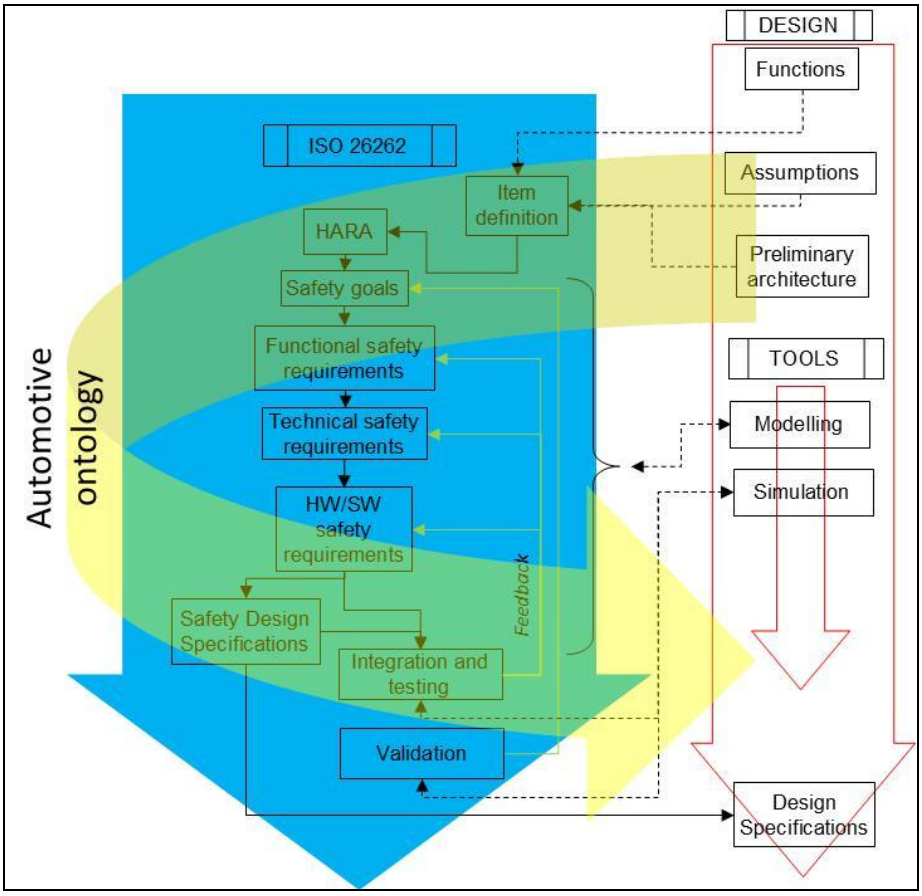


Figure 3-1: ISO 26262 versus automotive design

The ISO 26262 terminology could be considered as a basic reference for the deployment of the higher level ontology suitable for modelling a more general framework of a process, i.e., the automotive functional safety assessment process. This will be explained more precisely in the following.

If ontologies represent knowledge as a set of concepts within a specific domain of interest, using a shared vocabulary to denote the types, properties and interrelationships of those concepts, then ISO 26262 vocabulary (Part 1) influences the terminology by which the entire framework (structural framework for organizing information) of the functional safety process is built and organized in the automotive domain.

Furthermore, the other parts describe a structured and self-consistent process, with specific terminology, always related to the basic meaning founded on Part 1, but aiming to constitute a set of definitions interlaced and appropriate to the scope of the standard. By this point of view the ISO 26262 standard describes a specific world that consists of a set of types, properties, and relationship types constituting the skeleton of a structure for modelling the automotive functional safety assessment process (specific domain of interest).

Therefore, we can identify two levels of possible ontology in the standard: one constituted simply by the vocabulary, as previously shown, and one starting from the vocabulary and aiming to describe the entire workflow of the functional safety assessment process in the automotive domain, based on the structure of the process described by the standard itself. The second level can be managed by tools suitable for organizing the information in semiformal language. It influences the ontologies of the other automotive processes that interact with it, in particular the design. As a consequence, we could maintain this relationship across the more general classes and meta-classes that CRYSTAL platform would develop and support, where the functional safety aspects have to be considered. As an example the EAST-ADL meta-model would include definitions and properties from ISO 26262 standard when it deals with functional safety.

3.3 Technology: Autosar[7]

3.3.1 Short description

The AUTOSAR standard will serve as a platform upon which future vehicle applications will be implemented and will also serve to minimize the current barriers between functional domains. The AUTOSAR partnership is an alliance of OEM manufacturers and Tier 1 automotive suppliers working together to develop and establish a de-facto open industry standard for automotive E/E architecture which will serve as a basic infrastructure for the management of functions within both, future applications and standard software modules.

Motivation for this alliance is the management of E/E complexity associated with growth in functional scope and further the flexibility for product modification, upgrade, and update. Also the scalability of solutions within and across product lines and the improved quality and reliability of E/E systems is under this focus.

[7] The text is based on the official Autosar Homepage: www.autosar.org

The goals of this alliance are the fulfillment of future vehicle requirements, such as, availability and safety, SW upgrades/ updates, and maintainability. Furthermore, the scalability and flexibility to integrate and transfer functions should be increased with a higher penetration of "Commercial off the Shelf" SW and HW components across product lines. This leads to an improved containment of product and process complexity with risk and cost optimization of scalable systems. The technical goals can be summarized as **modularity**, **scalability**, **transferability**, and **re-usability** of functions. The *modularity* of automotive software elements allows individual software tailoring according to specific requirements. The *scalability* of functions supports the adaptability of common software modules to different vehicle platforms in order to prevent redundant software functionality. Furthermore, *transferability* of functions approaches an optimized availability of resources throughout the electronic architecture of vehicles. Via the *re-usability* of functions the product quality and reliability can be improved. Therefore, it reinforces the corporate brand image across product lines. AUTOSAR supports achieving the aforementioned technical goals by a common software infrastructure for automotive systems based on *standardized interfaces* for the different layers. Additionally, the functional interfaces are standardized across manufacturers and suppliers.

Figure 3-2 gives an overview about the Autosar-structure with the interlink of ASW-components *(*above the *Runtime Environment (RTE))* and the Basic software below the *RTE*. In addition, at system design level the *RTE* acts as a communication center for inter- and intra-ECU information exchange. The RTE provides a communication abstraction to *AUTOSAR Software Components* attached to it by providing the same interface and services whether inter-ECU communication channels are used (e.g. CAN, LIN, FlexRay, MOST,…) or communication stays intra-ECU. As the communication requirements of the software components running on top of the *RTE* are application dependent, the *RTE* needs to be tailored, partly by ECU-specific generation and partly by configuration. Thus, the resulting *RTE* will differ between one ECU and another.

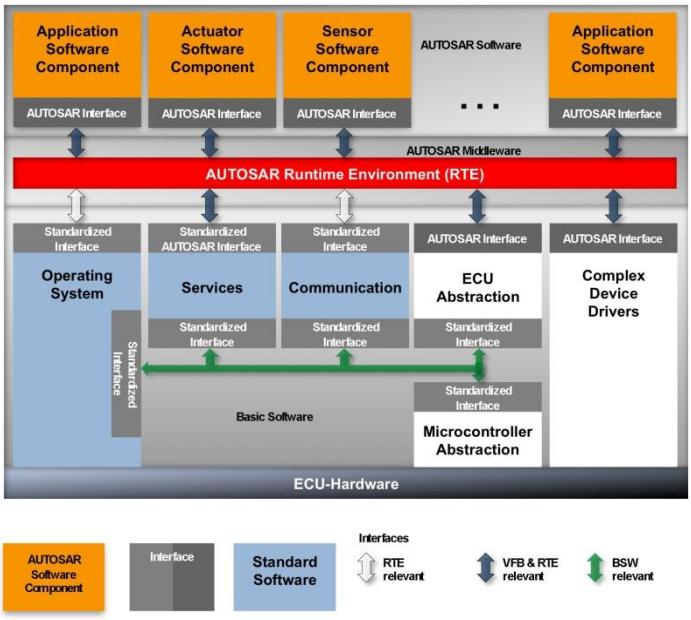


Figure 3-2: Autosar Runtime Environment and its interfaces for inter- and intra-ECU information exchange (www.autosar.org)

3.3.2 Ontology aspects

For building an ontology the overall glossary of AUTOSAR can be used as a starting point. It contains definitions of all major terms and notions used within AUTOSAR. It does not claim to be complete and it should be kept in mind that some CRYSTAL work packages have more specific terms defined within their domain specific glossary. The glossary terms are defined by properties such as name, definition, initiator, comment and reference. An example of an AUTOSAR glossary[8] entry is shown below.

[8] http://www.autosar.org/download/R4.1/AUTOSAR_TR_Glossary.pdf

System

Definition	An integrated composite that consists of one or more of the processes, hardware, software, facilities and people, that provides a capability to satisfy a stated need or objective.
Initiator	WP Virtual Functional Bus
Further Explanations	--
Comment	ITEA EAST uses IEEE 14407 standard. Here not applicable because of problem with the definition of function. One correct interpretation is: - it might be a composition of one or more ECUs
Example	Braking system
Reference	[ISO 12207]

3.4 Technology: EAST-ADL/EAST-ADL2

EAST-ADL[9] is an architecture description language for automotive systems engineering that was first developed in the EAST-EEA project (2001-2004). The language has then been successively refined through several research projects: ATESST (2006-2008), ATESST2 (2008-2010), and MAENAD (2010-2014). In order to support long-term maintenance of the language the *EAST-ADL Association* has been established. The current version of the language is 2.1.11. The EAST-ADL Association has the ambition to make EAST-ADL a defacto standard for systems modelling in the automotive domain in order to facilitate information exchange.

3.4.1 Short description

EAST-ADL captures the systems engineering information in a system model that is structured into different abstraction levels. Each abstraction level describes the entire system but at different level of detail. In addition to the structural modelling of the system, EAST-ADL contains modelling concepts to capture orthogonal aspects such as requirements, timing, variability, and dependability. See Figure 3-3 for an overview.

[9] Please read www.east-adl.info for more information.

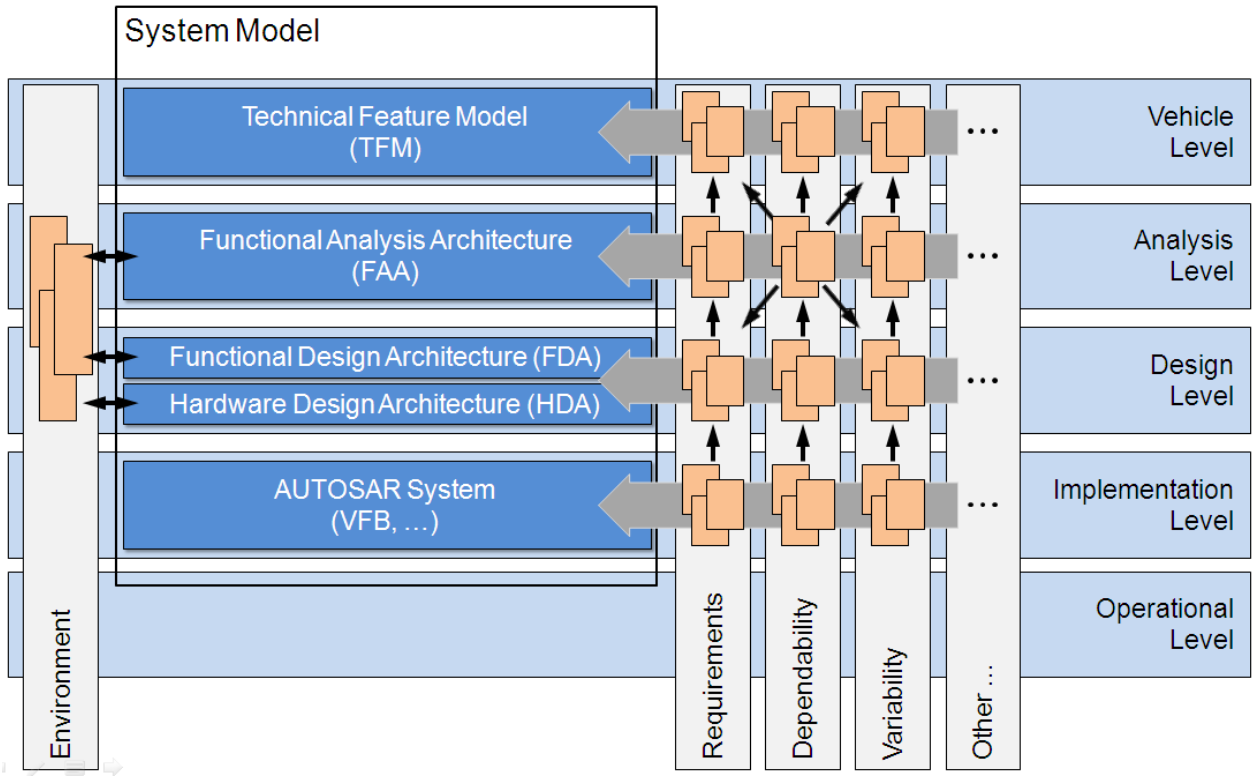


Figure 3-3: EAST-ADL overview (www.east-adl.info/Specification.html)

The models at the different abstraction levels represent information that is useful for the activities performed in the corresponding development phases. Together with the modelling concepts, EAST-ADL also provides a development methodology aligned with the commonly used V-model. At **vehicle level** the *Technical Feature Model (TFM)* is created to represent the stakeholder view of the system. That is, which *features* should the system contain. At **analysis level** the abstract *analysis functions* that will implement the features are modelled in a *Functional Analysis Architecture (FAA)*. At **design level** the analysis functions are further refined into *design functions* which represent the logical design of the system in a *Functional Design Architecture (FDA)*. This level also contains a model of the hardware and its topology in the *Hardware Design Architecture (HDA).* Moreover, the allocation of the design functions onto the hardware is done at this level. At **implementation level** the design functions are decomposed into *software components* which represent the system software. Since, automotive software architecture is covered by the AUTOSAR standard, EAST-ADL adopts the AUTOSAR approach for this level. The orthogonal modelling concepts can be applied at all abstraction levels and support traceability both within and between levels. As an example, requirements can be associated with structural elements using satisfy-links and the refinement of requirements through the abstraction levels can be modelled using derivation-links.

3.4.2 Ontology aspects

Since the EAST-ADL meta-model captures the modelling concepts and their relations there has been no need to define a separate ontology. Since AUTOSAR and ISO26262 are addressed by EAST-ADL, the terminology from these standards is included as well. The meta-model is structured into different packages based on aspects in order to support language modularity. That is, to make it easy to add extensions or replace existing packages. The packages available in v.2.1.11 are the following:

- Structure
- Environment
- Behaviour
- Variability
- Timing
- Requirements
- Dependability
- GenericConstraints
- Infrastructure

The ones most relevant from an ontology perspective will be briefly described in Annex II.

The packages and concepts available for the EAST-ADL meta-model can serve as a basis for an automotive domain ontology. Especially the fact that AUTOSAR and ISO 26262 are addressed by EAST-ADL supports compatibility between the different standards and technologies.

3.5 Tool: DODT (Domain Ontology Design Tool)

DODT (Domain Ontology Design Tool) is an Eclipse-based tool, which supports the elicitation of semi-formal requirements as well as an early analysis of requirements based on domain ontologies. It was developed in the CESAR project as a contribution to the Requirements Engineering subproject (SP2). In this document it will be used to show one potential application for a domain ontology.

3.5.1 Short description

DODT manages three kinds of information – a domain ontology, boilerplates, and requirements. The domain ontology represents formalized domain-specific knowledge. Boilerplates are templates with a fixed syntax

(constant part) and attributes (variable part) that have to be filled in by the requirements engineer. The fixed structure supports the specification of uniform requirements and the additional use of the domain ontology provides guidance for the requirements engineer. The variable part is connected to the domain ontology.

Ontology and boilerplates build the basis for the specification of semi-formal requirements with DODT. Hence, the ontology has to be created first – at least theoretically. One advantage of this tool is the support for the semiautomatic generation of ontology concepts. Therefore, documents are analyzed and the tool proposes concepts and relations which could be included in the ontology. The user can interact and decide which concepts and relations really should be included in the ontology.

The domain ontology is basically used for 3 reasons: (1) definition of the terminology – usage of a common, unambiguous set of terms, (2) guidance for the requirements engineer, and (3) the analysis of requirements, e.g., consistency and completeness checks based on the ontology.

The basic structure of the ontology is described by a set of concepts and their interrelations. This ontology description is here called attribute model. These attributes are not only used to describe the structure of the ontology, they are also used as placeholders for the boilerplates. This correspondence is illustrated in Figure 3-4.

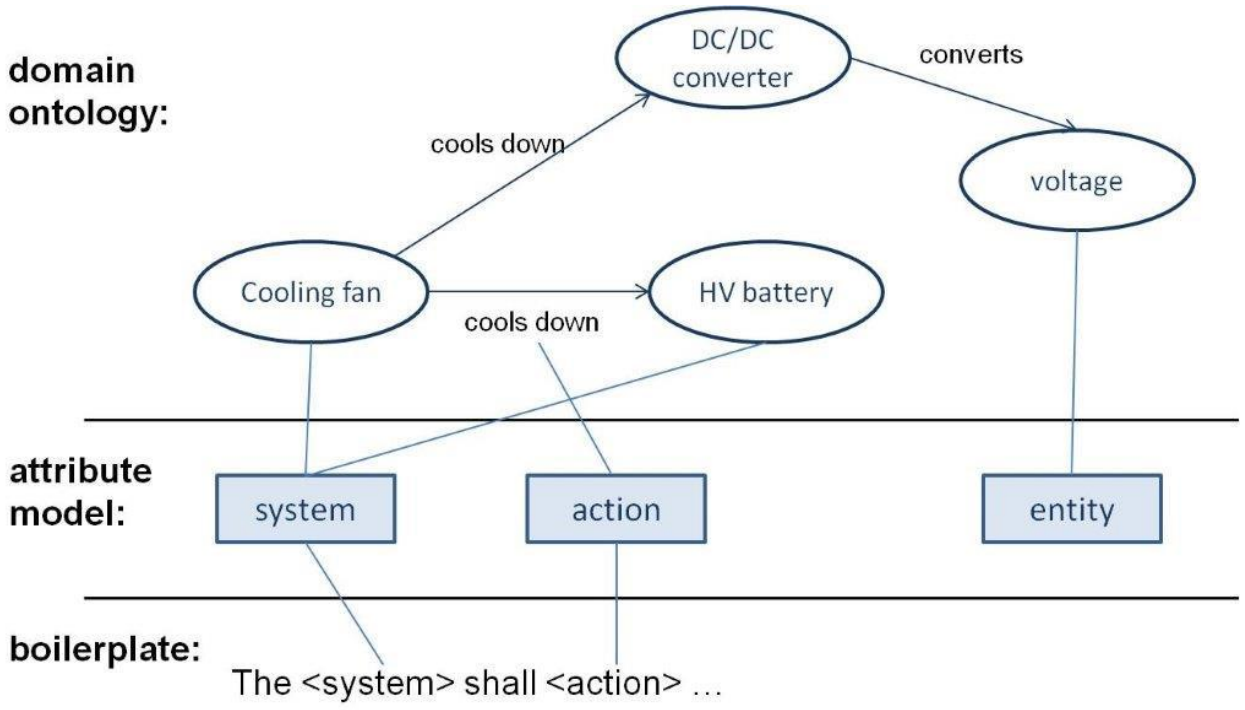


Figure 3-4: Structure of DODT consisting of boilerplates, an attribute model, and a domain ontology

The following ontology concepts have been defined in DODT:

- **System.** The system (or any subsystem, component, etc.) to be built.
- **Entity.** Entities are generic attributes for ontology concepts that can be used in case no other attribute fits, e.g. sensor, crash situation, exhaust air, etc.
- **Quantity** is a numerical value, usually followed by a <unit>.
- **Unit** typically represents measurement units for a quantity, usually following the <quantity> attribute in requirements, e.g. ms, amps, volt, etc.
- **User** is a person interacting with the system during development or operation, e.g. driver, maintenance man, etc.
- **State** is a conditions that hold for some time, e.g. acceleration, deceleration, etc.
- **Event** is a condition that happen spontaneously, e.g. critical fault detected, power down requested, etc.
- **Goal** is a high-level rationale for a requirement, e.g. high usability, etc.
- **Quality factor** is used to measure a non-functional property of the system, e.g. availability rate, etc.

Additionally, there are attributes in the attribute model, which are not mapped to concepts, but relations in the ontology. As an example, **action** is mapped to the combination <*relation*> <*concept*>. Taking an example from Figure 3-4, the combination relation *cools down* and the concept *DC/DC converter* would be mapped to action *cools down DC/DC converter*.

Axioms are a special fixed set of relations that are always true. DODT defines four of them:

- **Subclass of** describes a concept derived from the superclass concept. They can have different restrictions and different relations.
- **Equivalent** describes synonyms.
- **Contain** means that a concept consists of other concepts.
- **Contradictions** are used for concepts that might bear a conflict.

In order to analyze the quality of requirements, several types of analyses have been implemented based on the domain ontology [Malot, 2011]:

- **Completeness Analysis** - tries to find missing requirements.
- **Inconsistency Analysis** - identifies contradictions.
- **Ambiguity Analysis** - suggests replacements with more concrete subclasses.
- **Noise Analysis** – identifies nouns which have no corresponding concepts in the ontology
- **Opacity Analysis** – detects unrelated concepts in a requirement.
- **Redundancy Analysis** – detects duplicate requirements.

- **Obsoleteness Analysis** – concepts can be marked as obsolete, which means that they should not be used in Requirements. If they are used, this analysis will find them.

More information regarding the DODT method can be found in [Farfeleder, 2011a], [Farfeleder, 2011b], and [Sternudd, 2011].

3.5.2 Domain-specific requirements in CESAR

In the CESAR project, there have been ontologies for the different application domains: automotive, aerospace and avionics, railway, and industrial automation. There they tried to extract data from existing sources by applying natural language processing approaches. Therefore, they used existing requirements documents, existing ontologies and systematic top-down and bottom-up approaches. Figure 3-5 describes a subontology for the ACC system developed in CESAR and published in D_SP2_R2.2_M2 [Mitschke, 2010].

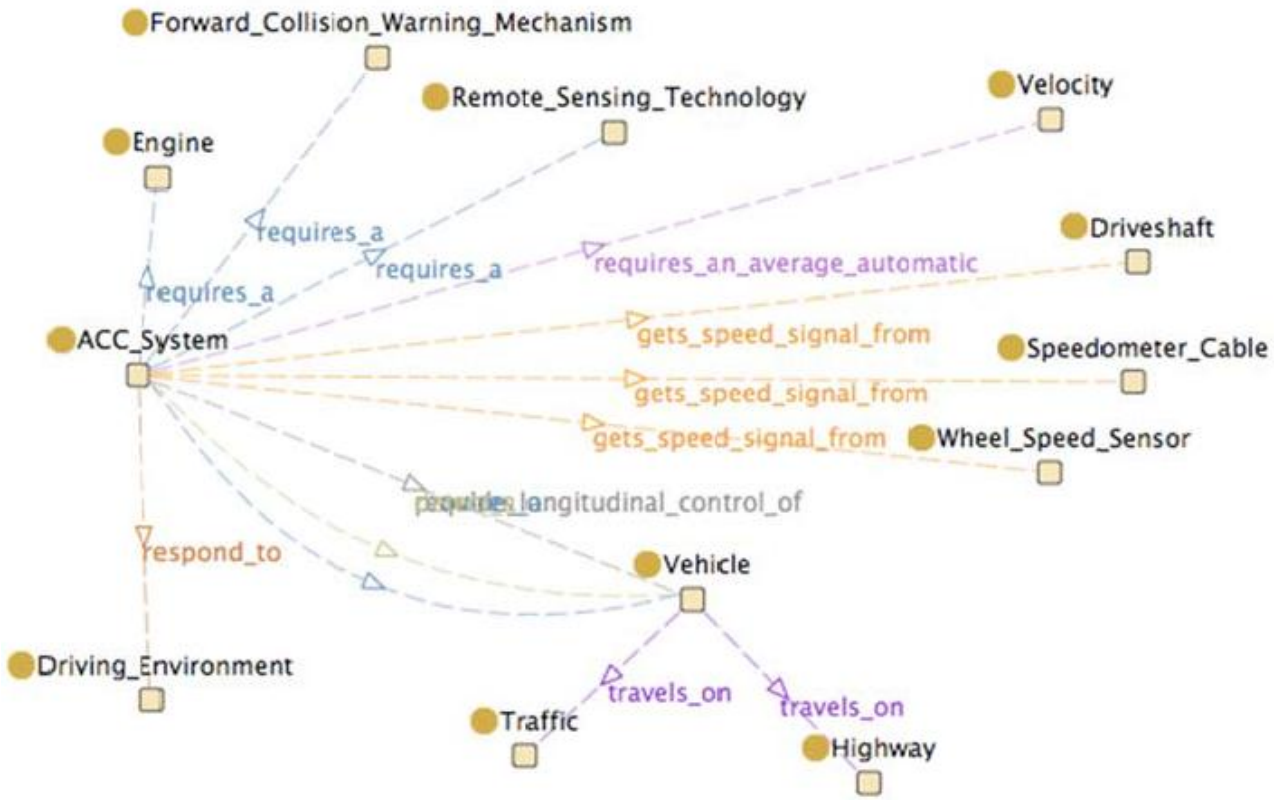


Figure 3-5: Subontology developed in the CESAR project [Mitschke, 2010]

3.6 Others

Additionally to the above-mentioned technologies, various glossaries exist containing domain vocabularies for many projects dealing with embedded system development. Some of them are mentioned below. Although, they are not as relevant for the automotive ontology as the above mentioned sections, they might be useful, to fill certain gaps that might arise.

The Artemis project **MBAT** (Combines Model-based analysis and Testing of Embedded Systems) focuses on verification and validation technologies for embedded systems and the development of a Reference Technology Platform (RTP) that facilitates efficient embedded system development. The ontological outcome of the project is a global glossary and a list of defined abbreviations and vocabulary based on the CESAR glossary. But, there is no defined ontology.

The Artemis project **iFEST** (industrial Framework for Embedded Systems Tools) focused on a tool integration framework for HW/SW co-design of heterogeneous and multi-core embedded systems. The automotive domain did not play a central role in this project. Also, an ontology was not considered in this project. Part of the outcome is a terminology of the general implementation aspects, a vocabulary that might be interesting for the entire IOS or for the specific tools and frameworks.

Other existing approaches for creating automotive-specific ontologies are the Volkswagen Ontology containing specific domain vocabulary (http://www.volkswagen.co.uk/vocabularies/vvo/ns) and an approach that describes an ontology for automotive human-machine-interaction [Feld, 2011].

4 Evaluation of state of the art

In this document we have examined various sources like projects and standards for parts that are usable in the construction of an automotive ontology. We search for answers to the following questions:

- (Q1) Which automotive ontologies are existing?
- (Q2) On which sources should an automotive ontology be based?
- (Q3) What should be the scope of the ontology?
- (Q4) How should the automotive ontology be represented?

By having a closer look into the purpose of the automotive ontology we can (partly) answer some of those questions. In our case, the primary purpose is to unify the wording throughout the automotive work packages and the resulting deliverables in the CRYSTAL project. The ontology could also be relevant for CRYSTAL SP 6 and the produced bricks, e.g., in requirements engineering. An optional secondary purpose is the extension of IOS / OSLC by the automotive ontology.

For the primary purpose the ontology should be based on standards and technologies which are accepted among the different stakeholders of the automotive domain, e.g., the ISO 26262 standard or EAST-ADL (which are also used in the different automotive workpackages) (Q2). Furthermore, the documents from the automotive work packages, especially from the public use case, should be considered as they can give a hint towards the minimum scope (Q2, Q3). We are also interested in restricting the ontological scope to wording that is relevant for CRYSTAL. In our case, this means that the scope should be limited to automotive system engineering. To enforce a unified wording throughout the produced documents, the ontology should be represented glossary-like, with keywords and textual descriptions (Q4). For the usage in bricks, which are pieces of software, there should be a machine readable version of the ontology (Q4).

For the optional second purpose the representation of the ontology should be conform to the IOS and OSLC. Since the IOS will probably be based on OSLC, and since OSLC utilizes RDF and OWL to represent its vocabularies, the machine readable version of the ontology should also be based on RDF and OWL (Q4).

With regard to question 1 we had a look into different Artemis projects as well as into different common approaches in automotive engineering. From the related Artemis projects domain ontologies have been considered only in CESAR, mainly as an input for the DODT tool. However, these domain ontologies have not been part of any deliverables.

Furthermore, we found the *Volkswagen Vehicles Ontology* which is scoped at the sales aspect for Volkswagen cars. Another paper on automotive ontologies aims at automotive human-machine-interaction. Both of them do not consider the automotive engineering aspect, which plays a major role in CRYSTAL.

Thus, we had to mainly search for good resources, from which we can extract major parts for an automotive domain ontology. As stated earlier, these resources should be well accepted within the automotive domain. We came up with three major resources, which could serve as a solid basis: the functional safety standard *ISO 26262*, the de-facto open industry standard E/E architecture *AUTOSAR*, and the architecture description language EAST-ADL, which also addresses the ISO 26262 and AUTOSAR and, therefore, indicates a good chance of compatibility between them. Each of these resources comes with glossaries/vocabularies and/or metamodels that we can use to extract basic concepts and relations between them.

5 Terms, Abbreviations and Definitions

CRYSTAL	**CR**itical S**YST**em Engineering **A**cce**L**eration
JU	Joint Undertaking
R	Report
P	Prototype
D	Demonstrator
O	Other
PU	Public
PP	Restricted to other program participants (including the JU).
RE	Restricted to a group specified by the consortium (including the JU).
CO	Confidential, only for members of the consortium (including the JU).
WP	Work Package
SP	Subproject
ISO	International Organization for Standardization
IEC	International Electrotechnical Commission
HARA	Hazard Analysis and Risk Assessment
OSLC	Open Services Lifecycle Collaboration
AUTOSAR	AUTomotive Open System ARchitecture
EAST-ADL	Electronics Architecture and Software Technology - Architecture Description Language
CESAR	Cost-efficient methods and processes for safety relevant embedded systems
DODT	Domain Ontology Design Tool
MBAT	Combines Model-based analysis and Testing of Embedded Systems
iFEST	industrial Framework for Embedded Systems Tools
OWL	Web Ontology Language
RDF	Resource Description Framework

Table 6-1: Terms, Abbreviations and Definitions

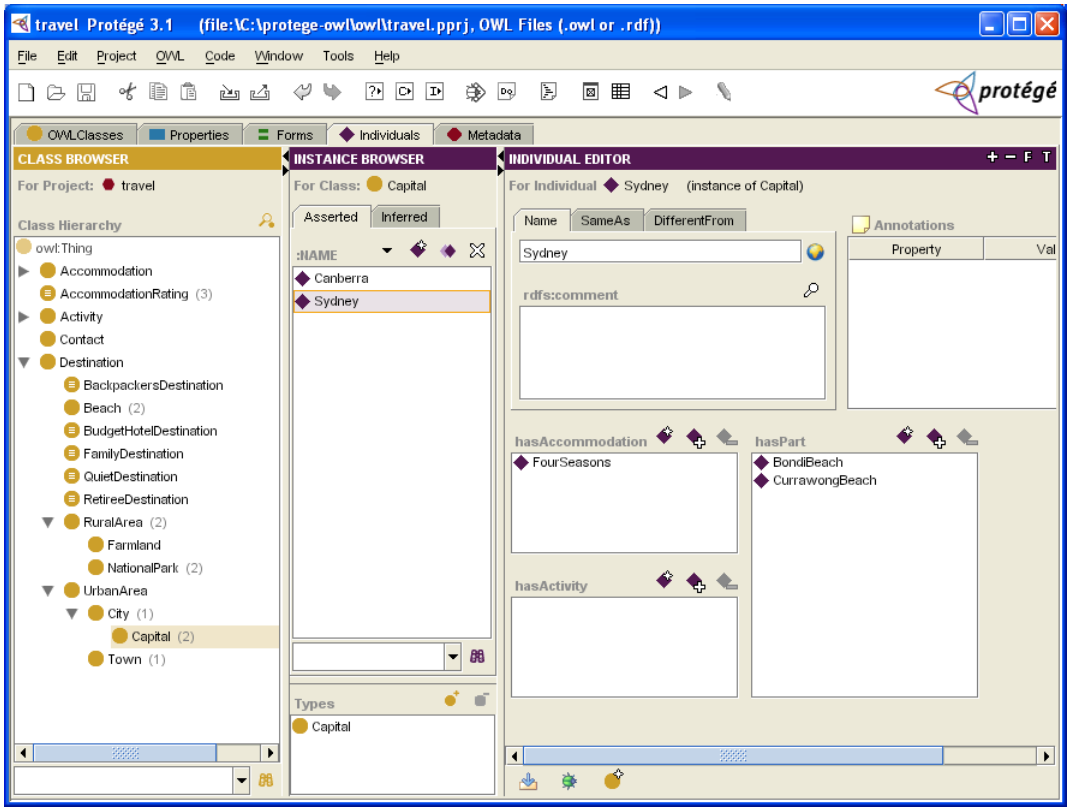
6 References

[Farfeleder, 2011a]	Farfeleder S., Moser T.; Krall A.; Stalhane T.; Zojer H.; Panis C.;*DODT: Increasing requirements formalism using domain ontologies for improved embedded systems development*, DDECS, May 2011
[Farfeleder, 2011b]	Farfeleder S., Moser T., Krall A., Stalhane T., Omoronyia I., Zojer H., *Ontology-Driven Guidance for Requirements Elicitation*, In: LNCS11, vol. 6644, pp. 212-226, 2011
[Feld, 2011]	Feld M., Müller C.: The automotive ontology: managing knowlwdge inside the vehicle and sharing it between cars, 2011
[Gruber, 1993]	Thomas R. Gruber: Toward principles for the design of ontologies used for knowledge sharing. Originally in N. Guarino and R. Poli, (Eds.), *International Workshop on Formal Ontology,* Padova, Italy. Revised August 1993. Published in *International Journal of Human-Computer Studies*, Volume 43, Issue 5-6 Nov./Dec. 1995, Pages: 907-928, special issue on the role of formal ontology in the information technology.
[Malot, 2011]	CESAR D_SP2_R3.3_M3_Vol 4 http://www.cesarproject.eu/fileadmin/user_upload/CESAR_D_SP2_R3.3_M3_Vol4_v1.000_PU.pdf
[Mitschke, 2010]	CESAR D_SP2_R2.2_M2 http://www.cesarproject.eu/fileadmin/user_upload/CESAR_D_SP2_R2.2_M2_v1.000_PU.pdf
[ISO, 2011-2012]	Technical Committee ISO/TC 22, Road vehicles, Subcommittee SC 3, Electrical and electronic equipment; *ISO 26262 standard: Road Vehicles – Functional Safety [Part1-10]*
[IEC, 2010]	IEC; *IEC 61508 standard: Functional safety of electrical/electronic/programmable electronic safety-related systems* (last rev. 2010)
[RDF, 2004]	RDF Primer, W3C Recommendation, http://www.w3.org/TR/2004/REC-rdf-primer-20040210/, 2004
[Sternudd, 2011]	Sternudd P., Master Thesis: *Unambiguous requirements in Functional Safety and ISO 26262: dream or reality?* pp. 63 – 79, 2011

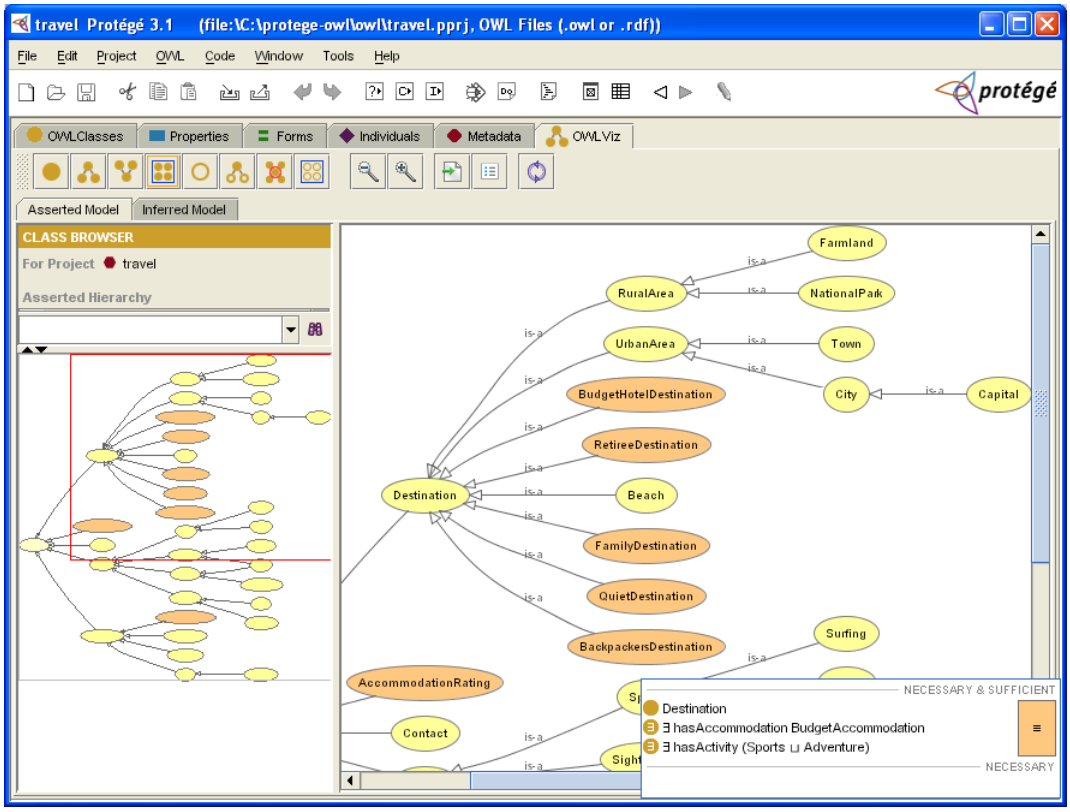
7 Annex

Annex I: Protégé

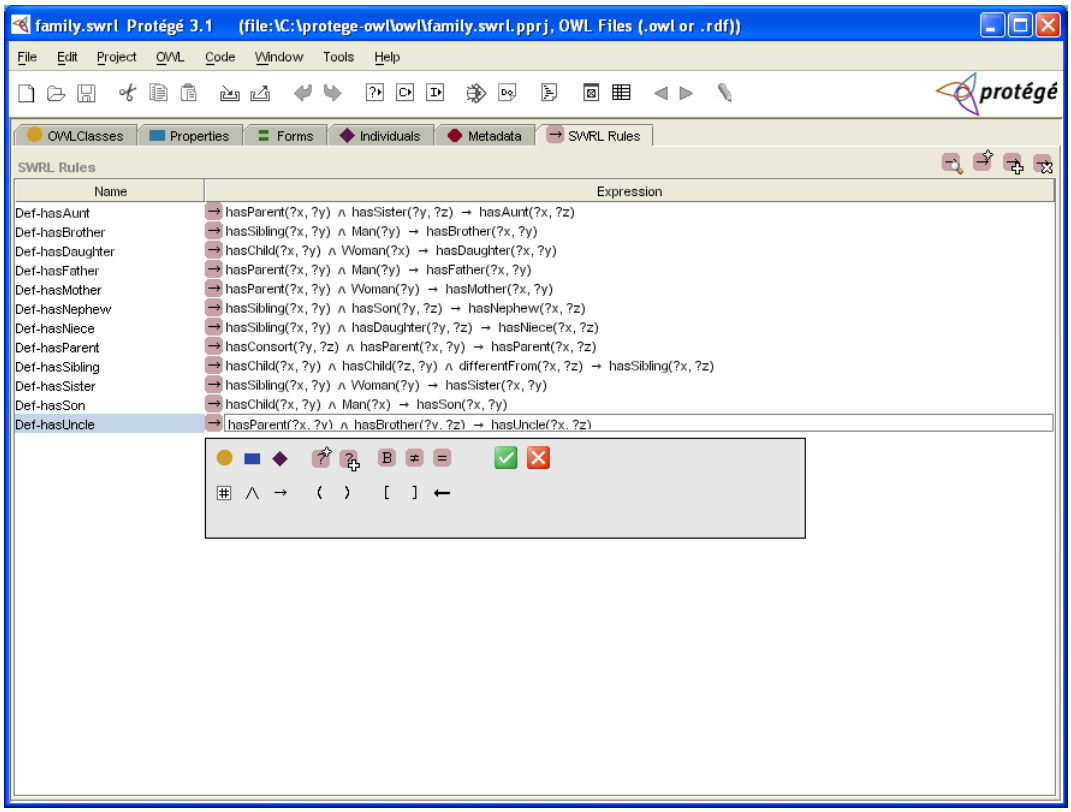
This section provides additional material related to section 2.2. The following screenshots provided by Stanford University show the functionality of Protégé (http://protege.stanford.edu/overview/po-screenshots.html).



Annex I-1: Based on the classes shown on the left, instances can be acquired on the right



Annex I-2: The created ontology can be displayed in a graph



Annex I-3: Protégé includes also some extensions - in this case an editor for the Semantic Web Rule Language

Annex II: Description of selected EAST-ADL packages

This section extends the information from section 3.4. The information in these sections has been retrieved from the website of the EAST-ADL Association[10].

Structure

The structure package contains the sub-packages SystemModeling, FeatureModeling, VehicleFeatureModeling, FunctionModeling, and HardwareModeling.

- **SystemModeling** contains the abstraction levels, whereas
- **FeatureModeling** contains concepts for feature modelling using feature trees, feature constraints, binding times, and variability.

[10] www.east-adl.info

- **VehicleFeatueModeling** introduces vehicle features as a representation of vehicle functionality at the vehicle level.
- **FunctionModeling** contains concepts for function modelling at both analysis and design level. This includes the concepts of types and prototypes, function connectors and ports, allocation of functions to hardware.
- **HardwareModeling** contains concepts such as sensors, actuators, nodes, hardware component types and prototypes, hardware connectors, ports and pins, busses and allocation targets.

Variability

The variability package is an extension to the structural modelling of EAST-ADL. It contains concepts to denote which structural elements that are variable and under which conditions the different variants are applicable. The conditions are typically linked to the feature modelling. Main concepts of interest are:

- **VariableElement** Marks the structural element (typically a function prototype as part of a function type) that is considered optional.
- **VariationGroup** Determines how variable elements may be combined, e.g. one requires the other.
- **ConfigurableContainer** Marks the structural element (typically a function type) which contains variable elements and is associated with a feature model.
- **ConfigurationDecision** A rule which determines when a configuration decision holds and what the effect is on the configuration.
- **ContainerConfiguration** Defines an actual configuration of the variable content of a configurable container.
- **FeatureConfiguration** Defines an actual feature configuration, i.e. selection/deselection of optional features.

Timing

The timing package contains timing concepts developed within the TIMMO[11] and TIMMO-2-USE[12] projects. The timing language resulting from these projects are also referred to as TADL and TADL2. The purpose of

[11] www.timmo-2-use.org/timmo/index.htm

[12] www.timmo-2-use.org

the timing constructs is to allow modelling of timing requirements and timing properties of the system. Main concepts of interest are:

- **TimingConstraint** captures timing information that can be considered as a requirement or property depending on role. Specific timing constraints include ExecutionTimeConstraint, DelayConstraint, RepetitionConstraint, SynchronizationConstraint, AgeConstraint and ReactionConstraint. Timing constraints are associated to events and event chains.
- **Event** Represents an identifiable state change in the system, e.g. a function has finished its execution.
- **EventChain** Acts as a container for two events that are interpreted as being causally related, i.e. a stimulus event and a response event. In TADL the event chain also contained the path between the events but this is no longer included in TADL2.

Requirements

The requirements package contains concepts for requirements modelling. This includes the formulation of requirements as well as the relationships between requirements and structural elements. Main concepts include:

- **Requirement** A text string stating a capability or condition of the system that must hold.
- **Satisfy** Marks that a structural element satisfies a requirement
- **Refine** Marks that a structural element refines a requirement
- **DeriveRequirement** Marks that a requirement is derived from another requirement
- **QualityRequirement** Represents a non-functional requirement

In addition, two sub-packages related to use cases and verification and validation are contained in the requirements package. The use-case package contains concepts for use-case modelling, i.e. actors, use cases, and extend as well as include relationships. Use cases are related to requirements since use-cases are another way to describe system capabilities. The verification and validation package contains concepts for linking of V&V activities and artefacts to requirements. For example it contains the relationship Verify which can be used to mark that a verification effort (e.g. test case) verifies a requirement.

Dependability

The dependability package contains concepts for modelling of information particularly related to functional safety. This includes hazard analysis and classification, derivation of safety requirements, fault propagation

and error modelling, organization of safety evidence in a safety case. The concepts are aligned with the ISO26262 standard. Main concepts include:

- **Item** Identifies the scope of the safety information.
- **Hazard** A state or condition of the system that may contribute to accidents. Caused by malfunctioning of the EE system.
- **HazardousEvent** A combination of a hazard and an operational situation (typically characterized by environment and traffic situation) and usage.
- **FeatureFlaw** Represents an abstract failure where the system cannot fulfil some of its requirements.
- **SafetyGoal** Top-level safety requirement. Defines how to mitigate one or more hazardous events.
- **SafetyConstraint** The qualitative integrity constraint on a fault or failure (i.e. ASIL). Can represent a required or actual integrity level depending on role.
- **FunctionalSafetyConcept** Represents the set of (functional safety) requirements that together fulfil a safety goal.
- **TechnicalSafetyConcept** Represents the set of (technical safety) requirements that together fulfil a functional safety concept and safety goal.