Document Title	Requirements on Methodology
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	362
Document Classification	Auxiliary

Document Status	Final
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	4.3.0

Document Change History			
Date	**Release**	**Changed by**	**Description**
2016-11-30	4.3.0	AUTOSAR Release Management	• Support for Data Exchange Points added • Minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation
2015-07-31	4.2.2	AUTOSAR Release Management	• Editorial changes
2014-10-31	4.2.1	AUTOSAR Release Management	• Support for Safety Extensions added • Support for Diagnostic Extract added
2013-10-31	4.1.2	AUTOSAR Release Management	• Layout update
2013-03-15	4.1.1	AUTOSAR Administration	• Requirement tables and requirement traceability reworked • Support for Abstract System Description added • Document converted to LaTeX
2011-12-22	4.0.3	AUTOSAR Administration	• Improved requirements tracing • Added support for Complex Drivers • New requirements numbering according to AUTOSAR definition
2010-02-02	3.1.4	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

Bibliography

[1] Standardization Template
AUTOSAR_TPS_StandardizationTemplate

[2] Main Requirements
AUTOSAR_RS_Main

1 Introduction

This document defines the requirements needed to specify the AUTOSAR methodology. Even if the AUTOSAR methodology specification already exists, this document will make the link between the needs and how it has been implemented. This document is a good tool to test the specification, see what is missing, what effort should be done in terms of planning and means.

1.1 Document Conventions

The representation of requirements in AUTOSAR documents follows the table specified in [TPS_STDT_00078], see Standardization Template, chapter Support for Traceability ([1]).

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 ([1]).

1.2 Acronyms and Abbreviations

Acronym	Description
SIL	Safety Integrity Level (IEC61508 definition)

Abbreviation	Description
ASAM MCD	Association for Standardization of Automation- and Measuring Systems Measurement, Calibration and Diagnostics
AUTOSAR	Automotive Open System Architecture
BSW	Basic Software
CPU	Central Processing Unit
DWARF	Debug With Arbitrary Record Format
ECU	Electronic Control Unit
MCAL	MicroController Abstraction Layer
OEM	Original Equipment Manufacture
OS	Operating System
RTE	Runtime Environment
SW	Software
SWC	Software Component
VFB	Virtual Functional Bus
XML	Extensible Markup Language

1.3 Requirements Tracing

The following table references the requirements specified in [2] and links to the fulfillments of these.

Requirement	Description	Satisfied by
[RS_Main_00030]	AUTOSAR shall support development processes for safety related systems	[RS_METH_00018] [RS_METH_00025] [RS_METH_00050] [RS_METH_00052] [RS_METH_00057] [RS_METH_00061] [RS_METH_00069] [RS_METH_00081]
[RS_Main_00060]	AUTOSAR shall provide a standardized software interface for communication between Applications	[RS_METH_00033]
[RS_Main_00080]	AUTOSAR shall provide means to describe a component model for Application Software	[RS_METH_00062] [RS_METH_00080]
[RS_Main_00130]	AUTOSAR shall provide an abstraction from hardware	[RS_METH_00032] [RS_METH_00033]
[RS_Main_00140]	AUTOSAR shall provide network independent communication mechanisms for applications	[RS_METH_00032] [RS_METH_00033]
[RS_Main_00150]	AUTOSAR shall support the deployment and reallocation of AUTOSAR Application Software	[RS_METH_00002] [RS_METH_00003] [RS_METH_00004] [RS_METH_00005] [RS_METH_00033] [RS_METH_00078] [RS_METH_00079]
[RS_Main_00190]	AUTOSAR shall support interoperability with non-AUTOSAR software on the same ECU	[RS_METH_00018]
[RS_Main_00220]	The functional interfaces of AUTOSAR shall be specified in standard C	[RS_METH_00015] [RS_METH_00038]
[RS_Main_00250]	AUTOSAR methodology shall provide a predefinition of typical roles and activities in work-share model	[RS_METH_00021] [RS_METH_00028] [RS_METH_00043] [RS_METH_00046] [RS_METH_00047] [RS_METH_00048] [RS_METH_00064] [RS_METH_00066]
[RS_Main_00260]	AUTOSAR shall provide diagnostics means during runtime, for production and services purposes	[RS_METH_00082]
[RS_Main_00290]	AUTOSAR shall support the verification of its specifications	[RS_METH_00025] [RS_METH_00069]

[RS_Main_00300]	AUTOSAR shall provide data exchange formats to support work-share in large inter and intra company development groups	[RS_METH_00002] [RS_METH_00003] [RS_METH_00004] [RS_METH_00005] [RS_METH_00006] [RS_METH_00018] [RS_METH_00020] [RS_METH_00025] [RS_METH_00033] [RS_METH_00050] [RS_METH_00051] [RS_METH_00052] [RS_METH_00054] [RS_METH_00057] [RS_METH_00061] [RS_METH_00067] [RS_METH_00069] [RS_METH_00077] [RS_METH_00078] [RS_METH_00079] [RS_METH_00080] [RS_METH_00081] [RS_METH_00082]
[RS_Main_00301]	AUTOSAR shall specify profiles for data exchange to support work-share in large inter- and intra-company development groups	[RS_METH_00083] [RS_METH_00084]
[RS_Main_00310]	AUTOSAR shall support hierarchical Application Software design methods	[RS_METH_00041]
[RS_Main_00330]	AUTOSAR shall support the principle of information hiding	[RS_METH_00032]
[RS_Main_00350]	AUTOSAR specifications shall be analyzable and support according methods to demonstrate the achievement of safety related properties.	[RS_METH_00025] [RS_METH_00041] [RS_METH_00050] [RS_METH_00051] [RS_METH_00052] [RS_METH_00054] [RS_METH_00081]
[RS_Main_00360]	AUTOSAR shall support management of vehicle diversity	[RS_METH_00062] [RS_METH_00074] [RS_METH_00075] [RS_METH_00076]
[RS_Main_00400]	AUTOSAR shall provide a layered software architecture	[RS_METH_00032] [RS_METH_00033]
[RS_Main_00490]	AUTOSAR processes shall be compliant to ISO26262	[RS_METH_00025] [RS_METH_00057] [RS_METH_00061] [RS_METH_00081]

2 Methodology Requirements

This chapter provides a definition of the requirements.

2.1 Main Requirements

[RS_METH_00006] Methodology shall explain how Autosar system is built ⌈

Type:	valid
Description:	Methodology shall explain how Autosar system is built using the templates and activities supported by guidance. It should be like a user manual to help an organization efficiently apply Autosar.
Rationale:	A strong methodology is necessary to effectively manage building a large system.
Dependencies:	–
Use Case:	Engineer would like to complete an activity and would like to know what inputs are needed, Guidance should be used, etc. Typical use cases involved to build an Autosar system include: • SWC implementation • ECU integration • System integration
Supporting Material:	–

⌋(*RS_Main_00300*)

[RS_METH_00033] Methodology should support VFB concept ⌈

Type:	valid
Description:	Virtual Functional Bus concept allows early checks between SW-C with a complete abstraction of the Hardware. The methodology should include this concept.
Rationale:	To improve the integration phases and the concurrent development.
Dependencies:	–
Use Case:	In AUTOSAR, an application is modeled as a composition of interconnected components. The "virtual functional bus" is the communication mechanism that allows these components to interact. Even if all the resources used by these components are not available (HW/Network) some basic checks can be done and early problems can be solve that will ease the integration phase later.
Supporting Material:	–

⌋(*RS_Main_00140*, *RS_Main_00060*, *RS_Main_00130*, *RS_Main_00150*, *RS_Main_00300*, *RS_Main_00400*)

[RS_METH_00041] Methodology shall support Bottom/Up Approach ⌈

Type:	valid
Description:	Methodology shall support the Bottom/Up (B/U) Approach. In this approach, all constraints coming from the Hardware in the B/U (ECUs/Sensors/Actuators) should be taken in account
Rationale:	To improve the integration phases, and to master the complexity in embedded RT distributed systems
Dependencies:	–
Use Case:	If in a given vehicle architecture, a new ECU is added or an existing ECU is replaced with a new one, all the new or modified resources from the ECU need to be included into the system configuration during integration.
Supporting Material:	–

⌋*(RS_Main_00310, RS_Main_00350)*

[RS_METH_00016] Methodology shall support building a system of both Autosar and Non-Autosar ECUs ⌈

Type:	valid
Description:	Methodology needs to show how to build a system of AUTOSAR compliant ECUs that are on the same architecture with non AUTOSAR compliant ECUs.
Rationale:	Provide a migration path to AUTOSAR for non-AUTOSAR applications.
Dependencies:	–
Use Case:	Legacy ECUs and LIN slaves need to interoperate with Autosar ECUs.
Supporting Material:	–

⌋*()*

[RS_METH_00017] Methodology shall clearly define what is standardized and what is not standardized ⌈

Type:	valid
Description:	Show which portions of the Methodology are normative and informative. The inputs/outputs, and content must be compatible among Autosar tools for certain core activities.
Rationale:	Necessary to ensure Interoperability of Tools to support all Autosar Activities
Dependencies:	–
Use Case:	Systems analysis tools are undergoing significant innovations that may not appropriate for standardization. Since the benefit of including these activities within the methodology still exists to help provide completeness, but should be clearly shown to be non-standardized.
Supporting Material:	–

⌋*()*

[RS_METH_00018] Methodology shall be modular ⌈

Type:	valid
Description:	Utilize process components. Sub processes shall be complete and testable on their own to allow usage of certain portions of the Methodology while still integrating with legacy tools and processes.
Rationale:	Easier to understand and verify all portions of the Methodology. Easier to manage modifications, encapsulates ripple effect due to changes to allow migration of current processes. Easier to utilize both legacy and Autosar activities. It should be possible to start from an intermediate activity and not necessarily from the beginning of the methodology. A modular Methodology facilitates organizations to migrate from or merge with their current processes. A modular Methodology allows organizations to insert intermediate activities such as quality gates, or other inspections, as well as collect metrics necessary to comply with CMMI processes and/or SIL-3.
Dependencies:	–
Use Case:	An organization is planning to introduce an Autosar ECU into their existing architecture, but is not planning to use the System Template and their respective activities and work products. Rather they plan to begin directly at the ECU level.
Supporting Material:	–

⌋*(RS_Main_00190, RS_Main_00300, RS_Main_00030)*

[RS_METH_00032] The methodology shall respect the different levels of Abstractions ⌈

Type:	valid
Description:	The methodology shall respect the Software Component, the System, and the ECU levels of Abstractions.
Rationale:	To improve the integration phases and to master the complexity in embedded RT distributed systems.
Dependencies:	–
Use Case:	AUTOSAR is using several abstractions levels to describe the information exchanged between the different players. In an early phase the "Virtual Functional Bus" is used in other phases we are working with the implementation of the SWC in several ECUs. The exchange between the "real" world and the Virtual world should be described and supported by the methodology.
Supporting Material:	–

⌋*(RS_Main_00130, RS_Main_00140, RS_Main_00330, RS_Main_00400)*

[RS_METH_00020] Methodology shall support iterations ⌈

Type:	valid
Description:	Need to support both small and large iteration loops.
Rationale:	Meet Autosar Quality requirements. Redoing work is error-prone. Support round trip engineering.
Dependencies:	[RS_METH_00062] < Methodology shall support configuration of parameters with different binding time >
Use Case:	Small iteration loops inside ECU Configuration Activity. Large iteration loops from System Design impacting ECU Configuration. Automotive systems are typically developed in several sample phases (A, B, C, etc). A Single Software Component is updated in a AUTOSAR System. The updated ECU Extract still matches the existing ECU Configuration (as long as no contradicting changes are made in the iteration).
Supporting Material:	–

⌋*(RS_Main_00300)*

[RS_METH_00062] Methodology shall support configuration of parameters with different binding time. ⌈

Type:	valid
Description:	Configuration of parameters can be performed in different process steps: pre-compile, link time, and post-build. The Autosar methodology must support system development with different combinations of these configuration classes.
Rationale:	–
Dependencies:	–
Use Case:	OEM configuration of post-build data after a release from a Tier1 supplier. Handling information related to different configuration classes as separate configuration items (units for version control).
Supporting Material:	–

⌋*(RS_Main_00080, RS_Main_00360)*

[RS_METH_00080] Exchange of Implicit Communication Behavior Description ⌈

Type:	valid
Description:	The AUTOSAR Methodology shall support the exchange of information to configure the Implicit Communication Behavior of the RTE according to the requirements of the Software Components. The information can be defined first time at the design of a Atomic Software Component but can be added as well if compositions are created. The design of a Atomic Software Component with respect to implicit communication behavior may be guided by blueprints of the Implicit Communication Behavior descriptions.
Rationale:	Define Implicit Communication Behavior requirements in a top down design approach
Use Case:	–
Dependencies:	–
Supporting Material:	[RS_SWCT_03065], [RS_STDT_00034]

⌋(*RS_Main_00080*, *RS_Main_00300*)

[RS_METH_00083] AUTOSAR methodology shall explain the description and handling of Data Exchange Points ⌈

Type:	valid
Description:	The methodology shall explain workflows for the development and use of Data Exchange Points. E.g. it shall describe which artifacts are provided by AUTOSAR that support the development of profiles of Data Exchange Points that can be used to analyze potential tool interoperability issues or to configure validation engines of AUTOSAR tools according to the described data exchange point.
Rationale:	Tool interoperability
Dependencies:	[RS_METH_00084]
Use Case:	• AUTOSAR specifies the contents of artifacts for different steps in the methodology. • A contract is establised between producing and consuming AUTOSAR tools with respect to exchanged artifacts. The producing tool assures its adherence to a an agreed profile and the consuming tool specifies its expectations using this profile.
Supporting Material:	–

⌋(*RS_Main_00301*)

2.2 Template Requirements

[RS_METH_00002] Methodology shall explain the typical usage of SW-C template ⌈

Type:	valid
Description:	The software component template includes numerous features including atomic software components, composition components, interfaces, ports, etc. The Methodology shall clearly show the activities to create/modify these and which activities require these to be completed before commencing.
Rationale:	Methodology Consistency using the SW-C template
Dependencies:	–
Use Case:	An existing software component is implemented and is to be integrated into a suppliers ECU. A description of the component is needed in order to correctly integrate that component into the ECU. A new system is being defined and requires a software component to contain specified interfaces. That specification is given to a supplier to implement the required software.
Supporting Material:	–

⌋(*RS_Main_00300*, *RS_Main_00150*)

[RS_METH_00003] Methodology shall explain the typical usage of BSW Module Template ⌈

Type:	valid
Description:	The basic software module template includes numerous features describing Interfaces and Data properties of Basic Software.
Rationale:	Methodology Consistency using the BSW Module description
Dependencies:	–
Use Case:	An existing BSW module is implemented and is to be integrated into an ECU. A description of the BSW Module is needed in order to correctly integrate that module into the ECU.
Supporting Material:	–

⌋(*RS_Main_00300*, *RS_Main_00150*)

[RS_METH_00004] Methodology shall explain the typical usage of the ECU Configuration template ⌈

Type:	valid
Description:	Depending on actors basic roles (supplier/customer), the tasks and the way to configure the ECU can be completely different. Basic Use Cases should clarify the way of using this template.
Rationale:	Methodology consistency using the ECU configuration
Dependencies:	–
Use Case:	Vendor of BSW is setting/giving recommended or predefined values. Conformance test use the Template to document which configuration is used in the Conformance Test. Use to describe the configuration when integrating a BSW module into an ECU.
Supporting Material:	–

⌋(*RS_Main_00300*, *RS_Main_00150*)

[RS_METH_00005] Methodology shall explain the typical usage of the System Template ⌈

Type:	valid
Description:	Depending on actors basic roles (supplier/customer), and the different phases, the tasks and the way to describe the System can be quite different. Basic use cases should clarify the way of using this template.
Rationale:	Methodology consistency using the System Templates
Dependencies:	–
Use Case:	OEM as integrator is fixing the ECUs, the topology and the list of the SW-C for one vehicle for the suppliers. Supplier is delivering to the OEMs a sub-system for integration.
Supporting Material:	–

⌋(*RS_Main_00300*, *RS_Main_00150*)

[RS_METH_00081] Methodology shall explain the typical usage of Safety Extensions ⌈

Type:	valid
Description:	The Safety Extensions include numerous features (Safety Requirement, Safety Measure, ASIL Attribute etc.) required for the design and development of safety critical systems in the AUTOSAR context. The AUTOSAR methodology shall describe the basic usage of the introduced features. Furthermore a basic method library for the Safety Extensions shall be provided.
Rationale:	Methodology Consistency using the Safety Extensions
Dependencies:	–
Use Case:	During the analysis, design and implementation of safety critical functions using AUTOSAR, the exchange of relevant information between involved parties shall be possible. The Safety Extensions allow a standardized exchange of safety information and provide the basis for consistent management among different vendors and tools as required by ISO 26262.
Supporting Material:	–

⌋(*RS_Main_00300*, *RS_Main_00030*, *RS_Main_00490*, *RS_Main_00350*)

[RS_METH_00082] Methodology shall explain the typical usage of Diagnostic Extract Template ⌈

Type:	valid
Description:	The Diagnostic Extract Template includes numerous features required for the decentralized configuration of diagnostic aspects in the AUTOSAR context. The AUTOSAR methodology shall describe the basic activities.
Rationale:	Methodology Consistency using the Diagnostic Extract Template
Dependencies:	–
Use Case:	The distributed nature of an AUTOSAR ECU development requires an optimized capturing of information. Different stakeholders shall be able to contribute to several parts of the diagnostic aspects. The Diagnostic Extract Template that represents a standardized exchange format on diagnostic functionality allows the decentralized configuration of diagnostic aspects.
Supporting Material:	–

⌋(*RS_Main_00300*, *RS_Main_00260*)

[RS_METH_00077] Methodology shall explain the typical interaction between OEMs and suppliers ⌈

Type:	valid
Description:	Methodology shall support use cases of interaction between OEM and supplier, where the OEM and the supplier have different views on the SW-C structure.
Rationale:	Methodology consistency using the System Templates
Dependencies:	–
Use Case:	The OEM hands over the initial System Extract to the supplier as a formal requirements specification. The supplier extends and refactors this System Extract. In the next development cycle the OEM hands over an updated System Extract to the supplier. Thereafter the supplier has to update his System Extract structure based on the updates made by the OEM. The amount of changes on the supplier side shall be minimized to the changes caused by OEM updates.
Supporting Material:	–

⌋(*RS_Main_00300*)

[RS_METH_00078] Methodology shall explain the typical usage of different views on the system of the OEM ⌈

Type:	valid
Description:	Methodology shall support use cases of the OEM, where the OEM has different views on the system.
Rationale:	Methodology consistency using the System Templates
Dependencies:	–
Use Case:	An OEM might structure the AUTOSAR software components from a functional point of view. However, for the concrete vehicle development project a topological view of structure of SW-Cs is needed. For better handling during the lifecycle, the SW-Cs from the functional decomposition are mapped to the topological view using appropriate mappings.

Supporting Material:	–

⌋(*RS_Main_00300*, *RS_Main_00150*)

[RS_METH_00079] Methodology shall explain the typical usage of different views on the system of the Supplier ⌈

Type:	valid
Description:	Methodology shall support use cases of the supplier where the supplier has different views on the system.
Rationale:	Methodology consistency using the System Templates
Dependencies:	–
Use Case:	The supplier needs to map different views of the system, e.g. a) the supplier already has an existing software architecture. Via software sharing some of the components are substituted by the ones delivered by the OEM. b) The supplier needs to formally describe changes between system descriptions representing different releases. c) The supplier develops one ECU for different OEMs and therefore needs to map the requirement-views of the OEMs to his solution view. d) The supplier realizes the OEMs definition for one ECU by 2 ECUs and therefore needs to map the system descriptions.
Supporting Material:	–

⌋(*RS_Main_00300*, *RS_Main_00150*)

[RS_METH_00084] AUTOSAR methodology shall relate templates to a distributed development process ⌈

Type:	valid
Description:	The AUTOSAR templates specify the language for describing an AUTOSAR-based software or system. The methodology shall support the specification of a subset of the templates, which is used for a specific work product in a distributed development process.
Rationale:	Exchange of AUTOSAR artifacts in distributed development
Dependencies:	[RS_METH_00083]
Use Case:	A (VFB) system description shall only contain relevant information for development of SW-Cs without deployment to an ECU network yet.
Supporting Material:	

⌋(*RS_Main_00301*)

2.3 Programming Language

[RS_METH_00015] Methodology shall be independent of programming language ⌈

Type:	valid
Description:	The methodology shall be independent of programming language by providing generic solutions. For portions that are necessarily dependent on the programming language, these sections shall be explicitly noted and modular such that the overall methodology can be tailored to accommodate other programming languages.
Rationale:	By appropriately structuring the methodology to support existing and emerging programming languages, the Methodology can be consistently and successfully applied across an entire vehicle.
Dependencies:	–
Use Case:	An ECU is built on a microcontroller optimized for programming language ABC. The Methodology explains when and how to specify and select the implementation of the software components deployed to that node that are compatible with that programming language.
Supporting Material:	–

⌋*(RS_Main_00220)*

[RS_METH_00038] Methodology shall support the C programming language ⌈

Type:	valid
Description:	Methodology shall show how to realize an Autosar System that uses the C programming language.
Rationale:	Presently C is the most common programming language used in the automotive embedded environment.
Dependencies:	–
Use Case:	A software component implemented in the C language is to be integrated into an Autosar ECU.
Supporting Material:	–

⌋*(RS_Main_00220)*

2.4 Activities

[RS_METH_00021] Methodology shall define Activities ⌈

Type:	valid
Description:	Methodology shall define the activities necessary to satisfy the Autosar Use Cases.
Rationale:	Allow tools from multiple vendors to be used to complete all Autosar Activities in order to build an Autosar compliant System. Provides a common language to communicate among Autosar Members.

Dependencies:	–
Use Case:	Activities include very specific tasks such as "Generate the RTE", and "Define Topology".
Supporting Material:	–

⌋(*RS_Main_00250*)

[RS_METH_00043] Activities shall have a purpose ⌈

Type:	valid
Description:	Each activity shall include a textual purpose explaining why the activity is needed, its objectives, and what can be expected upon completion.
Rationale:	A clearly defined purpose provides a common understanding of the activity and helps to ensure consistency between tool implementations that complete the activity.
Dependencies:	RS_METH_00021
Use Case:	Example activity description for "Generate the RTE": This purpose of this activity is to generate the RTE layer of an Autosar ECU. Example activity description for "Define Topology": The purpose of this activity is to select available Autosar ECUs and instantiate them onto a vehicle topology.
Supporting Material:	–

⌋(*RS_Main_00250*)

[RS_METH_00046] Activities shall have input work products ⌈

Type:	valid
Description:	Each activity shall include which work products are used to complete an activity. If no work products are needed, then this shall be explicitly stated.
Rationale:	Explicitly stating what work products are needed for an activity is necessary to ensure tool's supporting that activity are interoperable.
Dependencies:	RS_METH_00021
Use Case:	Example input work products for Generate RTE Activity include Software Components, ECU Communication Database, ECU Configuration.
Supporting Material:	–

⌋(*RS_Main_00250*)

[RS_METH_00047] Activities shall have output work products ⌈

Type:	valid
Description:	Each activity shall include which work products are produced or modified during an activity. If no work products are produced or modified, then this shall be explicitly stated.
Rationale:	Explicitly stating what work products are produced or modified during an activity is necessary to ensure tool's supporting that activity are interoperable.
Dependencies:	RS_METH_00021

Use Case:	Example output work products for Generate RTE include the RTE source files.
Supporting Material:	–

⌋(*RS_Main_00250*)

[RS_METH_00048] Activities shall include roles ⌈

Type:	valid
Description:	Each activity shall include the roles that are responsible to perform and, if any, to support it.
Rationale:	By assigning roles to activities, it is possible to take a view of the methodology from the point of view of a certain role. This answers questions such as "What are all the activities that I perform as a software developer?"
Dependencies:	[RS_METH_00021] < Methodology shall define Activities > [RS_METH_00028] < Methodology shall define Roles >
Use Case:	The Generate RTE Activity is performed by the role "Software Integration Engineer".
Supporting Material:	–

⌋(*RS_Main_00250*)

[RS_METH_00066] Methodology shall support activities that reference tools ⌈

Type:	valid
Description:	Activities may reference tools that help to complete the activity. The methodology shall describe these types of tools and when they are used.
Rationale:	By defining which tools are needed, the performers of the activity can ensure all the tools have been sourced and installed prior to beginning the activity. As well, the implementers of tools that are Autosar specific, have a clear understanding of what activities their tool should support and know what the input and output work products are available. This will help ensure interoperability of Autosar Tools.
Dependencies:	RS_METH_00021
Use Case:	The activity "Generate RTE" requires an RTE generator Tool and a compiler.
Supporting Material:	–

⌋(*RS_Main_00250*)

2.5 Work Products

[RS_METH_00025] Methodology shall define Work products ⌈

Type:	valid
Description:	Methodology shall define the work products necessary to build an Autosar System. Work products define a certain content and may exists as one or multiple files, which is not specified by Autosar.
Rationale:	A clear definition of what work products exists is necessary to allocate responsible roles and identify which activities consume, produce, or modify that work product.
Dependencies:	–
Use Case:	According to SPEM, "A work product or artifact is anything produced, consumed, or modified by a process. It may be a piece of information, a document, a model, source code, and so on." Example work products include a software component description, an ECU instance description, a topology description, or the RTE source files.
Supporting Material:	–

⌋(*RS_Main_00030*, *RS_Main_00290*, *RS_Main_00300*, *RS_Main_00350*, *RS_Main_00490*)

[RS_METH_00050] Work products shall have a description ⌈

Type:	valid
Description:	Each work product shall have a textual description describing its contents.
Rationale:	A description of the work product is necessary to form a common understanding of what the work product contains.
Dependencies:	[RS_METH_00025] < Methodology shall define Work products >
Use Case:	The AtomicSoftwareComponentType is the highest (most abstract) description level that corresponds to the Virtual Function Bus view. This Work Product includes the data types and interfaces. These components do not contain any other sub-components as they define the smallest granularity of components.
Supporting Material:	–

⌋(*RS_Main_00300*, *RS_Main_00350*, *RS_Main_00030*)

[RS_METH_00051] Work products shall have a reference(s) to metaclass(es) in the Autosar Metamodel. ⌈

Type:	valid
Description:	The work product shall references to the Autosar Metaclasses that are used to exchange it. If no Metaclass exists, such as for source files, or requirements documents, then this requirement is exempt.
Rationale:	The Autosar Metaclasses provide a standard mechanism to exchange Autosar specific work products.
Dependencies:	[RS_METH_00025] < Methodology shall define Work products >
Use Case:	The AtomicSoftwareComponentType is exchanged using the AtomicSoftwareComponentType MetaClass.

Supporting Material:	–

⌋(*RS_Main_00300*, *RS_Main_00350*)

[RS_METH_00052] It must be possible to avoid duplication of data in Work Products ⌈

Type:	valid
Description:	Work Products defined should avoid duplicating data already defined in other work products. Work Products defined in the methodology are "Logical" work products, not "physical" work products. So in your own process you can decide to include, for example, the interfaces and software component description in 1 physical file or duplicated in several files. However, in the Methodology it is defined only once, with duplications avoided.
Rationale:	Data duplication causes problems when iterating and changes are made inconsistently across all affected work products. Having one source of data eliminates this problem.
Dependencies:	[RS_METH_00025] < Methodology shall define Work products >
Use Case:	Two atomic software components exchange information via a Port Interface. Since the port interface referenced by both software components, it should treated as a separate work product in order to avoid defining it twice. Exceptions to this requirement arise when a work product undergoes a rule-based transformation such as compiling or code generation. In this case, the duplication in the resulting work products always remains consistent with the original source. For example, type definitions in object files are continuously updated based on the c type definition via the compiler. However it would be unwise to define the same type definition twice in two separate c header files.
Supporting Material:	–

⌋(*RS_Main_00030*, *RS_Main_00300*, *RS_Main_00350*)

[RS_METH_00054] Work Products shall not have circular references with other work products ⌈

Type:	valid
Description:	A circular reference exists when several work products sequentially reference each other and the last work product references the first. Work Products shall be defined such that circular references do not exist.
Rationale:	Work products with circular references are especially difficult to manage and exchange between parties and create problems when defining activities that create them.
Dependencies:	[RS_METH_00025] < Methodology shall define Work products >
Use Case:	A composition software component A aggregates other software component prototypes B and C, which may also be composition software components. These aggregated software components B and C and the software components they refer to can never reference the composition software component A.
Supporting Material:	–

⌋(*RS_Main_00300*, *RS_Main_00350*)

[RS_METH_00061] Methodology shall describe the change of existing work products. ⌈

Type:	valid
Description:	Methodology shall include scenarios in which the output of an activity is not created newly, but updated from an older version of a work product.
Rationale:	Updating an existing work product may lead to a different process then creating a new one. For example, version information must be handled.
Dependencies:	[RS_METH_00020] < Methodology shall support iterations >
Use Case:	Software update: The methodology shall describe the workflow needed for SW updates and required conversions of already existing data (e.g. NVRAM data) of an AUTOSAR ECU.
Supporting Material:	–

⌋(*RS_Main_00030*, *RS_Main_00300*, *RS_Main_00490*)

[RS_METH_00069] It shall be possible to add precise and human readable documentation to each work product. ⌈

Type:	valid
Description:	The methodology shall allow that precise and human readable documentation be added to each work product. This documentation shall be either part of the work product or uniquely referred.
Rationale:	This is necessary in order to document design decisions or restrictions, which cannot obviously be deduced from the formal content, e.g. from names. Such documentation will increase the traceability which is demanded by quality or safety standards.
Dependencies:	[RS_METH_00025] < Methodology shall define Work products >
Use Case:	Choosing a redundancy mechanism, e.g. in the configuration for a NVRAM data block, may be related to a safety requirement. This may need verbal explanation.
Supporting Material:	–

⌋(*RS_Main_00030*, *RS_Main_00290*, *RS_Main_00300*)

2.6 Guidance

[RS_METH_00027] Methodology shall define unambiguous guidance terminology ⌈

Type:	valid
Description:	Activities can refer on guidance to assist execution of the activity. Guidance can take the form of checklists, tutorials, or tools. When tools are used, a name shall be given to it. An actual tool implementation may perform many other tasks, but within the context of an activity, the tool only performs the named guidance.

Rationale:	Facilitate sourcing and comparison of various tools, especially when they perform many, likely different, sets of Autosar Activities.
Dependencies:	–
Use Case:	For example, there is likely a need to have a guidance that helps configure the RTE, the RTE Generator. Similarly, there is likely a need to have a guidance that actually generate the RTE, the RTE generator. It can be foreseen that both of these guidance can be completed by the same tool or separate tools.
Supporting Material:	–

⌋()

[RS_METH_00042] Methodology shall incorporate the usage of industry standard tools ⌈

Type:	valid
Description:	Where industry standard tools, such as compilers and linkers exist, the Methodology shall incorporate them.
Rationale:	AUTOSAR should not required the use of tools when industry standard tools already exist.
Dependencies:	–
Use Case:	Compilers are industry standard tools.
Supporting Material:	–

⌋()

2.7 Roles

[RS_METH_00028] Methodology shall define Roles ⌈

Type:	valid
Description:	Methodology shall capture the typical roles people act within Autosar
Rationale:	The definition of Roles will allow distinguishing the responsibilities of different parties working in the AUTOSAR activities and assign ownership to a Work Product. It will also help to define the level of granularity for the activities and work products by considering the "owner" and the "performer".
Dependencies:	–
Use Case:	Example roles include, "Software Developer", "Module Integrator", and "Electrical Architect".
Supporting Material:	–

⌋(*RS_Main_00250*)

[RS_METH_00064] Roles shall have a description ⌈

Type:	valid
Description:	Methodology shall capture the typical roles people act within Autosar
Rationale:	–
Dependencies:	[RS_METH_00028] < Methodology shall define Roles >
Use Case:	A "Software Developer" is an engineer proficient in modelling or programming languages and is capable of translating requirements into software and resolving defects.
Supporting Material:	–

⌋(*RS_Main_00250*)

2.8 Process Requirements

[RS_METH_00056] AUTOSAR methodology shall not be bound to a particular lifecycle model ⌈

Type:	valid
Description:	AUTOSAR methodology shall not be bound to a particular lifecycle model. Activities must be independent with respect to the time and phase of the development process they are executed.
Rationale:	Connection to company specific lifecycle model: The methodology shall enable the use of different lifecycle models such as e.g. V-Model, Rational Unified Process.
Dependencies:	–
Use Case:	If e.g. extreme programming is used, the test cases are created prior to the implementation. For most other development processes, the implementation is generated prior to the creation of test cases.
Supporting Material:	–

⌋()

[RS_METH_00057] AUTOSAR methodology shall support traceability to external artifacts ⌈

Type:	valid
Description:	External artifacts are artifacts such as requirements or test cases that are not defined in the context of AUTOSAR. The mapping of external artifacts onto AUTOSAR work products must be possible.
Rationale:	Achieve traceability during the creation of the product.
Dependencies:	–
Use Case:	E.g. the mapping of requirements to • (the implementation of) a software component and • the configuration of a software component

Supporting Material:	–

⌋(*RS_Main_00300*, *RS_Main_00030*, *RS_Main_00490*)

2.9 Development Requirements

[RS_METH_00009] Methodology should be modeled ⌈

Type:	valid
Description:	Methodology should be modeled using consistent relationships.
Rationale:	Modelling the methodology will allow the relationships to be precise and will allow a tool to automatically generate the published document.
Dependencies:	–
Use Case:	
Supporting Material:	–

⌋()

[RS_METH_00010] Methodology should define rules to translate methodology model into a document ⌈

Type:	valid
Description:	Templates and rules can be applied to the modeled methodology to convert it into a document for publishing.
Rationale:	Using precise rules to automatically generate the methodology document will eliminate many sources of errors and inconsistencies.
Dependencies:	[RS_METH_00009] < Methodology should be modeled >
Use Case:	Possible to generate Latex version of the document.
Supporting Material:	–

⌋()

[RS_METH_00067] Methodology document shall include hyperlinks between Activities, Roles, Work Products, and Guidance. ⌈

Type:	valid
Description:	Cross references between the Activities, Roles, Work Products, and Guidance shall be hyperlinked.
Rationale:	By creating hyperlinks to all references, the Methodology can quickly answer many common queries such as "Show me all the work products Role X owns", or "Show me all activities that require work product Y as an input". This is very helpful to quickly learn the Methodology or to conduct a downstream impact analyses in order to identify the stakeholders.
Dependencies:	[RS_METH_00009] < Methodology should be modeled > [RS_METH_00010] < Methodology should define rules to translate methodology model into a document >
Use Case:	Possible to generate Latex version of the Methodology.
Supporting Material:	–

⌋*(RS_Main_00300)*

2.10 Variant Handling Requirements

[RS_METH_00074] Methodology shall specify Binding times ⌈

Type:	valid
Description:	AUTOSAR Methodology shall specify particular points in the workflow on which variation can be resolved
Rationale:	Need for a stable reference on Binding times
Dependencies:	–
Use Case:	During the development of an Autosar System and ECU, specific variants need to be created, and eventual chosen, e.g pre compile, or post build.
Supporting Material:	–

⌋*(RS_Main_00360)*

[RS_METH_00075] Methodology shall specify the tasks of resolving variant ⌈

Type:	valid
Description:	AUTOSAR Methodology shall specify particular tasks/activities in which variation will be resolved
Rationale:	Need for clarification of methodology of variants
Dependencies:	–
Use Case:	If two software components provide the same interface in different variants of the system, a task is needed to select the one provider to resolve that system variant.
Supporting Material:	–

⌋(*RS_Main_00360*)

[RS_METH_00076] Methodology shall specify a work product for values of variant selectors ⌈

Type:	valid
Description:	AUTOSAR Methodology shall specify particular work products to maintain the values of variant selectors.
Rationale:	This makes it clear where the values for variant selectors are stored and maintained.
Dependencies:	–
Use Case:	The possible variants are known up front: they are created at a certain time and owned as a work product, and finally consumed when the variant is selected.
Supporting Material:	–

⌋(*RS_Main_00360*)

3 Change History

3.1 Change History for AUTOSAR 4.2.1 against 4.1.2

3.1.1 Removed RS Items

N/A

3.1.2 Changed RS Items

N/A

3.1.3 Added RS Items

Number	Heading
[RS_METH_00081]	Methodology shall explain the typical usage of Safety Extensions
[RS_METH_00082]	Methodology shall explain the typical usage of Diagnostic Extract Template

Table 3.1: Added Specification Items in 4.2.1

3.2 Change History for AUTOSAR 4.2.2 against 4.2.1

No changes.

3.3 Change History for AUTOSAR 4.3.0 against 4.2.2

3.3.1 Removed RS Items

Number	Heading
RS_METH_00063	Work Products shall be capable to be version controlled

Table 3.2: Removed Specification Items in 4.3.0

3.3.2 Changed RS Items

Number	Heading
[RS_METH_00066]	Methodology shall support activities that reference tools
[RS_METH_00009]	Methodology shall be modeled

Table 3.3: Changed Specification Items in 4.3.0

3.3.3 Added RS Items

Number	Heading
[RS_METH_00083]	AUTOSAR methodology shall explain the description and handling of Data Exchange Points
[RS_METH_00084]	AUTOSAR methodology shall relate templates to a distributed development process

Table 3.4: Added Specification Items in 4.3.0