Document Title	Requirements on Standardization Template
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	536
Document Classification	Auxiliary

Document Version	1.0.0
Document Status	Final
Part of Release	4.0
Revision	3

Document Change History			
Date	**Version**	**Changed by**	**Description**
30.10.2011	1.0.0	AUTOSAR Administration	Initial Release

Disclaimer

Advice for users

AUTOSAR specifications 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 specifications for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such specifications, 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

References

[1] Software Component Template
AUTOSAR_TPS_SoftwareComponentTemplate.pdf

[2] Key words for use in RFCs to Indicate Requirement Levels
http://www.ietf.org/rfc/rfc2119.txt

[3] Specification of ECU Configuration
AUTOSAR_TPS_ECUConfiguration.pdf

[4] Specification of Platform Types
AUTOSAR_SWS_PlatformTypes.pdf

[5] Generic Structure Template
AUTOSAR_TPS_GenericStructureTemplate.pdf

[6] Table of Application Interfaces
AUTOSAR_MOD_AITable.pdf

[7] Standardization Template
AUTOSAR_TPS_StandardizationTemplate.pdf

[8] SW-C and System Modeling Guide
AUTOSAR_TR_SWCModelingGuide.pdf

1 Introduction

AUTOSAR models are in many cases not created from scratch but existing content is taken as the basis. The existing content could be contributed by the AUTOSAR initiative itself in form of standardized model elements.

This document specifies the requirements for the Standardization Template. This template is intended to support the delivery of standardized model elements by AUTOSAR.

AUTOSAR 4.0 already specifies the blueprint approach for standardization. This approach is continued and refined by the Standardization Template. It thereby will replace Appendix A in Software Component Template ([1]).

As an particular example, let us consider the standardization of application interfaces. That is, in terms of the AUTOSAR meta-model the standardization mainly applies to the definition of `PortPrototype`s for specific purposes.

Due to the structure of the AUTOSAR meta-model it is not possible to merely express a standardized `PortPrototype` because for good reasons the latter does not exist on its own but is always owned by a `SwComponentType`.

The Standardization Template specifies the approach to overcome this situation.

1.1 Conventions used in this Document

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

Initiator:	Initiator (e.g. WP Methodology and Configuration)
Date:	Date of last change
Requirement:	Short description (same as above)
Description:	Detailed description
Rationale:	Why is this requirement important, what its omission could cause?
Use Case:	A scenario that makes the requirement necessary or useful
Dependencies:	References to other requirements which this requirement depends on
Conflicts:	References to other requirements which this requirement is in conflict with
Supporting Material:	References to other documents, models etc.
Comment:	Comments

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 follows, based on [2].

Note that the requirement level of the document in which they are used modifies the force of these words.

- MUST: This word, or the adjective "LEGALLY REQUIRED", means that the definition is an absolute requirement of the specification due to legal issues.
- MUST NOT: This phrase, or the phrase "MUST NOT", means that the definition is an absolute prohibition of the specification due to legal issues.
- SHALL: This phrase, or the adjective "REQUIRED", means that the definition is an absolute requirement of the specification.
- SHALL NOT: This phrase means that the definition is an absolute prohibition of the specification.
- SHOULD: This word, or the adjective "RECOMMENDED", means 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", means 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, SHALL 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, SHALL be prepared to interoperate with another implementation, which does not include the option (except, of course, for the feature the option provides.)

1.2 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.

1.3 Use Case Tracing

Following table references the use cases specified in and links to the related requirements.

Use Case	Description	Satisfied by
[UC_STDT_0001]	Support Application Interfaces	[RS_STDT_0001] [RS_STDT_0003] [RS_STDT_0005] [RS_STDT_0006] [RS_STDT_0007] [RS_STDT_0016] [RS_STDT_0019] [RS_STDT_0020] [RS_STDT_0021] [RS_STDT_0022] [RS_STDT_0026]
[UC_STDT_0002]	Express Parts of SWS	[RS_STDT_0001] [RS_STDT_0002] [RS_STDT_0018]
[UC_STDT_0003]	Standardize ECUCParamdefs	[RS_STDT_0001] [RS_STDT_0010] [RS_STDT_0029]
[UC_STDT_0004]	Express predefined Paths	[RS_STDT_0001] [RS_STDT_0013] [RS_STDT_0030]
[UC_STDT_0005]	Express PlatformTypes	[RS_STDT_0001]
[UC_STDT_0006]	Express Examples of applied Standards	[RS_STDT_0001]
[UC_STDT_0007]	Support Verification if an implementation adheres to defined Standard	[RS_STDT_0001] [RS_STDT_0008] [RS_STDT_0009] [RS_STDT_0015] [RS_STDT_0017]
[UC_STDT_0008]	Support reusable Documentation	[RS_STDT_0001] [RS_STDT_0002] [RS_STDT_0003] [RS_STDT_0023] [RS_STDT_0026]
[UC_STDT_0009]	Define name conventions	[RS_STDT_0001] [RS_STDT_0004] [RS_STDT_0014] [RS_STDT_0023] [RS_STDT_0024] [RS_STDT_0025]
[UC_STDT_0010]	Perform Standardization on Levels beyond the AUTOSAR Scope	[RS_STDT_0011] [RS_STDT_0012] [RS_STDT_0024] [RS_STDT_0025]
[UC_STDT_0011]	Derive Objects from Blueprints by manually changing properties	[RS_STDT_0015] [RS_STDT_0029]
[UC_STDT_0012]	Derive Objects from Blueprints in a completly standardized Way	[RS_STDT_0015] [RS_STDT_0029]
[UC_STDT_0013]	Integrate compile test	[RS_STDT_0027]
[UC_STDT_0014]	Generate BSW ”Standard AUTOSAR Interface” description from model	[RS_STDT_0028]

2 Use Cases

This chapter describes use-cases for the Standardization Template. The intention of these uses cases is to point out the potential applications of the Standardization Template. In general, the use case of the Standardization Template is to express items standardized by AUTOSAR as AUTOSAR XML artifact. This artifact can subsequently be used to support the development of AUTOSAR compliant products.

Each use-case defined in this document has its unique identifier starting with the prefix "UC-STDT-" (meaning Standardization Template Use Case).

2.1 Support of Application Interfaces

[UC_STDT_0001] Support Application Interfaces ⌈ AUTOSAR : provides standardized, openly disclosed interfaces for different domains such as chassis, powertrain, body etc. The definition of these interfaces can be handled on various levels of deepness:

L8 Complete description of the SW-Cs including behavioral model and ports

L7 Complete description of all ports including interface behavior and data qualities

L6 Complete description of all ports including textual description of interface behavior and the data qualities of the interfaces

L5 Complete description of all ports including data qualities of the interfaces

L4 Complete description of all ports including within-AUTOSAR-agreed data qualities

L3 Partial description of ports/interfaces of a SWC including within-AUTOSAR-agreed data qualities

Note that this partial description includes the fact that only some of the ports are described, as well as the fact that this description of a port is incomplete and is also separated from the applicable component. This is also known as PortBluePrint.

L2 Dictionary of interfaces including a set of within-AUTOSAR-agreed data qualities

L1 Dictionary of data elements including types and ranges.

L0 Dictionary of names

As of Release 4.0, AUTOSAR standardization covers the level L0 ... L3. Nevertheless vendor internal applications might use Standardization Template for higher levels too.

This use case mainly covers the application software aspects.

Applying this formal description will improve consistency and usability of the AUTOSAR Application Interfaces and empower formal checks e.g. of backward compatibility of application interfaces. ⌋

2.2 Express parts of SWS

[UC_STDT_0002] Express Parts of SWS ⌈ The Standardization Template shall allow to express parts of SWS for basic software modules formally using the AUTOSAR schema. This includes (but is not restricted to):

- Standardized interfaces (i.e. C-APIs)
- Standardized AUTOSAR Interfaces (Ports, PortInterfaces, ...)
- Definition of ECUC-Parameters (see [UC_STDT_0003])

Applying this formal description will improve consistency and usability of the AUTOSAR SWS and empower formal checks e.g. of backward compatibility of interfaces. ⌋

2.3 Standardize ECUCParamdefs

[UC_STDT_0003] Standardize ECUCParamdefs ⌈ Part of the AUTOSAR SWS is also the set of ECU configuration parameter definitions. These parameter definitions are the basis of the so called vendor specific parameters which are used in particular AUTOSAR implementations.

Even if this is specified in great detail in [3] it is also in the scope of Standardization Template. ⌋

2.4 Express predefined Paths

[UC_STDT_0004] Express predefined Paths ⌈ Development partners may mutually agree on a particular package layout and thus share AUTOSAR artifacts in a later phase of the development. For this use case it is helpful to initially express a set of predefined resp. partly predefined reference paths respective `referenceBase` which can be loaded in individual AUTOSAR authoring tools.

This use case covers the beginning or the end of a reference path. For example the usecase is to standardize the substructure after a variable path: `<My-Path>/PortInterfaces`. In this case only `PortInterfaces` is standardized. ⌋

2.5 Express PlatformTypes

[UC_STDT_0005] Express PlatformTypes ⌈ The platform types defined in [4] need to be available in processable format for AUTOSAR development tools. This approach improves consistency and quality of AUTOSAR products.

The details of [5] chapter 3.1 apply. ⌋

2.6 Express Examples of applied Standards

[UC_STDT_0006] Express Examples of applied Standards ⌈ In addition to the application interfaces ([6]) AUTOSAR provides examples how to apply the standardized elements, in particular blueprints.

The details of [5] chapter 3.1 apply. ⌋

2.7 Support Verification if an implementation adheres to defined Standard

[UC_STDT_0007] Support Verification if an implementation adheres to defined Standard ⌈ When an AUTOSAR product is developed an initial verification can be performed by verifying the product against the formalized standard. This includes

- the check of compatibility rules between blueprints and derived model elements. These compatibility rules are to be defined for each meta class eligible to blueprints.
- tracing between model elements and SWS respectively blueprints in order to check if all blueprints were implemented.

Note that this use case is a very initial verification and does not compete or even replace conformance test specification. It rather contributes to conformance test.

The compatibility rules need only to be described in the document, e.g. in form of constraints. There is no formal representation of the compatibility rules in the meta-model.

The compatibility rules are specified individually for each meta-class eligible for blueprinting. For example all port blueprints follow the same compatibility rules. ⌋

2.8 Support reusable Documentation

[UC_STDT_0008] Support reusable Documentation ⌈ Parts of the AUTOSAR SWS may be published such that it can be reused for the actual product documentation. The vendor of an implementation then takes such parts out of an Instance of a Standardization Template and incorporates it in his own software documentation.

The same approach may apply to the explanation of Application Interfaces. ⌋

2.9 Define name conventions

[UC_STDT_0009] Define name conventions ⌈ AUTOSAR has modeling guides and naming conventions. If these conventions are published as instance of the Standardization Template, they can be utilized to configure e.g. modeling tools.

The use case also covers the ability to express various levels of obligation. This may for example be expressed similar to the keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL". ⌋

2.10 Perform Standardization on Levels beyond the AUTOSAR Scope

[UC_STDT_0010] Perform Standardization on Levels beyond the AUTOSAR Scope ⌈ The Standardization Template shall be applicable for company internal standardization respectively for mutual agreements which go beyond the AUTOSAR standardization on meta level[1] M1 (see [UC_STDT_0001]). ⌋

2.11 Derive Objects from Blueprints by manually changing properties

[UC_STDT_0011] Derive Objects from Blueprints by manually changing properties ⌈ The user makes a kind of copy of the blueprint and is allowed to add his own things (e.g. adding own field to a standardized enum-type). Example is the usage of ApplicationInterfaces and ConfigurationParamaters.

In case of Standardized PortInterfaces of AUTOSAR Services the Standardization template shall mention the use case that in this case a PortInteface based on the "Standardized PortInterface Blueprint" might contain a subset of ClientServerOperations. ⌋

2.12 Derive Objects from Blueprints in a completly standardized Way

[UC_STDT_0012] Derive Objects from Blueprints in a completly standardized Way ⌈ The user can only configure or otherwise influence the content of the copied "blueprint" in a completely standardized way, (e.g. configuring the fields of an enum-type according to the "needs" of the software) but he cannot add own things.

[1] For more details of Meta levels see Chapter 2.2 in [5]

This could even go so far, that only the rules of configuration are standardized (like in the case of DCM-PortInterfaces) - but this nonetheless completely determines the outcome in a concrete project. ⌋

2.13 Integrate compile test

[UC_STDT_0013] Integrate compile test ⌈ Until Release 4.0 all APIs of the BSW are modeled and chapter 8 of the SWS is mainly generated out of the model. Additionally we propose to generate empty C functions (and data structures/consts/...) out of the model and link all these functions together. If the compile or link process fails the consistency (e.g. between different SWS) is violated and needs to be fixed. ⌋

2.14 Generate BSW ”Standard AUTOSAR Interface” description from model

[UC_STDT_0014] Generate BSW ”Standard AUTOSAR Interface” description from model ⌈ Until Release 4.0 the "Standard AUTOSAR Interface" is part of each SWS which offers this interface (Typically contained in an own chapter of subchapter of 7 or 8). The description is mostly plain text with some pseudo language to show the usage of the interface (including constants, etc.). Furthermore the description of the services often uses "elements" from the meta-model which are not up-to-date or their meaning has changed.

It is intended to standardize this part of the SWS, e.g. via an own model (and then the generated descriptions can be imported into the SWS like chapter 8) OR via a -standardized- language to clarify the understanding of the interface and allow an automatic conversation for RTE purposes. ⌋

3 Requirements

This chapter describes all requirements driving the work to define the *Standardization Template* specification [7].

Each requirement in this document has its unique identifier starting with the prefix "RS_STDT_" (meaning Requirement Specification for Standardization Template).

3.1 Shall support and explain Blueprints in general

[RS_STDT_0001] Shall support and explain Blueprints in general ⌈

Initiator:	WP "Coordination of Appl. Interfaces"
Date:	2010-04-23
Description:	The standardization template shall support blueprints. Blueprint is a kind of "incomplete" model" which is copied and refined lateron. The principles of blueprints shall be defined: • "Instantiation" is done by copy rather than referenced. Downstream processing excludes the usage of blueprints. • Define proper terminology for Blueprints and blueprinted model elements. • How are the elements named that are created out of blueprints? • Shall clearly define which parts of the meta-model are eligible for blueprinting. Blueprinting non-eligible parts of the meta-model shall count as a "validation error". • define the rules how to derive objects from blueprints, in particular the strategy, which properties may be added / removed / redefined. These rules shall be defined individually for each meta-class eligible for blueprinting. Necessary facilities of the Meta Model shall be defined: • specific blueprints • mapping blueprints and derived objects
Rationale:	This helps to understand the concept and application of blueprints as blueprints are the main mean of standardization.
Dependencies:	–
Conflicts:	–
Supporting Material:	–
Comment:	–

⌋*(UC_STDT_0001, UC_STDT_0002, UC_STDT_0003, UC_STDT_0004, UC_STDT_0005, UC_STDT_0006, UC_STDT_0007, UC_STDT_0008, UC_STDT_0009)*

3.2 Formalized description of BSW SWS

[RS_STDT_0002] Formalized description of BSW SWS ⌈

Initiator:	WP 1.1.1
Date:	2010-04-23
Requirement:	Shall allow to represent formalized parts of BSW SWS
Description:	The standardization Template shall be able to publish formalized parts of a SWS which then acts as a blueprint of the specified Module. The Standardization Template shall provide means to describe standardized Interfaces (C-APIs). The Standardization Template shall allow the description standardized AUTOSAR Interfaces. The Standardization Template must support the specification of variants of the interfaces.
Rationale:	Especially the "Standard AUTOSAR Interface" is part of each SWS which offers this interface (Typically contained in an own chapter of subchapter of 7 or 8). The description is mostly plain text with some pseudo language to show the usage of the interface (including constants, etc.). Furthermore the description of the services often uses "elements" from the meta-model which are not up-to-date or their meaning has changed. The current state causes several problems when the RTE "routes" calls between the BSWs and SWC. The pseudo language must be manually transferred into some sort of "SWC-Description". If people try to mix modules from different vendors it is not clear how this can work. In our understanding the format needs to be standardized. We propose to standardize this part of the SWS, e.g. via an own model (and then the generated descriptions can be imported into the SWS like chapter 8) OR via a -standardized- language to clarify the understanding of the interface and allow an automatic conversation for RTE purposes.
Dependencies:	–
Conflicts:	–
Supporting Material:	–
Comment:	–

⌋*(UC_STDT_0002, UC_STDT_0008)*

3.3 Shall allow to represent port blueprints

[RS_STDT_0003] Shall allow to represent port blueprints ⌈

Initiator:	WP "Coordination of Appl. Interfaces"
Date:	2010-04-23
Requirement:	Shall allow to represent port blueprints.
Description:	AUTOSAR standardizes so called "Application Interfaces". These Interfaces in fact result in port blueprints.
Rationale:	AUTOSAR publishes standardized Models as ARXML.
Dependencies:	–
Conflicts:	–

Supporting Material:	–
Comment:	–

⌋*(UC_STDT_0001, UC_STDT_0008)*

3.4 Shall allow to represent `shortName` patterns

[RS_STDT_0004] Shall allow to represent `shortName` patterns ⌈

Initiator:	WP "Coordination of Appl. Interfaces"
Date:	2010-04-23
Requirement:	Shall allow to represent `shortName` patterns
Description:	–
Rationale:	AUTOSAR publishes the Application Interfaces Modeling guide
Dependencies:	TR_SWCModelingGuide [8] might need to be adapted.
Conflicts:	–
Supporting Material:	–
Comment:	–

⌋*(UC_STDT_0009)*

3.5 Shall support keywords and keyword abbreviations

[RS_STDT_0005] Shall support keywords and keyword abbreviations ⌈

Initiator:	WP "Coordination of Appl. Interfaces"
Date:	2010-04-23
Requirement:	Shall support name parts
Description:	In [8] AUTOSAR publishes building rules for `ShortName` as sequence of Keywords. These keywords need to be expressed by Standardization Template. The existing Keyword identifiable should be extended with a ShortLabel. Semantics of SHORT-LABEL: used instead of ShortName. Necessary in case there are several nampe parts that are to be abbreviated by the same keyword abbreviation. As Keywords are identifiables, this would lead to a conflict.
Rationale:	Support AUTOSAR publication
Dependencies:	TR_SWCModelingGuide [8] might need to be adapted.
Conflicts:	–
Supporting Material:	–
Comment:	–

⌋*(UC_STDT_0001)*

3.6 Shall be implemented without compatibility problems to existing templates

[RS_STDT_0006] Shall be implemented without compatibility problems to existing template ⌈

Initiator:	WP 1.2
Date:	2010-04-23
Requirement:	Shall be implemented without compatibility problems to existing template
Description:	–
Rationale:	Maintenance of backwards compatibility of the Schema as requested for 4.0
Dependencies:	–
Conflicts:	–
Supporting Material:	–
Comment:	–

⌋*(UC_STDT_0001)*

3.7 Shall be based on the AUTOSAR schema

[RS_STDT_0007] Shall be based on the AUTOSAR schema ⌈

Initiator:	WP 1.2
Date:	2010-04-23
Requirement:	Shall be based on the AUTOSAR schema
Description:	–
Rationale:	General approach of having one Meta Model for all templates
Dependencies:	–
Conflicts:	–
Supporting Material:	–
Comment:	–

⌋*(UC_STDT_0001)*

3.8 Shall provide means to support analyzing the conformity of implementations with the AUTOSAR standards

[RS_STDT_0008] Shall provide means to support analyzing the conformity of implementations with the AUTOSAR standards ⌈

Initiator:	WP 1.1.1
Date:	2010-04-23
Requirement:	Shall provide means to support analyzing the conformity of implementations with the AUTOSAR standards

Description:	The Standardization Template should enable the implementer to check if its description (e.g. BSWM) is in conformance with AUTOSAR standards specified on M1 level (SWS).
Rationale:	This establishes traceability between AUTOSAR Implementations and defined standard. And is also a precondition to check the application compatibility between different releases.
Dependencies:	–
Conflicts:	–
Supporting Material:	–
Comment:	–

⌋*(UC_STDT_0007)*

3.9 Shall be able to represent requirements stated in SWS

[RS_STDT_0009] Shall be able to represent requirements stated in SWS ⌈

Initiator:	WP 1.1.1
Date:	2010-04-23
Requirement:	Shall be able to represent requirements stated in SWS.
Description:	To improve requirements traceability a formalized description of a SWS shall contain textual representatives of each requirement contained in the respective SWS document (specification items of SWSs). The statements shall be categorized as one of {Requirement, Specification, Implementation, Constraint}
Rationale:	This feature enables a automated tracking of changes of requirements, which is the basis for a improved change management and compatibility assessment.
Dependencies:	–
Conflicts:	–
Supporting Material:	–
Comment:	–

⌋*(UC_STDT_0007)*

3.10 Shall refer to ECUC parameter definition

[RS_STDT_0010] Shall refer to ECUC parameter definition ⌈

Initiator:	WP 1.1.1
Date:	2010-04-23
Requirement:	Shall refer to ECUC parameter definition

Description:	ECUC parameter definitions are also standardized by AUTOSAR. Therefore it is in the scope of the Standardization Template. Strictly speaking, these standardizes ECUC Parameter Definitions act as Blueprints for the vendor specific parameters, even if these are not mapped using the BluePrintMappingSet. Standardization Template shall not change the approaches at least for AUTOSAR 4.0, but reflect the relationships.
Rationale:	This maintains the overall scope and the applied patterns.
Dependencies:	–
Conflicts:	–
Supporting Material:	[3]
Comment:	–

⌋*(UC_STDT_0003)*

3.11 Shall be able to standardize components

[RS_STDT_0011] Shall be able to standardize components ⌈

Initiator:	Bosch
Date:	2010-04-30
Description:	STDT shall be able to express standardization of components, even if AUTOSAR does not standardize components. This requirement covers a set of individual components. No compatibility rules shall hardwired in STDT. Support is provided by the fact that it only allows to specify SwComponentType as eligible for blueprinting.
Rationale:	This allows to leverage AUTOSAR standardization principles inside a company.
Dependencies:	–
Conflicts:	–
Supporting Material:	–
Comment:	–

⌋*(UC_STDT_0010)*

3.12 Shall be able to standardize architecture

[RS_STDT_0012] Shall be able to standardize architecture ⌈

Initiator:	Bosch
Date:	2010-04-30
Requirement:	STDT shall be able to express standardization of components and communication, even if AUTOSAR does not standardize architecture of application software. This requirement covers a set of communicating components.
Rationale:	This allows to leverage AUTOSAR standardization principles inside a company.

Dependencies:	–
Conflicts:	–
Supporting Material:	–
Comment:	–

⌋(*UC_STDT_0010*)

3.13 Shall be able to express parts of reference paths resp. package hierarchies

[RS_STDT_0013] Shall be able to express parts of reference paths resp. package hierarchies ⌈

Initiator:	Bosch
Date:	2010-04-30
Requirement:	STDT shall be able to express standardized reference paths resp. parts of eference paths. It shall be possible to specify the beginning respectively the end of a package hierarchy. See UC_STDT_004 for an example.
Rationale:	This allows to leverage AUTOSAR standardization principles inside a company.
Dependencies:	–
Conflicts:	–
Supporting Material:	–
Comment:	–

⌋(*UC_STDT_0004*)

3.14 Shall be able to express levels of obligation

[RS_STDT_0014] Shall be able to express levels of obligation ⌈

Initiator:	Bosch
Date:	2010-04-30
Requirement:	The level of obligation shall be expressed by key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL".
Rationale:	This allows to use Standardization Models to evaluate conformity of an implementation
Dependencies:	–
Conflicts:	–
Supporting Material:	–
Comment:	–

⌋(*UC_STDT_0009*)

3.15 Shall support different Approaches to derive from Blueprints

[RS_STDT_0015] Shall support different Approaches to derive from Blueprints ⌈

Initiator:	Bosch
Date:	2010-05-12
Requirement:	Shall support different Approaches to derive from Blueprints. Such different approaches are 1. The user makes a kind of copy of the blueprint and is allowed to add his own things (e.g. adding own field to a standardized enum-type). 2. The user can only configure or otherwise influence the content of the copied "blueprint" in a completely standardized way, (e.g. configuring the fields of an enum-type according to the "needs" of the software) but he cannot add own things. 3. In case of Standardized `PortInterface`s of AUTOSAR Services the Standardization template shall mention the use case that in this case a `PortInterface` based on the "Standardized PortInterface Blueprint" might contain a subset of `ClientServerOperation`s.
Rationale:	This allows to use Standardization Models to evaluate conformity of an implementation. Conformity depends on the approach to derive objects
Dependencies:	–
Conflicts:	–
Supporting Material:	–
Comment:	–

⌋(*UC_STDT_0007, UC_STDT_0011, UC_STDT_0012*)

3.16 Shall be able to express information about the state of model elements

[RS_STDT_0016] Shall be able to express information about the state of model elements ⌈

Initiator:	Bosch
Date:	2010-05-12
Requirement:	It would be beneficial, if the STDT would also support information about the state of model elements. Example: Due to backward compatibility it will be difficult to delete exiting application interfaces for future releases. If some of the model elements are no longer "state of the art", they can be marked as being e.g. "obsolete".
Rationale:	This supports a continues evolution of a standard
Dependencies:	–

Conflicts:	–
Supporting Material:	–
Comment:	–

⌋(*UC_STDT_0001*)

3.17 Shall cover the compatibility of blueprints and derived objects

[RS_STDT_0017] Shall cover the compatibility of blueprints and derived objects ⌈

Initiator:	Bosch
Date:	2010-05-12
Requirement:	STDT shall describe the compatibility rules between blueprints and derived objects. These compatibility shall be described individually for each meta class eligible for blueprinting.
Rationale:	This supports a continues evolution of a standard
Dependencies:	[RS_STDT_0001]
Conflicts:	–
Supporting Material:	–
Comment:	–

⌋(*UC_STDT_0007*)

3.18 Shall allow to describe the dependencies of APIs (e.g. invocation and callback/polling interfaces)

[RS_STDT_0018] Shall allow to describe the dependencies of APIs (e.g. invocation and callback/polling interfaces) ⌈

Initiator:	WP "Software Architecture and OS"
Date:	2010-05-26
Requirement:	STDT shall allow to describe the dependencies of invocation interfaces and the corresponding callback or polling interfaces.
Rationale:	Standardized interfaces consist in many cases of invocation interfaces (C-APIs) and callback or polling interfaces. In many cases it is configurable, which communication pattern is used. This configurable dependency and the parameters shall be described via blueprints.
Dependencies:	[RS_STDT_0002]
Conflicts:	–
Supporting Material:	–
Comment:	–

⌋(*UC_STDT_0002*)

3.19 Shall define the mandatory semantics for a Blueprint

[RS_STDT_0019] Shall define the mandatory semantics for a Blueprint ⌈

Initiator:	WP "Coordination of Appl. Interfaces"
Date:	2010-05-26
Requirement:	The STDT shall define the mandatory semantics for a Blueprint.
Description:	For a given model element,the template must define which attributes of the model element must be standardized to be entitled as a blueprint. For e.g which information of a PortInterface must be present to be called as Blueprint ? In case of Standardized PortInterfaces of AUTOSAR Services the Standardization template shall mention the use case that in this case a PortInteface based on the "Standardized PortInterface Blueprint" might contain a subset of ClientServeOperations.
Rationale:	Helps to have a common understanding on blueprint model element.
Dependencies:	–
Conflicts:	–
Supporting Material:	–
Comment:	–

⌋(*UC_STDT_0001*)

3.20 Shall support variants of a VariableDataprototype

[RS_STDT_0020] Shall support variants of a VariableDataprototype ⌈

Initiator:	WP "Coordination of Appl. Interfaces"
Date:	2010-05-26
Requirement:	The STDT shall support variants of a VariableDataprototype.
Description:	The PortBlueprint should be able to map to different VariableDataprototype of the same instance of PortInterface.
Rationale:	Variant handling for WP10.3 is mostly intended for reusing the definition between passenger cars and trucks. Therefore it's probably useful to have variants at data type level instead of creating new blueprints.
Dependencies:	–
Conflicts:	–
Supporting Material:	–
Comment:	–

⌋(*UC_STDT_0001*)

3.21 Shall support multiple instantiation for an example SWC with PortBlueprint

[RS_STDT_0021] Shall support multiple instantiation for an example SWC with PortBlueprint ⌈

Initiator:	WP "Coordination of Appl. Interfaces"
Date:	2010-05-26
Requirement:	The STDT shall support multiple instantiation for an example SWC with PortBlueprint.
Description:	It should be possible for the PortBlueprint to support multiple instantiation.
Rationale:	–
Dependencies:	–
Conflicts:	–
Supporting Material:	–
Comment:	–

⌋*(UC_STDT_0001)*

3.22 Means of exchange format between stakeholders for blueprints

[RS_STDT_0022] Means of exchange format between stakeholders for blueprints ⌈

Initiator:	WP "Coordination of Appl. Interfaces"
Date:	2010-05-26
Requirement:	Means of exchange format between stakeholders for blueprints.
Description:	AUTOSAR methodology shall define the exchange of the PortInterfaceMapping for a given SWCdescription file, i.e The RTE and the VFB in principle ignores the Blueprints but how should the exchange be established between the stakeholders with the Blueprint information, while creating the PortPrototypes out of it?.
Rationale:	–
Dependencies:	–
Conflicts:	–
Supporting Material:	–
Comment:	–

⌋*(UC_STDT_0001)*

3.23 Shall be able to standardize Alias Names

[RS_STDT_0023] Shall be able to standardize Alias Names ⌈

Initiator:	Bosch
Date:	2010-06-02
Requirement:	Means of exchange format between stakeholders for blueprints.
Description:	STDT shall be able to standardize alias names.
Rationale:	e.g. used for system constants in measurement and calibration system Necessary if system constants will be standardized in future (what is not yet decided)
Dependencies:	–
Conflicts:	–
Supporting Material:	–
Comment:	–

⌋(*UC_STDT_0008, UC_STDT_0009*)

3.24 Shall be able to standardize Unique Names and Display Names

[RS_STDT_0024] Shall be able to standardize Unique Names and Display Names ⌈

Initiator:	Bosch
Date:	2010-06-02
Requirement:	Shall be able to standardize Unique Names and Display Names
Description:	STDT shall be able to standardize Unique Names and Display Names e.g. in documentation (the complete instance reference would not be readable) and in measurement and calibration systems (standardized measurement and calibration formats like A2L require unique names for sw signals).
Rationale:	support standardization of unique names, e.g. for • documentation • calibration and measurement tools
Dependencies:	–
Conflicts:	–
Supporting Material:	–
Comment:	–

⌋(*UC_STDT_0009, UC_STDT_0010*)

3.25 Shall be able to standardize life cycle states

[RS_STDT_0025] Shall be able to standardize life cycle states ⌈

Initiator:	Bosch
Date:	2010-06-02

Requirement:	Shall be able to standardize life cycle states
Description:	STDT shall be able to standardize life the states of a particular lifecycle).
Rationale:	Since AUTOSAR has the goal of being backward compatible it is not possible to just delete a standardized model element and add a new one or - even worse - to change the model element without notification.
Dependencies:	–
Conflicts:	–
Supporting Material:	–
Comment:	–

⌋*(UC_STDT_0009, UC_STDT_0010)*

3.26 Shall allow to represent port interface blueprints

[RS_STDT_0026] Shall allow to represent port interface blueprints ⌈

Initiator:	WP "Coordination of Appl. Interfaces"
Date:	2010-04-23
Requirement:	Shall allow to represent port interface blueprints.
Description:	AUTOSAR standardizes so called "Application Interfaces". These Interfaces in fact result in port blueprints and appropriate port interface blueprints
Rationale:	AUTOSAR publishes standardized Models as ARXML.
Dependencies:	–
Conflicts:	–
Supporting Material:	–
Comment:	–

⌋*(UC_STDT_0001, UC_STDT_0008)*

3.27 Shall allow to evaluate the integrity of Blueprints

[RS_STDT_0027] Shall allow to evaluate the integrity of Blueprints ⌈

Initiator:	WP "Software Architecture and OS"
Date:	2010-02-25
Requirement:	Shall allow to evaluate the integrity of Blueprints
Description:	Until Release 4.0 all APIs of the BSW are modeled and chapter 8 of the SWS is mainly generated out of the model. Additionally we propose to generate empty C functions (and data structures/consts/...) out of the model and link all these functions together. If the compile or link process fails the consistency (e.g. between different SWS) is violated and needs to be fixed.
Rationale:	In the past we had often problems that some SWS assume specific services (of structs/consts/...) from other modules and the interface did not match (or even did not exists at all). In the compile test such errors will be found.
Dependencies:	–

Conflicts:	–
Supporting Material:	–
Comment:	–

⌋(*UC_STDT_0013*)

3.28 Shall allow to generate BSW ”Standard AUTOSAR Interface” description from model

[RS_STDT_0028] Shall allow to generate BSW ”Standard AUTOSAR Interface” description from model ⌈

Initiator:	WP "Software Architecture and OS"
Date:	2010-02-25
Requirement:	Shall allow to generate BSW "Standard AUTOSAR Interface" description from model
Description:	"Standard AUTOSAR Interface" is part of each SWS which offers this interface (Typically contained in an own chapter of subchapter of 7 or 8). The description is mostly plain text with some pseudo language to show the usage of the interface (including constants, etc.). Furthermore the description of the services often uses "elements" from the meta-model which are not up-to-date or their meaning has changed. STDT shall provide support to standardize this part of the SWS, e.g. via an own model (and then the generated descriptions can be imported into the SWS like chapter 8) OR via a -standardized- language to clarify the understanding of the interface and allow an automatic conversation for RTE purposes.
Dependencies:	–
Conflicts:	–
Supporting Material:	–
Comment:	–

⌋(*UC_STDT_0014*)

3.29 Shall be able to represent further Blueprints

[RS_STDT_0029] Shall be able to represent further Blueprints ⌈

See also [RS_STDT_0003], [RS_STDT_0026]

Initiator:	WP "Software Architecture and OS" Markus Bechter
Date:	2011-02-22 WP 1.2 Meeting
Requirement:	Shall be able to represent defined Blupeprints

Description:	STDT shall be able to represent Blueprints for the following elements: • AliasNameSet (see RS_STDT_023) • ApplicationDatatype • BswModuleEntry • BaseType • BswModuleDescription • CompuMethod - to enhance enumerators • DataConstr - to widen ranges as long as it fits in the BaseType not allowed to restrict : Wp: Not allowed to change the range • DatatypeMappingSet - mapped types in derived Mappings set must be the derived ones from Blueprint • EcucModuleDef • EcucDefintionCollection • ImplementationDatatype • ModeDeclarationGroup - to add addtional modes • PortInterfaces (for sender receiver and client server interfaces) (see RS_STDT_026) • PortPrototypeBlueprints (see RS_STDT_003) • SwComponentType
Dependencies:	–
Conflicts:	–
Supporting Material:	–
Comment:	This is a collector requirement which was added later. It summarizes all required blueprints even if some of those were already covered by other requirements.

⌋*(UC_STDT_0003, UC_STDT_0012, UC_STDT_0011)*

3.30 Shall allow to standardize package structures

[RS_STDT_0030] Shall allow to standardize package structures ⌈

Initiator:	WP "Coordination of Appl. Interfaces"
Date:	2011-08-12
Requirement:	Shall allow to standardize package structures
Description:	STDT shall be able to represent blueprints of package structures in particular to predefine access paths.
Dependencies:	–
Conflicts:	–
Supporting Material:	–
Comment:	–

⌋*(UC_STDT_0004)*

A Change History for AUTOSAR 4.0.3 against 4.0.2

Number	Heading
[UC_STDT_0001]	Support Application Interfaces
[UC_STDT_0002]	Express Parts of SWS
[UC_STDT_0003]	Standardize ECUCParamdefs
[UC_STDT_0004]	Express predefined Paths
[UC_STDT_0005]	Express PlatformTypes
[UC_STDT_0006]	Express Examples of applied Standards
[UC_STDT_0007]	Support Verification if an implementation adheres to defined Standard
[UC_STDT_0008]	Support reusable Documentation
[UC_STDT_0009]	Define name conventions
[UC_STDT_0010]	Perform Standardization on Levels beyond the AUTOSAR Scope
[UC_STDT_0011]	Derive Objects from Blueprints by manually changing properties
[UC_STDT_0012]	Derive Objects from Blueprints in a completly standardized Way
[UC_STDT_0013]	Integrate compile test
[UC_STDT_0014]	Generate BSW ”Standard AUTOSAR Interface” description from model

Table A.1: Added Use Cases

Number	Heading
[RS_STDT_0001]	Shall support and explain Blueprints in general
[RS_STDT_0002]	Formalized description of BSW SWS
[RS_STDT_0003]	Shall allow to represent port blueprints
[RS_STDT_0004]	Shall allow to represent `shortName` patterns
[RS_STDT_0005]	Shall support keywords and keyword abbreviations
[RS_STDT_0006]	Shall be implemented without compatibility problems to existing template
[RS_STDT_0007]	Shall be based on the AUTOSAR schema
[RS_STDT_0008]	Shall provide means to support analyzing the conformity of implementations with the AUTOSAR standards
[RS_STDT_0009]	Shall be able to represent requirements stated in SWS
[RS_STDT_0010]	Shall refer to ECUC parameter definition
[RS_STDT_0011]	Shall be able to standardize components
[RS_STDT_0012]	Shall be able to standardize architecture
[RS_STDT_0013]	Shall be able to express parts of reference paths resp. package hierarchies
[RS_STDT_0014]	Shall be able to express levels of obligation
[RS_STDT_0015]	Shall support different Approaches to derive from Blueprints
[RS_STDT_0016]	Shall be able to express information about the state of model elements
[RS_STDT_0017]	Shall cover the compatibility of blueprints and derived objects
[RS_STDT_0018]	Shall allow to describe the dependencies of APIs (e.g. invocation and callback/polling interfaces)
[RS_STDT_0019]	Shall define the mandatory semantics for a Blueprint
[RS_STDT_0020]	Shall support variants of a VariableDataprototype
[RS_STDT_0021]	Shall support multiple instantiation for an example SWC with PortBlueprint
[RS_STDT_0022]	Means of exchange format between stakeholders for blueprints
[RS_STDT_0023]	Shall be able to standardize Alias Names
[RS_STDT_0024]	Shall be able to standardize Unique Names and Display Names
[RS_STDT_0025]	Shall be able to standardize life cycle states
[RS_STDT_0026]	Shall allow to represent port interface blueprints
[RS_STDT_0027]	Shall allow to evaluate the integrity of Blueprints
[RS_STDT_0028]	Shall allow to generate BSW ”Standard AUTOSAR Interface” description from model

[RS_STDT_0029]	Shall be able to represent further Blueprints
[RS_STDT_0030]	Shall allow to standardize package structures

Table A.2: Added Requirements