Document Title	Requirements on Interoperability of Autosar Tools
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	101

Document Status	Final
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	4.4.0

Document Change History

Date	Release	Changed by	Description
2018-10-31	4.4.0	AUTOSAR Release Management	• editorial changes
2017-12-08	4.3.1	AUTOSAR Release Management	• editorial changes
2016-11-30	4.3.0	AUTOSAR Release Management	• added use case and requirements for the description of data exchange points
2015-07-31	4.2.2	AUTOSAR Release Management	• added use case section that was part of the TR_IOAT
2014-10-31	4.2.1	AUTOSAR Release Management	• added requirement for naming conventions [RS_IOAT_00003] • minor editorial changes
2013-03-15	4.1.1	AUTOSAR Administration	• improved requirements traceability • harmonized document structure
2011-05-13	3.2.1	AUTOSAR Administration	• Legal disclaimer revised
2008-08-13	3.1.1	AUTOSAR Administration	• Legal disclaimer revised
2007-12-21	3.0.1	AUTOSAR Administration	• Document meta information extended • Small layout adaptations made

2007-01-24	2.1.15	AUTOSAR Administration	• "‘Advice for users"’ revised • "‘Revision information"’ added
2006-11-28	2.1	AUTOSAR Administration	• Legal disclaimer revised
2006-05-16	2.0	AUTOSAR Administration	• Initial release

Disclaimer

Table of Contents

1 Introduction 6

1.1 Scope of this document . . . 7
1.2 Terminology . . . 8
1.3 Document Conventions . . . 9
1.4 Guidelines . . . 10

2 Requirements Tracing 11

3 Use-Cases (non normative) 12

3.1 Usage within the different steps of top-down functional development . 12
3.2 Support for subcontracting . . . 13
3.3 Support of different Versions of Meta-Model . . . 14
3.4 Concurrent modeling . . . 14
3.4.1 Renaming model elements . . . 15
3.4.2 Updating of model elements . . . 16
3.4.3 Moving of elements from one namespace to another . . . 17
3.4.4 Parallel development of models . . . 17
3.5 Direct exchange of AUTOSAR model in tool-chain . . . 17
3.6 Shipment of AUTOSAR models and related artifacts . . . 19
3.7 Filter and merge AUTOSAR models . . . 20
3.8 Handling of identical double definitions . . . 20
3.9 Description of Data Exchange Points . . . 21
3.9.1 Support for Detection of Incompatibilities . . . 22
3.9.2 Validation of Models . . . 24
3.9.3 Machine Readable Language . . . 26
3.9.4 Authoring and Lifecycle . . . 27

4 Requirements 33

[RS_IOAT_00001] Support data exchange . . . 33
[RS_IOAT_00002] Standardize the handling of errors in AUTOSAR models . 33
[RS_IOAT_00003] Provide naming conventions . . . 33
[RS_IOAT_00004] Standardized Authoring Support Data . . . 34
[RS_IOAT_00007] AUTOSAR Example Data Exchange Point Description . . 34
[RS_IOAT_00008] AUTOSAR Baseline of Data Exchange Point . . . 34

A Glossary 36

References

[1] Interoperability of AUTOSAR Tools
AUTOSAR_TR_InteroperabilityOfAutosarTools

[2] Meta Model
AUTOSAR_MMOD_MetaModel

[3] Standardization Template
AUTOSAR_TPS_StandardizationTemplate

[4] Main Requirements
AUTOSAR_RS_Main

[5] Requirements on AUTOSAR Features
AUTOSAR_RS_Features

[6] Methodology
AUTOSAR_TR_Methodology

[7] Generic Structure Template
AUTOSAR_TPS_GenericStructureTemplate

[8] Container Catalog XML Model Specification
http://www.asam.net

[9] Interoperability Of Autosar Tools Supplement
AUTOSAR_TR_InteroperabilityOfAutosarToolsSupplement

[10] Software Process Engineering Meta-Model Specification
http://www.omg.org/spec/SPEM/2.0/

1 Introduction

1.1 Scope of this document

This document collects the requirements on the `Interoperability of Autosar Tools` specification (IAOT) [1].

1.2 Terminology

1. The AUTOSAR metamodel[2] 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.
2. 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.
3. 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 and defines the AUTOSAR data exchange format.
4. 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.
5. An AUTOSAR tool is a software tool which supports interpreting, processing and/or creating of AUTOSAR XML descriptions
6. An Metadata includes pertinent information about data, including information about the authorship, versioning, access-rights, timestamps etc.

1.3 Document Conventions

The representation of requirements in AUTOSAR documents follows the table specified in [TPS_STDT_00078], see Standardization Template, chapter Support for Traceability ([3]).

The verbal forms for the expression of obligation specified in [TPS_STDT_00053] shall be used to indicate requirements, see Standardization Template, chapter Support for Traceability ([3]).

1.4 Guidelines

Existing specifications shall be referenced (in form of a single requirement). Differences to these specifications are specified as additional requirements. All Requirements shall have the following properties:

- Redundancy
  Requirements shall not be repeated within one requirement or in other requirements.
- Clearness
  All requirements shall allow one possibility of interpretation only. Used technical terms that are not in the glossary must be defined.
- Atomicity
  Each Requirement shall only contain one requirement. A Requirement is atomic if it cannot be split up in further requirements.
- Testability
  Requirements shall be testable by analysis, review or test.
- Traceability
  The source and status of a requirement shall be visible at all times.

2 Requirements Tracing

The following table references the requirements specified in [4], [5] and links to the fulfillments by this document.

Requirement	Description	Satisfied by
[RS_BRF_01028]	AUTOSAR shall provide naming conventions for symbols in its documentation	[RS_IOAT_00003]
[RS_Main_00300]	AUTOSAR shall provide data exchange formats to support work-share in large inter and intra company development groups	[RS_IOAT_00001] [RS_IOAT_00002]
[RS_Main_00301]	AUTOSAR shall specify profiles for data exchange to support work-share in large inter- and intra-company development groups	[RS_IOAT_00004] [RS_IOAT_00007] [RS_IOAT_00008] [UC_IOAT_00011] [UC_IOAT_00012] [UC_IOAT_00013] [UC_IOAT_00014] [UC_IOAT_00015] [UC_IOAT_00018] [UC_IOAT_00022] [UC_IOAT_00030] [UC_IOAT_00041] [UC_IOAT_00090] [UC_IOAT_00092] [UC_IOAT_00100]

Table 2.1: RequirementsTracing

3 Use-Cases (non normative)

This chapter describes use-cases for the interoperability of AUTOSAR tools. The intention of these use cases is to point out 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 “UC_IOAT” (meaning Use Case - Interoperability of AUTOSAR Tools).

Please note the different levels between the use cases described here and the use cases (also called “capability patterns”) described in the AUTOSAR methodology model (see [6]).

The use cases in the methodology focus on the logical tasks and their work products and do not address the aspects of using different tools, as the use cases do which are listed below.

The same task of the methodology may be performed by various AUTOSAR tools. On the other hand one particular AUTOSAR tool may be used to perform several different tasks. When different tools are used in a project, the logical data flow described by the methodology must nonetheless be provided.

3.1 Usage within the different steps of top-down functional development

[UC_IOAT_00005] 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 may 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 may be edited by another tool and/or person.
  (Action: exchange of incomplete models between tools within a company)
- The AUTOSAR model may 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 partial models between tools that are used in different companies)

- It may be the case that only a subset of the whole AUTOSAR model is passed to a supplier. The supplier may need to make sure that all required information is available. The possible partitioning of an AUTOSAR model is defined using ≪atpSplitable≫. Therefore AUTOSAR tools shall be able to handle partial models (e.g. with dangling references)
(Action: extraction of an AUTOSAR model out of the full AUTOSAR model, so that 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).

  This use-case has different aspects: partial model can mean:

  1. a subset of model elements (e.g. only some components)
  2. a subset of the specification of a particular model element (all components, but not all data of the components e.g. not Internal Behavior)
  3. a combination of these two: Only some components with only a subset of the component description.

- A supplier may be contracted to implement an AUTOSAR software component. The supplier 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

  (Action: 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 in time some AUTOSAR models may need to be integrated / merged. Potential collisions need to be resolved. Two cases need to be distinguished:

  – “integrated” means that a component is incorporated into an existing system. Even if a component can be considered as a partial model of an system wide AUTOSAR model, there are still some tasks to be performed as defined in the AUTOSAR Methodology (e.g. define mappings). ([6])

  – “merged” means that a partial model is integrated into an existing model. The main motivation for this is concurrent modeling, variant handling, different responsibilities along e.g. a component development process.

  Please refer also to [UC_IOAT_00009], [UC_IOAT_00004], [UC_IOAT_00010]

3.2 Support for subcontracting

[UC_IOAT_00001] Integrate extracts from an AUTOSAR model of an OEM passed for further refinement and implementation to a supplier ⌈ 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 have been made on the models 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 (the details are explained in [7]).

Such mechanisms can easily be based on proper distribution to sub-models (using the stereotype ≪`atpSplitable`≫. In this case the meta data in the ASAM catalog (see [8]) can indicate the changed artifacts.

A more fine-grain control can be performed using specific `Collection`s.

This use-case applies in particular to AUTOSAR authoring tools which are used to create and maintain AUTOSAR XML descriptions.

3.3 Support of different Versions of Meta-Model

[UC_IOAT_00002] Dealing with changes of the AUTOSAR meta model over time ⌈ The AUTOSAR meta-model and the derived AUTOSAR data exchange format will change over time. It SHALL be predictable whether tools (potentially with different underlying meta model versions) can exchange AUTOSAR models.

⌋*()*

3.4 Concurrent modeling

[UC_IOAT_00004] Allowing for concurrent work on the same mode ⌈ 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. ⌋*()*

Concurrent development is restricted to `PackageableElement`s. It should always be clear who is responsible for a particular `PackageableElement`. The representation of this responsibility is not in the scope of AUTOSAR. It could be handled in the catalog.

In addition to this, concurrent development can be handled by split the work into artifacts such that one party can handle the artifact on his own. This is supported by application of the stereotype ≪`atpSplitable`≫. There is no more concurrency than provided by ≪`atpSplitable`≫.

3.4.1 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.

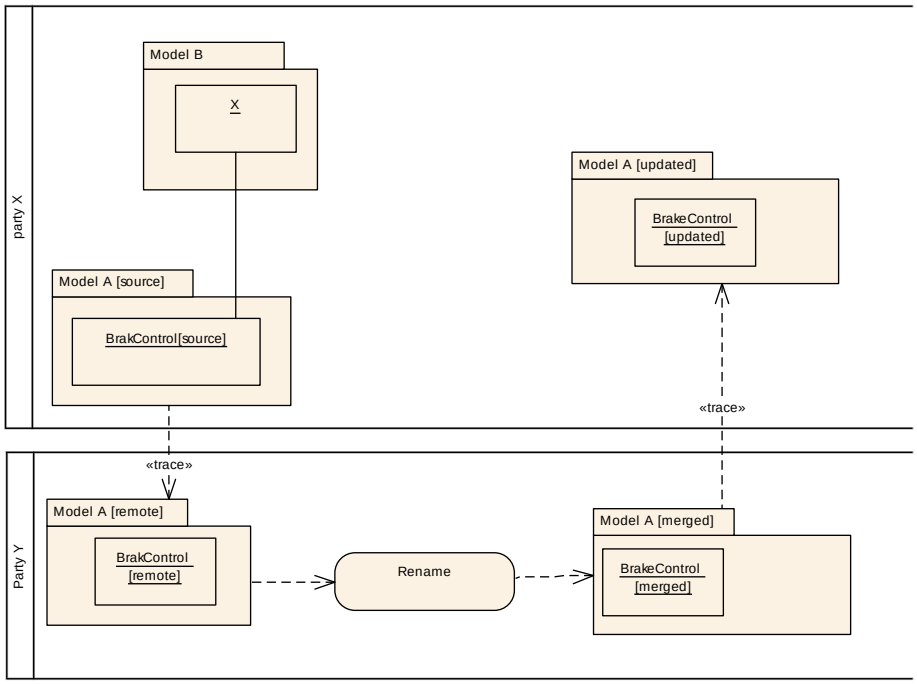

Figure 3.1: Concurrent modeling - renaming of elements

3.4.2 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

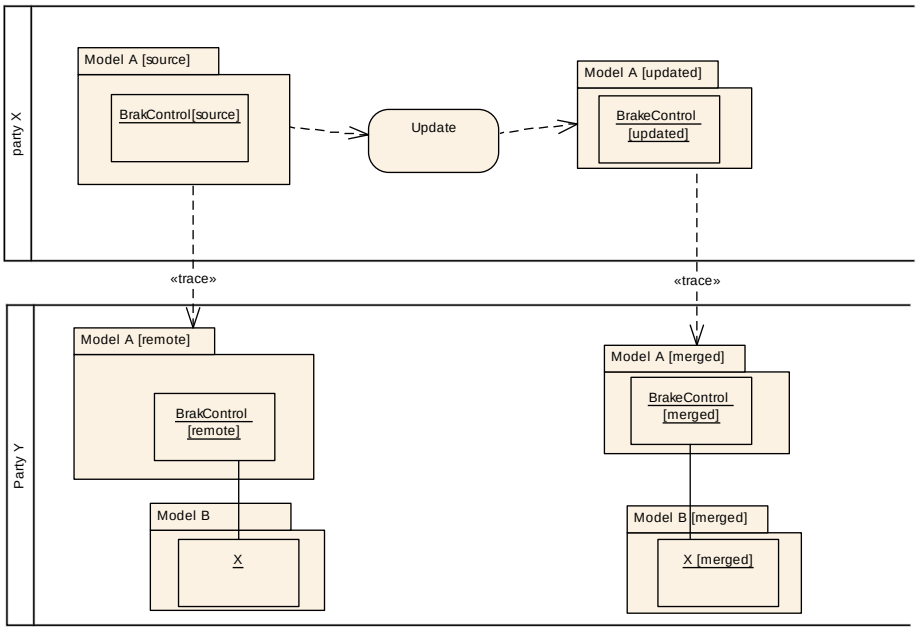

Figure 3.2: Concurrent modeling - updating of elements

3.4.3 Moving of elements from one namespace to another

If an element is moved from one AUTOSAR name space to another, this is basically the same as a rename, since model elements are identified by their fully qualified name which is the concatenation of all `shortName`s up to the root of the model.

This scenario is basically similar to the scenarios described in sections 3.4.1 and 3.4.2.

3.4.4 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.

3.5 Direct exchange of AUTOSAR model in tool-chain

[UC_IOAT_00006] Support for direct exchange of AUTOSAR models in a tool-chain ⌈ This use case describes how information could be exchanged between au-

thoring tools. In this use case each tool exports the AUTOSAR model to an XML description which is then directly 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.

This exchange is not limited to file exchange but can also be performed using cut and paste. Regardless of the physical level, AUTOSAR XML description is the only standardized exchange format for model elements.

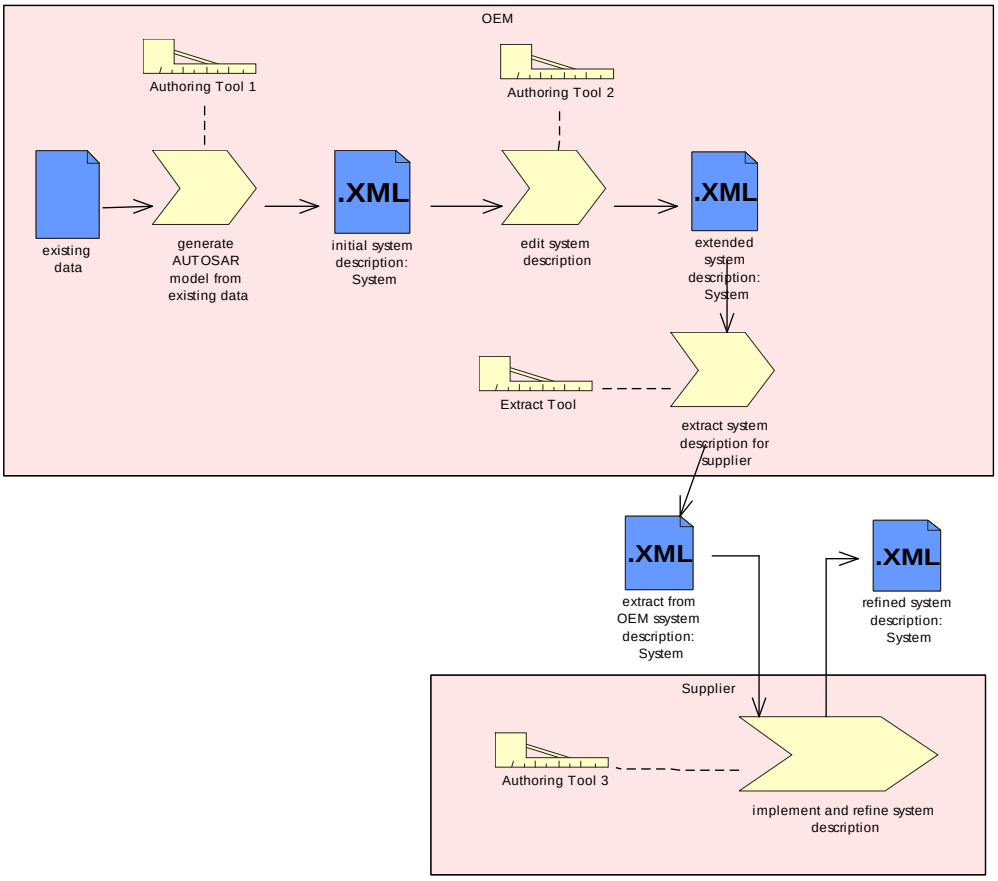

Figure 3.3: Tool Chain

3.6 Shipment of AUTOSAR models and related artifacts

[UC_IOAT_00008] An AUTOSAR model and related artifacts are shipped from one party to another. ⌈ If two parties exchange an AUTOSAR model, the receiver needs to know the AUTOSAR model that can be shipped via several files has been received correctly.

As an example, 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 meta-data 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 work flow describes how an AUTOSAR model could be handled, if additional meta information for data exchange is available:

- 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 AUTOSAR model is very likely split into sub-models. The relevant artifacts and roles of the sub-model needs to be specified. This needs to be mutually agreed between the involved parties.
- In a collaborative exchange scenario, the sender also wants to submit version information and information about new / deleted artifacts.
- The sender might also want to add some meta-data that describes which parts of the AUTOSAR model may be changed and which are not allowed to be changed.

Workflow at the OEM's site In this case an OEM gathers a collection of model elements to be submitted to a supplier. The meta-data 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 `AtomicSwComponentTypes`. The implementation of the behavior has an impact on the `Implementation` description of `AtomicSwComponentTypes`. Therefore, the version of the `Implementation` must be changed. It is not subject of Interoperability to determine how and by whom the version is changed.

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.

3.7 Filter and merge AUTOSAR models

[UC_IOAT_00009] Filter and merge AUTOSAR models ⌈ 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.

The possible subsets are limited to the application of ≪`atpSplitable`≫.

If the model contains variants, it may not be possible that all variants can be bound before the merging, depending on the binding time. Hence, an AUTOSAR tool needs to be aware that the model that has been modified by the supplier contains variants that need to be bound at a later time. ⌋*()*

3.8 Handling of identical double definitions

[UC_IOAT_00010] Handling of identical double definitions ⌈ When working on one particular component, there are `ARElements` which need to be known but do not directly belong to the component. In this case the deliverable of the component development step may contain these objects such that it is "self contained".

By this, the component also documents how it was built. But in the integration step, this leads to duplicate elements which in fact are no duplicates. ⌋*()*

When integrating such self contained components, these definitions may appear in the deliverable of all these components. As long as they are identical this is not a problem per se. But it violates the constraint that only the `atpSplitkey`s and their containers may be repeated in the different partial models. Nevertheless this use case needs to be handled properly.

This use case relates to `ARElement` such as `PortInterface`, `CompuMethod`, `SwBaseType`, `ApplicationDataType`, `ImplementationDataType`, `Unit`, `PhysicalDimension`, `DataConstr`, `PortPrototypeBlueprint`.

Please refer also to [UC_IOAT_00005].

3.9 Description of Data Exchange Points

The tools and guidelines that are referred in this section are described in chapter Glossary. The following actors of use cases are referenced in this section:

Autosar Specification Author An engineer that authors AUTOSAR standard specifications. E.g.: The author of the AUTOSAR Generec Structure Template or AUTOSAR SWS COM.

Profile Author An engineer that authors profiles for data exchange points.

Tool Vendor The vendor of the `AUTOSAR Tool`

Tool Vendor of Producing Tool The vendor of the `AUTOSAR Tool` that produces an `AUTOSAR Model`.

Tool Vendor of Consuming Tool The vendor of the `AUTOSAR Tool` that consumes an `AUTOSAR Model`.

Data Exchange Point Harmonization Group A group of engineers and managers that is responsible for the harmonization of `Data Exchange Point`s.

Profile Analyzer An engineer that analyzes profiles for data exchange points. He analyzes e.g. the consistency and completeness of a single profile or checks multiple profiles for potential incompatibilities.

Group of Tool Vendors and Users A group that discusses interoperability issues and tries to jointly identify fixes.

Producer of AUTOSAR Model (M1) An user that produces an `AUTOSAR Model` (M1) using an `AUTOSAR Tool`.

Consumer of AUTOSAR Model (M1) An user that consumes an `AUTOSAR Model` (M1) using an `AUTOSAR Tool`.

3.9.1 Support for Detection of Incompatibilities

Goal: Get the tool chain running.

In early phases of a project it is often not yet possible to validate the interoperability between partners and tools by providing complete AUTOSAR models, which make use of all relevant features. The profile approach SHALL help identifying potential interoperability issues and responsibilities in early phases of the project. E.g. the profile approach could provide a checklist that can be filled in by the partners in order to describe the provided and expected data.

This section describes use cases that illustrate how to identify incompatibilities between tools or tools and reference profiles:

- [UC_IOAT_00011] Support for Detection of Incompatibilities between Tools
- [UC_IOAT_00012] Support for Detection of Incompatibilities between a Tool and a Reference
- [UC_IOAT_00013] Identify Incompatibilities of Profiles (sub use case that is invoked by the aforementioned use cases)

[UC_IOAT_00011] Support for Detection of Incompatibilities between Tools ⌈

Description Find potential interoperability issues between tools and organizations by checking compatibility of their data exchange points. This is possible even before the final AUTOSAR model is available.

Post Conditions Risk of interoperability issues reduced. Differences in profiles of producing and consuming tools have been identified, agreed-upon plan exists on how to deal with the differences.

Actors `Tool Vendor of Consuming Tool`, `Tool Vendor of Producing Tool`, `Group of Tool Vendors and Users`

Guidelines `Profile Authoring Tool`, `Profile Compatibility Checker Tool`

Basic Flow

1. Describe the profile that defines the requirements on the Autosar-Model of the consuming tool.
   Tools / Guidelines: `Profile Authoring Tool`
2. Describe the profile that defines the assurance with regards to the `Autosar Model` of the producing tool.
   Tools / Guidelines: `Profile Authoring Tool`
3. Identify incompatibilities and unspecified aspects.
   Invokes [UC_IOAT_00013]
   Tools / Guidelines: `Profile Compatibility Checker Tool`
4. Analyze and discuss differences

5. Fix incompatibilities

⌋(*RS_Main_00301*)

[UC_IOAT_00012] Support for Detection of Incompatibilities between a Tool and a Reference Profile ⌈

Description Reduce risk of interoperability issues by identification of incompatibilities of tool with a harmonized reference profile.

Tool vendors can focus on the implementation of the harmonized aspects in order to ensure applicability in many tool chains. If multiple tools are compatible with a single reference profile, then it is more likely that they can successfully exchange data later on.

The reference profile can be created within the scope of the AUTOSAR standardization, a group of tool vendors and users or even within the scope of an individual project.

Post Conditions Risk of interoperability issues reduced by increased support for harmonized exchange points. Conformance of a tool w.r.t. a reference profile has been evaluated as OK or as failed. Incompatibilities have been identified and a decision has been made on where to adapt the tool and on where to adapt the reference profile.

Actors `Data Exchange Point Harmonization Group, Tool Vendor of Producing Tool, Tool Vendor of Consuming Tool`

Guidelines `Profile Compatibility Checker Tool`

Basic Flow
1. Identify and agree on suitable reference profile by GroupToolVendorsAndUsers
2. Describe the data that can be produced or consumed by the tool. Invokes [UC_IOAT_00030]
3. Identify incompatibilities and unspecified aspects. Invokes [UC_IOAT_00013]
4. Analyze and Discuss Differences
5. Fix incompatibilities

⌋(*RS_Main_00301*)

[UC_IOAT_00013] Identify Incompatibilities of Profiles ⌈

Description Identify the incompatibilities between two profiles. Start comparison on simple, high-level aspects of the profiles in order to find the conceptual incompatibilities first. If the profiles are incompatible on this level, then it is not required to spend effort for analysis of the details. If no severe incompatibilities are identified on higher levels, then a more detailed analysis can follow.

It is assumed that the identification of incompatibilities involves manual as well as automated steps.

As a side effect, this use case identifies parts of the profiles which are not comparable since essential information is missing and should be added by a ProfileAuthor.

E.g. Compare the step(s) in the methodology and the intended use and continue with required meta-model coverage, etc.

Post Conditions Differences in profiles have been identified and parts of the profiles are identified which are not comparable due to missing information.

Actors `Profile Author`

Guidelines `Profile Compatibility Checker Tool`

Basic Flow 1. Load the profiles

2. The `Profile Compatibility Checker Tool` matches the various properties of the profiles. E.g. show the matching properties "methodology deliverable", "intended use" and "meta-classes" of the profiles side by side.
3. For formalized properties the `Profile Compatibility Checker Tool` calculates some analysis hints. E.g. it calculates, if the consuming tool requires data that is not produced by the producing tool.
4. The ProfileAnalyzer starts visual inspection of the high-level descriptions.
5. And continues with the details.
6. He potentially request additional information from the tool vendors in order to be able to complete the analysis.

⌋*(RS_Main_00301)*

3.9.2 Validation of Models

Goal: Configure tool according to contract, continuously validate contract.

A profile for a specific data exchange point can describe the completeness and correctness of an AUTOSAR model. This includes the distinction between data that is relevant for the follow-up activities and data that might be in the model and can safely be ignored. E.g.: An RTE generator might require completeness of software components that are mapped to the current ECU while it may ignore incomplete software components that are mapped to other ECUs. The profile could declare a filter based on the used meta-class and additional queries that take the context within the AUTOSAR model into consideration. E.g.: A SenderReceiverInterface is relevant if it is referenced by a Port of a SoftwareComponent that is mapped to a specific ECU.

Additionally: If AUTOSAR allows different modeling patterns for describing the same semantics, the profile should be able select the recommended pattern(s). E.g.: The semantics of a linear compu method can be described using a `CompuMethod` with category "LINEAR" or category "RAT_FUNC". The profile should that only one alternative is allowed. E.g. never use "RAT_FUNC" for linear `CompuMethod`s.

The following use cases are described in this section:

- [UC_IOAT_00014] Validate Compliance of Models with Respect to
- [UC_IOAT_00041] Communicate Requirements on Data to Upstream Tool

[UC_IOAT_00014] Validate Compliance of Models with Respect to Profile ⌈

Description AUTOSAR models are validated against a defined profile using the AutosarValidationTool. This validation can be used to e.g.:

- Check if the produced data conforms to the data exchange point that was agreed between the partners
- Test the data exchange point description of a producing tool by checking the produced AUTOSAR models against the profile.
- Test the data exchange point description of a consuming tool by checking the consumable AUTOSAR models against the profile.
- Test the producing tool by checking the conformance of the produced AUTOSAR models against an agreed profile.
- Test the consuming tool by checking the conformance of the consumable AUTOSAR models against an agreed profile.

Post Conditions List of validation messages which describe violations with respect to profile. For rules that are not implemented by the tool or require visual inspection by an engineer, the AutosarValidationTool lists the locations in the AUTOSASR model that require manual analysis.

Actors `Producer of AUTOSAR Model (M1), Consumer of AUTOSAR Model (M1)`

Guidelines `AUTOSAR Validation Tool`

Basic Flow

1. Read Profile for Data Exchange Point
2. Configure validation rules according to the profile (it is assumed that the actual implementation of the rules is provided by the `AUTOSAR Validation Tool`)
3. Read AUTOSAR model
4. Run validation
5. List violations against profile
6. List locations that require visual inspection by an engineer

⌋(*RS_Main_00301*)

[UC_IOAT_00041] Communicate Requirements on Data to Upstream Tool ⌈

Description Make sure that the upstream tool knows what the downstream tool expects.

Post Conditions The upstream tool knows what is expected or not supported by the downstream tool. The Producer of the Autosar Model (M1) can get guidance so that he knows what data to fill in and which modeling patterns to avoid.

Actors `Producer of Autosar Model (M1)`

Guidelines `AUTOSAR Authoring Tool`

Basic Flow
1. Read agreed Profile for Data Exchange Point that describes the capabilities of the consuming tool.
2. Select / Deselect / Configure validation rules of `AUTOSAR Authoring Tool` according to the profile (it is assumed that the actual implementation of the rules is provided by the validation engine of the tool)
3. (optional) Configure UI in order to avoid not allowed patterns
4. Read AUTOSAR model
5. Run validation
6. List violations against the profile
7. List locations that require visual inspection by an engineer (e.g. because a rule was not implemented or it is not formalized)

⌋(*RS_Main_00301*)

3.9.3 Machine Readable Language

The properties of the language for describing Data Exchange Points are related to aspects of AUTOSAR descriptions that depend on the step in the methodology.

The following use case is described in this section:

- [UC_IOAT_00030] Describe Data Exchange Point

[UC_IOAT_00030] Describe Data Exchange Point ⌈

Description Machine readable language that allows the description of a data exchange point. This language essentially selects a subset of the AUTOSAR templates for an intended use.

Post Conditions The description of a Data Exchange Point is available

Actors `Profile Author`

Guidelines `Profile Authoring Tool`

Basic Flow

1. Describe Overview and Admin Data
   (e.g. tool, in/out, referenced assets, referenced AUTOSAR revision(s))
2. Describe Artifact in the Methodology
   (e.g. reference to artifact "ECU System Description")
3. Describe Intended Use
   (e.g. informally describe as free text "Create an ECU Extract and configure the CAN Communication Stack for unsegmented sending and receiving")
4. Describe relevant subset of the meta-model
   (e.g. list of meta-class and meta-attribute names that might be relevant for the intended use)
5. Describe relevant constraints
   (e.g. list of relevant constraint IDs)
6. Describe relevant spec items
   (e.g. list of relevant spec item IDs)
7. Describe relevant subset of the model
   (e.g. identification of root elements and queries that show how to find the relevant parts of the model)
8. Describe excluded patterns
   (e.g. explicitly exclude meta-classes, meta-attributes, enumeration values, CATEGORYs, etc.)
9. Describe completeness
   (e.g. description of min multiplicities or constraints)
10. Describe default values
    (e.g. condition when to apply the AUTOSAR defined default value. "do not apply", "always apply if value is missing", "apply on revision update from model that was not able to express the value", ...)
11. Describe responsibilities
    (e.g. via queries that identify the subset of the model the consumer guarantees completeness for)
12. Document rationales of decisions for maintainability and reuse
    (e.g. a set of references to parts of the profile which belong together. The group contains a documentation such as "required for feature a")

⌋(*RS_Main_00301*)

3.9.4 Authoring and Lifecycle

Goals: Simplify the authoring and maintenance of profiles for data exchange points

[UC_IOAT_00030] describes some parameters that can describe a data exchange point on different levels of abstraction. Especially the configuration of a data exchange point from scratch can be very time consuming. The following use cases address the topic of authoring and maintenance:

- [UC_IOAT_00090] Authoring Support for Profile for Data Exchange Point
- [UC_IOAT_00092] Validate Consistency of Profile. Support Quality
- [UC_IOAT_00022] Compose Existing Profiles
- [UC_IOAT_00018] Gradual Refinement / Incomplete Descriptions
- [UC_IOAT_00015] Support for Harmonization of Data Exchange Points
- [UC_IOAT_00100] Cherry-Picking in newer AUTOSAR revisions

[UC_IOAT_00090] Authoring Support for Profile for Data Exchange Point ⌈

Description Support the authoring of profiles for data exchange points by e.g.:

- Providing a list of referable constraints, names of meta-classes or meta-attributes, etc. per AUTOSAR Revision
- Providing guidance on how derive hints for parts of a profile description from existing .arxml test or example files (e.g. calculation of used meta classes and meta-attributes)
- Providing Reusable assets (`Blueprint`s) from which a profile can be composed or derived.
- Providing hints on relevant constraints or meta-classes by leveraging cross-references (TechnicalTerms) in descriptions of constraints.

Post Conditions Effort for creation of profiles reduced. Risk of incorrect descriptions reduced by reuse

Actors `Profile Author`

Guidelines `Profile Authoring Tool`, `Profile Authoring Support Data`

Basic Flow

1. Load `Profile Authoring Support Data` using `Profile Authoring Tool`
2. Select the AUTOSAR revision which the profile is relates to
3. Search for existing assets that can be reused using `Profile Authoring Support Data`
4. Customize blueprints as needed
5. Compose assets to an overall profile.
   Invokes [UC_IOAT_00022]
6. Add additional description of the profile.
   Invokes [UC_IOAT_00002]

7. Validate the profile (e.g. check if references to meta-classes are valid in the configured AUTOSAR revision).
   Invokes [UC_IOAT_00092] Validate Consistency of Profile. Support Quality Assurance

⌋(*RS_Main_00301*)

[UC_IOAT_00092] Validate Consistency of Profile. Support Quality Assurance ⌈

Description Ensure the quality of profile by running consistency checks. E. g.:

- Identify dangling references to not existing meta-classes, constraints
- Identify conflicting statements like "I need this meta-class" versus "this meta-class is not supported"

Post Conditions Inconsistencies in the profile are identified. Locations that require manual review identified.

Actors `Profile Author`, ProfileAnalyzer

Guidelines ProfileConsistencyCheckerTool (can also be a feature of a `Profile Authoring Tool`), `Profile Authoring Support Data`

Basic Flow 1. Load profile

2. Run validation
3. List violations against consistency rules
4. List locations in the profile that might require manual review

⌋(*RS_Main_00301*)

[UC_IOAT_00022] Compose Existing Profiles ⌈

Description Compose two or more existing profiles to a new profile

Post Conditions
- New Composed profile.
- Documentation its ancestry so that some hints are available that help redoing the composition if new versions of existing profiles are released.
- List of locations in the new profile where a composition was not possible or manual interaction is required.

Actors `Profile Author`

Guidelines `Profile Authoring Tool`, `Profile Consistency Checker Tool`, `Profile Authoring Support Data`

Basic Flow 1. Read the profiles

2. Match the content of the profiles and show it side by side
3. For formal content calculate merged content or report conflicts

4. For non formalized content list the locations that require manual interaction
5. Document the ancestry of the content of the new profile. E.g.: Document from which profile what content in the new profile came from. Document locations where manual fixes or decisions were required.
6. Save the new profile
7. Save a list of locations that require manual interaction.

⌋(*RS_Main_00301*)

[UC_IOAT_00018] Gradual Refinement / Incomplete Descriptions ⌈

Description In order to reduce the effort for discussion about interoperability issues and responsibilities, the profile approach SHALL allow the reuse / refinement of already existing profiles and profile blueprints. This reuse SHALL be enabled for `Profile Author`s that create profiles and blueprints for the AUTOSAR standard as well as for `Profile Author`s that refine the AUTOSAR provided profiles and blueprints for real life projects outside of AUTOSAR.

Thus, the tooling and Authoring support data that is used by AUTOSAR Specification Authors should also be accessible in these real life projects.

E.g. AUTOSAR could standardize some profiles that describe (at least partially) the data that is expected at some selected data exchange points. These harmonized and standardized profiles could be refined gradually by other organizations and the actual development projects.

Post Conditions Refined profile. Initial Tooling Prototype and `Profile Authoring Support Data` available in AUTOSAR and in Real Life projects outside of AUTOSAR

Actors `Profile Author`

Guidelines `Profile Authoring Tool`, `Profile Authoring Support Data`

Basic Flow 1. Load existing profile blueprint

2. Copy content
3. Document that the profile is derived from a specific blueprint
4. Tailor the new profile
5. Document the changes
6. Save the new refined profile

⌋(*RS_Main_00301*)

[UC_IOAT_00015] Support for Harmonization of Data Exchange Points ⌈

Description Reduce the risk of tool interoperability issues by providing support for the harmonization of data exchange points. E.g. by

- providing harmonized and standardized blueprints that can be reused and tailored to assemble the overall profile. These blueprints could e.g. document the meta-classes and meta-attributes that are required to be configured for a new AUTOSAR concept.
- providing harmonized and standardized profiles for dedicated data exchange points. Tool vendors could first focus on the common features required by the profile.
- marking locations in the existing template specifications where multiple modeling patterns can be used to describe the same semantics.

Post Conditions Harmonization of Data Exchange Points is supported. Effort for creation of profiles reduced.

Actors `AUTOSAR Specification Author`, `Profile Author`

Guidelines `Profile Authoring Tool`, `Profile Authoring Support Data`

Basic Flow 1. Describe blueprints of partial profiles that e.g. show which meta-classes and meta-attributes are involved in order to configure a given feature (by `AUTOSAR Specification Author`)

2. Compose profile from blueprints (by `Profile Author`)

⌋(*RS_Main_00301*)

[UC_IOAT_00100] Cherry-Picking in newer AUTOSAR revisions ⌈

Description In real life projects the customers often request compatibility of the exchanged AUTOSAR models with a specific (old) AUTOSAR revision. (E.g. because the selected tool chain is known to work best with e.g. AUTOSAR 4.0.3 models)

However, some innovations require features or bug fixes that are only available in newer AUTOSAR revisions or are not yet standardized at all. This cherry-picking scenario requires means for exchanging data that is required for configuring the new or fixed functionality using the old data model.

The profile describes how to configure the new features using the AUTOSAR extension mechanisms SDGs and custom CATEGORYs.

Post Conditions A profile that describes the semantics of the custom CATEGORYs and the "Schema" of the SDGs.

Actors `Profile Author`

Guidelines `Profile Authoring Tool`

Basic Flow 1. Create new or open existing profile

2. Describe applicablility of new custom CATEGORY

3. Describe semantics and potentially additional constraints for new CATEGORY
4. Describe "Schema" and applicablility of SDGs
5. Describe semantics and potentially additional constraints for the SDGs
6. Save new or modified profile.

⌋(*RS_Main_00301*)

4 Requirements

This chapter provides a definition of the relevant requirements.

[RS_IOAT_00001] Support data exchange ⌈

Type:	valid
Description:	AUTOSAR SHALL define requirements on AUTOSAR tools AND requirements on the data exchange format which allow for seamless exchange of data between different AUTOSAR tools. The concept SHALL allow for exchanging of AUTOSAR models even if the AUTOSAR tools do not support all features defined in the AUTOSAR metamodel or methodology.
Rationale:	Within the AUTOSAR methodology AUTOSAR models will be exchanged between different parties. Each party could use different AUTOSAR tools which best fit to the step in the methodology. In order to facilitate seamless exchange of AUTOSAR models, a standardized AUTOSAR data exchange format is required. Additionally further requirements need to be defined on the AUTOSAR tools in order to keep AUTOSAR models consistent.
Dependencies:	–
Use Case:	–
Supporting Material:	–

⌋(*RS_Main_00300*)

[RS_IOAT_00002] Standardize the handling of errors in AUTOSAR models ⌈

Type:	valid
Description:	AUTOSAR SHALL provide a concept for a standardized mechanism for handling errors in AUTOSAR models. This concept SHALL not only be implemented by all AUTOSAR tools which interpret, modify or create AUTOSAR models.
Rationale:	Without a standard collection of possible errors, each tool would have its own sets, but the difference between these could cause relations created by one tool in a tool-chain be reported later as fatal errors by another tool.
Dependencies:	–
Use Case:	–
Supporting Material:	–

⌋(*RS_Main_00300*)

[RS_IOAT_00003] Provide naming conventions ⌈

Type:	valid
Description:	The TR_IAOT shall provide naming conventions. This especially includes requirement ids, module abbreviations, meta data and configuration symbols used in the document of a release and in AUTOSAR models
Rationale:	Avoid ambiguities and name clashes inside the specification and AUTOSAR models. Provide a consistent uniform presentation of meta data to the reader of the specification. Allow automatic processing of specification elements. Improve Interoperability between AUTOSAR tools.
Dependencies:	–
Use Case:	–
Supporting Material:	–

⌋(*RS_BRF_01028*)

[RS_IOAT_00004] Standardized Authoring Support Data ⌈

Type:	valid
Description:	The Interoperability of AUTOSAR Tools Supplement [9] SHALL provide Profile Authoring Support Data such as lists of referable constraints, spec items, requirements, meta-classes, meta-attributes, etc.
Rationale:	Leverage existing information and make it easily accessible by profile authoring tools.
Dependencies:	–
Use Case:	[UC_IOAT_00015], [UC_IOAT_00030], [UC_IOAT_00090], [UC_IOAT_00092], [UC_IOAT_00018]
Supporting Material:	–

⌋(*RS_Main_00301*)

[RS_IOAT_00007] AUTOSAR Example Data Exchange Point Description ⌈

Type:	valid
Description:	The Interoperability of AUTOSAR Tools Supplement [9] SHALL provide an example profile that illustrates the use of the profile language.
Rationale:	Learn how to describe a profile by analyzing or extending an existing profile.
Dependencies:	–
Use Case:	[UC_IOAT_00011], [UC_IOAT_00012], [UC_IOAT_00013], [UC_IOAT_00014], [UC_IOAT_00015], [UC_IOAT_00018], [UC_IOAT_00100], [UC_IOAT_00022], [UC_IOAT_00030], [UC_IOAT_00041], [UC_IOAT_00090], [UC_IOAT_00092]
Supporting Material:	–

⌋(*RS_Main_00301*)

[RS_IOAT_00008] AUTOSAR Baseline of Data Exchange Point ⌈

Type:	valid
Description:	The Interoperability of AUTOSAR Tools Supplement [9] SHALL provide a baseline of data exchange point that can be used as a starting point for further detailing.
Rationale:	AUTOSAR already provides some valuable information that can be used as a starting point for a profile. Making this information available as a profile will most likely reduce the effort for the creation of customized profiles. E.g.: Information such as the lower multiplicities can be reused
Dependencies:	–
Use Case:	[UC_IOAT_00015], [UC_IOAT_00018], [UC_IOAT_00022], [UC_IOAT_00090]
Supporting Material:	–

⌋(*RS_Main_00301*)

A Glossary

Artifact This is a Work Product Definition that provides a description and definition for tangible work product types. Artifacts may be composed of other artifacts ([10]).

At a high level, an artifact is represented as a single conceptual file.

AUTOSAR Tool This is a software tool which supports one or more tasks defined as AUTOSAR tasks in the methodology. Depending on the supported tasks, an AUTOSAR tool can act as an authoring tool, a converter tool, a processor tool or as a combination of those (see separate definitions).

AUTOSAR Authoring Tool An AUTOSAR Tool used to create and modify AUTOSAR XML Descriptions. Example: System Description Editor.

AUTOSAR Converter Tool An AUTOSAR Tool used to create AUTOSAR XML files by converting information from other AUTOSAR XML files. Example: ECU Flattener

AUTOSAR Definition This is the definition of parameters which can have values. One could say that the parameter values are Instances of the definitions. But in the meta model hierarchy of AUTOSAR, definitions are also instances of the meta model and therefore considered as a description. Examples for AUTOSAR definitions are: `EcucParameterDef`, `PostBuildVariantCriterion`, `SwSystemconst`.

AUTOSAR XML Description In AUTOSAR this means "filled Template". In fact an AUTOSAR XML description is the XML representation of an AUTOSAR model.

The AUTOSAR XML description can consist of several files. Each individual file represents an AUTOSAR partial model and shall validate successfully against the AUTOSAR XML schema.

AUTOSAR Meta-Model This is an UML2.0 model that defines the language for describing AUTOSAR systems. The AUTOSAR meta-model is an UML representation of the AUTOSAR templates. UML2.0 class diagrams are used to describe the attributes and their interrelationships. Stereotypes, UML tags and OCL expressions (object constraint language) are used for defining specific semantics and constraints.

AUTOSAR Meta-Model Tool The AUTOSAR Meta-Model Tool is the tool that generates different views (class tables, list of constraints, diagrams, XML Schema etc.) on the AUTOSAR meta-model.

AUTOSAR Model This is a representation of an AUTOSAR product. The AUTOSAR model represents aspects suitable to the intended use according to the AUTOSAR methodology.

Strictly speaking, this is an instance of the AUTOSAR meta-model. The information contained in the AUTOSAR model can be anything that is representable according to the AUTOSAR meta-model.

AUTOSAR Partial Model In AUTOSAR, the possible partitioning of models is marked in the meta-model by ≪atpSplitable≫. One partial model is represented in an AUTOSAR XML description by one file. The partial model does not need to fulfill all semantic constraints applicable to an AUTOSAR model.

AUTOSAR Processor Tool An AUTOSAR Tool used to create non-AUTOSAR files by processing information from AUTOSAR XML files. Example: RTE Generator

AUTOSAR Specification Element An AUTOSAR Specification Element is a named element that is part of an AUTOSAR specification. Examples: requirement, constraint, specification item, class or attribute in the meta model, methodology, deliverable, methodology activity, model element, bsw module etc.

AUTOSAR Template The term "Template" is used in AUTOSAR to describe the format different kinds of descriptions. The term template comes from the idea, that AUTOSAR defines a kind of form which shall be filled out in order to describe a model. The filled form is then called the description.

In fact the AUTOSAR templates are now defined as a meta-model.

AUTOSAR Validation Tool A specialized AUTOSAR Tool which is able to check an AUTOSAR model against the rules defined by a profile.

AUTOSAR XML Schema This is a W3C XML schema that defines the language for exchanging AUTOSAR models. This Schema is derived from the AUTOSAR meta-model. The AUTOSAR XML Schema defines the AUTOSAR data exchange format.

Blueprint This is a model from which other models can be derived by copy and refinement. Note that in contrast to meta model resp. types, this process is *not* an instantiation.

Instance Generally this is a particular exemplar of a model or of a type.

Life Cycle Life Cycle is the course of development/evolutionary stages of a model element during its life time.

Meta-Model This defines the building blocks of a model. In that sense, a Meta-Model represents the language for building models.

Meta-Data This includes pertinent information about data, including information about the authorship, versioning, access-rights, timestamps etc.

Model A Model is an simplified representation of reality. The model represents the aspects suitable for an intended purpose.

Partial Model This is a part of a model which is intended to be persisted in one particular artifact.

Pattern in GST : This is an approach to simplify the definition of the meta model by applying a model transformation. This transformation creates an enhanced model out of an annotated model.

Profile Authoring Support Data Data that is used for efficient authoring of a profile. E.g. list of referable constraints, meta-classes, meta-attributes or other reusable model assets (blueprints)

Profile Authoring Tool A specialized `AUTOSAR Tool` which focuses on the authoring of profiles for data exchange points. It e.g. provides support for the creation of profiles from scratch, modification of existing profiles or composition of existing profiles.

Profile Compatibility Checker Tool A specialized `AUTOSAR Tool` which focuses on checking the compatibility of profiles for data exchange. Note that this compatibility check includes manual compatibility checks by engineers and automated assistance using more formal algorithms.

Profile Consistency Checker Tool A specialized `AUTOSAR Tool` which focuses on checking the consistency of profiles.

Property A property is a structural feature of an object. As an example a “connector” has the properties “receive port” and “send port”

Properties are made variant by the ≪`atpVariation`≫.

Prototype This is the implementation of a role of a type within the definition of another type. In other words a type may contain Prototypes that in turn are typed by "Types". Each one of these prototypes becomes an instance when this type is instantiated.

Type A type provides features that can appear in various roles of this type.

Value This is a particular value assigned to a “Definition”.

Variability Variability of a system is its quality to describe a set of variants. These variants are characterized by variant specific property settings and / or selections. As an example, such a system property selection manifests itself in a particular “receive port” for a connection.

This is implemented using the ≪`atpVariation`≫.

Variant A system variant is a concrete realization of a system, so that all its properties have been set respectively selected. The software system has no variability anymore with respect to the binding time.

This is implemented using `EvaluatedVariantSet`.

Variation Binding A variant is the result of a variation binding process that resolves the variability of the system by assigning particular values/selections to all the system’s properties.

This is implemented by `VariationPoint`.

Variation Binding Time The variation binding time determines the step in the methodology at which the variability given by a set of variable properties is resolved.

This is implemented by `vh.LatestBindingtime` at the related properties .

Variation Definition Time The variation definition time determines the step in the methodology at which the variation points are defined.

Variation Point A variation point indicates that a property is subject to variation. Furthermore, it is associated with a condition and a binding time which define the system context for the selection / setting of a concrete variant.

This is implemented by `VariationPoint`.