Document Title	Specification of Interoperability of Authoring Tools
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	204
Document Classification	Auxiliary

Document Version	1.3.1
Document Status	Final
Part of Release	3.2
Revision	1

Document Change History

Date	Version	Changed by	Change Description
23.03.2011	1.3.1	AUTOSAR Administration	Legal disclaimer revised
29.01.2010	1.3.0	AUTOSAR Administration	• Updated of semantics of identifier w.r.t. lower/upper case • Legal disclaimer revised
23.06.2008	1.2.1	AUTOSAR Administration	Legal disclaimer revised
14.11.2007	1.2.0	AUTOSAR Administration	• Added description on how to merge models • Removed dependencies to no longer existing documents • Added requirements on extension mechanism • Added requirements on handling / exchanging errors • Document meta information extended • Small layout adaptations made
31.01.2007	1.1.0	AUTOSAR Administration	NonSplitableElements are the minimum granularity of most AUTOSAR descriptions. In case a smaller granularity is required, the Metamodel may explicitly define SplitableElements • Legal disclaimer revised • Release Notes added • “Advice for users” revised • “Revision Information” added
12.12.2005	1.0.0	AUTOSAR Administration	Initial release

Special Note:

This specification is not yet precise enough to ensure that any involved party will interpret it in the same way. In particular the Methodology [3] needs to be refined further with respect to the role of UUID and ShortName.

Disclaimer

Advice for users

AUTOSAR Specification Documents may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).

Any such exemplary items are contained in the Specification Documents for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such Specification Documents, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

Table of Contents

1 Introduction

1.1 Aspects of AUTOSAR authoring tools (non-normative)

The AUTOSAR methodology document [3] describes the major steps of a development of system with AUTOSAR: from the system level to the generation of an ECU executable. It describes the dependencies of work-products and activities. Each authoring tool can support one or more activities.

The term *AUTOSAR authoring tool* refers to all tools that support the activities of interpretation, modification and creation of AUTOSAR models which describe a system and its configuration as defined in e.g. the

- Software-Component Template [4],
- ECU Resource Template [6],
- System Template [5],
- Basic Software Module Description Template [8],
- ECU Configuration Template [7],

In particular, AUTOSAR authoring tools are required to be able to interpret, create or modify AUTOSAR XML descriptions (i.e. the XML representation of AUTOSAR models, see [12]).

Figure 1-1 sketches the descriptions that can be maintained by AUTOSAR authoring tools within the AUTOSAR methodology (for a detailed description of the notation see [3]). According to the AUTOSAR methodology, the System Configuration Input consists of models describing software-components, ECU hardware and some system constraints.

The formal description of AUTOSAR software-components does not include a complete formal description of the behavior of the software-component. The latter is intentionally left to dedicate Behavior Modeling Tools (BMT). It is therefore necessary to bridge the gap between a software-component model and the corresponding behavior model created by a particular BMT. This task is carried out by the "Coupling Tool" mentioned in Figure 1-1.

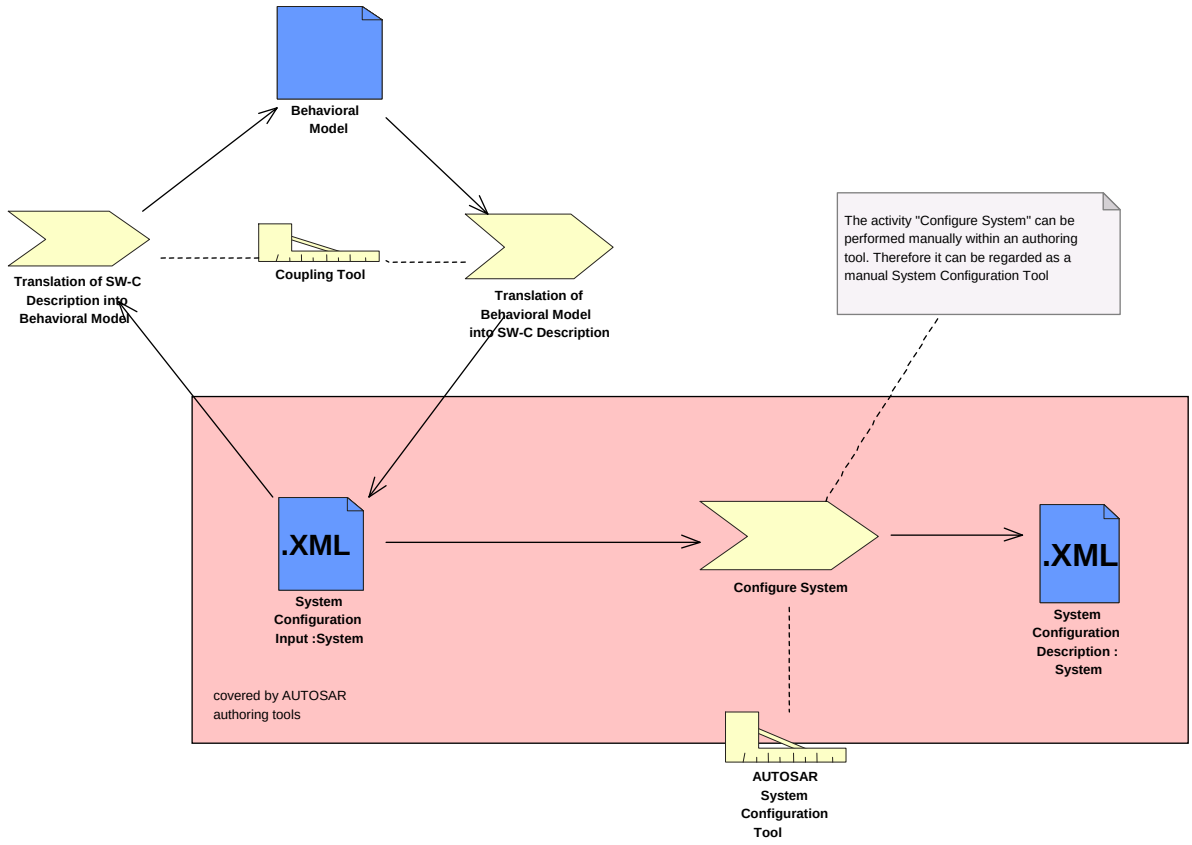

Figure 1-1: Descriptions which can be created and modified by AUTOSAR authoring tools

A further aspect of AUTOSAR authoring tools is the configuration of the System that (as sketched in Figure 1-1) produces the System Configuration Description as an output. Please note that this task can be carried out manually or (to some extent) automatically.

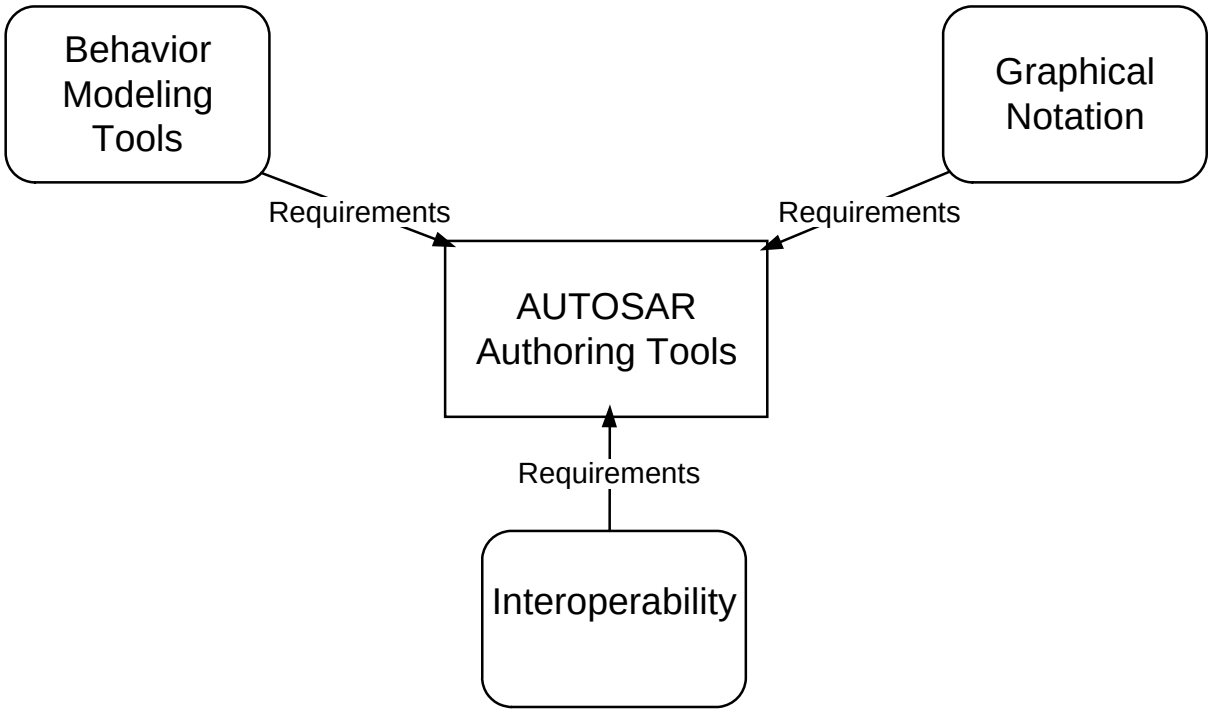

Figure 1-2: Aspects of AUTOSAR authoring tools

The description of AUTOSAR authoring tools covers several important aspects as depicted in Figure 1-2. Please note that the description of all these aspects results in the formulation of requirements on AUTOSAR authoring tools.
Each aspect as depicted in Figure 1-2 is described in a separate AUTOSAR document. In other words: beyond the scope of this document at hand, a separate discussion of specific aspects of AUTOSAR authoring tools (as depicted in Figure 1-2) is available (in a separate document for each aspect):

- **Specification of Interoperability of Authoring Tools (this document)**
  This document emphasizes on issues that might come up when exchanging AUTOSAR models between different tools. After describing some basic concepts of data exchange this document sketches strategies on how these issues can be resolved. Requirements on AUTOSAR authoring tools for ensuring interoperability are defined.
- **Specification of Interaction with Behavioral Models [10]**
  The document ”AUTOSAR Interaction with Behavioral Models” lists use-cases for behavior modeling within AUTOSAR. Parts of the AUTOSAR meta model which are relevant for behavior modeling are identified. Requirements for the interaction of software component descriptions with behavior models are derived.
- **Specification of Graphical Notation [11]**
  The “Graphical Notation” document defines the graphical AUTOSAR notation for AUTOSAR authoring tools. For example, the document provides a comprehensive schema for graphically modeling `CompositionTypes`. The graphical notation should be used as a guideline for implementing AUTOSAR authoring tools.

It is advised to read all of these documents in order to understand the overall concept of AUTOSAR authoring tools.
The dependency relationships of this document are depicted in Figure 1-3.

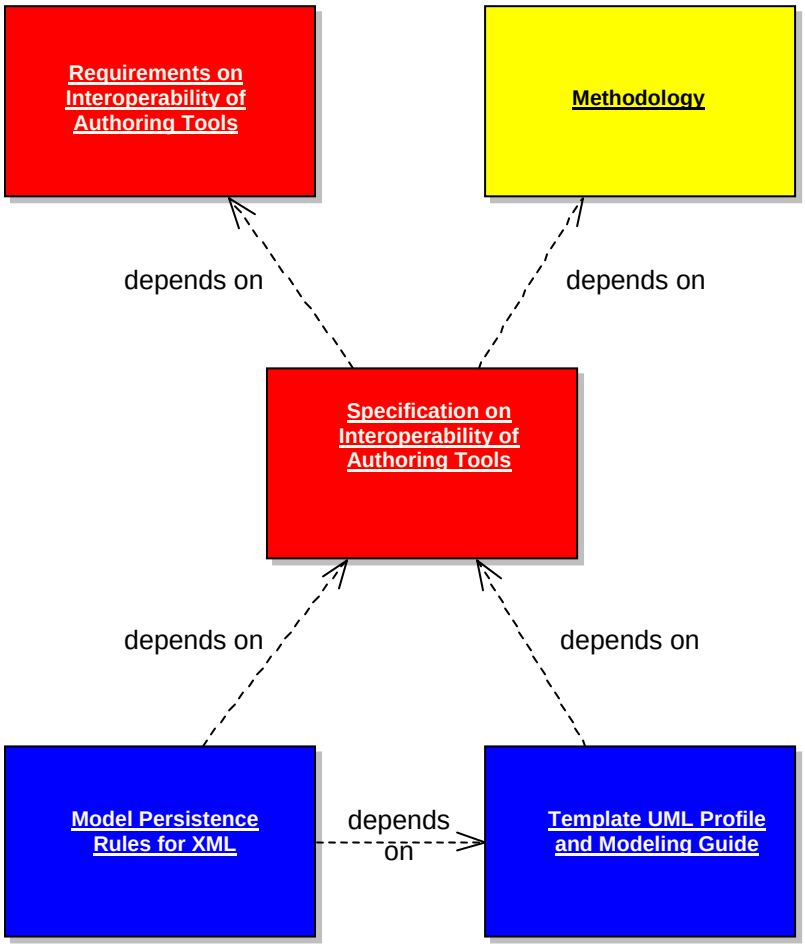

Figure 1-3: Document dependencies.

According to Figure 1-3, this document depends on the "Requirements on Interoperability of Authoring Tools" [2] and the "Methodology" [3]. This document defines requirements on AUTOSAR authoring tools, the document "Template UML-Profile and Modeling-Guide" [13] and the document "Model Persistence Rules for XML" [12].

1.2 Origins and goals (non-normative)

Whenever data is exchanged between different parties they need to agree on a common understanding about the wording and the semantics. Otherwise there will be misunderstandings. This observation applies to communication between different tools as well.
AUTOSAR formally defines the structure and semantics of data by means of UML[1] class diagrams and OCL[2]. In addition to a common data exchange language

[1] UML: Unified Modeling Language [27]

(specified by the AUTOSAR XML schema) further issues need to be considered in order to support a successful communication.
This document points out potential problems when exchanging models between different tools and companies and defines strategies on how these problems could be solved.
Based on the "Requirement on Interoperability of authoring tools" [2] and a set of use-cases (chapter 3) four kinds of requirements are defined:

- Requirements on AUTOSAR authoring tools
- Requirements on the AUTOSAR XML schema
- Requirements on the AUTOSAR metamodel
- Requirements on metadata for data exchange

All requirements described in this document are summarized in chapters 8.1 to 8.4.
These requirements mainly focus on interoperability of AUTOSAR authoring tools. **However, they are also valid for other tools which need to interpret or create AUTOSAR XML descriptions.**

This document is structured as follows:

- Chapter 1 "Introduction" (this chapter) gives an overview of the documents that deal with AUTOSAR authoring tools.

- Chapter 2 "Requirements tracing" lists the requirements on this document and associates the chapters where these requirements are addressed.

- Chapter 3 "Use-Cases (non-normative)" gives some examples on information interchange between tools. The intention is to point out potential problems while using different tools.

- Chapter 4 "Use-case tracing (non-normative)" shows which use-cases are covered in which sections of this document.

- Chapter 5 "Basic concepts" describes some basic concepts which are used in the following chapters.
  - The subchapter 5.1 "Data representation" explains the relationship between AUTOSAR template, AUTOSAR XML schema, AUTOSAR models, AUTOSAR XML descriptions, etc.
  - The following subchapter 5.2 "Abstraction levels of information exchange via XML" shall help to understand the tasks which need to be done when exchanging AUTOSAR models via XML. These tasks are illustrated by different levels. Note: tools are not forced to implement this architecture.

- Chapter 6 "Requirements on AUTOSAR authoring tools " explains concepts for tool interoperability and defines requirements on AUTOSAR authoring tools. E.g.:
  - Integration of tools which do not support the full set of information defined in the metamodel: Those tools only need to import and export the information that can be internally represented. Merging the results

[2] OCL: Object Constraint Language[28]

of the input information with the output exported by the authoring tool results in an updated overall model. All authoring tools are required to support this merge for the set of information that is internally supported.
  - Migration between different versions of the AUTOSAR metamodel: Each tool should support manual or automatic upgrades of AUTOSAR XML descriptions which were created with respect to the last major version of the AUTOSAR metamodel.
  - Support for concurrent modeling of AUTOSAR systems: e.g. each authoring tool should support working on incomplete AUTOSAR models. It should be possible to merge several AUTOSAR models together.

- Chapter 7 “Requirements on the AUTOSAR data exchange format” defines requirements on the *AUTOSAR metamodel*, the *AUTOSAR XML schema* which is derived from the *AUTOSAR metamodel* and *metadata for data exchange*.

- Chapter 08 ”Compliance” discusses the compliance of AUTOSAR authoring tools and summarizes the requirements defined in this document: A tool may be called AUTOSAR compliant if it implements all mandatory requirements on authoring tools defined in this document. Compliant tools should be able to be used within the AUTOSAR methodology. However, a seamless exchange of AUTOSAR XML descriptions between different tools is only possible if they support the same set of information.

- Chapter 9 "References” finally lists the documents that are referenced by this document.

1.3 Terminology

- The **AUTOSAR metamodel** is a UML2.0 model that defines the language for describing AUTOSAR systems. The AUTOSAR metamodel is a graphical representation of a template. UML2.0 class diagrams are used to describe the attributes and their interrelationships. Stereotypes and OCL (object constraint language) are used for defining specific semantics and constraints.
- An **AUTOSAR model** is an instance of the AUTOSAR metamodel. The information contained in the AUTOSAR model can be anything that is representable according to the AUTOSAR metamodel. The AUTOSAR model can be stored in many different ways: it might be a set of files in a file system, an XML stream, a database or memory used by some running software tools, etc.
- The **AUTOSAR XML Schema** is a W3C XML schema that defines the language for exchanging AUTOSAR models. This Schema is derived from the AUTOSAR metamodel. The AUTOSAR XML Schema defines the AUTOSAR data exchange format.
- An **AUTOSAR XML description** describes the XML representation of an AUTOSAR model. The AUTOSAR XML description can consist of several fragments (e.g. files). Each individual fragment must validate successfully against the AUTOSAR XML schema.

- An **AUTOSAR authoring tool** is a software tool which supports interpreting, processing and creating of AUTOSAR XML descriptions.
- **Metadata** includes pertinent information **about** data, including information about the authorship, versioning, access-rights, timestamps etc.

2 Requirements tracing

The requirements on this document are described in the document “Requirements on Interoperability of authoring tools” [2]. Table 1 contains a requirements trace matrix that indicates where these requirements are covered in this document.

Requirement	Satisfied by
ATIREQ_001 Support data exchange	Sections 5, 6, 7
ATIREQ_002 Standardize the handling of errors in AUTOSAR models	Section 6.7

Table 1: Requirements trace matrix

2.1 About requirements

Each requirement defined in this document has its unique identifier starting with the prefix "ATI" (meaning **A**uthoring **T**ool **I**nteroperability).

2.1.1 Structure

Each requirement is defined as a table. The structure of the tables is as follows:

Initiator:	< number of originating work package, company, etc >
Date:	< date of last change >
Requirement:	< the normative text of the requirement >
Description:	< detailed description of the requirement >
Rationale:	< why is this necessary, what its omission could cause >
Use Case:	< example to a scenario that makes the requirement necessary or useful >
Dependencies:	< reference to depending and depended-on requirements >
Conflicts:	< reference to conflicting requirement >
Supporting Material:	< links to other documents >
Comment:	< additional remarks >

2.1.2 Conventions used

In requirements, the following specific semantics are used (taken from Request for Comments RFC 2119 from the Internet Engineering Task Force IETF):
The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119. Note that the requirement level of the document in which they are used modifies the force of these words.

- MUST: This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification.
- MUST NOT: This phrase, or the phrase "SHALL NOT", means that the definition is an absolute prohibition of the specification.
- SHOULD: This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
- SHOULD NOT: This phrase, or the phrase "NOT RECOMMENDED" mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
- MAY: This word, or the adjective "OPTIONAL", means that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation, which does not include a particular option, MUST be prepared to interoperate with another implementation, which does include the option, though perhaps with reduced functionality. In the same vein an implementation, which does include a particular option, MUST be prepared to interoperate with another implementation, which does not include the option (except, of course, for the feature the option provides.)

3 Use-Cases (non-normative)

This chapter describes use-cases for interoperability of AUTOSAR authoring tools. The intention of these uses cases is to point out the potential problems that might occur when exchanging AUTOSAR models (represented as AUTOSAR XML descriptions) in a development process. It is NOT intended to define a standardized AUTOSAR process. **The use-cases are EXAMPLES that are intended to highlight potential problems while exchanging AUTOSAR models.**
Each use-case defined in this document has its unique identifier starting with the prefix “ATUC” (meaning **A**uthoring **T**ool **U**se **C**ase).

3.1.1 [ATUC_005] Usage within the different steps of top-down functional development

Short description
Usage within the different steps of top-down functional development

When developing a system using the top-down approach the AUTOSAR models are first initiated as outlines, then refined and updated by the OEM or the supplier through successive iteration loops as the development of the network and the ECUs progresses. This development process includes for example the following steps which are related to tool interoperability:

- The initial AUTOSAR model might be automatically generated out of an existing proprietary database or created manually from scratch.
  (Action: conversion of data from a proprietary database to the AUTOSAR format)
- The incomplete result might be edited by another tool and/or person.
  (Action: exchange of incomplete models between tools within a company)
- The AUTOSAR model might be created to a given level of granularity by the OEM and then passed over to another department or to a supplier.
  (Action: exchange of incomplete models between tools that are used in different companies)
- It might be the case that only a subset of the whole AUTOSAR model is passed to a supplier. The supplier might need to make sure that all required information is available.
  (Action: extraction of an AUTOSAR model out of the full AUTOSAR model , The extracted model only contains the information that is required by another party; check if model was changed while it was sent to another party)
- A supplier might be contracted to implement an AUTOSAR software component. He needs to return a complete AUTOSAR model for the implemented component. The OEM might need to evaluate if the AUTOSAR model is complete in order to facilitate further processing in the AUTOSAR tool chain.
  (check if a model that is returned from another party only contains valid

changes. E.g. only the AUTOSAR software component was changed. The interface descriptions were not changed)

- At some point of time some AUTOSAR models might need to be integrated / merged. Potential collisions need to be resolved.
  (merge models)

3.1.2 [ATUC_001] Support for subcontracting

Short description
Extracts from an AUTOSAR model of an OEM is passed for further refinement and implementation to a supplier. The results need to be integrated.

Automotive systems are developed by several companies. An OEM could develop a system until a given granularity is reached and then pass the further development to one or more suppliers.
For example an OEM defines a coarse grained architecture of software components, their interfaces and connectors between them. This architecture is refined and implemented by some suppliers. The suppliers are not allowed to change any interfaces which were defined by the OEM. Otherwise this would lead to problems during the integration phase. The OEM needs to find out which changes on the models have been made by the suppliers. Therefore a tool that checks differences between models is required.
Additionally a formal mechanism for explicitly describing which parts of a model may be modified or extended by suppliers could avoid misunderstandings and conflicts during integration of the results. Authoring tools could evaluate the access rights and warn the user if he tries to modify elements he is not allowed to edit.

3.1.3 [ATUC_002] Versioning of metamodels

Short description
Dealing with changes of the AUTOSAR metamodel over time.

The AUTOSAR metamodel and the derived AUTOSAR data exchange format will change over time. It should be predictable if tools (potentially with different underlying metamodel versions) can exchange AUTOSAR models.

3.1.4 [ATUC_003] Versioning of models

Short description
Dealing with changes of AUTOSAR models, AUTOSAR metamodel and AUTOSAR tools over time.

Versioning of the AUTOSAR models:
If a "bug" in an interface description is found, it needs to be ensured that all models (that are for example in the hands of suppliers) are updated consistently to reflect the corrected version. It is not intended to have several versions of the same model element within the same model.

3.1.5 [ATUC_004] Concurrent modeling

Short description
Allowing for concurrent work on the same model

A complete system can be represented as a big AUTOSAR model. Several co-workers, departments or even companies are concurrently working on parts of the model. The following sections describe some more detailed scenarios.

3.1.5.1 Renaming model elements

Parties X and Y work on model A. Elements of the model are connected to elements of Model B, which is local to party X (see Figure 3-1).

- Party X has model A, which contains an element “BrakControl”
- Model A is passed on to party Y for further refinement (indicated by the “<<trace>>” arrow in the diagram
- Party Y modifies the model and renames “BrakControl” to “BrakeControl”
- Party Y returns modified element to party X. Party X has to merge modified data. X faces a problem: Party X uses the element “BrakControl”, which is no longer existent in the new model.

If party X identifies elements by their name, party X has no way to decide if “BrakControl” was deleted and a completely new independent element “BrakeControl” has been introduced or if “BrakControl” was renamed. In the latter case, keeping all the original associations of “BrakControl” to other model elements would make sense, in the former it would not.

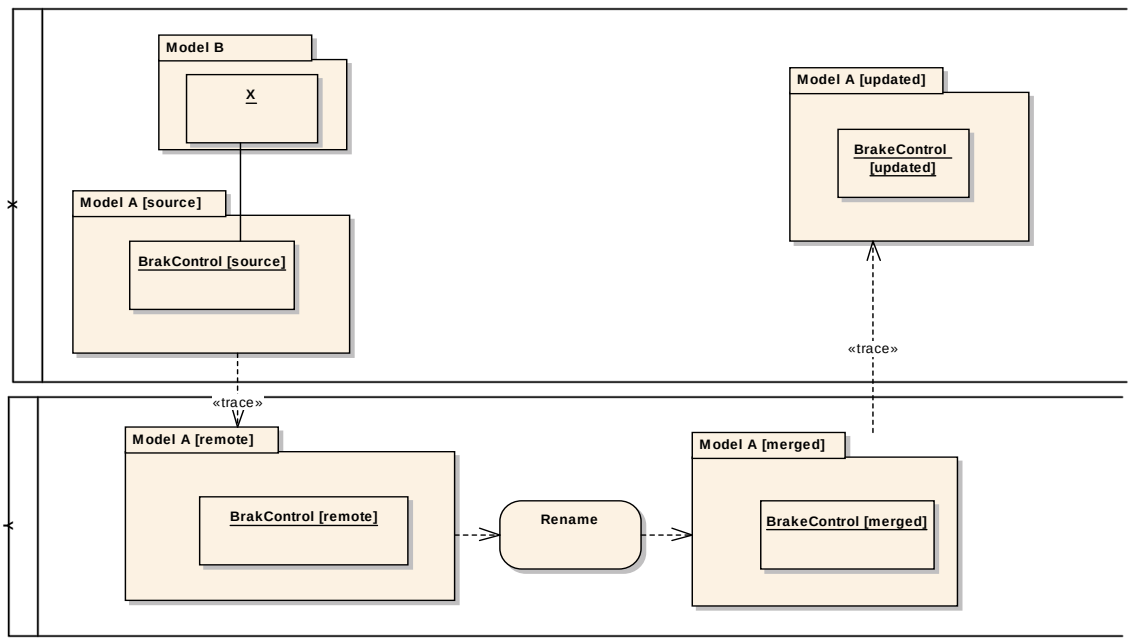

Figure 3-1: Concurrent modeling - renaming of elements

3.1.5.2 Updating of model elements

This scenario is similar to the renaming scenario; it differs only in the workflow (see Figure 3-2):

- Party X has the Model A, which contains an element “BrakControl”
- Model A is passed on to Party Y for further refinement (indicated by the “<<trace>>” arrow in the diagram
- Party Y uses the model and connects model elements to element of its own model B
- While party Y is using model A, party X detects a problem in its model, fixes it and wants to provide the updated model to party Y.
- Party Y has to merge the modified data. Y faces the same problems as in the first renaming scenario: Party Y uses the element “BrakControl”, which is no longer existent in the new model.

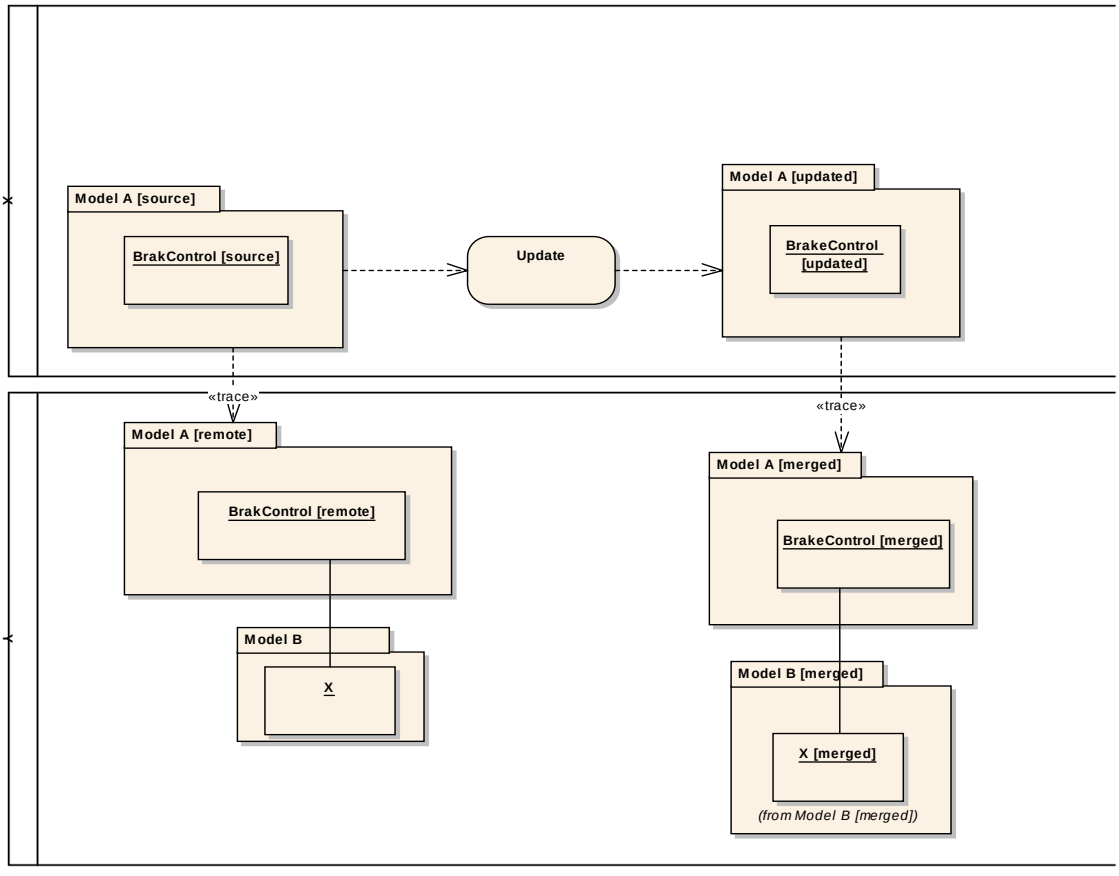

Figure 3-2: Concurrent modeling - update of elements

3.1.5.3 Moving of elements from one namespace to another

If an element is moved from one AUTOSAR namespace to another, this is basically the same as a rename, since model elements are identified by their fully qualified

name (i.e. the full name, including the hierarchy). This scenario is basically similar to the scenarios described in sections 3.1.5.1 and 3.1.5.2.

3.1.5.4 Parallel development of models

Several developers might create models in parallel. Each of them works on his local version of a model. At some point of time it turns out, that developer A needs some model elements that are in the responsibility of developer B. It should be possible that developer A can create a reference to the elements of developer B, even if the content is not available in his local copy.
Another issue which might occur is that developer A and developer B both model the same content. The authoring tool should support merging the models of developer A and B. It should be able to detect potential conflicts.

3.1.6 [ATUC_006] Direct exchange of AUTOSAR model in tool-chain

Short description
Support for direct exchange of AUTOSAR models in a tool-chain.

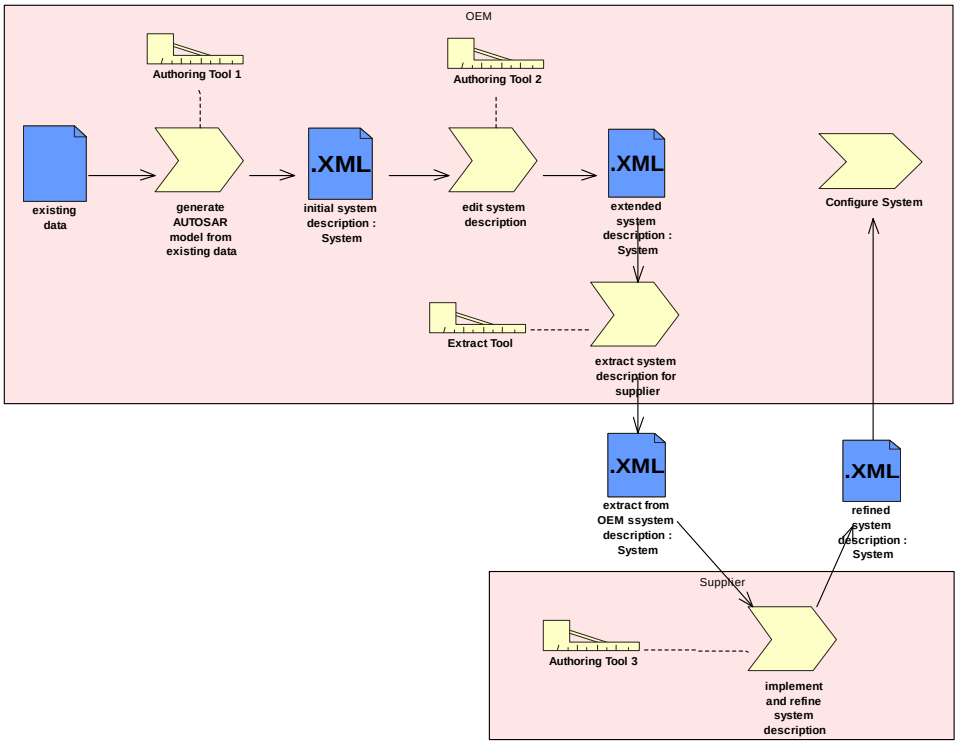

Figure 3-3: Tool-chain

This use case and use case ATUC_007 describe how information could be exchanged between authoring tools. In this use case each tool exports the AUTOSAR model to an XML description which is then directly imported by the next tool. In ATUC_007 tools do not directly exchange models; instead each exported model is stored in a repository before it is imported by the next tool.

A scenario for a direct exchange is:

An OEM might import some data from an existing database and create an initial AUTOSAR model using “Authoring tool 1”. The result is extended by the “Authoring tool 2”. An extract of the AUTOSAR model is passed to a supplier for further refinement. This scenario implies that each tool in the tool chain is able to handle all information created by any other tool which was used in the chain before.

3.1.7 [ATUC_007] Repository for AUTOSAR models

Short description
A company’s AUTOSAR models are stored in a repository. Different tools modify the model.

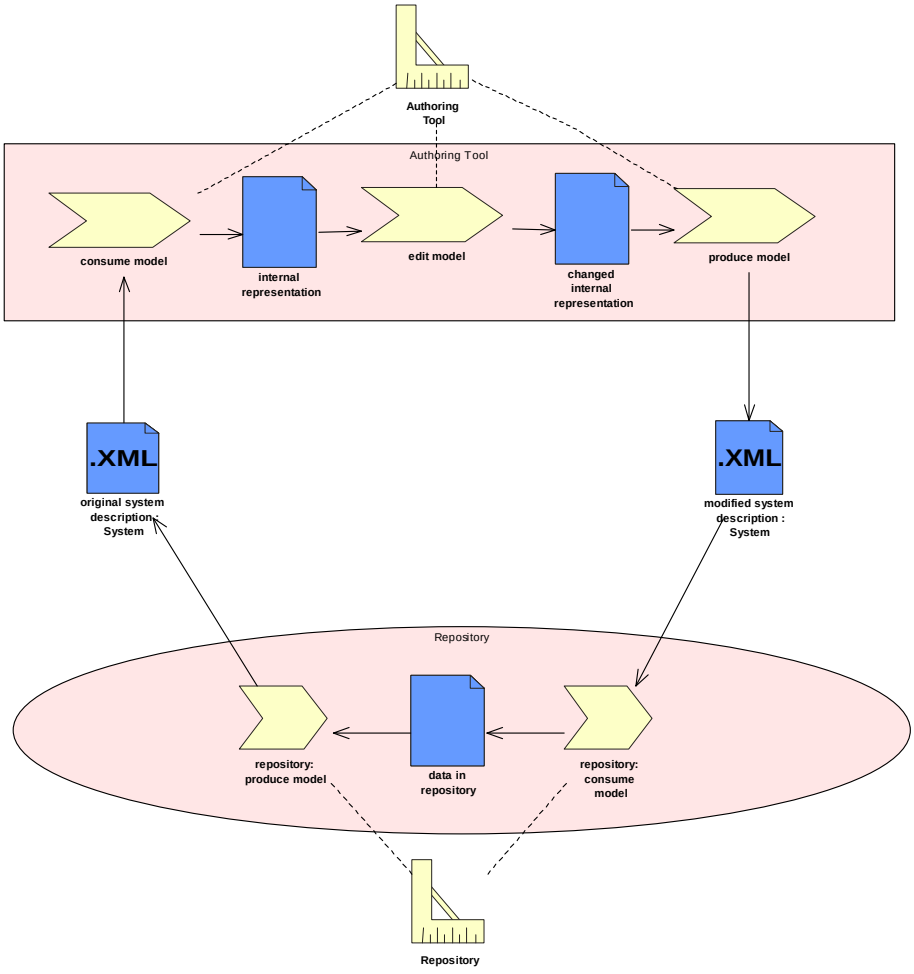

Figure 3-4: Repository

In addition to direct data exchange authoring tools might exchange information via a repository which could handle version management, access rights, etc. In this use-case all AUTOSAR models of a company are stored in a repository and can be accessed by the authoring tools. Extracts of the stored models might be exported to AUTOSAR XML descriptions and edited by different tools, persons or departments. After having performed the changes and imported the changed model in the

repository, it should be possible to check if the overall model which is stored in the repository is still consistent with respect to the structural and semantic rules defined in the AUTOSAR metamodel.
In practice more than one repository exists: Each company often has at least one repository. Some departments within one company might have their own repository. These repositories are often combined with a configuration management system.
When exchanging models with other companies the repository needs to create an extract from the repository that only represents the information that is required or is allowed to be visible by the other party.
The internal data structure of the repository can be different from the data structure defined by AUTOSAR. However, the repository should not perform any not documented changes on the model while the model is being checked in our checked out: i.e.: The semantics of a model must not be changed by the repository (or any other tool within an AUTOSAR tool chain) unless it is explicitly intended by the user.

3.1.8 [ATUC_008] Shipment of AUTOSAR models and related artifacts

Short description
An AUTOSAR model and related artifacts are shipped from one party to another. The receiver wants to check if all information was received correctly.

If two parties exchange an AUTOSAR model, the receiver needs to know the AUTOSAR model that can be ship via several files has been received correctly. In addition to the AUTOSAR model itself, additional artifacts in electronic form need to be shipped as well, e.g. the component's object code or a model of a behavior modeling tool. The sender might also want to add some metadata that describes which parts of the AUTOSAR model may be changed and which are not allowed to be changed.

3.1.9 [ATUC_009] Filter and merge AUTOSAR models

Short description
A filtered subset of an AUTOSAR model is passed to a supplier. The modified model needs to be merged back into the original model after being modified by the supplier.

An OEM creates an AUTOSAR system model and passes a filtered subset to a supplier. The filtered subset only contains information that is relevant for a specific ECU. The supplier modifies and extends the AUTOSAR model and passes the model back to the OEM who needs to merge the modified model into the full AUTOSAR system model.

4 Use-case tracing (non-normative)

Use-case	Satisfied by
ATUC_001 Support for subcontracting	6.2
ATUC_002 Versioning of metamodels	6.5
ATUC_003 Versioning of Models	6.2, 6.6
ATUC_004 Concurrent modeling	6.2
ATUC_005 Top-down functional development	6
ATUC_006 Direct exchange of AUTOSAR models in tool-chain	6.1
ATUC_007 Repository for AUTOSAR models	5
ATUC_008 Shipment of AUTOSAR models and other artifacts	7.2.1.4
ATUC_009 Filter and merge AUTOSAR models	6.2.2

5 Basic concepts

This chapter is intended to provide the reader with a more detailed insight into the exchange of information among different AUTOSAR authoring tools.
AUTOSAR has chosen XML as a language for exchange of data between different AUTOSAR authoring tools. Therefore AUTOSAR authoring tools SHALL be able to interpret and create AUTOSAR XML descriptions. The following section describes the different data representations which will be used in AUTOSAR and clarifies the relationship between the AUTOSAR metamodel, XML and the internal data structure of AUTOSAR authoring tools. Additionally, common tasks that an AUTOSAR authoring tool needs to perform are defined.

5.1 Data representation

In a development process, many different tools with different representation of AUTOSAR models are used (Excel Sheets, Modeling Tools, UML, XML, etc.). Each tool and its underlying representation of data have their advantages and disadvantages. These tools and representations can be grouped into technological spaces.
A technological space is a working context with a set of associated concepts, body of knowledge, tools, required skills, and possibilities [30]. Examples for technological spaces which are used within AUTOSAR are: Metamodel, XML and AUTOSAR authoring tools (see Figure 5-1).
Technological spaces (e.g. Metamodel and XML) are no islands. There are bridges between several technological spaces. The deliverable “AUTOSAR Interaction with Behavioral Models” explains how the AUTOSAR metamodel and the AUTOSAR concepts can be mapped to behavioral models and back again. The document “AUTOSAR model persistence rules for XML” [12] for example defines how to map an AUTOSAR metamodel to a W3C XML Schema.
Using XML and UML within AUTOSAR combines the strength of both technological spaces:
AUTOSAR defined templates for data that is exchanged in AUTOSAR. Since XML is widely accepted as a standard for representation and exchange of structured data it was chosen to be the basis for the exchange of AUTOSAR models.
Due to the complexity of the data and its interrelationships a manual creation of a consistent AUTOSAR XML schema turned out to be time-consuming and error prone. In addition the expressive power of XML schema is not sufficient for expressing content related constraints between data entities.
Therefore a metamodel based approach was chosen to graphically describe the templates by means of UML2.0 class diagrams. Constraints that cannot be formulated graphically are described textually in OCL (Object constraint language). The UML model which defines all data entities and interrelationships that can be used for describing AUTOSAR systems and related artifacts is called AUTOSAR metamodel. An instance of the metamodel, i.e. a concrete description of software components, etc., is called AUTOSAR model.
Figure 5-1 depicts the aforementioned technological spaces. The meta-levels (M0 to M4) show the correspondence of concepts in the different technological spaces. All concepts within one meta-level are strongly related.

Unlike the classical four-layer architecture used by the OMG, five metalevels are shown. Starting at the lowest, most concrete metalevel those are:

M0: AUTOSAR objects / run-time objects

These are the (run-time) instances of specific AUTOSAR entities, for example instances of a windshield wiper software component. Those instances typically allocate physical resources like memory, processing time or space needed by hardware.

M1: AUTOSAR models

Models on this metalevel are built by the AUTOSAR end-user (automotive engineers). They may define a software component called “windshield wiper” with a certain set of ports that is connected to another software component and so on. On this level all artifacts required to describe an AUTOSAR system are detailed, including re-usable types as well as specific instances.

M2: AUTOSAR metamodel

Here it is defined that in AUTOSAR we have an entity called “software component” and another entity called “port”. The relation between those entities as well as their semantics is part of such an overall model.

M3: UML profile for AUTOSAR templates

The templates on M2 are built with the metamodel defined on M3. As discussed before this is UML plus a particular UML profile to better support template modeling work. Formally a template on M2 is still an instance of UML, but at the same time the template profile is *applied*, i.e. that additionally rules set out by the stereotypes in the profile need to be observed.

M4: Meta Object Facility

Just for completeness, OMG’s MOF sits on the final metalevel M4. No further metalevels are required since MOF is designed to be reflective.

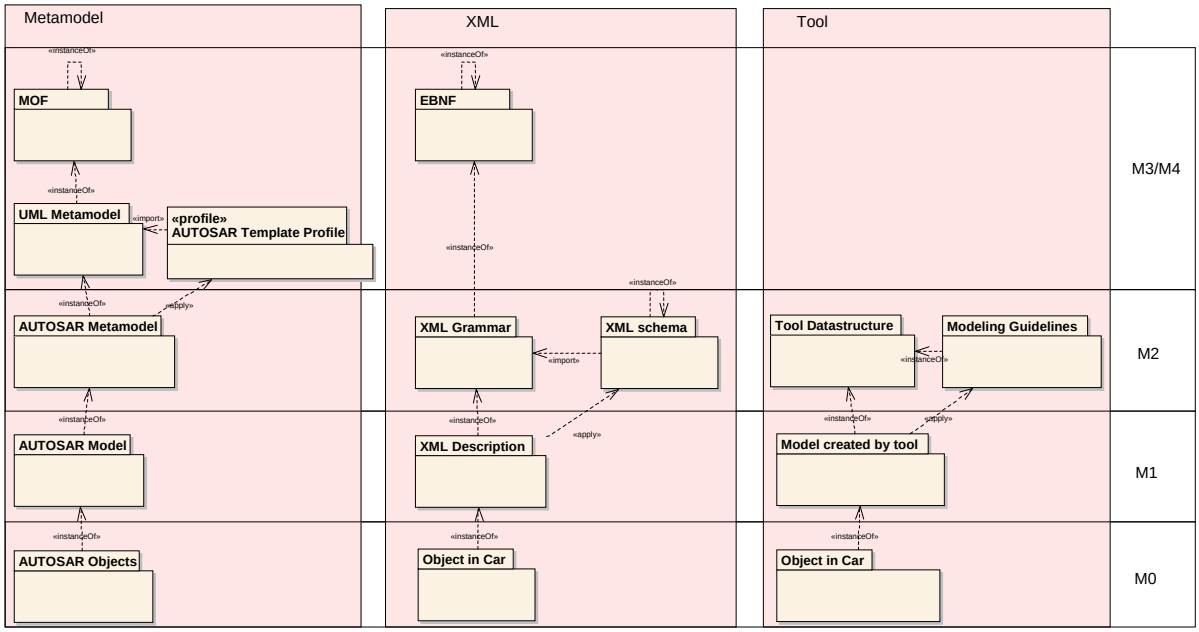

Figure 5-1: Technological Spaces

5.1.1 Metamodel

The Technological Space “Metamodel” relates to the Model Driven Architecture (MDA) approach which was recently proposed by the OMG. According to MDA, the software development process is populated with a number of different models, each representing a particular view on the system being built. Models are written in the language of their meta-model.
The left part in Figure 5-1 shows the Metamodel Technological Space as it is used in AUTOSAR. The lowest part called M0 corresponds to the real world. All AUTOSAR models are at the level M1 and the definition of the language (AUTOSAR Metamodel) used to create these AUTOSAR models is at the level M2. The M2 AUTOSAR Metamodel[3] is described by means of UML2.0 class diagrams and OCL constraints. A detailed description of the language that can be used for creating the AUTOSAR Metamodel is given on the level M3 by the UML2.0 Metamodel and the AUTOSAR Template Profile. The profile defines which UML2.0 elements may be used (e.g.: only class diagrams and OCL, see [13] for more information on the AUTOSAR Template Profile). The UML2 Metamodel is defined by MOF which constitutes the level M4.

5.1.2 XML

Extensible Markup Language (XML) is a markup language standardized by W3C. It is widely accepted as a standard for representation and exchange of structured and semi structured data. The XML description is the central concept in the XML technological space. Descriptions are written in a syntax constrained by well-formedness and validity constraints. Well-formedness constraints are defined by the

[3] A graphical representation of the AUTOSAR template is the AUTOSAR Metamodell

XML grammar rules, whereas the validity constraints are defined in a separate document called XML schema, which is written in a given schema language (W3C XML DTD [16], W3C XML Schema [15], etc.). In other words: The XML grammar describes that a XML description contains opening and closing tags, etc. The XML schema defines e.g. which tags may be used in which combinations.
The middle part of Figure 5-1 illustrates the relations between an XML description, the XML grammar and a XML schema.
The technological space XML can be considered as a low level technological space: The AUTOSAR metamodel can be mapped to a XML schema [12]. But the original AUTOSAR metamodel cannot be reconstructed out of the XML schema.

5.1.3 Tool

Each authoring tool has its internal data structure which implements the concepts that can be used within the authoring tool. This internal data structure is located at the metamodel level (M2) and defines the language which can be used to create descriptions or models (M1). The code that can be generated out of the model is again a different representation of the model (e.g. C). The runtime-instances of the code that are executed on an ECU in a car are represented on metalevel M0.
In most cases the internal data structure of an authoring tool is different from the structure defined by the AUTOSAR metamodel, e.g. for performance or historical reasons. In order to allow interoperability it is required to map the models represented by the internal data structure of the authoring tool on an AUTOSAR XML description.

5.2 Abstraction levels of information exchange via XML

Table 2 depicts several abstraction levels[4] which will be used in this document for structuring the requirements on authoring tool interoperability and the AUTOSAR data exchange format. Each abstraction level is based on the underlying levels. For each abstraction level the mechanism that must be supported is described – beginning with the physical level (sets of files) until the semantic layer (semantic constraints formally specified in the AUTOSAR metamodel). The abstraction levels “presentation level” and “application level” are not relevant for basic authoring tool interoperability but might have impact on the exchangeability of AUTOSAR authoring tools which perform a similar functionality (e.g.: if AUTOSAR authoring tools use a common graphical notation and provide similar mechanism for modifying AUTOSAR models, the effort for introducing another authoring tool is reduced).

In other words: each AUTOSAR authoring tool SHALL support the exchange of AUTOSAR models based on sets of XML-descriptions that can be distributed over several files. Each file in the set of files SHALL validate successfully against the AUTOSAR XML schema that is generated out of the AUTOSAR metamodel. The exchanged model SHOULD NOT violate semantic constraints. When exchanging AUTOSAR models an authoring tool needs to create an AUTOSAR XML description

[4] These abstraction levels were inspired by the OSI reference model (Open System Interconnection). They are intended for providing a structure for the requirements on tool interoperability. These levels do NOT define a communication stack for AUTOSAR tools.

out of the internal data structure. Another authoring tool needs to interpret the AUTOSAR XML description and create its internal data-representation.
In addition to the common mechanism for exchanging AUTOSAR models, authoring tools can implement additional mechanisms. E.g.: An authoring tool could provide a plugin-interface which allows direct access to its internal data-structure. In this case a plugin and the authoring tool would be interoperable on the content level. However, even if an authoring tool supports additional mechanisms for exchanging AUTOSAR models at least the mechanisms defined in Table 2 SHALL be supported.
A more elaborated discussion about the requirements on the AUTOSAR XML data exchange format and the interoperability of AUTOSAR authoring tools is given in the chapters 7 and 6.

Abstraction Level	***Minimum supported mechanism for authoring tool interoperability***	***Document that describes further information***
Application Level	e.g.: Advanced automatic features	Not specified by AUTOSAR
Presentation Level	Graphical notation	Partly specified by AUTOSAR • Graphical Notation [11]
Semantic Level	Semantic constraints precisely described in the metamodel (e.g.: criteria for compatibility of interfaces)	• AUTOSAR metamodel [9], • AUTOSAR Template UML Profile and Modeling Guide [13]
Content Level	Internal data-structure	• This document
Data Format Level	AUTOSAR XML schema	• Model Persistence Rules for XML [12]
Physical Level	Sets of files	• This document

Table 2: Data Exchange Abstraction Levels

5.2.1 Physical level

The physical level defines the physical characteristics of the communication path. A physical representation could be one or more files or a data-stream.

5.2.2 Data format level

The data format level defines the format of the exchanged data. In AUTOSAR the data exchange format between different authoring tools is XML. The exchanged data SHALL be well-formed as defined in the W3C XML 1.1 Specification [16]. Additionally the exchanged XML descriptions SHALL be *valid* with respect to the AUTOSAR XML schema. We refer to this kind of data as “XML descriptions”. This level can be implemented by off-the-shelf XML-parsers.

5.2.3 Content level

The content level defines the amount of information that can be exchanged. On the one hand it defines, which information is allowed to be given (boundedness). On the other hand, this level defines which information must at least be available (coverage). Authoring tools which base on this level can assume that at least a minimum and not more than a maximum of information is available. This level can be partly implemented by off-the-shelf validating XML-parsers which validate against a strict W3C XML schema (which can be generated out of the metamodel if the subset is formally defined). Some checks such as resolving of references must be implemented by the AUTOSAR authoring tool.

Example: An authoring tool might be specialized in the description of interfaces. This authoring tool might not understand any additional XML data. The content level needs to make sure that no data is transferred to the authoring tool which it does not understand.
Additionally the import and export of the AUTOSAR XML descriptions into the AUTOSAR authoring tool internal data structure is realized in this level. If the AUTOSAR XML description was split up over several files, the represented AUTOSAR model needs to be constructed. During this construction merge conflicts can occur and need to be resolved.

Example: references are resolved and potential conflicts such as multiple definitions of the same element are detected.

5.2.4 Semantic level

The definition of a standardized XML based exchange format is only a first step towards successful interoperability of different authoring tools. The AUTOSAR XML data exchange format is defined by the AUTOSAR XML schema and therefore can only cover the “data format level” (validity according to the AUTOSAR XML schema).

In order to allow for seamless authoring tool interoperability all authoring tools must have the same interpretation of the semantics of the AUTOSAR models. E.g. all tools must have a common interpretation of the compatibility of instances of `PortInterfaces`.
The validation of semantic constraints is not only mandatory for the exchange of AUTOSAR models – it is mandatory in any case (i.e. even if the information would not be exchanged among different tools) because it must be possible to check the consistency of AUTOSAR models **during** their creation and maintenance and **before** the model is exported to an AUTOSAR XML description.

The template specifications released by AUTOSAR [4][5][6] already contain discussions of particular semantic constraints. However, the description in the template specifications is not complete and not precise enough for the implementation of tools.

Therefore the semantic constraints must be precisely formulated within the AUTOSAR metamodel that is specified by means of UML. This includes constraints

that are implied by the semantics of UML (e.g.: no loopback relation to the same port is allowed) AND constraints that are defined as OCL constraints. By use of OCL, ambiguities introduced by natural language can be avoided and automated checking can be directly derived from the formally defined semantic constraints.
The way how to model OCL constraints in the metamodel is defined in the AUTOSAR Template UML Profile and Modeling Guide [13].

Example: Compatibility of interfaces

5.2.5 Presentation level

In this level a more abstract access to the AUTOSAR model is supported. This can e.g. be realized by an API for a programming language, by a set of text tables and/or a graphical editor. This level does not support any automatic support for editing the AUTOSAR models. The API and representation of the model is highly depending on the AUTOSAR tool. Therefore AUTOSAR only specifies the graphical notation [11].

Example: editor with no automation support

5.2.6 Application level

The application supports automatic features such as algorithms for tool supported mapping of software components onto ECUs. This document does not specify the behavior of tools on this level, since the behavior is highly depending on the implementation of the tool. Additionally the “automatic” features have no impact on tool interoperability.

Example: semi-automatic algorithms for mapping signals onto buses

6 Requirements on AUTOSAR authoring tools

6.1 Support for AUTOSAR XML data exchange format

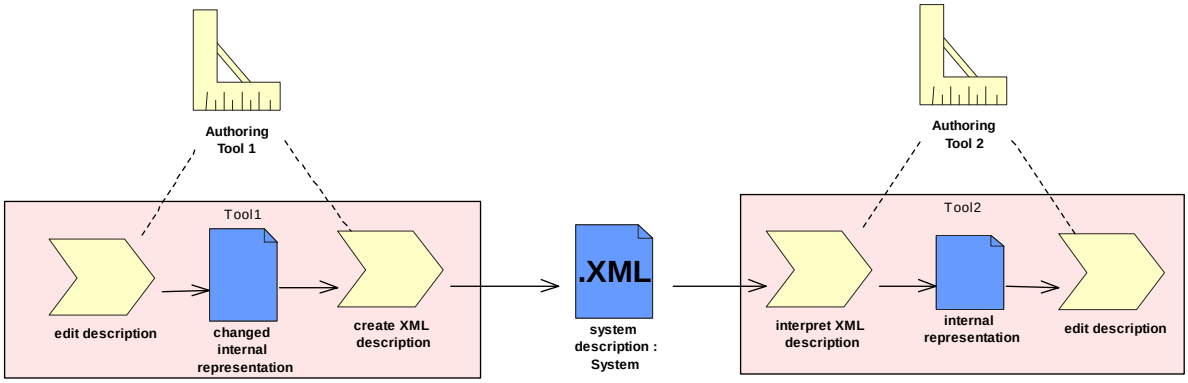

Figure 6-1: Support for AUTOSAR data exchange format

When exchanging data between different departments or companies all involved parties need to agree on a common wording for the exchanged information. Otherwise all parties have a different understanding of the information. The same holds for exchanging AUTOSAR models between AUTOSAR authoring tools: Whenever AUTOSAR models are exchanged they need to be represented as AUTOSAR XML descriptions.
If the tools in a tool-chain are all able to deal with the full set of information as described in the AUTOSAR metamodel, they can perfectly exchange their AUTOSAR models. Of course this requires that all tools have implemented the functionalities which are defined in

- The physical level (e.g. they support files),
- the data format level (e.g. the files are valid with respect to the AUTOSAR XML schema),
- the content level (e.g. the minimum amount of data is zero and the maximum amount of data is defined by the AUTOSAR metamodel), and
- the semantic level (e.g. ALL semantic constraints which are defined for data that is already available evaluate to TRUE).

Optionally these tools can use the same graphical notation as defined in the presentation level.
Using the aforementioned tool architecture consisting of multiple levels the following requirements need to be fulfilled:

6.1.1 Physical level

6.1.1.1 [ATI0010] Authoring tool SHALL support sets of files

Initiator:	WP1.2
Date:	20.04.05
Importance:	High
Requirement:	AUTOSAR authoring tools SHALL support the reading and writing of single files and **sets of files** that are stored in a file system.
Description:	AUTOSAR authoring tools SHALL support for reading and writing single files and of sets of files that are stored in a file system. The tool SHALL provide a mechanism to select a specific files and **sets of files** in the file system.
Rationale:	An AUTOSAR XML description can be shipped in several files. Some files could contain datatypes – others could contain interfaces, etc.
Use Case:	This allows the transport of models via CD, DVD, Email, etc. Splitting up an AUTOSAR model (represented as AUTOSAR XML descriptions) over several files supports concurrent modeling (chapter 6.2) and a more fine-grained versioning (chapter 6.6)
Dependencies:	ATI0036, ATI0023
Conflicts:	
Supporting Material:	
Comment:	See [ATI0042] for a related use case. The set of files could be described by means of metadata for data exchange which might be transmitted with the AUTOSAR XML description (see section 7.3) This requirement does not require a tool to restrict itself to files as the only representation on the physical level. It could also be possible to exchange AUTOSAR XML descriptions via data-streams if all involved parties support this feature. However at least files SHALL be supported.

6.1.2 Data format level

6.1.2.1 [ATI0012] Authoring tool SHALL support AUTOSAR XML descriptions

Initiator:	WP1.2
Date:	20.04.05
Importance:	High
Requirement:	AUTOSAR authoring tool SHALL support the interpretation and creation of AUTOSAR XML descriptions.
Description:	AUTOSAR authoring tools SHALL support the interpretation and creation of AUTOSAR XML descriptions. These descriptions SHALL be “well-formed” and “valid” as defined by the XML recommendation [15][16], whether used with or without the document’s corresponding AUTOSAR XML schema(s). In other words: Even if the tool does not use standard XML mechanisms for validating the XML descriptions it SHALL ensure that the XML descriptions can be successfully validated against the AUTOSAR XML schema.
Rationale:	Although it is optional not to transmit and/or validate an XML description with its AUTOSAR XML schema(s), the XML description SHALL still conform as if the check had been performed.
Use Case:	
Dependencies:	Requirements on the AUTOSAR XML schema are defined in: ATI0019, ATI0023, ATI0025, ATI0027, ATI0028, ATI0029, ATI0030, ATI0031, ATI0032, ATI0029 A specialization of this requirement is defined in ATI0033.
Conflicts:	
Supporting Material:	[15][16]
Comment:	

6.1.2.2 [ATI0033] Authoring tool SHALL be able to import and export supported model elements as AUTOSAR XML descriptions

Initiator:	WP1.2
Date:	23.06.05
Importance:	High
Requirement:	Authoring tool SHALL be able to import and export supported model elements as AUTOSAR XML descriptions.
Description:	For all model elements that are defined by AUTOSAR and are supported by an authoring tool, the tool SHALL provide the user with the possibility to import them from XML descriptions and export them as XML descriptions that validate successfully against the AUTOSAR XML schema. Even if the tool has non AUTOSAR exchange possibility of models it SHALL nonetheless support the creation of the corresponding AUTOSAR XML description.
Rationale:	It is possible that between two tools or independent components of the same tool exists some data exchange mechanism. Such mechanism might make it possible for the tool to bypass AUTOSAR XML descriptions even if the model can be represented by AUTOSAR XML descriptions.
Use Case:	Avoid proprietary exchange formats that bypass the standardized AUTOSAR XML descriptions and therefore endanger the interoperability of tools from different vendors.
Dependencies:	ATI0012
Conflicts:	
Supporting Material:	
Comment:	This requirement is a refinement of ATI0012. It not only requires the tool to be able to interpret and create AUTOSAR XML descriptions. It requires that whenever the tool supports a concept that is presentable by means of an AUTOSAR XML description, then it shall be able to interpret and create those XML descriptions. Please not that an authoring tool is only required to import the content from an XML description that can be internally represented (see chapter 6.4).

6.1.3 Content level

6.1.3.1 [ATI0007] Authoring tool SHALL NOT change model contents without the intention of the user

Initiator:	WP1.2
Date:	21.02.05
Importance:	High
Requirement:	Authoring tool SHALL NOT change model contents without the intention of the user.
Description:	Authoring tool SHALL NOT perform any changes on the AUTOSAR model while interpreting and creating AUTOSAR XML descriptions: If the user doesn't explicitly trigger any changes then the semantics of the AUTOSAR model represented by the original XML description SHALL be equivalent to the semantics of the model represented by the created XML description. This includes that an authoring tool SHALL preserve references even if the target was not available in the input XML description. This also includes that uuids are not allowed to be removed or changed.
Rationale:	
Use Case:	The metamodel contains some elements that are marked by {ordered}. The order in the XML representation may not change without the intention of the user when interpreting and creating the XML description. Other use-cases may apply as well.
Dependencies:	ATI0024
Conflicts:	
Supporting Material:	
Comment:	An AUTOSAR authoring tool is not required to support the full AUTOSAR metamodel (see chapter 6.4). Information that is not internally supported by the tool can be ignored while interpreting an AUTOSAR XML description. Of course those tools can only create AUTOSAR XML descriptions that represent the supported information. Those tools SHALL ensure that the semantics of the supported information is not changed while interpreting and creating XML descriptions.

6.1.3.2 [ATI0035] Authoring tool SHALL support exchange of partial information

Initiator:	WP1.2
Date:	18.07.2005
Importance:	High
Requirement:	Authoring tool SHALL support for exchange of partial information.
Description:	An AUTOSAR authoring tool SHALL support the exchange of AUTOSAR models that are not complete.
Rationale:	It SHALL be possible to exchange intermediate work products.
Use Case:	ATUC_001, ATUC_005
Dependencies:	ATI0024, ATI0019.
Conflicts:	
Supporting Material:	
Comment:	

6.1.3.3 [ATI0048] Authoring tool SHOULD support AUTOSAR extension mechanism

Initiator:	WPII-1.2 (Interoperability Group)
Date:	13.07.2007
Importance:	Medium
Requirement:	Authoring tool SHOULD support the AUTOSAR extension mechanism.
Description:	Authoring tool SHOULD support the AUTOSAR extension mechanism. Tools that do not need the additional information for its intended purpose MAY ignore that information. If several extensions are defined then the tool SHOULD support the relevant extensions and MAY ignore the irrelevant extensions.
Rationale:	For some use-cases it is required to exchange information that is not (yet) captured in the standard data exchange format. This additional information is often specific to a tool (e.g. allow for round-trip engineering) or a tool-chain within a development process.
Use Case:	
Dependencies:	Requirements on the AUTOSAR extension mechanism are described [ATI0031], [ATI0052], [ATI0053], [ATI0054]
Conflicts:	
Supporting Material:	
Comment:	

6.1.4 Semantic level

6.1.4.1 [ATI0003] Authoring tool SHALL support validity checks

Initiator:	WP1.2
Date:	21.02.05
Importance:	Medium
Requirement:	AUTOSAR authoring tool SHALL support validating the consistency of the AUTOSAR model to the AUTOSAR metamodel, including the semantic constraints.
Description:	AUTOSAR authoring tool SHALL support validating the consistency of the AUTOSAR model to the AUTOSAR metamodel, including the semantic constraints. The validation SHALL be performed when interpreting and creating AUTOSAR models. A tool SHOULD allow the user to trigger validity checks manually. The tool SHALL allow for interpreting and creating AUTOSAR XML-descriptions that are not valid with respect to the semantic constraints. If the tool is not able to interpret the file due to the violation of some semantic constraints it SHALL notify the user and indicate the location of the violating elements.
Rationale:	
Use Case:	
Dependencies:	Requirement on the AUTOSAR metamodel concerning semantic constraints are described in ATI0034.
Conflicts:	
Supporting Material:	
Comment:	The semantic constraints should preferably be defined by means of a formal language such OCL. However, a tool is not required to have a generic OCL validator. The validation of the semantic constraints can be implemented in a tool specific manner. In order to allow for the integration of specialized tools, only the information that is internally supported need to be validated. E.g. if a tool is specialized on the description of software components it is optional if this tool could detect errors in parts of AUTOSAR models that are not relevant for the intended purpose.

6.1.4.2 Semantic of Identifier

The attribute `shortName` unambiguously identifies the object within the context given by the first ancestor `Identifiable`. The content of `shortName` therefore needs to be unique (case-insensitive) within a given `Identifiable`.

Note that the check for uniqueness of `shortName` must be performed **case-insensitive**. This supports the good practice that names should not differ in upper / lower case only which would cause a lot of confusion.

The term “case insensitive” indicates that the characters in the sets

```
{a b c d e f g h i j k l m n o p q r s t u v w x y z}
{A B C D E F G H I J K L M N O P Q R S T U V W X Y Z}
```

are respectively considered to be the same. In other words case-insensitive check for uniqueness of `shortNames` result in the fact that e.g. elements with `shortName` "X" and "x" are considered the same and may **not** exist in the same Package.

On the other hand references in AUTOSAR models are created by specifying a **case-sensitive** path of `ShortNames`.

Note that the term ``case-sensitive'' indicates that the characters in the sets

```
{a b c d e f g h i j k l m n o p q r s t u v w x y z}
{A B C D E F G H I J K L M N O P Q R S T U V W X Y Z}
```

are respectively considered to be different. In other words case-sensitive paths “/X/Y” and “/x/y” are **not** the same.

6.1.5 Presentation level

6.1.5.1 [ATI0015] Authoring tool SHOULD use AUTOSAR graphical notation

Initiator:	WP1.2
Date:	20.04.05
Importance:	Optional
Requirement:	Software SHOULD use the AUTOSAR Graphical Notation for displaying AUTOSAR models.
Description:	AUTOSAR authoring tools SHOULD use the AUTOSAR graphical notation (a set of graphical symbols and diagrams) for representing AUTOSAR models.
Rationale:	Common look and feel when working with AUTOSAR models using different tools. Learning curve less steep for users when switching between authoring tools.
Use Case:	Engineers who are used to use different AUTOSAR authoring tools should be able to discuss about AUTOSAR models without having to learn new graphical notations.
Dependencies:	
Conflicts:	
Supporting Material:	See “AUTOSAR graphical notation” [11] for more details on the AUTOSAR graphical notation and additional requirements on exchange of layout information.
Comment:	

6.1.5.2 [ATI0014] Authoring tool SHALL support standardized error codes for errors in AUTOSAR models

Initiator:	WP1.2
Date:	21.02.05
Importance:	High
Requirement:	Authoring tools SHALL support standardized error codes for errors in AUTOSAR models.
Description:	An AUTOSAR authoring tool SHALL support standardized error codes for errors in AUTOSAR models. The text of each error message can freely be chosen.
Rationale:	The main benefit of this approach is the comparability of tool behavior in case of errors.
Use Case:	Different project partners carry out an AUTOSAR software project by means of different tools for e.g. structural design. By using standardized error codes, the partners can be sure that they talk about the same issue.
Dependencies:	
Conflicts:	
Supporting Material:	More details on error codes are described in section 6.7
Comment:	The text of each error message can freely be chosen since most AUTOSAR authoring tools will use an internal data-structure that is different from the AUTOSAR metamodel. The user of the tool would be absolutely confused if he would get an error message on some concepts that are hidden to the user (e.g. for complexity reasons). It would be more helpful if the error-message contains some information on how the error can be resolved by means of the currently used authoring tool. However, the error-code is required to figure out, if two engineers who are using different tools are talking about the same issue.

6.2 Support for concurrent modeling

During the development of AUTOSAR systems, models will be passed between different parties (e.g. from OEM to supplier and back). The fact that it is not possible or wanted for all involved parties to work on the same repository, those parties will work on different instances of the same model, e.g. enrich the models, add new elements, implement the model etc. At some point in time, these models will be merged into one complete model for the entire system: The consistency of the merged model must be ensured and conflicts need to be resolved.

Since AUTOSAR models can be stored as AUTOSAR XML descriptions in several files (which could be modified independently), an AUTOSAR authoring tool must support merging the models stored in these files into a consistent internal data representation. This includes providing a mechanism for manually or rule-based resolving of merge conflicts.

Figure 6-2 shows the concept of handling AUTOSAR models which can be split up over several files. The AUTOSAR authoring tool needs to interpret the contents stored in each single file and needs to merge them into the tool internal data structure. After having finished modifying the model an XML description is created which can again be split up over several files (the granularity of information that can be split up over several files is defined in [ATI0038]).

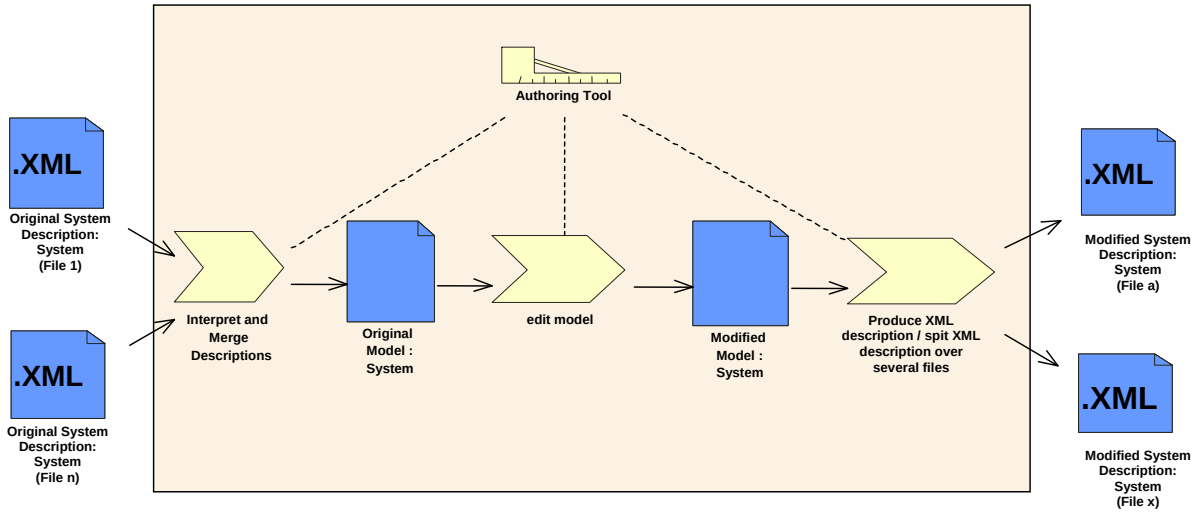

Figure 6-2: Concept of handling AUTOSAR models which can be split up over several files

6.2.1 Detection of differences between models

6.2.1.1 [ATI0043] Authoring tool SHOULD provide a mechanism for showing differences between AUTOSAR models

Initiator:	WP1.2
Date:	22.07.2005
Importance:	Medium
Requirement:	Authoring tool SHOULD provide a mechanism for showing differences between AUTOSAR models.
Description:	Authoring tool SHOULD provide a mechanism for showing differences between AUTOSAR models. These differences could be represented in textual or in graphical ways.
Rationale:	Identification of differences between two versions of an AUTOSAR model
Use Case:	1) OEM wants to find out which parts of the model have been modified by the supplier. 2) The user wants to check if a merge of two models was executed as expected.
Dependencies:	
Conflicts:	
Supporting Material:	
Comment:	This feature could be handled by a separate tool. However, a representation of the differences in the syntax of the used authoring tool is preferred.

6.2.1.2 Definition of differences

The reconciliation of two independently modified models involves several activities.

1. Identification of model elements which are only available in one model.
2. Identification of model elements that are available in both models and are identical in both models. Two model elements are considered identical if
    a. their attributes values are equal,
    b. their short-name references to other elements are equal and
    c. they are composed of identical elements (recursiveness).
3. Identification of model elements that are available in both models and are different in the models.

The activities 2 and 3 require a precise definition of the concept of a “changed” model element.

In the following section we refer to metaclasses that are specializations of metaclass Identifiable as identifiables.
An identifiable is considered as “changed” if any of the following is changed:

- Any of its attributes of simple datatype changed its value.

- The identifiable or any parts (recursive) that are not identifiables themselves are modified. So the granularity of possible detection of changes is the identifiable. The rationale for this is that the Identifiable is considered as having an identity of its own and this will also be the element that has an uuid[5]. The uuid is an attribute defined by the AUTOSAR metamodel class Identifiable and is only used for supporting merging models and calculating differences between them.

The following section explains this definition in more details:

6.2.1.3 Definition of differences - aggregation

Figure 6-3 illustrates the granularity of modification detection: A is a specialization of `Identifiable`. B is aggregated by A (B is a part of A) and B1 is aggregated by B (B1 is part of B). B and B1 are not specializations of `Identifiables`. A is considered changed (indicated by orange color) if the underlying structure or any of the structure's elements that are not specializations of `Identifiable` change.

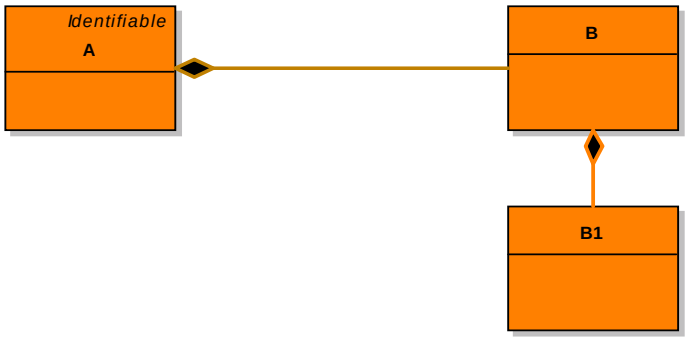

Figure 6-3: Modification detection – aggregated metaclasses that are not identifiable

If an `Identifiable` A aggregates other `Identifiables` (see Figure 6-4), then A is only considered changed, if A aggregates another Instance of C (i.e. elements are added/removed). A is not considered changed, if an aggregated instance of the `Identifiable` C changes.
If the instance of C changes, the change is not propagated to A. C is an `Identifiable` of its own and the change is marked for that instance.

Figure 6-4: Modification detection - aggregated metaclasses that are identifiable

[5] UUID: Universal Unique IDentifier

6.2.1.4 Definition of differences - references

If an `Identifiable` A has references to other `Identifiables` (see Figure 6-5), A is considered changed, if a new reference is added/deleted or the string that internally represents the reference is changed. A is not considered changed, if the referenced element changes. Please note that the AUTOSAR references are based on the `shortName` referencing mechanism as explained in the Model Persistence Rules for XML document [12] and therefore changing the `shortName` of an instance of C implies updating all references to the instance. In this special case A is considered changed if the `shortName` of C is updated (the string representing the reference needed to be updated too).

Figure 6-5: Modification detection - references

6.2.1.5 Algorithm for comparison of model elements

The aforementioned definition highly depends on identification of model elements in the compared models. This identification can only be performed unambiguously if they have a universal unique id (`uuid`). This `uuid` is NOT allowed to be changed during the lifetime of a model element.

The concrete algorithm of comparison is left open to the implementation, but it is required that it detects all of the changes described above. Possible implementation could be a comparison on a per-attribute basis or the calculation and comparison of a signature value (e.g. calculated by the MD5 algorithm [31]) for any Identifiable. Since somebody might change the content of an XML description with an XML editor without updating the signature values, an authoring tool should not rely on the correctness of the checksums that are transmitted with an XML description. However, if a user can ensure that the checksum values fit to the content then they can be used in order to increase the performance of the comparison.

Please see section 6.2.2.8 for more information on the `uuids` and checksums.

6.2.1.6 [ATI0024] Authoring tool SHALL support unique identification of model elements

Initiator:	MMT
Date:	14.12.04, updated 22.07.2005
Importance:	High
Requirement:	Authoring tool SHALL support unique identification of model elements.
Description:	The AUTOSAR authoring tool SHALL continue interpreting an AUTOSAR XML description even if `uuids` are missing. An AUTOSAR authoring tool SHALL not change the `uuid` while processing an AUTOSAR model unless it is explicitly intended by the user. An AUTOSAR authoring tool SHOULD create XML descriptions which contains `uuids` for each `Identifiable`.
Rationale:	Ability to trace model elements independently of content modifications.
Use Case:	Export of data from a database with unique `uuids`, change data in external tools and then re-import into the database. During the re-import the `uuids` aid the unambiguous assignment of model elements to data in the database.
Dependencies:	ATI0042, ATI0035, ATI0007
Conflicts:	
Supporting Material:	
Comment:	

6.2.1.7 Examples of differences between models (non normative)

The following figures show an original and a changed model. The model elements that are marked in green indicate which element is considered to be changed:

Figure 6-6 shows the original model. The `SenderReceiverInterface` “interface” and the `IntegerType` “velocityType” are defined within the `ARPackage` “OEM”. The `SenderReceiverInterface` “interface” has a `DataElementPrototype` “velocity” which is of type “velocityType”. The range of the `IntegerType` “velocityType” contains all numbers between 0 and 10000.

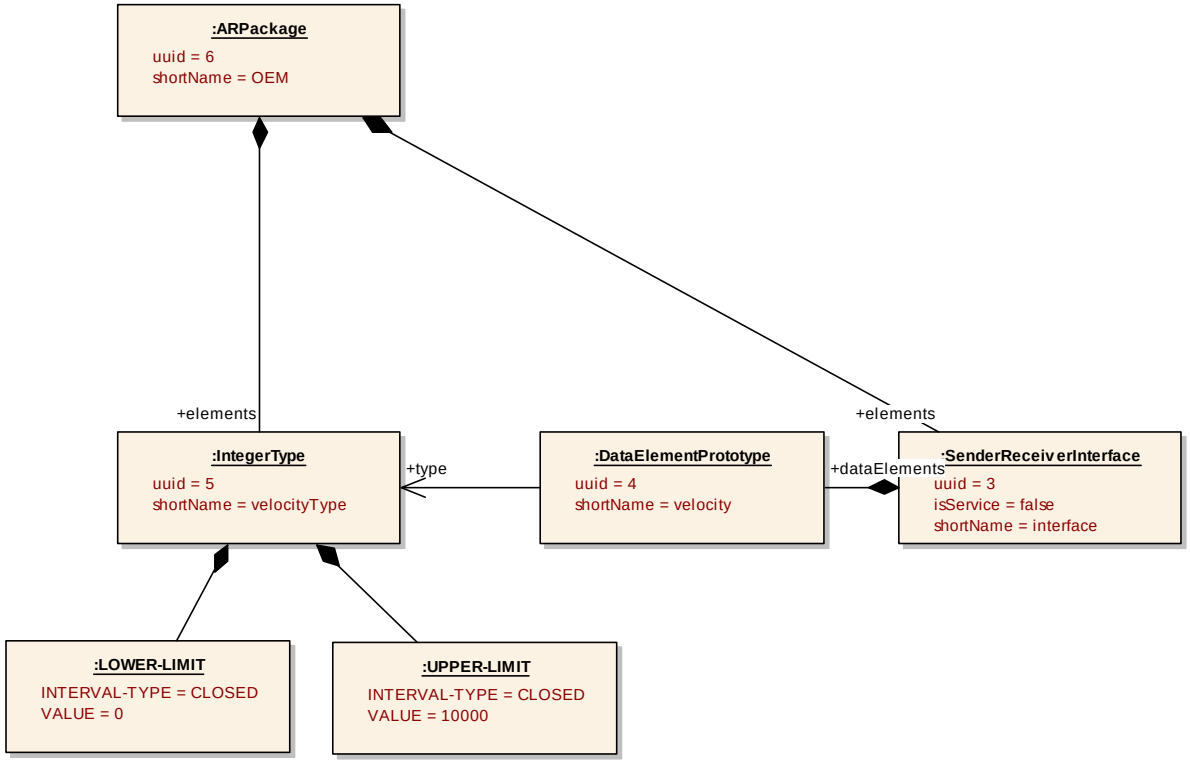

Figure 6-6: Original model

The model which is described in Figure 6-7 contains two additional model elements: `DataElementPrototype` “buttonPressed” and the `BooleanType` “buttonPressedType”. These elements are marked as changed because they were added. The `ARPackage` “OEM” and the `SenderReceiverInterface` “interface” are marked as changed because aggregations to the new model elements have been added.

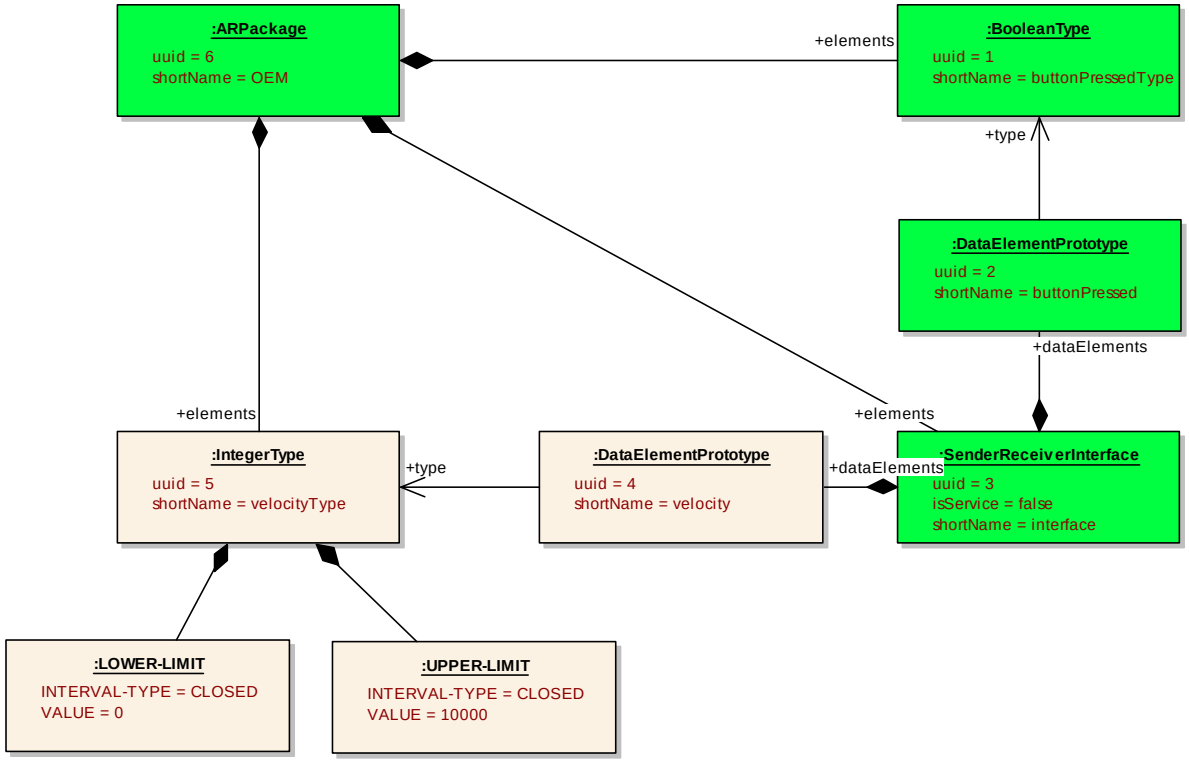

Figure 6-7: Modified model - DateElementPrototype and BooleanType added

In Figure 6-8 the range of `IntegerType` “velocityType” was extended. The change in the object :`UPPER-LIMIT` is propagated to the `Identifiable IntegerType`.

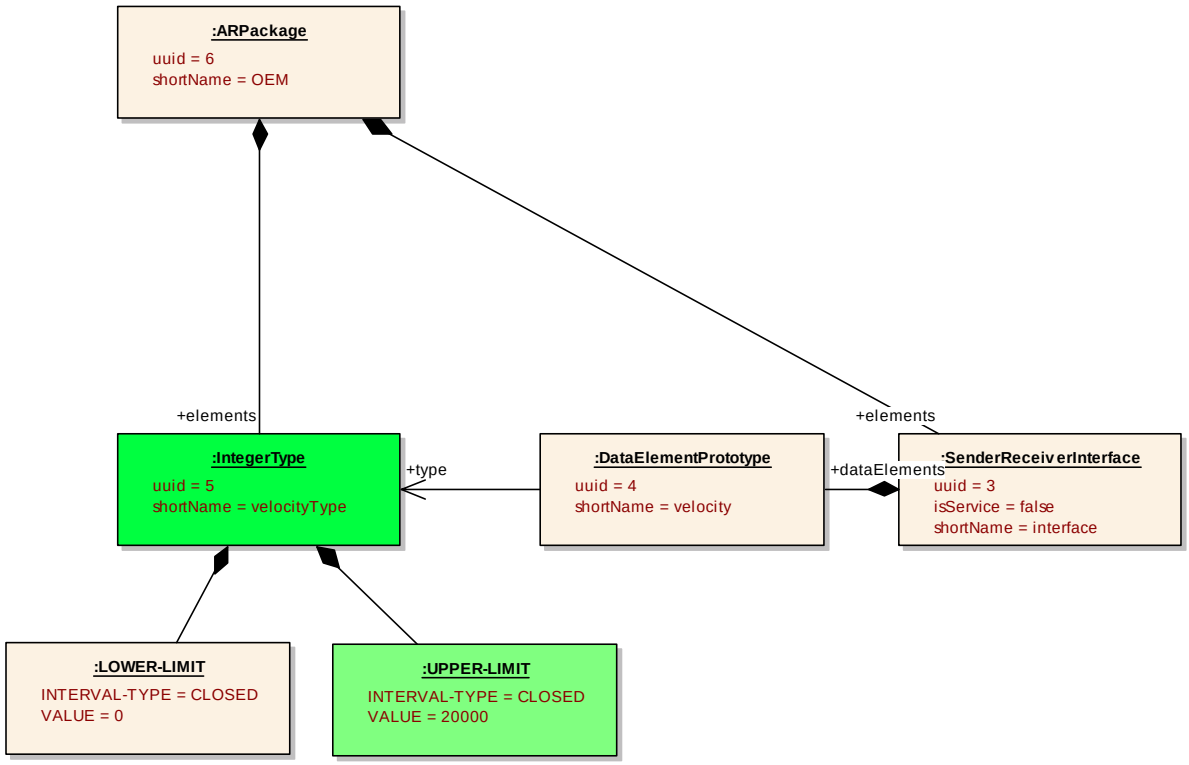

Figure 6-8: Modified model - range of IntegerType changed

The shortName of the IntegerType in Figure 6-9 was changed from “velocityType” to “speedType”. Changing the shortName of an object implies a change of the references to this object. (The attribute shortName has a special semantics within AUTOSAR. Within the XML descriptions the shortName is used for identification of the target of a reference. See [12] for more details on the referencing mechanism within the AUTOSAR XML descriptions)

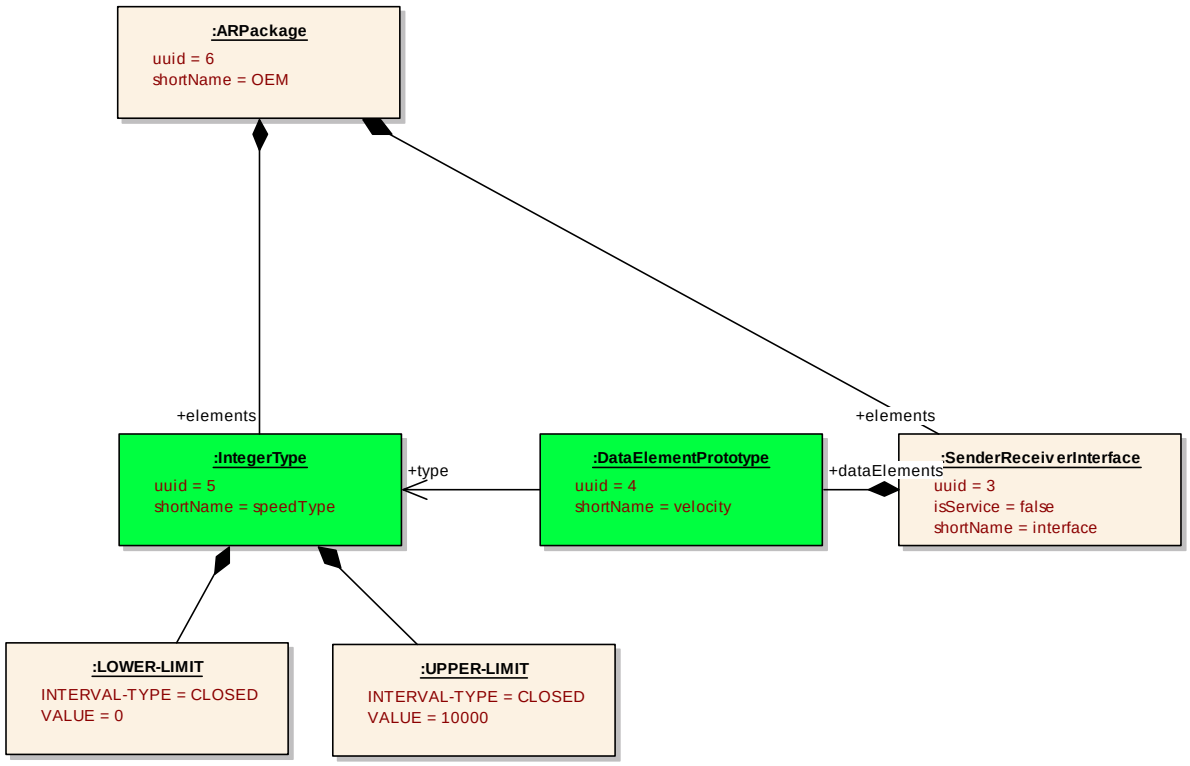

Figure 6-9: Modified model - shortName of IntegerType changed.

If a model element is moved to a different ARPackage then only the source and the target ARPackage are changed (Their list of elements changes). This is described in Figure 6-10. Please note that all references that point to moved elements need to be changed as well. This is required because AUTOSAR uses absolute short name paths for referencing elements. If references would have been based on uuids then the references would not change.

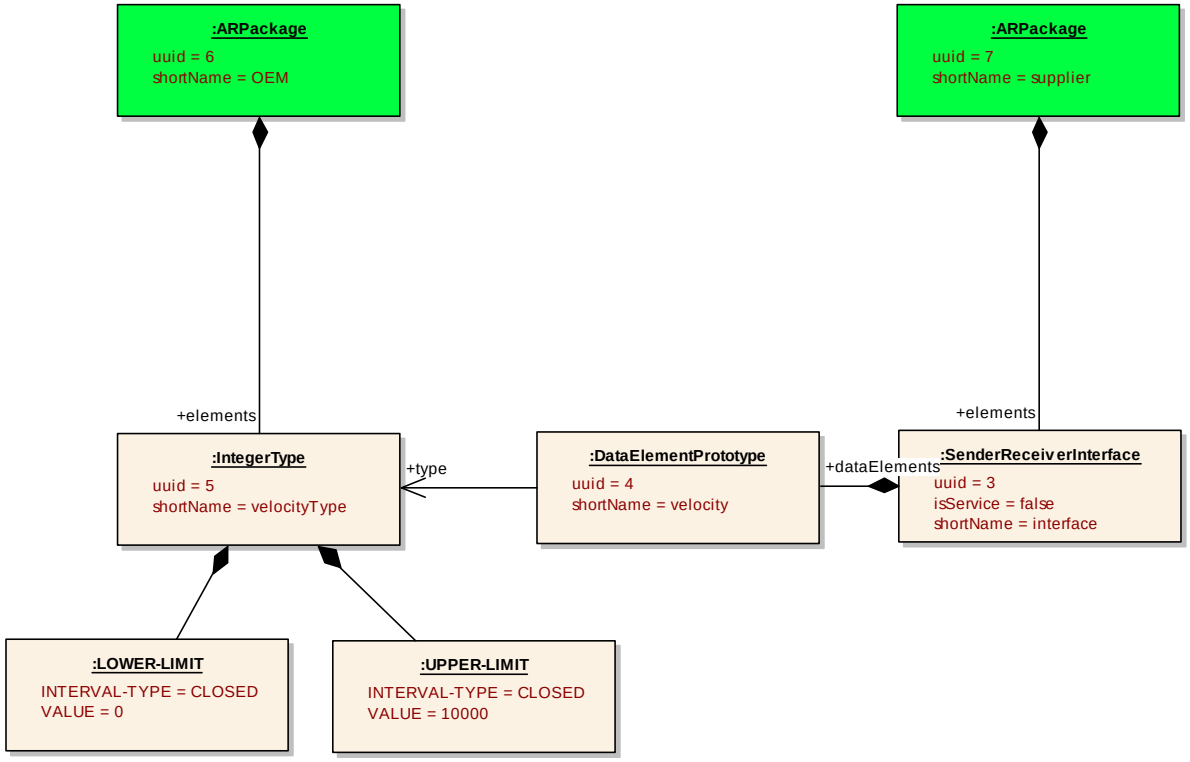

Figure 6-10: Modified model - SenderReceiverInterface moved to different ARPackage

6.2.2 Merging models

The following sections describe requirements on interpreting AUTOSAR models that have been split up into several submodels, each stored in an individual file. In order to create an overall model the content described in the submodels need to be merged. The following algorithm describes a 2-way merge. Due to missing experience with handling complex AUTOSAR models in development processes, strategies for resolving merge conflicts are left open to implementations of tools.
In order to reduce manual interaction for resolving merge conflicts, tools may take additional information into consideration. This information can e.g. be derived out of an earlier version of the overall model (origin model, 3-way merge) or out information stored in metadata for data exchange (see section 7.3).

6.2.2.1 [ATI0042] Authoring tool SHALL support for merging of AUTOSAR models

Initiator:	WP1.2
Date:	22.07.2005
Importance:	High
Requirement:	Authoring tool SHALL support for merging of AUTOSAR models.
Description:	AUTOSAR authoring tools SHALL support for merging of AUTOSAR models that have been split up and stored in multiple submodels. The minimum granularity of an AUTOSAR model is explicitly modeled in the AUTOSAR metamodel: If an aggregation is marked as `<<splitable>>`, then the aggregated elements MAY be described in different files. If the aggregation is not marked as `<<splitable>>`, then the aggregated content SHALL be stored in the same file as the aggregating element. Please note that the merge and split functionality is not required to be integrated into the authoring Tool. It may be implemented as a standalone tool that can be used with the authoring tool.
Rationale:	Allow for splitting up AUTOSAR models over several sub models which can be stored, versioned, and modified independently.
Use Case:	When storing an AUTOSAR model as an AUTOSAR XML description in several files, each file needs to be valid with respect to the AUTOSAR XML schema: i.e. each file contains a full path beginning from the root element AUTOSAR. Let's assume that an AUTOSAR model defines an `Atomic-SoftwareComponentType` *A* and a `SenderReceiver-Interface` *S* within the same `ARPackage` *pkg*. The XML description of *A* can be located in another file than the XML description of *S*. Each file would contain the description of the `ARPackage` *pkg*. While interpreting the contents of the two files the AUTOSAR authoring tool must make sure that only one instance of *pkg* is created in the internal representation of the AUTOSAR model.
Dependencies:	This requirement is refined by ATI0044, ATI0043, ATI0040 and ATI0024.
Conflicts:	See [13] for detailed description on how to mark an aggregation as `<<splitable>>`.
Supporting Material:	
Comment:	

6.2.2.2 Algorithm for merging models

The algorithm for merging models can be decomposed into the following steps. Please note that identification of model elements by `uuid` or absolute `shortName` path doesn't change the following algorithm. However, if identification via absolute

`shortName` paths is used and the shortName was changed or elements were moved, special attention is required:

1. The merge algorithm is based on matching `uuids` of `Identifiable` model elements. If a `uuid` is not available, the absolute short name path of model elements can be used. Since not all elements are `Identifiable`, the elements that are not `Identifiable` are grouped with their owning `Identifiable`. This can be achieved by the following strategy:
   Separate the models into disjunctive groups of model elements. All model elements shall be covered. Each group contains exactly one root node or `Identifiable`. Please note that these groups correspond to the groups that are defined for calculating differences between models in section 6.2.1.
   For each root node and `Identifiable` create a group that contains a root node or `Identifiable` and additional elements that are calculated by the following algorithm:
   a. Recursively add aggregated elements that are not `Identifiable`
   b. Add attributes, aggregations and references of all elements in the group
2. Theses groups are merged individually. The order of groups to be merged is defined by starting at the root node and recursively navigating to the contained groups (groups that are aggregated by elements within the current group).

The algorithm described above is illustrated in Figure 6-11.

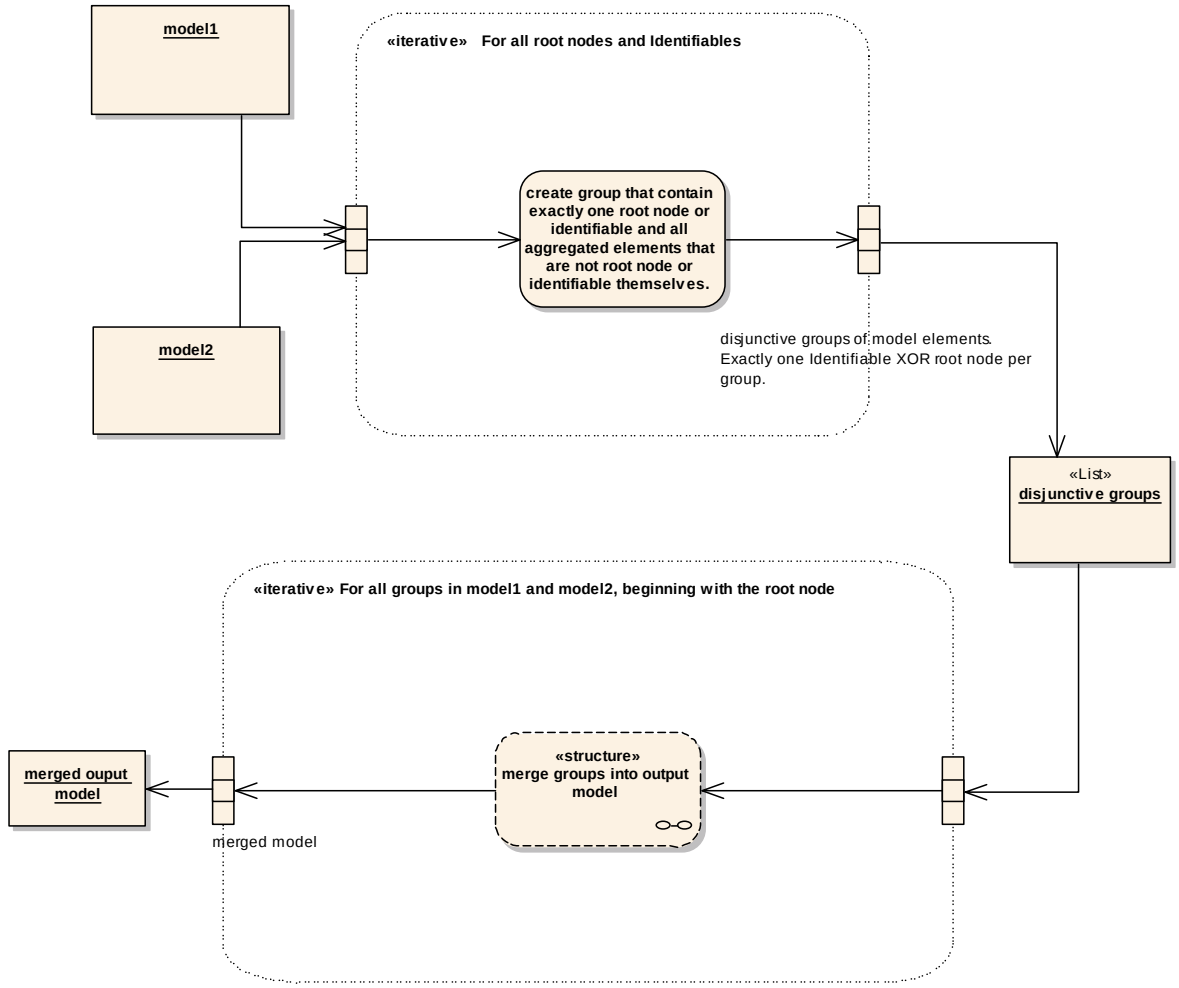

Figure 6-11: Algorithm for merging models

The algorithm for merging the disjunctive groups in the model is decomposed into the following steps:

1. If a group is contained in the input models (model1 or model2) but not in the output model then the group is copied into the output model.
2. If a group is already contained in the output model, then the input group and the output group are compared based on all attributes, elements and relations (aggregations and references) between elements in the groups. At this step relations to elements outside of the group are not considered. If a difference is detected, then the conflict needs to be resolved.
3. Then all relations (aggregations and references) that navigate out of the group are merged.

This algorithm described above is illustrated in Figure 6-12.

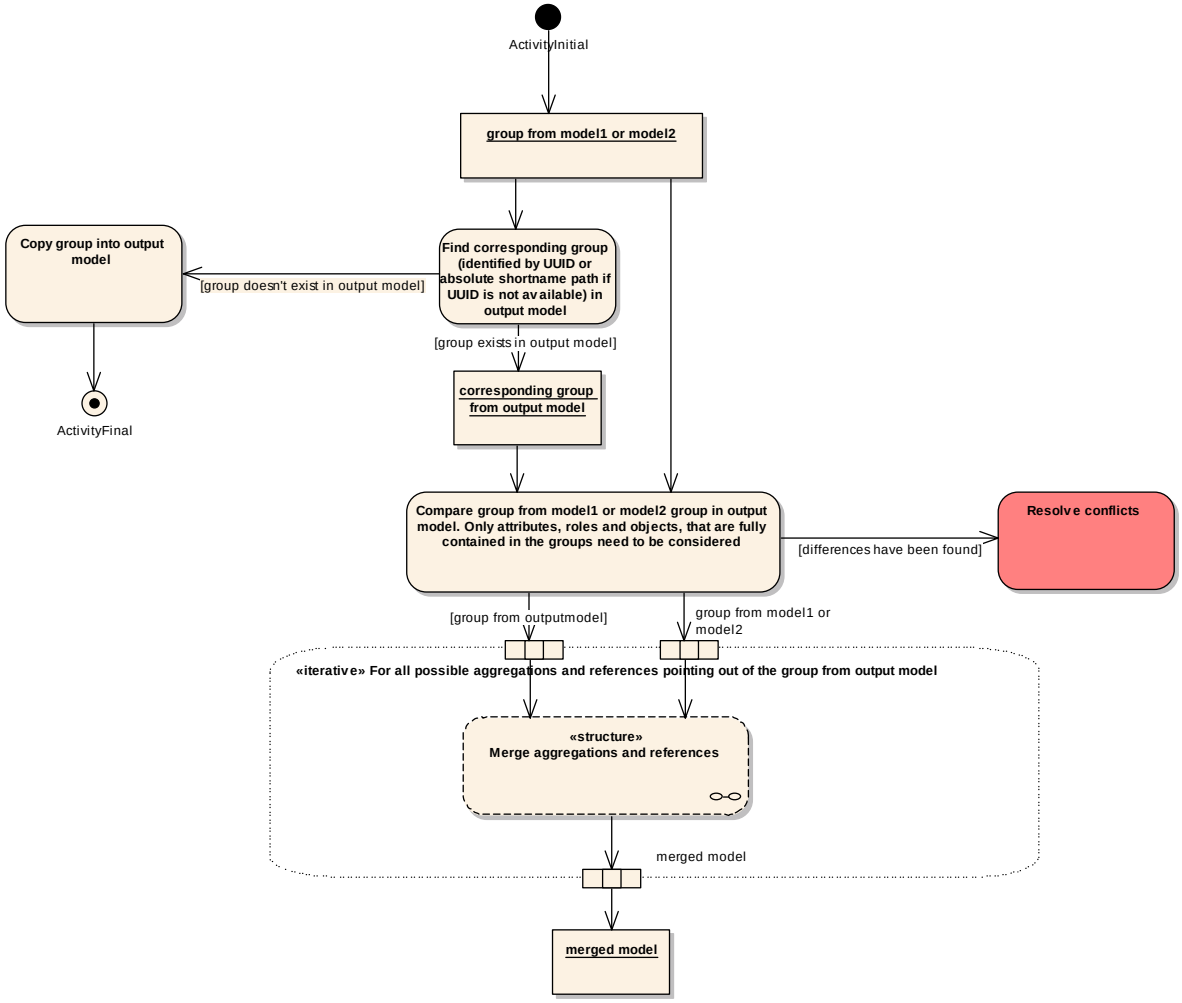

Figure 6-12: Algorithm for merging disjunctive groups in the model

AUTOSAR distinguishes between elements in the model that may be split up over several models, and elements that need to be defined self-contained and therefore shall be described in exactly one model. For each attribute, aggregation and reference the metamodel shall describe if it is `<<splitable>>`. The following section describes the semantics of `<<splitable>>`:

- If the metamodel marks an **aggregation** as `<<splitable>>` then the aggregated elements may be described in different models. If the aggregation is not `<<splitable>>`, then all aggregated element(s) shall be described in the same model as the aggregating element.
- If the metamodel marks a **reference** as `<<splitable>>`, then the description of the links may be split up over several models. If a reference is not `<<splitable>>`, then the description of the links shall be described completely within one model. Please note:
  1. References that are not `<<splitable>>` do not require that the referenced elements are contained in the same model.
  2. If more than one model describes the reference, then both descriptions shall be identical.

- If an attribute is <<splitable>>, then one model may contain the value of an attribute while another model doesn't. If it is not defined as <<splitable>>, then the value of an attribute shall be described completely in one model. If the values of the attribute are different in the input models, than a merge conflict is detected.

The algorithm for aggregations is illustrated in Figure 6-13

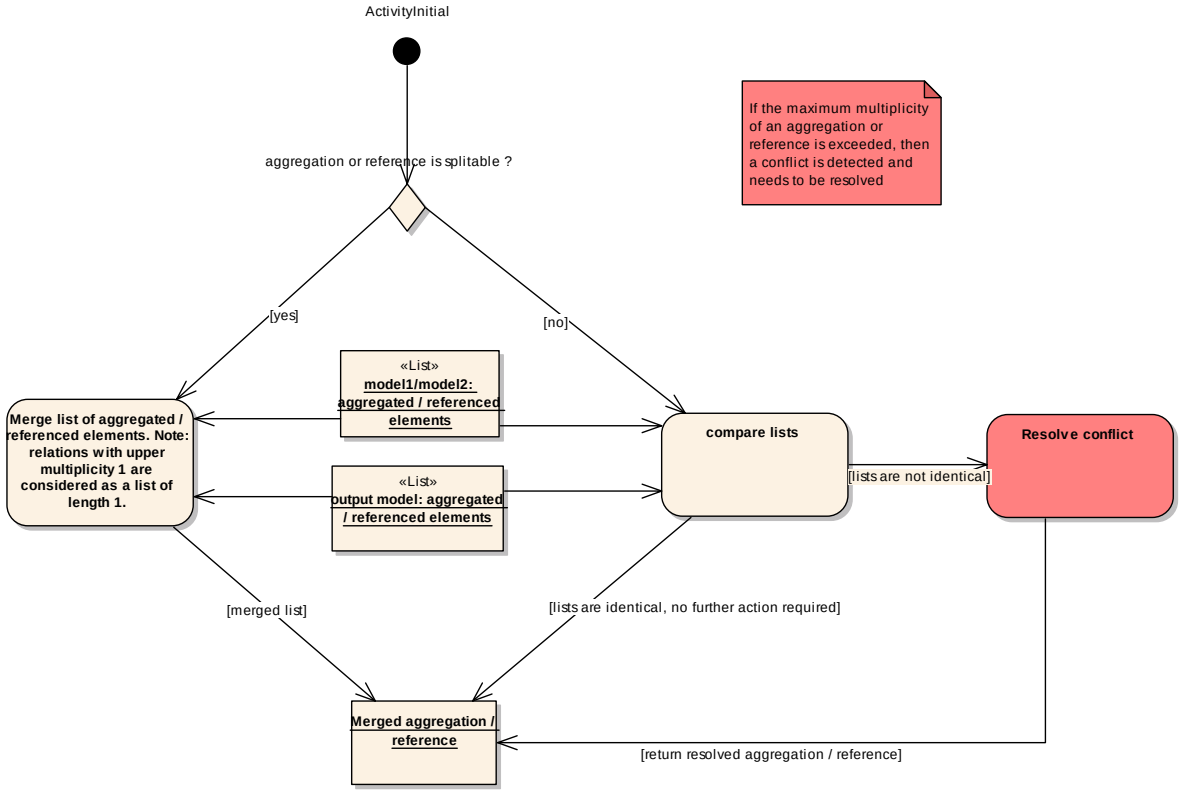

Figure 6-13: Algorithm for merging relations (aggregations or references)

6.2.2.3 Handling Conflicts

In case of conflicts, these need to be resolved with human interaction, since semantic knowledge is needed in most cases for the decisions. Merging tools might infer suggestions from additional information. If the submodels are derived from a common origin, the 3-way merge algorithms can be applied. Additionally metadata for data exchange could be used. E.g.: A timestamp could be used to indicate the time of modification of an element (the initial creation of an instance is considered as a modification, too), where modification means any change as described in the section above. Timestamp is an optional field, since its existence is not critical for the process.

6.2.2.4 Handling merge conflicts: optimistic approach

In the case of an optimistic approach for resolving conflicts, work is allowed on copies of a model and only when the models are synchronized/integrated potential conflicts are be resolved. Whenever a merge conflict is detected (i.e. two AUTOSAR models contain model elements which have the same `uuid` but a different description) the conflict needs to be resolved. This could be implemented by the tool by interactively asking the user to choose the right model element.
Additional metadata on the model elements can help finding out the correct model element. E.g. if a timestamp is assigned to a model element it can be decided which information is newer than the other. Another approach for resolving or even avoiding merge conflicts is described in section 6.2.2.6.

6.2.2.5 [ATI0044] Authoring tool SHOULD provide a mechanism for resolving merging conflicts

Initiator:	WP1.2
Date:	22.07.2005
Importance:	Medium
Requirement:	Authoring tool SHOULD provide a mechanism for resolving merging conflicts
Description:	AUTOSAR authoring tools SHOULD provide a mechanism for resolving merging conflicts. This mechanism could be e.g. implemented by an interactive user interface which allows for choosing from conflicting elements or by using further metadata such as the timestamp (e.g. the latest version of an element shall be used), etc.
Rationale:	Allow for integration of models which have been modified by different parties.
Use Case:	When integrating models which have been modified and extended by different parties merge conflicts are likely to occur.
Dependencies:	
Conflicts:	
Supporting Material:	Error code on merge conflicts: ARCont00004
Comment:	

6.2.2.6 Handling merge conflicts: access control approach

The probability of merge conflicts can be reduced by a well defined work-flow. E.g. in a top down development process [ATUC_005] an OEM could decompose a system down to a given granularity. The refinement of the decomposition could be done by different suppliers. If each supplier modifies a disjunctive part of the model no merge conflicts will occur when merging the results into the OEMs model of the full system.
In order to support this strategy the supplier needs to know which parts of a model are allowed to be changed and which parts are not allowed to be changed.

These access rights SHOULD be exchanged together with metadata for supporting data exchange. See requirement on the metadata for data exchange ATI0036, ATI0038.
If access rights are defined an AUTOSAR authoring tool SHOULD prohibit the user from modifying model elements that are marked as read-only.

6.2.2.7 [ATI0040] Authoring tool SHOULD prohibit the user from modifying model elements that are marked read-only

Initiator:	WP1.2
Date:	21.07.2005
Importance:	Medium
Requirement:	Authoring tool SHOULD prohibit the user from modifying model elements that are marked read-only.
Description:	A tool SHOULD only allow the user to modify, create or delete model elements which he is allowed to change. The information if a model element is allowed to be modified should be represented to the user.
Rationale:	Prohibit a user from modifying model elements he is not allowed to change.
Use Case:	OEM wants to send information to tier-1 supplier. OEM wants to lock certain model elements so that the tier-1 is not allowed to change them.
Dependencies:	Metadata for data exchange could define the access-rights on model elements, see ATI0038. The access-policies can support the merging of models, see ATI0042.
Conflicts:	
Supporting Material:	
Comment:	

6.2.2.8 Metamodel attributes for enabling merging of AUTOSAR models

This section describes attributes of the metaclass `Identifiable` which support merging AUTOSAR models: `uuid`, checksum, timestamp.

uuid

Universal Unique Identifier. The purpose of this attribute is to provide a universal unique identifier for an instance of a metaclass. The values of this attribute should be universal unique strings prefixed by the type of identifier. For example, to include a DCE[6] `uuid` as defined by The Open Group[7], the `uuid` would be preceded by "DCE:". The values of this attribute may be used to support merging of different AUTOSAR models.

[6] http://www.opengroup.org/dce/
[7] http://www.opengroup.org/

The DCE standard is widely used, including by Microsoft for COM (GUIDs) and by many companies for DCE, which is based on CORBA. The method for generating these 128-bit IDs is published in the standard and the effectiveness and uniqueness of the IDs is not in practice disputed.

If the id namespace is omitted, DCE is assumed.

For example "DCE:2fac1234-31f8-11b4-a222-08002b34c003" is equivalent to “2fac1234-31f8-11b4-a222-08002b34c003".
The `uuid` is intended to support merging of models. They are not intended to be used for referencing in XML descriptions. The AUTOSAR referencing mechanism is explained in the Model Persistence Rules for XML [12].

Checksum

The checksum is calculated out of all attributes, references and aggregations of an `Identifiable` model element. The detailed scope of this calculation is described above (6.2.1.1). The checksum can be used for comparing different model elements.

Timestamp

The timestamp describes the time of the creation or modification of an `identifiable.`

6.2.2.9 Example on merging models

Figure 6-15, Figure 6-16 and Figure 6-17 describe two example models and the result after merging them. The underlying metamodel for those examples is described in Figure 6-14. Splitable aggregations are marked by the stereotype <<splitable>>.

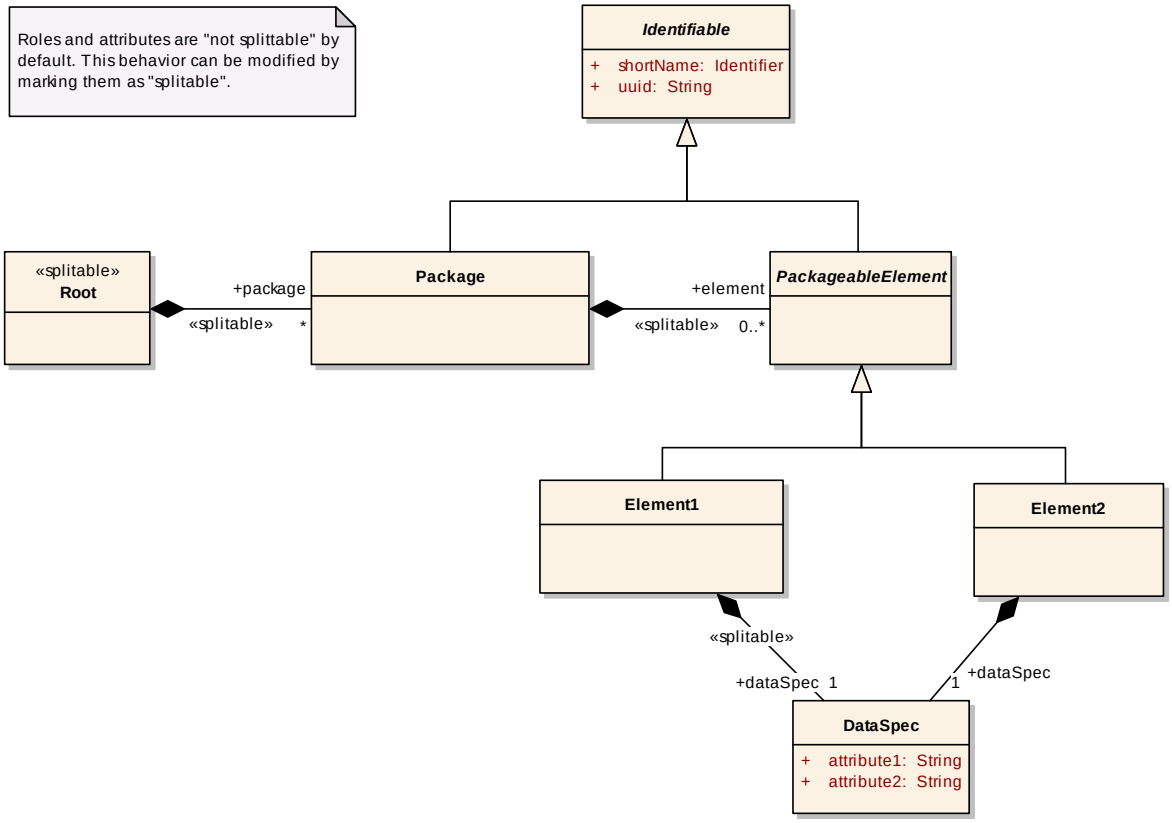

Figure 6-14: Metamodel of the models described in this section

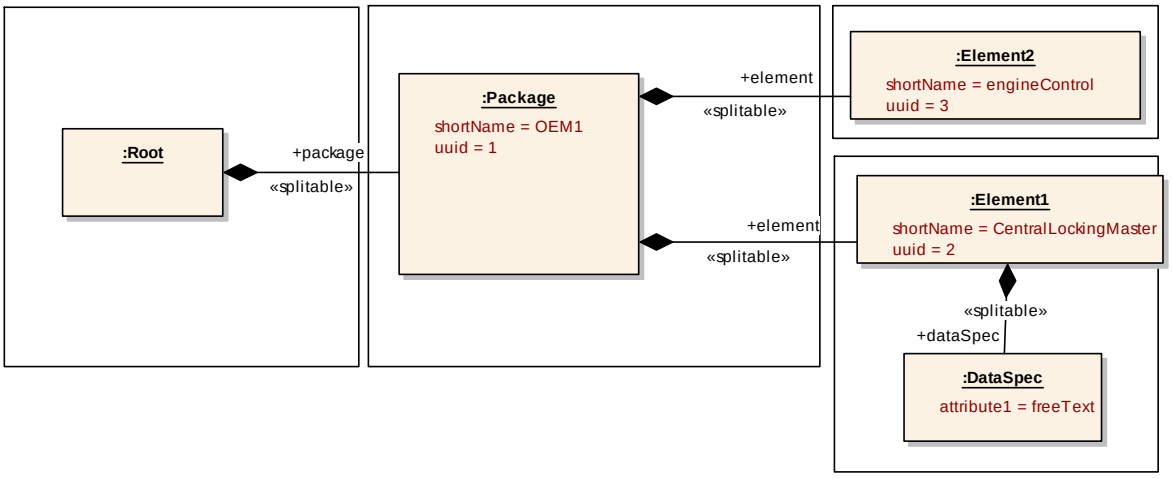

Figure 6-15: Separation of the model1 into disjunctive groups

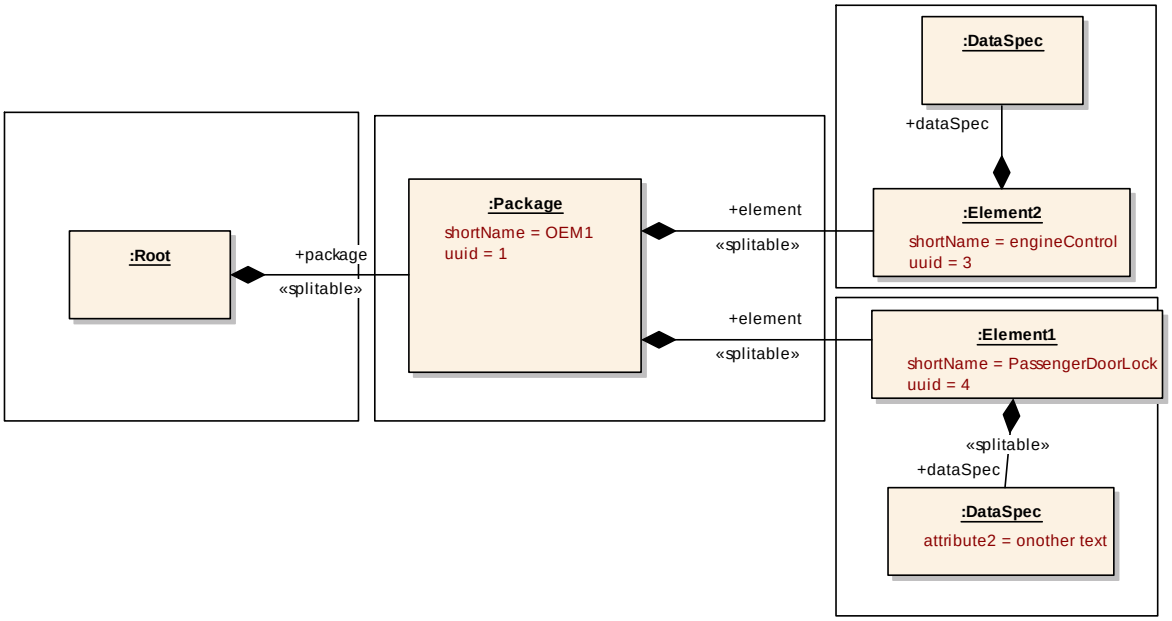

Figure 6-16: Separation of the slave model into disjunctive groups

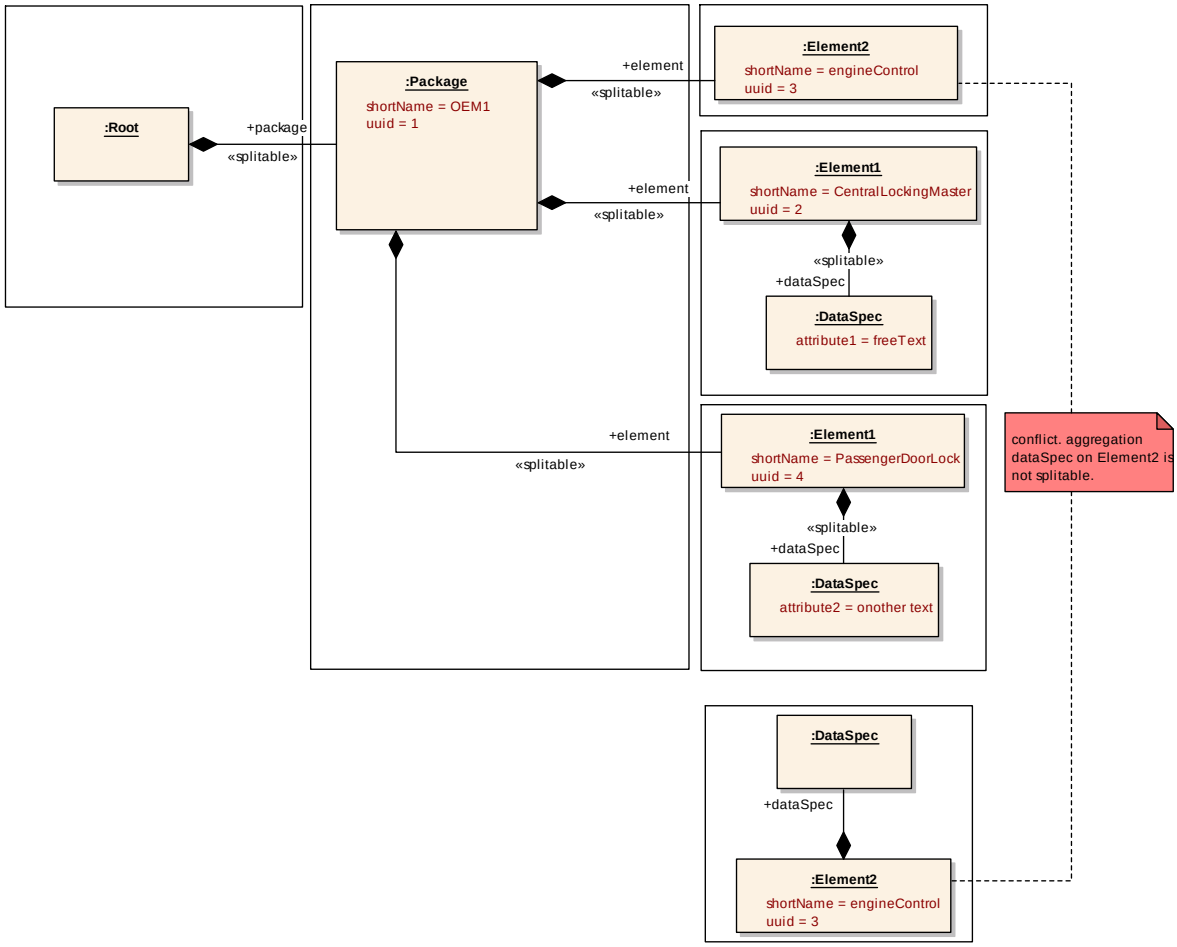

Figure 6-17: Merged model. The conflict between the elements with uuid=3 needs to be resolved

6.3 Shipment of AUTOSAR models and related artifacts

As described in the sections above, the minimum required data exchange mechanism of AUTOSAR authoring tools is the interpretation and creation of AUTOSAR XML descriptions which can be split up over several files.
The separation over several files imposes the risk that a file gets lost during the data exchange or a file was accidentally transmitted incompletely. Listing all files that belong to a shipment as metadata for data exchange would allow the receiver to check if all files have been received.

The metadata for data exchange should contain information that supports the exchange of AUTOSAR models and related artifacts:

- List of physical artifacts (e.g. files) that belong to the shipment in order to detect missing or superfluous artifacts.
- Checksum for each file in order to allow for detection of modification.
- Access rights on model elements which allow for transmitting information which model elements are allowed to be changed by the receiver.
- Explicit annotation of deleted, moved and added model elements in order to support merging.

6.3.1.1 [ATI0036] Authoring tool SHOULD be able to interpret and create metadata for data exchange

Initiator:	WP1.2
Date:	20.07.2005
Importance:	High
Requirement:	AUTOSAR authoring tool SHOULD be able to interpret and create a metadata for data exchange
Description:	AUTOSAR authoring tool SHOULD be able to interpret and create metadata for data exchange that contains information on the exchange model and related artifacts.
Rationale:	Support the exchange of additional information about exchanged XML-descriptions and their usage. e.g. access policies, checksum, list of files that contain the XML description.
Use Case:	OEM wants to send information to tier-1 supplier. OEM wants to lock certain model elements so that the tier-1 is not allowed to change them. The tier-1 needs to find out if all information has been transmitted correctly. The metadata for data exchange could additionally list further files that are not specified by AUTOSAR, e.g. specific model files of behavior modeling tools. The following workflow describes how an AUTOSAR model could be handled, if additional meta information for data exchange is available. *Workflow at the OEM's site* In this case an OEM gathers a collection of model elements to be submitted to a supplier. The metadata for data exchange is stored into a file when the collection is complete. This file could be called a manifest or catalog. The OEM could create a specific folder in the model repository and names it after the characteristics of the information exchange. The catalog file is checked into this folder. Now, all versions of model files that are part of the exchange are shared into the folder. As a result, the OEM gets a comprehensive description of the model interchange without touching the model files themselves. The catalog file could contain information about which parts of the AUTOSAR model are allowed to be changed. Now the OEM repeats the same activity for model interchange with a different supplier who is responsible for refinement of another part of the AUTOSAR model and therefore the access rights for the second supplier are different. Let's assume that the collection of model files submitted to the two suppliers is identical with respect to the version of the models. The OEM is now capable of recognizing that model files submitted to different suppliers are exactly identical although the access rights might be different. *Workflow at the supplier's site*

	The supplier receives the catalog file and feeds it to the AUTOSAR authoring tool in use. The latter takes the catalog file as the basis for the actual import of AUTOSAR models. Access rights as well as other meta-information would most likely be taken over by the AUTOSAR authoring tool. The supplier now implements the behavior of received `AtomicSoftwareComponentTypes`. The implementation of the behavior has an impact on the `Implementation` description of `AtomicSoftwareComponentTypes`. Therefore, the version of the `Implementation` must be increased. The supplier then exports the work results to the common AUTOSAR model format. In addition, the AUTOSAR authoring tool would create a new catalog file that indicates which file contains the extended version of the `Implementation`. Now the supplier submits catalog file in combination with model files back to the OEM. The latter takes the received catalog file and checks it for differences with the submitted file. Of course this only allows for checking of differences on file-level. However the OEM does only have to check the parts of the AUTOSAR model that is stored in changed files. See also [ATUC_008]
Dependencies:	More requirements on the metadata for data exchange are described in ATI0037, ATI0038, ATI0039. Per shipment all metadata for data exchange must be stored in a single file. Metadata for data exchange is not splitable. Otherwise metadata for metadata would be required.
Conflicts:	
Supporting Material:	
Comment:	

6.4 Interoperability with specialized tools

Within a tool-chain tools that modify AUTOSAR models that have been created in earlier phases should be able to interpret and understand all information which has been defined before. Formally described this means:

Let's assume that

- A and B are AUTOSAR tools which are used in a tool-chain($A, B \in AUTOSARToolsInChain$),
- the set of models which can be created by tool A is described by $L_{create}(A)$
- and the set of models which can be interpreted by tool B is described by $L_{interpret}(B)$.

With these definitions we can formally describe the observation by the following expression which should evaluate to TRUE:

$$\forall A, B \in AUTOSARToolsInChain:$$
$$L_{create}(A) \subseteq L_{interpret}(B)$$

Having a deeper look at the description before, the following questions come up:

- How can the partners be sure, that the tools can exchange the AUTOSAR model? In other words how do I validate, that $L_{create}(A) \subseteq L_{interpret}(B)$? (see section 6.4.1)
- There are many specialized tools which are very well established in current development processes. These tools are not able to interpret or create the full set of information which is described in the AUTOSAR metamodel. How can be ensured that those tools can be used in an AUTOSAR development process? In other words: What happens if $L_{create}(A) \not\subset L_{interpret}(B)$? (see section 6.4.2)

6.4.1 Requirements for predictable tool interoperability

It should be predictable that two tools are able to exchange their models. In an early phase in the development process it should be possible to find out if two tools are able to exchange models which will be created in later phases. This could be a criterion for choosing tools that are used in an AUTOSAR development process. As shown in the section above an AUTOSAR model can be exchanged from one tool to another, if $L_{create}(A) \subseteq L_{interpret}(B)$.

The formal validation of this expression would require all AUTOSAR tools to formally describe the supported subset of the AUTOSAR metamodel. It needs to be checked if the structure and semantics of all metaclasses that are supported by the tool *A* is also supported by the tool *B.*

This approach would check if two tools can exchange information if all supported features are used. Usually not all features are always used and therefore the results of this formal compatibility check would become questionable.

A more pragmatic approach would be to compare features which describe the supported functionalities in a more abstract way. A starting point for these functionalities are activities defined in the “AUTOSAR methodology” document [3]. AUTOSAR should precisely define the required inputs and provided outputs of those activities. See requirement [ATI0051].

A more elaborate compatibility test can then be performed. This test can be based on a number of test-models that are created in early phases of the development of an AUTOSAR system.

In practice the information if two tools can exchange models is not the only criterion for choosing a special tool (consider criteria such as usability, know how with existing tools, etc). It is not very likely that e.g. a supplier switches to another tool because it is not completely compatible with the tool of an OEM. Instead he would approach the tool-vendor and ask for implementation of the missing features.

In order to get a set of AUTOSAR authoring tools which have comparable functionalities and are able to exchange their models, a set of compliance classes can help. (See chapter 8.5.1 for more information on those compliance classes.)

6.4.1.1 [ATI0016] Documentation of authoring tool SHOULD describe supported features

Initiator:	WP1.2
Date:	20.04.05, updated 18.07.2007
Importance:	Medium
Requirement:	Documentation of authoring tools SHOULD describe supported features
Description:	The documentation of AUTOSAR authoring tools SHOULD describe the supported features. The description of the features SHOULD be based on the activities defined in the AUTOSAR methodology document [3]
Rationale:	When choosing specialized Authoring tools that do not support the full set information described in the AUTOSAR metamodel for use in an AUTOSAR tool chain, it SHALL be predictable if AUTOSAR models can be exchanged between different tools.
Use Case:	
Dependencies:	
Conflicts:	
Supporting Material:	
Comment:	In current practice the compatibility of tools is checked by exchanging example models.

6.4.2 Requirements on the integration of specialized tools

The sections 6.1, 6.2 and 6.3 described some general concepts and requirements on tool interoperability. The following list describes some observations on handling of AUTOSAR models with special focus on tools that do not cover all information that is represented by the AUTOSAR metamodel:

- AUTOSAR models are exchanged via AUTOSAR XML descriptions which can be split up over several files. While interpreting an AUTOSAR XML description an authoring tool must merge the contents of the different files into the internal data-structure of the tool (see chapter 6.1.3 Content level).
- While merging AUTOSAR models conflicts can occur. Often user-interaction is required for the resolving of merge conflicts. A tool can usually only provide interaction mechanisms for model elements that are internally represented.
- While merging AUTOSAR models which were created by different parties (e.g.: integration of several independently developed models to an overall model) violations of semantic constraints can show up. Each AUTOSAR model could be valid in itself. However, violations can show up if all sub models are merged and checked together. These violations can only be solved by tools that are able to modify the violating information.
- A tool can very well perform its intended function even if some errors exist in the model. Only the information that is required for performing the intended function needs to be valid.

These observations lead to the following tool architecture.

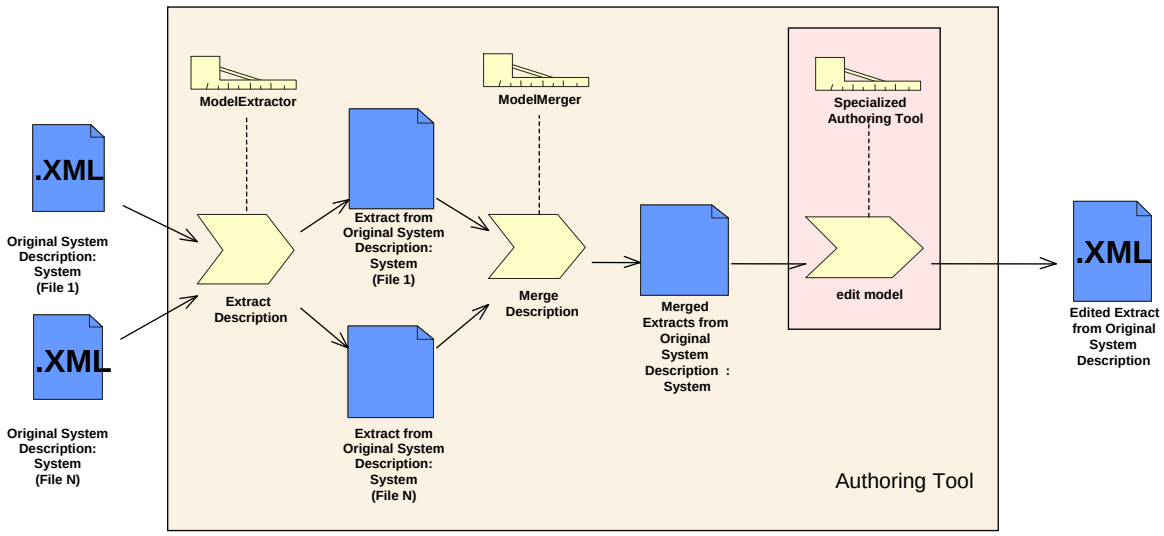

Figure 6-18: Integration of specialized authoring tools

Figure 6-18 describes an architecture which enables the integration of specialized AUTOSAR authoring tools into the AUTOSAR methodology:

1. First the tool extracts the information that is supported by the internal data-structure from the files that contain the original XML description. Information that is not supported is ignored. This information doesn't get lost, since it is still available in the original XML descriptions.
2. In a second step the extracted descriptions are merged into the overall internal representation of the tool. The tool can implement an intelligent merger since all information that needs to be merged is represented by means of the tool internal data structure.
3. The tool can provide mechanisms to the user that allows adding and changing the model content.
4. The tool can create an AUTOSAR XML description out of the internal data structure. The fragmentation of the resulting XML description can be based on the original fragmentation or another fragmentation can be chosen by the tool.
5. If the authoring tool was not able to interpret the full input model then it only can export the supported model elements to an XML description. However information doesn't get lost since a merge of the original model with the exported model will result in an updated overall model. (This assumes that the merger overwrites older model elements with newer model elements).

The architecture ensures that the specialized tool does not need to care about model-elements that are not supported by the internal data structure.

Additionally, a specialized tool might detect errors in the input model that it might not be able to fix. In this case the tool should export information about the errors. See requirements [ATI0049] and [ATI0050]

6.5 Migration between different versions of the metamodel

The implementation of the “Data Format Level”, “Content level”, “Semantic level” and all higher levels highly depend on the metamodel.
Changes in the metamodel can be classified in two different classes. The criteria for these classes are based on the effort that is usually required for adaptation of tools.

Minor changes

Extensions in the metamodel that don’t influence relations and constraints between existing classes in the metamodel are considered as minor changes. Existing models are still valid when used in a tool that conforms to a new version of the metamodel.

Major changes

Changing the structure, semantics and/or the relations in the metamodel are considered as major changes. Existing models are no longer valid in newer versions of the tool. They need to be updated.

The following sections describe these two types of changes and their impact on the aforementioned levels in more details.

6.5.1 Minor changes in the metamodel

These changes don’t influence the existing structure, interrelationships and constraints within the metamodel: Existing instances of an old metamodel remain valid with respect to a new metamodel.

The following changes result in minor changes:

- Adding new metaclasses
- adding new (optional) aggregations
- adding new (optional) references
- adding new (optional) attributes
- adding semantic constraints which do not effect existing metaclasses and relations
- Setting a class from abstract to concrete

These minor changes in the metamodel SHOULD result in minor changes in the XML representation (existing XML descriptions remain valid with respect to a new AUTOSAR XML schema)[8].

AUTOSAR authoring tools that are implemented according to a new version of the AUTOSAR metamodel (only minor changes have been performed) can interpret existing AUTOSAR XML descriptions. The following sections describe the effect on the different abstraction levels:

- “Physical level”: changes on the metamodel don’t have any affect on the physical level.

[8] See requirement on the AUTOSAR XML schema ATI0030

- "Data format level": The new AUTOSAR XML schema contains additional XML elements which represent the new content defined by the metamodel. Since AUTOSAR authoring tools and the respective AUTOSAR XML schema shall allow for exchanging partially described AUTOSAR models[9], existing AUTOSAR XML descriptions remain valid with respect to the new AUTOSAR XML schema.
  In other words: It is only possible to exchange partially defined AUTOSAR XML descriptions if most of the content is considered to be optional in the AUTOSAR XML schema. Adding a new optional element doesn't violate the validity of existing AUTOSAR XML descriptions.
- "Content level": It is expected that new versions of the tool implementations are more powerful with respect to the coverage of metaclasses defined by the AUTOSAR metamodel. Therefore they will be able to interpret existing XML descriptions and create the internal data-representation.
- "Semantic level": The additional constraints are limited to the extensions. An existing AUTOSAR model is still valid with respect to the new constraints since these constraints have by definition of minor changes no impact on existing metaclasses.

Please note: An old AUTOSAR authoring tool can still fully interpret an AUTOSAR XML description if the new features have not been used in the description. Additionally, AUTOSAR authoring tools are not required to support the full set of information defined in the metamodel (see section 6.4 for more details). They can ignore information they do not understand. Therefore an old tool could still interpret the supported features and ignore the additional features. Of course this old tool is usually not able to modify features that have been introduced in the extended metamodel.

An overview over the compatibility of AUTOSAR tools and AUTOSAR XML descriptions in case of **minor changes** is listed in Table 3.

[9] See requirement on AUTOSAR authoring tools ATI0035 and on AUTOSAR XML schema ATI0019

	Old AUTOSAR authoring tool	***New AUTOSAR authoring tool***
Old AUTOSAR XML description	compatible	compatible
New AUTOSAR XML description (minor change)	Fully compatible only if new features are not used in the new XML description.[10] Otherwise an authoring tool can still interpret the new XML description. However it is not able to modify information that was not available in the old AUTOSAR format.	compatible

Table 3: compatibility matrix for minor changes

6.5.2 Major changes in the metamodel

Major changes are modifications on the metamodel which go beyond minor changes. These include:

- Adding constraints for existing metaclasses and relations
- Removing of metaclasses and relations
- Changes on the semantics
- Renaming of metaclasses or relations
- Changing the upper multiplicity of attributes, references or composite associations from 1 to a value bigger than one or vice versa (Those changes usually require changing the internal data structure of tools. Lists of elements need to be supported)
- Adding new specializations to metaclasses which did not already have specializations before

Existing AUTOSAR models can no longer be used without modifications in a tool that supports the new version of the metamodel. Authoring tool SHALL reject the import or provide means to automatically or manually update the model to the new version. Information on update should be given to the user.

Please note, that major changes in the metamodel only have an effect on a tool if the changed structures are supported by the tool. E.g. If a tool is specialized on the creation of `software components` and it doesn't support aspects described in the ECU resource template [5] then any major change in the metamodel concerning the

[10] Please note that the XML namespace might change in later versions of the AUTOSAR XML Schema. If the tool uses XML Schema validation it should be able to apply the old XML Schema to the new XML description even if the XML namespace has changed.

ECU descriptions doesn't have any impact on that tool: From the point of view of this specialized tool, old AUTOSAR models remain compatible.

An overview over the compatibility of AUTOSAR tools and AUTOSAR XML descriptions in case of **major changes** is listed in Table 4.

	Old AUTOSAR authoring tool	***New AUTOSAR authoring tool***
Old AUTOSAR XML description	compatible	Compatible only if changes in the metamodel do not affect the content described in the old XML description. Otherwise an upgrade mechanism should be provided by the authoring tool.
New AUTOSAR XML description (major change)	compatible only if changes in the metamodel do not affect the content described in the new XML description. Otherwise the tool SHOULD reject the XML description.	compatible

Table 4: Compatibility matrix for major changes

These observations lead to the following requirement on AUTOSAR authoring tools:

6.5.2.1 [ATI0005] Authoring tool SHOULD support upgrading AUTOSAR models

Initiator:	WP1.2
Date:	21.02.05
Importance:	Medium
Requirement:	Authoring tool SHOULD support upgrading AUTOSAR models.
Description:	AUTOSAR authoring tools SHOULD support upgrading AUTOSAR models from at least the last major version of the metamodel. It is not required that an authoring tool supports the update of arbitrary AUTOSAR models. Only the content that is internally supported by the tool SHOULD be upgradeable. If the tool doesn't support automatic or manual update of models then it SHALL reject the import of old XML description. It is not required that a tool creates AUTOSAR XML descriptions that are valid with respect to older AUTOSAR XML schema.
Rationale:	Reuse of existing AUTOSAR models.
Use Case:	
Dependencies:	The upgradeability of AUTOSAR XML descriptions is supported by ATI0041, ATI0027 and ATI0021
Conflicts:	
Supporting Material:	
Comment:	This requirement is really hard to implement. This functionality is NOT required to be implemented by all AUTOSAR authoring tools. Instead a dedicated tool specialized on converting models can be defined.

6.6 Support for versioning of AUTOSAR models

6.6.1 Granularity of AUTOSAR models

The minimum granularity of an AUTOSAR model SHALL be defined in the metamodel. If aggregations in the metamodel are marked as <<splitable>>, then the aggregated elements may be stored in different models.
As a rule of thumb: Most PackageableElements (e.g. ARElements) need to be described completely within a single file. However, in the AUTOSAR ECU Configuration Template a more fine-grained granularity is explicitly modeled. Please note that each individual XML file SHALL be valid with respect to the AUTOSAR XML schema.

6.6.2 Annotation of AUTOSAR model elements by version information

The AUTOSAR metamodel supports metadata for storing version information and authorship for each element that is derived from the metaclass Identifiable.

6.7 Standardized error handling

In order to be able to exchange information about errors in models it is required to have a common vocabulary for model errors. This allows for discussing about those errors while using different tools.
As a rationale for this proposal, please consider a scenario where different project partners carry out an AUTOSAR software project by means of different tools for e.g. structural design. Let, for example, developers at different organizations using different AUTOSAR tools work with an AUTOSAR model that contains semantic inconsistencies. Now let the error messages according to the inconsistencies be reported by the particular tools. How can the partners be sure that they talk about the same issue if each of the tools reports a different error without a hint to a standardized error case?

When working with an AUTOSAR authoring tool the validity of an AUTOSAR model can be checked at a wide variety of user interactions. E.g. the model could be checked whenever:

- AUTOSAR XML descriptions are interpreted or created,
- the user inserts new data,
- the user inserts some specific data or
- the user explicitly triggers a validation of the model.

In order to enable the exchange of AUTOSAR models, an AUTOSAR authoring tool SHALL validate the models when interpreting and creating AUTOSAR XML descriptions. Additionally, the tool SHOULD support the validation of models on user request. Further validations are not specified by AUTOSAR and are left over to the implementation of the tool.

6.7.1 Standardized error codes

AUTOSAR does not standardize the **text** of each error message but requires tools to enclose a **reference to a standardized error code** by which means different stakeholders could identify error messages of different tools as referring to the identical problem.
Example:
Tool A: "interfaces are not compatible (ARSema02012)"
Tool B: "serious problems detected, data types are inconsistent (ARSema02012)"

The following severities of errors are defined:

- **Fatal error**: in case a fatal error is detected while interpreting an AUTOSAR XML description the authoring tool SHALL reject it. In case the fatal error occurs while creating an AUTOSAR XML description the tool SHALL stop creating it and SHALL warn the user.
  Trying to recover from those errors would most likely result in unpredictable results and is therefore not reasonable for safety critical software.
- **Critical error**: in case a critical error occurs the tool SHOULD try to recover from the error situation and SHALL display a warning to the user. The tool is

not required to continue consuming or producing the AUTOSAR XML description.

- **Uncritical Error**: in case an uncritical error occurs the tool SHALL try to recover from the error situation and SHALL display a warning message to the user. The tool is required to proceed consuming or producing the AUTOSAR XML description.

AUTOSAR error codes for authoring tools are structured in the following manner:
AR<Level><ID>
The level is defined by the following literals:

Literal	*Level*
Phys	Physical level
Data	Data format level
Cont	Content level
Sema	Semantic level
Pres	Presentation level
Appl	Application level

Example:
An error with id 1 on the content level is indicated by error code
ARCont00001

6.7.1.1 Error codes on physical level

Not standardized by AUTOSAR:
Standardization of error codes on this level would not increase the interoperability. Exchanging error codes such as “hard disk full” would not help in discussions about the content of a model.

6.7.1.2 Error codes on data format level

Error Code	*Criticality*	*Description*
ARData00001	fatal	The XML description is not well-formed. In addition to the error-code the error text SHOULD give more information on the location and reason of the error in the XML description.
ARData00002	critical	The XML description is not valid with respect to the AUTOSAR XML schema (which covers all metaclasses available in the metamodel). The error text SHOULD give more information on the location of the error in the XML description. Furthermore the text SHOULD show the violated validity rule as defined in the W3C specification ([16][15]). Note: this error is not considered to be fatal because even if a validation against the AUTOSAR XML schema is not successful the data that is relevant to the tool could still be valid. See section 6.4 for more details on the integration of specialized tools.

6.7.1.3 Error codes on content level

Error Code	*Criticality*	*Description*
ARCont00001	critical	A reference in the XML description was not resolved. The tool SHOULD show which reference was broken.
ARCont00002	uncritical	The XML description contains more information than is internally represented by the tool. The tool SHOULD indicate which elements are not supported.
ARCont00003	critical	Some information is missing. The tool SHOULD show which data is missing. Note: Authoring tools which are intended to produce model frames out of an AUTOSAR model (coupling of AUTOSAR models to behavioural models) require a minimum set of information. Otherwise they cannot create the model frames. These tools are allowed to reject the AUTOSAR model in case they do not support filling in the missing information. Note: Usually AUTOSAR authoring tools allow for creating AUTOSAR models from scratch. Incomplete models should be exchangeable. For these tools this is not an error at all.
ARCont00004	critical	More than one element is identified by the same short-name hierarchy. These elements have the same `uuid`. The tool MUST check if these elements represent exactly the same content (identity rules are defined in chapter 6.2.1.1): 1) If the content is identical then the tool SHOULD indicate that the element is defined more than once and MUST continue the import. 2) If the content is not identical the tool SHOULD provide a mechanism that allows for choosing between the different alternatives. If this interaction is not provided the tool may reject the AUTOSAR model.
ARCont00005	critical	More than one element is identified by the same short-name hierarchy. These elements have a different `uuid`. The tool SHOULD provide a mechanism that allows for choosing between the different alternatives. If this interaction is not provided the tool may reject the AUTOSAR model.
ARCont00006	uncritical	`uuid` is missing in the AUTOSAR XML description
ARCont00007	critical	The fragmentation of the model is not consistent with the metadata for data exchange. The tool should indicate the inconsistencies to the user. Examples for inconsistencies are: a part of a model is missing, a part of a model is corrupt (checksum is not correct)

6.7.1.4 Error codes on semantic level

Authoring tools shall check the validity of AUTOSAR models against semantic constraints (see requirement ATI0003). If semantic constraints are violated, an error-code needs to be created.
The semantic constraints shall be defined precisely in the AUTOSAR metamodel. For each semantic constraint the following information shall be available (see requirement ATI0034):

For each constraint defined in the metamodel the following information shall be available:

- **Unique id**: a unique id which SHALL be displayed in case the constraint is violated.
- **Constraint**: the formal definition of the constraint using the object constraint language.
- **Severity**: The severity {fatal, critical, uncritical} defines the behavior of the tool in case the constraint is violated.
- **Description**: The human readable description of the constraint.

Constraints are often not directly visible in the UML diagrams of the metamodel. Therefore a special view on the metamodel should be created which lists all defined constraints in form of simple tables. In addition to the fields defined above the generated view should show the name and path of the element in the metamodel.

6.7.1.5 Error codes on presentation level

Not standardized by AUTOSAR.
Standardization of error codes on this level would not increase the interoperability.

6.7.1.6 Error codes on application level

Not standardized by AUTOSAR
Standardization of error codes on this level would not increase the interoperability.

6.7.2 Guidelines for standardized error reporting

6.7.2.1 [ATI0049] Interactive authoring tool SHOULD guide the user to the locations of errors

Initiator:	WPII-1.2 (Interoperability Group)
Date:	13.07.2007
Importance:	Medium
Requirement:	Interactive authoring tool SHOULD guide the user to the locations of errors
Description:	An interactive authoring tool SHOULD guide the user to the locations of errors. If possible the error messages SHOULD contain a hint that describes how the error can be fixed.
Rationale:	Support user while fixing errors in an AUTOSAR model
Use Case:	
Dependencies:	
Conflicts:	
Supporting Material:	
Comment:	

6.7.2.2 [ATI0050] Authoring tool SHOULD support exchanging information about errors

Initiator:	WPII-1.2 (Interoperability Group)
Date:	13.07.2007
Importance:	Medium
Requirement:	Authoring tool SHOULD support exchanging information about errors
Description:	Authoring tool SHOULD support exchanging information about errors.
Rationale:	Some tools (e.g.: highly specialized batch tools) might indicate an error but might only provide limited means for fixing the error. A standardized representation of information of the errors including the location in the model that can be read into another tool can help identifying and fixing the problems.
Use Case:	
Dependencies:	[ATI0055] Metadata for data exchange SHOULD contain information about errors in the model
Conflicts:	
Supporting Material:	.
Comment:	

7 Requirements on the AUTOSAR data exchange format

A common data exchange format is a crucial precondition for tool interoperability. This chapter lists requirements on the AUTOSAR data exchange format.

7.1 Requirements on the AUTOSAR XML schema

The AUTOSAR metamodel defines the language that is used for describing and exchanging AUTOSAR systems. The AUTOSAR XML schema represents this language in XML notation.

7.1.1 General requirements

7.1.1.1 [ATI0025] AUTOSAR XML schema SHALL be consistent with the AUTOSAR metamodel

Initiator:	MMT
Date:	30.03.05, updated 06.07.2005
Importance:	High
Requirement:	AUTOSAR XML schema SHALL be consistent with the AUTOSAR metamodel.
Description:	The AUTOSAR XML schema SHALL NOT contain any information that is not defined in the metamodel. Additionally the AUTOSAR XML schema SHALL support representing all data that can be expressed by means of the AUTOSAR metamodel. In rare cases the AUTOSAR XML schema MAY refer to externally defined XML element definitions. The references to those externally defined XML element definitions need to be explicitly annotated in the metamodel.
Rationale:	The AUTOSAR metamodel contains more than 1000 information entities. If the data exchange format is fully consistent with the metamodel, some parts of tools can (automatically) be generated out of the metamodel. This reduces the threshold for implementing tools that cover bigger subsets of the metamodel. The explicit annotation of definitions of XML-elements that are not represented in the metamodel (e.g. via XML Namespace) is required for avoiding conflicts in XML-element names. Additionally a tool can clearly find out which XML-elements of an XML description are represented in the metamodel.
Use Case:	Reduce the threshold for implementation of tools that cover big parts of the metamodel or even the full metamodel. The integration of externally defined XML-elements allows for the integration of widely implemented existing standard languages such as XHTML, DocBook or MSR-Report for documentation purposes.
Dependencies:	
Conflicts:	
Supporting Material:	
Comment:	It shall be possible to integrate externally declared XML-schema elements which are referenced by the AUTOSAR metamodel. These XML elements shall be explicitly marked within the AUTOSAR XML schema by means of XML namespaces.

7.1.1.2 [ATI0046] AUTOSAR SHALL provide documentation on the Schema generation process and all involved documents and tools

Initiator:	WPII-1.2
Date:	06.06.2007
Importance:	High
Requirement:	AUTOSAR SHALL provide documentation on the Schema generation process and all involved documents and tools
Description:	AUTOSAR SHALL provide documentation on the Schema generation process and all involved documents and tools. This includes detailed information about the versions of the documents
Rationale:	If errors in the XML-Schema are detected it must be possible to identify the documents and tools that have impact on the XML schema.
Use Case:	Fix bugs in the XML-Schema.
Dependencies:	[ATI0025] AUTOSAR XML schema SHALL be consistent with the AUTOSAR metamodel
Conflicts:	
Supporting Material:	
Comment:	

7.1.1.3 [ATI0032] AUTOSAR XML schema SHALL support for unambiguous mapping to metamodel instances

Initiator:	MMT
Date:	30.03.05, updated 06.07.2005
Importance:	High
Requirement:	AUTOSAR XML schema SHALL support for unambiguous mapping to metamodel instances and vice versa.
Description:	A model that is represented by an XML description SHALL always contain all information that is required to map it to instances of the AUTOSAR metamodel. This includes the representation of metaclasses, aggregations, references and attributes. Additionally each AUTOSAR XML description should contain information about the version of the AUTOSAR metamodel and AUTOSAR schema it was using. The mapping SHOULD be computable by simple algorithms.
Rationale:	Unambiguous reproduction of AUTOSAR models which have been represented as AUTOSAR XML descriptions.
Use Case:	An AUTOSAR model could contain a `CompositionType` and an `AtomicSoftwareComponentType` without any further specification of `PortPrototypes`, etc. If the data exchange format uses a common XML element without further annotations for both `ComponentTypes` it is not possible to find out which object needs to be created for that Component within a tool: A `CompositionType` that was created within one tool could be interpreted as an `AtomicSoftwareComponentType` in another tool.
Dependencies:	
Conflicts:	
Supporting Material:	
Comment:	

7.1.1.4 [ATI0028] AUTOSAR XML schema SHALL support for strict XML validation

Initiator:	WP1.2
Date:	06.07.2005
Importance:	High
Requirement:	Data exchange format SHALL support for strict XML validation.
Description:	The data exchange format SHALL support for strict XML validation using standardized and proven XML techniques.
Rationale:	A crucial part for exchanging XML descriptions between different AUTOSAR authoring tools is a common interpretation of the exchanged models. Supporting strict validation of XML descriptions by means of standardized proven XML techniques reduces the risk of having different interpretations of a valid model within different tools.
Use Case:	A timestamp can be expressed in very many different formats. E.g.: “2005-07-07#00:05:13”, “123859483”, “Tuesday, 08.06.2005”. If an XML schema allows all kinds of strings as a timestamp then a tool might not understand the timestamp that was created by another tool. Using strict XML validation the allowed format for a timestamp could be defined and all tools have to use this format in the XML descriptions.
Dependencies:	
Conflicts:	
Supporting Material:	XML schema definition languages are for example: • W3C DTD [16] • W3C XML Schema [15] • OASIS Relax NG [19]
Comment:	

7.1.1.5 [ATI0019] AUTOSAR XML schema SHALL support for description of incomplete models

Initiator:	WP1.2
Date:	07.07.2005
Importance:	High
Requirement:	AUTOSAR XML schema SHALL support description of incomplete models.
Description:	AUTOSAR XML schema SHALL support for describing of incomplete models. However, at least an identifier SHALL be assigned to each instance of a metaclass that is a specialization of the metaclass `Identifiable`. These instances are potential targets of references.
Rationale:	Exchange of work products independent of their level of completion.
Use Case:	In an iterative development process it shall be possible to exchange intermediate work products.
Dependencies:	This requirement is motivated by ATI0035 and ATI0012.
Conflicts:	
Supporting Material:	
Comment:	

7.1.1.6 [ATI0027] AUTOSAR XML schema MAY use XML namespace

Initiator:	MMT
Date:	30.03.05
Importance:	Low
Requirement:	AUTOSAR XML schema MAY use XML Namespace.
Description:	AUTOSAR XML schema MAY use the XML Namespace feature. This also includes supporting multiple namespaces.
Rationale:	The full AUTOSAR metamodel contains more than 1000 data entities (metaclasses and attributes). Modularization of the Metamodel into smaller parts improves the maintainability. The XML namespace feature represents this modularization in the XML representation. Additionally the XML namespace feature can be used to include externally defined XML element definitions into the AUTOSAR data exchange format. E.g. XML elements for documentation purposes such as XHTML or MSR-Report.
Use Case:	Some parts of the metamodel are stable while other parts are still under development. The modularization allows for decoupling these parts: The rather stable parts might only need to be updated once in 1 year. The parts that are still under development can be updated more often. This also allows for faster integration of new feature into the metamodel.
Dependencies:	
Conflicts:	
Supporting Material:	XML namespace is defined in [17].
Comment:	

7.1.1.7 [ATI0023] AUTOSAR XML schema SHALL allow for flexible distribution of XML descriptions over several XML files and referencing between them

Initiator:	WP2.1.1.1
Date:	08.12.04
Importance:	High
Requirement:	AUTOSAR XML schema SHALL allow flexible distribution of XML descriptions over several XML files and referencing between them.
Description:	The AUTOSAR XML schema SHALL allow certain flexibility in mapping the AUTOSAR XML descriptions on one or more XML-files. Even if the AUTOSAR XML description is split up over several files each single file SHALL be valid with respect to the AUTOSAR XML schema. It is necessary that different elements in the AUTOSAR XML descriptions can refer to each other regardless if the element resides in the same file or another. The referencing into elements of external files (like e.g. MS Word) is not scope of this requirement.
Rationale:	Due to a distributed development process the different AUTOSAR XML descriptions will be created at different points in time and with different responsibilities. It shall be accomplished that e.g. the System Generation does not have to combine all descriptions to one single file before processing.
Use Case:	The `InternalBehavior` part of a SW-component can be delivered in a different XML-file than the description of the `Implementation`. Still there are relationships between the two descriptions that need to be described with references between the XML files.
Dependencies:	
Conflicts:	
Supporting Material:	
Comment:	Please note, the metamodel explicitly defines [13] how an AUTOSAR model may be split up over several individual models that are stored in individual files.

7.1.2 Migration between different versions of the AUTOSAR metamodel

7.1.2.1 [ATI0030] AUTOSAR XML schema SHALL ensure upwards compatibility of existing XML descriptions in case on minor changes in the metamodel

Initiator:	MMT
Date:	18.04.05, updated 07.07.2005
Importance:	Medium
Requirement:	AUTOSAR XML schema SHOULD ensure upwards compatibility of existing XML descriptions in case on minor changes in the metamodel.
Description:	The AUTOSAR XML schema SHALL preserve the compatibility of existing XML descriptions in case minor changes have been performed on the metamodel.
Rationale:	Increase the life-cycle of existing XML descriptions.
Use Case:	When adding a new attribute in the metamodel the resulting data exchange format should require no changes in the existing XML instances in order for the latter to be valid according to the new schema.
Dependencies:	ATI0021
Conflicts:	
Supporting Material:	Minor changes are described in chapter 6.5.1.
Comment:	

7.1.2.2 [ATI0029] AUTOSAR XML schema SHALL contain the version information of the metamodel it was generated from

Initiator:	MMT
Date:	30.03.05
Importance:	High
Requirement:	AUTOSAR XML schema SHALL contain the version information of the metamodel it was generated from.
Description:	The version number of the metamodel SHALL be generated into the AUTOSAR XML schema.
Rationale:	The schema MUST contain information that clearly identifies the version of the metamodel it was generated from.
Use Case:	
Dependencies:	
Conflicts:	
Supporting Material:	
Comment:	

7.2 Requirements on the AUTOSAR metamodel

The AUTOSAR metamodel should be the central repository for all language elements that can be used to describe AUTOSAR systems. This section describes requirements on the AUTOSAR metamodel which support the interoperability of AUTOSAR authoring tools (see ATI0034) and the maintainability of the data exchange format.

7.2.1.1 [ATI0034] AUTOSAR metamodel SHALL precisely describe semantic constraints

Initiator:	WP1.2
Date:	15.07.2005
Importance:	High
Requirement:	AUTOSAR metamodel SHALL precisely describe semantic constraints.
Description:	AUTOSAR metamodel SHALL precisely describes semantic constraints, preferably by means of a standardized formal language such as OCL. In addition to the formal description of the semantic constraint the following information SHALL be available: unique id, severity, informal description.
Rationale:	Avoid ambiguous descriptions of semantic constraints which could be interpreted differently by different tool vendors.
Use Case:	Tool vendors need a precise source for the implementation of semantic constraints.
Dependencies:	
Conflicts:	
Supporting Material:	
Comment:	

7.2.1.2 [ATI0021] AUTOSAR SHOULD provide for upward compatibility detection

Initiator:	MMT
Date:	18.04.05
Importance:	Medium
Requirement:	AUTOSAR SHOULD provide for upward compatibility detection.
Description:	AUTOSAR SHOULD develop a mechanism that identifies which changes in the metamodel lead to incompatible XML descriptions. A tool SHOULD check if a generated schema is compatible with a previous version. Compatibility means that descriptions valid for the old schema are also valid for the new one.
Rationale:	Identifying major changes in the metamodel which result in incompatible XML descriptions.
Use Case:	
Dependencies:	
Conflicts:	
Supporting Material:	
Comment:	

7.2.1.3 [ATI0041] AUTOSAR metamodel SHOULD mark concepts that will be removed in future versions

Initiator:	WP1.2
Date:	21.07.2005
Importance:	Medium
Requirement:	AUTOSAR metamodel SHOULD mark concepts that will be removed in future versions.
Description:	Metamodel elements (e.g. associations, metaclasses, attributes, aggregations) SHOULD be explicitly marked as ‘deprecated’ if they will be removed in future versions.
Rationale:	Decreasing the effort of model conversion in case of major changes.
Use Case:	An authoring tool could warn the user if he opens a model that uses deprecated concepts.
Dependencies:	
Conflicts:	
Supporting Material:	
Comment:	

7.2.1.4 [ATI0047] All artifacts in the AUTOSAR metamodel that are part of the AUTOSAR standard SHALL be documented

Initiator:	WPII-1.2
Date:	05.06.2007
Importance:	High
Requirement:	All artifacts in the metamodel that are part of the standard SHALL be documented
Description:	The AUTOSAR metamodel is a graphical representation of the AUTOSAR data exchange format. All artifacts that are part of the AUTOSAR standard SHALL be documented. The documentation of all standardized artifacts SHALL be available as a printable document.
Rationale:	Improve quality of standard by explicitly making all artifacts that belong to the standard visible to the reviewer. Avoid artifacts that are not intended to be part of the standard.
Use Case:	
Dependencies:	
Conflicts:	
Supporting Material:	
Comment:	

7.2.1.5 [ATI0051] The AUTOSAR metamodel SHALL indicate which metaclasses, attributes, references and aggregations are required for an activity in the AUTOSAR methodology

Initiator:	WPII-1.2
Date:	13.07.2007
Importance:	High
Requirement:	The AUTOSAR metamodel SHALL indicate which meta classes, attributes, references and aggregations are required for an activity in the AUTOSAR methodology
Description:	The AUTOSAR methodology describes the sequence of activities within an AUTOSAR development process. For each activity the metamodel SHALL precisely identify which metaclasses, attributes, references and aggregations are required for each activity.
Rationale:	Improve quality of standard by explicitly making all artifacts that belong to the standard visible to the reviewer. Avoid artifacts that are not intended to be part of the standard.
Use Case:	
Dependencies:	
Conflicts:	
Supporting Material:	
Comment:	

7.2.1.6 [ATI0031] AUTOSAR metamodel SHOULD provide extension mechanism

Initiator:	MMT
Date:	18.04.05, updated 07.07.2005
Importance:	Medium
Requirement:	AUTOSAR metamodel SHOULD provide extension mechanism.
Description:	AUTOSAR metamodel SHOULD provide a mechanism that allows for annotating an AUTOSAR model by additional information. For these extensions only concepts are allowed that are explicitly defined in the AUTOSAR metamodel. E.g. it SHALL not be possible to include arbitrary elements that are not available in the AUTOSAR metamodel.
Rationale:	Several standards already facilitate mechanisms to provide extension points in their specifications.
Use Case:	This extension mechanism SHALL be mainly used to capture extension that are not yet part of the standard (however the aim is the migration of those into the standard later on). A valid example would be the MSR Special-Data-Groups or UML-tagged values.
Dependencies:	
Conflicts:	
Supporting Material:	
Comment:	

7.2.1.7 [ATI0052] The AUTOSAR extension mechanism SHOULD support formal definition of extensions

Initiator:	WPII-1.2 (Interoperability Group)
Date:	13.07.2007
Importance:	Medium
Requirement:	The AUTOSAR extension mechanism SHOULD support formal definition of extensions.
Description:	The AUTOSAR extension mechanism SHOULD support the formal definition of elements, attributes and structure of extensions.
Rationale:	Formal language for describing the additional information in order to support tool implementations.
Use Case:	If additional information needs to be exchanged that is not (yet) standardized in the AUTOSAR metamodel, then partners can agree on a structure of the additional information and formally describe that structure using a concept provided by the AUTOSAR extension mechanism
Dependencies:	
Conflicts:	
Supporting Material:	See e.g. UML profile mechanism [27]. Defining an extension can be compared with defining a UML profile.
Comment:	

7.2.1.8 [ATI0053] The AUTOSAR extension mechanism SHOULD support validating extended AUTOSAR models

Initiator:	WPII-1.2 (Interoperability Group)
Date:	13.07.2007
Importance:	Medium
Requirement:	The AUTOSAR extension mechanism SHOULD support validating extended AUTOSAR models
Description:	The AUTOSAR extension mechanism SHOULD support validating extended AUTOSAR models based on the definition of the extension.
Rationale:	Check if additional information complies to the agreed structure
Use Case:	If project partners agree on an extension the receiving tool should be able to check if the received extended AUTOSAR model complies to the agreed extension.
Dependencies:	
Conflicts:	
Supporting Material:	See e.g. UML profile mechanism [27]. Validating a model can be compared with validating a model against a profile
Comment:	

7.2.1.9 [ATI0054] The AUTOSAR extension mechanism SHOULD support multiple individual extensions in a single AUTOSAR model

Initiator:	WPII-1.2 (Interoperability Group)
Date:	13.07.2007
Importance:	Medium
Requirement:	The AUTOSAR extension mechanism SHOULD support multiple individual extensions in a single AUTOSAR model
Description:	The AUTOSAR extension mechanism SHOULD support multiple individual extensions in a single AUTOSAR model. For each additional information the extension mechanism SHOULD be able to find out the corresponding extension definition.
Rationale:	Allow for exchanging additional information for several use cases in a single AUTOSAR model. Use case specific tools may ignore extensions that are not required for fulfilling the supported activity.
Use Case:	In a development process the partners might have identified the need for extensions have agreed on an extension definition. The extensions are implemented by several tools which require additional extensions for supporting round-trip engineering. This will lead to multiple extensions in a single AUTOSAR description.
Dependencies:	
Conflicts:	
Supporting Material:	See e.g. UML profile mechanism [27]. Defining several extensions for a single model can be compared with applying several profiles to a single model.
Comment:	

7.3 Requirements on metadata for data exchange

The availability of metadata for data exchange supports the interoperability of AUTOSAR authoring tools. This metadata is e.g. required for checking the completeness of a shipment or for giving additional information on what the receiver of an AUTOSAR model is allowed to do with the model. This chapter describes requirements on the metadata for data exchange.

7.3.1.1 [ATI0037] Metadata for data exchange SHALL be based on existing standards

Initiator:	WP1.2
Date:	20.07.2005
Importance:	High
Requirement:	The metadata for data exchange SHALL be based on existing standards and SHALL be defined by AUTOSAR.
Description:	The metadata for supporting data exchange SHALL be based on existing standards.
Rationale:	Use of existing standardized (existing) solutions for interpretation and creation of metadata.
Use Case:	Reduce costs for implementation of tools/libraries that are capable of interpreting and creating of metadata for data exchange.
Dependencies:	
Conflicts:	
Supporting Material:	Examples of existing standards for describing metadata e.g. for data exchange are: OMG Reusable Assets Specification – OMG RAS [24] ASAM Container Catalog – ASAM CC [25][26]
Comment:	

7.3.1.2 [ATI0038] Description of access rights SHOULD allow for being mapped to data structures that are different from the AUTOSAR metamodel

Initiator:	WP1.2
Date:	20.07.2005
Importance:	Medium
Requirement:	Description of access rights SHOULD allow for being mapped to data structures that are different from the AUTOSAR metamodel.
Description:	The description of access rights SHOULD allow for being mapped to data structures that are different from the AUTOSAR metamodel.
Rationale:	The internal data structure of AUTOSAR authoring tools will very likely be different to the structure of the AUTOSAR metamodel. Tools can only represent information about the access rights to the user if the access rights defined on instances of the AUTOSAR metamodel can be mapped to instances of the tool internal datastructure.
Use Case:	A tool might want to represent a system on a high level of abstraction. E.g. the tool only shows some connections between a `CanBus` and an `ECUInstance`. The existence of `PhysicalMediumSegments` and `ECUCommunicationPortInstance` might be hidden to the user. If the access rights are defined in a very complex manner, then it might not be possible to decide which impact these right have to model elements in the tool with a different internal data structure.
Dependencies:	
Conflicts:	
Supporting Material:	
Comment:	It SHOULD be possible to map the aspect of access rights to file-based configuration management systems. In order to limit the complexity only the values “read-only” and “read-write” are allowed.

7.3.1.3 [ATI0039] Metadata for data exchange SHALL NOT change the content of AUTOSAR models

Initiator:	WP1.2
Date:	20.07.2005
Importance:	High
Requirement:	Metadata for data exchange SHALL NOT change the content of AUTOSAR models.
Description:	The metadata for data exchange SHALL be independent from the content of an AUTOSAR model. The content of an AUTOSAR model SHALL remain identical if it is exchanged together with or without the metadata.
Rationale:	Allow for integration of tools that do not support metadata for data exchange. Avoiding the creation of new versions of the AUTOSAR model for each data exchange.
Use Case:	[ATUC_008]
Dependencies:	
Conflicts:	
Supporting Material:	
Comment:	e.g.: the metadata for data exchange shall not define any order for reading files that contain AUTOSAR XML descriptions.

7.3.1.4 [ATI0055] Metadata for data exchange SHOULD contain information about errors in the model

Initiator:	WPII-1.2 (Interoperability Group)
Date:	20.07.2007
Importance:	Medium
Requirement:	Metadata for data exchange SHOULD contain information about errors in the model.
Description:	Metadata for data exchange SHOULD contain information about errors in the model in a standardized format. This format SHOULD contain the standardized error code, the informal error message and the location of the detected error in the model.
Rationale:	Exchange information about errors contained in a model
Use Case:	Within an AUTOSAR development process several kinds of tools might be used. E.g.: Interactive tools, batch tools, tools that do not support all elements and constraints in the metamodel. If error-log messages that are created by a specialized batch-tool are created in a standardized format, then this information can interpreted by other tools for fixing the errors.
Dependencies:	Error codes
Conflicts:	
Supporting Material:	
Comment:	

7.3.1.5 [ATI0056] Metadata for data exchange SHOULD contain information about deleted, changed and moved elements

Initiator:	WPII-1.2
Date:	11.09.2007
Importance:	Medium
Requirement:	Metadata for data exchange SHOULD contain information about deleted, changed and moved elements.
Description:	Metadata for data exchange SHOULD contain information that supports merging models. It SHOULD contain information about added, changed and moved elements.
Rationale:	Support automated merge without user-interaction
Use Case:	Two models have been created out of a common model and contain redundant information. E.g.: both models contain an element with uuid=3. In one model that element has been removed explicitly. While merging the two models a tool needs to know if an element was removed explicitly or if one model was incomplete.
Dependencies:	
Conflicts:	
Supporting Material:	
Comment:	

8 Compliance

An AUTOSAR authoring tool may be called "*AUTOSAR compliant*" if it implements the mandatory requirements on AUTOSAR authoring tools defined in this document. For better readability the requirements of this document are summarized in the following sections.

8.1 Summary of requirements on AUTOSAR authoring tools

The following table lists all requirements on AUTOSAR authoring tools which are required for tool interoperability. The mandatory requirements are marked by grey background color.
Each requirement is assigned to a number of abstraction levels (marked by 'x'). This allows for e.g. easily identifying all requirements that are relevant on the content level and all higher level. A plugin for an AUTOSAR authoring tool could directly access the internal data structure of the authoring tool: Plugin and authoring tool would exchange information on content level. For this communication requirements on lower levels are not relevant.

Requirement[11] on AUTOSAR authoring tools	***See page***	***Abstraction Level***				
		Physical	***Data format***	***Content***	***Semantic***	***Presentation***
[ATI0003] Authoring tool SHALL support validity checks	36				x	
[ATI0005] Authoring tool SHOULD support upgrading AUTOSAR models	69		x	x		
[ATI0007] Authoring tool SHALL NOT change model contents without the intention of the user	34			x		
[ATI0010] Authoring tool SHALL support sets of files	31	x				
[ATI0012] Authoring tool SHALL support AUTOSAR XML descriptions	32		x			
[ATI0014] Authoring tool SHALL support standardized error codes	39		x	x	x	x
[ATI0015] Authoring tool SHOULD use AUTOSAR graphical notation	38					x
[ATI0016] Documentation of authoring tool SHOULD describe supported features	63					
[ATI0024] Authoring tool SHALL support unique identification of model elements	44			x		
[ATI0033] Authoring tool SHALL be able to import and export supported model elements as AUTOSAR XML descriptions	33			x		
[ATI0035] Authoring tool SHALL support exchange of partial information	35			x		
[ATI0036] Authoring tool SHOULD be able to interpret and create metadata for data exchange	60	x		x		
[ATI0040] Authoring tool SHOULD prohibit the user from modifying model elements that are marked read-only	55					x
[ATI0042] Authoring tool SHALL support for merging of AUTOSAR models	49			x		
[ATI0044] Authoring tool SHOULD provide a mechanism for resolving merging conflicts	54			x		x
[ATI0043] Authoring tool SHOULD provide a mechanism for showing differences between AUTOSAR models	41			x		x
[ATI0048] Authoring tool SHOULD support AUTOSAR extension mechanism	35			x		
[ATI0049] Interactive authoring tool SHOULD guide the user to the locations of errors	75					x
[ATI0050] Authoring tool SHOULD support exchanging information about errors	75			x		

Table 5: Requirements on AUTOSAR authoring tools

[11] Requirement ids are not continuous since some concepts have been changed during the creation of this document and the requirements have been removed.

8.2 Summary of requirements on AUTOSAR XML schema

Requirement ATI0012 requires an AUTOSAR authoring tool to support the AUTOSAR XML schema. The requirements on the AUTOSAR XML schema are listed in the following table. The AUTOSAR XML schema is completely located at on the Data Format Level. Therefore all other levels are grayed out:

Requirement on the AUTOSAR XML schema and the AUTOSAR metamodel	***See page***	***Abstraction Level***				
		Physical	***Data format***	*Content*	*Semantic*	*Presentation*
[ATI0019] AUTOSAR XML schema SHALL support for description of incomplete models	81		x			
[ATI0023] AUTOSAR XML schema SHALL allow for flexible distribution of XML descriptions over several XML files and referencing between them	83		x			
[ATI0025] AUTOSAR XML schema SHALL be consistent with the AUTOSAR metamodel	77		x			
[ATI0027] AUTOSAR XML schema MAY use XML namespace	82		x			
[ATI0028] AUTOSAR XML schema SHALL support for strict XML validation	80		x			
[ATI0029] AUTOSAR XML schema SHALL contain the version information of the metamodel it was generated from	84		x			
[ATI0030] AUTOSAR XML schema SHALL ensure upwards compatibility of existing XML descriptions in case on minor changes in the metamodel	84		x			
[ATI0032] AUTOSAR XML schema SHALL support for unambiguous mapping to metamodel instances	79		x			

Table 6: Requirements on the AUTOSAR XML schema

8.3 Summary of requirements on the AUTOSAR metamodel

The requirements on the AUTOSAR metamodel are listed in the following table:

Requirement on the AUTOSAR metamodel	***See page***	***Abstraction Level***				
		Physical	***Data format***	***Content***	***Semantic***	***Presentation***
[ATI0021] AUTOSAR SHOULD provide for upward compatibility detection	86		x	x	x	
[ATI0034] AUTOSAR metamodel SHALL precisely describe semantic constraints	85				x	
[ATI0031] AUTOSAR metamodel SHOULD provide extension mechanism	88		x	x	x	
[ATI0041] AUTOSAR metamodel SHOULD mark concepts that will be removed in future versions	86		x	x	x	
[ATI0047] All artifacts in the AUTOSAR metamodel that are part of the AUTOSAR standard SHALL be documented	87			x		
[ATI0051] The AUTOSAR metamodel SHALL indicate which metaclasses, attributes, references and aggregations are required for an activity in the AUTOSAR methodology	87			x		
[ATI0052] The AUTOSAR extension mechanism SHOULD support formal definition of extensions	89		x	x	x	
[ATI0053] The AUTOSAR extension mechanism SHOULD support validating extended AUTOSAR models	89		x	x	x	
[ATI0054] The AUTOSAR extension mechanism SHOULD support multiple individual extensions in a single AUTOSAR model	90		x	x	x	

Table 7: Requirements on the AUTOSAR metamodel

8.4 Summary of requirements on metadata for data exchange

The requirements on the metadata for data exchange listed in the following table:

Requirement on metadata for data exchange	***See page***	***Abstraction Level***				
		Physical	***Data format***	***Content***	***Semantic***	***Presentation***
[ATI0037] Metadata for data exchange SHALL be based on existing standards	91	x	x	x	x	
[ATI0038] Description of access rights SHOULD allow for being mapped to data structures that are different from the AUTOSAR metamodel	92			x		
[ATI0039] Metadata for data exchange SHALL NOT change the content of AUTOSAR models	93		x	x	x	
[ATI0055] Metadata for data exchange SHOULD contain information about errors in the model	94		x	x	x	
[ATI0056] Metadata for data exchange SHOULD contain information about deleted, changed and moved elements	95		x	x		

Table 8: Requirements on the AUTOSAR metadata for data exchange

8.5 Notes on compliance

8.5.1 Compliance classes based on coverage of the metamodel

In order to allow for seamless tool interoperability, AUTOSAR authoring tools should support a common set of features - the tools should implement a common coverage of the AUTOSAR metamodel.
Otherwise one tool would generate AUTOSAR models that cannot be interpreted by another tool. The definition of these common sets of features highly depends on the intended work-flow. The compliance classes should be based on the support of activities in the AUTOSAR Methodology [3]

Examples for compliance classes that are based on the coverage of the metamodel are:

- RTE-generation compliant (all information that is required for generating the RTE can be created by the authoring tool)
- SW component to ECU mapping compliant (all information that is required for mapping SW components onto ECUs can be created and modified by the authoring tool)

8.5.2 Testing the compliance of an AUTOSAR authoring tool

The compliance of a given tool could be tested by defining a set of AUTOSAR XML descriptions which need to be processed by the AUTOSAR authoring tools. These compliance tests should be performed in early phases of an AUTOSAR system development by the stakeholders who need to exchange AUTOSAR models. This document doesn't define any models for testing the interoperability of AUTOSAR authoring tools.

9 References

9.1 Normative References to AUTOSAR documents

[1] Glossary
AUTOSAR_Glossary.pdf

[2] Requirements on Interoperability of Authoring Tools
AUTOSAR_RS_InteroperabilityAuthoringTools.pdf

[3] Methodology
AUTOSAR_Methodology.pdf

[4] Software Component Template
AUTOSAR_SoftwareComponentTemplate.pdf

[5] Specification of System Template
AUTOSAR_SystemTemplate.pdf

[6] Specification of ECU Resource Template
AUTOSAR_ECUResourceTemplate.pdf

[7] Specification of ECU Configuration Template
AUTOSAR_ECU_Configuration.pdf

[8] Specification of Basic Software Module Description Template
AUTOSAR_BSWMDTemplate.pdf

[9] Metamodel
AUTOSAR_Metamodel.eap

[10] Specification of Interaction with Behavioral Models
AUTOSAR_InteractionBehavioralModels.pdf

[11] Specification of Graphical Notation
AUTOSAR_GraphicalNotification.pdf

[12] Model Persistence Rules for XML
AUTOSAR_ModelPersistenceRulesXML.pdf

[13] Template UML Profile and Modeling Guide
AUTOSAR_TemplateModelingGuide.pdf

9.2 Normative References to external documents

[14] XML Information Set (Second Edition), W3C Recommendation 4 February 2004,
http://www.w3.org/TR/xml-infoset/

[15] XML Schema 1.1,
http://www.w3.org/XML/Schema

[16] Extensible Markup Language (XML) 1.1,
http://www.w3.org/TR/xml11

[17] Namespaces in XML 1.1, W3C Recommendation 4 February 2004,
http://www.w3.org/TR/xml-names11

[18] Extensible Markup Language (XML) Conformance Test Suites, http://www.w3.org/XML/Test/

[19] RELAX NG Specification, http://www.relaxng.org/spec-20011203.html

[20] MSR-TR-CAP, http://www.msr-wg.de/medoc/download/msr-tr-cap/msr-tr-cap.pdf

[21] XML Metadata Interchange (XMI), v1.2, http://www.omg.org/cgi-bin/apps/doc?formal/02-01-01.pdf

[22] XML Metadata Interchange (XMI), v2.0, http://www.omg.org/cgi-bin/apps/doc?formal/05-05-01.pdf

[23] XML Metadata Interchange (XMI) v2.1, http://www.omg.org/cgi-bin/apps/doc?formal/05-09-01.pdf

[24] Reusable Asset Specification, http://www.omg.org/cgi-bin/apps/doc?ptc/05-04-02.pdf

[25] MSR CC, http://www.msr-wg.de/medoc/download/msrcc/v20/msrcc-eadoc-en/msrcc-eadoc.pdf

[26] ASAM CC (Successor of MSR CC) http://www.asam.net/03_standards_06.php.

[27] Unified Modeling Language: Superstructure, Version 2.0, OMG Available Specification, ptc/05-07-04. http://www.omg.org/cgi-bin/apps/doc?formal/05-07-04.pdf

[28] Unified Modeling Language OCL, Version 2.0, OMG Available Specification, ptc/05-06-06. http://www.omg.org/cgi-bin/apps/doc?ptc/05-06-06.pdf

9.3 Other References

[29] Tony Clark, Andy Evans, Paul Sammut, James Willans. *Applied Metamodelling, A Foundation for Language Driven Development*. Version 0.1

[30] Ivan Kurtev, Jean Bézivin, Mehmet Aksit. *Technological Spaces: an Initial Appraisal.* http://www.sciences.univ-nantes.fr/lina/atl/www/papers/PositionPaperKurtev.pdf

[31] MD5 – Wikipedia entry, http://en.wikipedia.org/wiki/MD5

[32] David Carlson, Modeling XML Applications with UML: Practical e-Business Applications, Addison Wesley, 2001. supporting material to the book available at http://www.xmlmodeling.com/