

<b>Document Title</b>	Glossary
<b>Document Owner</b>	AUTOSAR
<b>Document Responsibility</b>	AUTOSAR
<b>Document Identification No</b>	55

<b>Document Status</b>	published
<b>Part of AUTOSAR Standard</b>	Foundation
<b>Part of Standard Release</b>	R20-11

<b>Document Change History</b>			
<b>Date</b>	<b>Release</b>	<b>Changed by</b>	<b>Change Description</b>
2020-11-30	R20-11	AUTOSAR Release Management	Added new terms: <ul style="list-style-type: none"> <li>• E2E protection alive counter</li> <li>• E2E protection sequence counter</li> <li>• Vehicle State Manager</li> <li>• Health Indicator</li> <li>• System Health Monitor</li> <li>• Wake-up and sleep on dataline</li> <li>• Foundation</li> <li>• Intrusion Detection System</li> <li>• Onboard Security Event</li> </ul>
2019-11-28	R19-11	AUTOSAR Release Management	Removed FlexRay specific terms Added new terms: <ul style="list-style-type: none"> <li>• Secure channel</li> <li>• Abstract Platform</li> <li>• Raw Data Stream</li> <li>• Signal Service Translation</li> </ul> <p>Changed Document Status from Final to published</p>
2019-03-29	1.5.1	AUTOSAR Release Management	Editorial changes
2018-10-31	1.5.0	AUTOSAR Release Management	Extended abbreviations Added terms: <ul style="list-style-type: none"> <li>• AUTOSAR Run-Time Interface</li> <li>• Bus Mirroring</li> <li>• Cluster</li> <li>• Executable Entity Cluster</li> <li>• Execution Order Constraint</li> <li>• Execution Time</li> <li>• LIN Bus Idle</li> </ul>

Document Change History			
Date	Release	Changed by	Change Description
			<ul style="list-style-type: none"> <li>• Log and Trace</li> <li>• Logical Execution Time</li> <li>• Mappable Element</li> <li>• Security Event</li> <li>• Synchronization Points</li> <li>• Timed Communication</li> </ul> <p>Changed OSEK references</p> <p>Incorporated concepts as draft:</p> <ul style="list-style-type: none"> <li>• AUTOSAR Run-Time Interface</li> <li>• MCAL Multicore Distribution</li> <li>• Transport Layer Security</li> </ul>
2018-03-29	1.4.0	AUTOSAR Release Management	<p>Added terms:</p> <ul style="list-style-type: none"> <li>• Access Control Policy</li> <li>• Access Control Decision</li> <li>• MetaDataItem</li> <li>• Policy Decision Point (PDP)</li> <li>• Policy Enforcement Point (PEP)</li> <li>• Identity and Access Management (IAM)</li> <li>•</li> </ul> <p>Removed terms:</p> <ul style="list-style-type: none"> <li>• FlexRay Global Time</li> <li>• Meta Model</li> <li>• MetaDataLength</li> <li>• Model</li> <li>• Multiple Configuration Sets Shipping</li> <li>• Template</li> <li>• Variation Definition Time</li> <li>•</li> </ul> <p>Changed terms:</p> <ul style="list-style-type: none"> <li>• AUTOSAR Definition</li> <li>• AUTOSAR Metamodel</li> <li>• AUTOSAR Model</li> <li>• AUTOSAR Service</li> <li>• AUTOSAR XML description</li> <li>• Link Time Configuration</li> </ul>

Document Change History			
Date	Release	Changed by	Change Description
			<ul style="list-style-type: none"> <li>• Manifest</li> <li>• PDU MetaData</li> </ul>
2017-12-08	1.3.0	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• No content changes</li> </ul>
2017-10-27	1.2.0	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• No content changes</li> </ul>
2017-03-31	1.1.0	AUTOSAR Release Management	<p>Added terms:</p> <ul style="list-style-type: none"> <li>• Adaptability</li> <li>• Adaptive Application</li> <li>• Adaptive Platform Foundation</li> <li>• Adaptive Platform Services</li> <li>• ASIL Decomposition</li> <li>• Audit</li> <li>• AUTOSAR Adaptive Platform</li> <li>• AUTOSAR Runtime for Adaptive Applications</li> <li>• Cascaded Switch</li> <li>• Cascading Failure</li> <li>• Classic Platform</li> <li>• Common Cause Failure</li> <li>• Dependent Failures</li> <li>• Diagnostic Coverage</li> <li>• Diversity</li> <li>• Ethernet Switch Port Groups</li> <li>• Executable</li> <li>• External Port</li> <li>• Failure Mode</li> <li>• Fault Reaction Time</li> <li>• Fault Tolerant Time Interval</li> <li>• Freedom from Interference</li> <li>• Functional Cluster</li> <li>• Functional Safety Concept</li> <li>• Functional Safety Requirement</li> <li>• Host ECU</li> <li>• Host Port</li> <li>• Hypervisor</li> <li>• Independence</li> </ul>

Document Change History			
Date	Release	Changed by	Change Description
			<ul style="list-style-type: none"> <li>• Independent Failures</li> <li>• Internal Port</li> <li>• Link State Accumulation</li> <li>• Machine</li> <li>• Manifest</li> <li>• Master Switch</li> <li>• Microcontroller</li> <li>• Performance</li> <li>• Plausibility</li> <li>• Predictability</li> <li>• Proven In Use Argument</li> <li>• Recovery</li> <li>• Safe State</li> <li>• Safety Case</li> <li>• Safety Goal</li> <li>• Safety Measure</li> <li>• Safety Mechanism</li> <li>• Service Discovery</li> <li>• Service Instance</li> <li>• Service Interface</li> <li>• Service Oriented Communication</li> <li>• Service Proxy</li> <li>• Service Skeleton</li> <li>• Slave Switch</li> <li>• Software package</li> <li>• Software Unit</li> <li>• Systematic Fault</li> <li>• Uplink Port</li> <li>• Virtualization</li> </ul> <p>Removed terms:</p> <ul style="list-style-type: none"> <li>• Accreditation Body</li> <li>• Accreditation</li> <li>• Attestation</li> <li>• Conformance Declaration</li> <li>• Conformance Test Agency (CTA)</li> <li>• First party</li> <li>• Implementation Conformance Statement</li> </ul>

Document Change History			
Date	Release	Changed by	Change Description
			<ul style="list-style-type: none"> <li>• Interrupt Logic</li> <li>• Partial Model</li> <li>• Surveillance</li> <li>• Third party</li> </ul> <p>Changed terms:</p> <ul style="list-style-type: none"> <li>• Automotive Safety Integrity Levels</li> <li>• Availability</li> <li>• Acceptance Test Suite</li> <li>• Electronic Control Unit</li> <li>• Error</li> <li>• Fail-safe</li> <li>• Fail-silent</li> <li>• Failure Rate</li> <li>• Failure</li> <li>• Fault Tolerance</li> <li>• Fault</li> <li>• FlexRay Bus</li> <li>• FlexRay Cycle</li> <li>• FlexRay L-PDU-Identifier</li> <li>• FlexRay L-SDU-Identifier</li> <li>• FlexRay Matrix</li> <li>• FlexRay Slot Multiplexing</li> <li>• Graceful Degradation</li> <li>• Fail-degraded</li> <li>• Implementation Conformance Class 1 (ICC1)</li> <li>• Implementation Conformance Class 2 (ICC2)</li> <li>• Implementation Conformance Class 3 (ICC3)</li> <li>• Link Time Configuration</li> <li>• Partitioning</li> <li>• Protocol Control Information</li> <li>• Protocol Data Unit</li> <li>• Post-build Time Configuration</li> <li>• Pre-Compile Time Configuration</li> <li>• Probability of Failure</li> <li>• Redundancy</li> </ul>

Document Change History			
Date	Release	Changed by	Change Description
			<ul style="list-style-type: none"> <li>• Risk</li> <li>• Safety</li> <li>• Service Data Unit</li> </ul>
2016-11-30	1.0.0	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• --Migration of document to standard "Foundation"--</li> <li>• Following terms added:</li> <li>• AUTOSAR Blueprint (3.19)</li> <li>• Bypassing (3.38)</li> <li>• Hook (3.137)</li> <li>• OS Event (3.174)</li> <li>• Post-build Hooking (3.185)</li> <li>• Pre-build Hooking (3.187)</li> <li>• Rapid Prototyping (RP) (3.195)</li> <li>• Rapid Prototyping Memory Interface (3.196)</li> <li>• Rapid Prototyping Tool (3.197)</li> <li>• Reentrancy (3.200)</li> <li>• Standardized AUTOSAR Blueprint (3.236)</li> <li>• Standardized Blueprint (3.238)</li> <li>• Following terms changed:</li> <li>• Asset (3.11)</li> <li>• Asynchronous Function (3.13)</li> <li>• AUTOSAR Application Interface (3.17)</li> <li>• Availability (3.31)</li> <li>• ECU Abstraction Layer (3.77)</li> <li>• Feature (3.100)</li> <li>• Function (3.127)</li> <li>• Microcontroller Abstraction Layer (MCAL) (3.162)</li> </ul>
2015-07-31	4.2.2	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Following terms changed:</li> <li>• ECU Abstraction Layer (3.77)</li> <li>• Standardized AUTOSAR Interface (3.237)</li> <li>• Following terms removed:</li> <li>• Software Module</li> </ul>

<b>Document Change History</b>			
<b>Date</b>	<b>Release</b>	<b>Changed by</b>	<b>Change Description</b>
2014-10-31	4.2.1	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Following terms changed:</li> <li>• Data Variant Coding (3.67)</li> <li>• OS-Application (3.173)</li> <li>• Post-build time configuration (3.186)</li> <li>• Standardized AUTOSAR Interface (3.237)</li> </ul>
2014-03-31	4.1.3	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Extended Abbreviations (0)</li> <li>• Following terms changed:</li> <li>• Software Component (SW-C) (3.229)</li> </ul>
2013-03-15	4.1.1	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Extended Abbreviations (0)</li> <li>• Following terms added:</li> <li>• Application Interface (3.4)</li> <li>• Asynchronous Functions (3.13)</li> <li>• AUTOSAR Application Interface (3.17)</li> <li>• Dynamic PDU (3.73)</li> <li>• Life Cycle (3.155)</li> <li>• MetaDataLength (3.161)</li> <li>• PDU MetaData (3.179)</li> <li>• Pretended Networking (3.189)</li> <li>• Synchronous Functions (3.245)</li> </ul>
2011-12-22	4.0.3	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Extended Abbreviations (0)</li> <li>• Following terms added:</li> <li>• Callback (3.40)</li> <li>• Callout (3.41)</li> <li>• ECU (3.76)</li> </ul>
2009-12-18	4.0.1	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Following terms added:</li> <li>• AUTOSAR Partial Model (3.25)</li> <li>• Bus Wake-Up (3.37)</li> <li>• Empty Function (3.82)</li> </ul>

<b>Document Change History</b>			
<b>Date</b>	<b>Release</b>	<b>Changed by</b>	<b>Change Description</b>
2010-02-02	3.1.4	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Following terms added:</li> <li>• Automotive Safety Integrity Levels (ASIL) (3.16)</li> <li>• Bit Position (3.34)</li> <li>• Category 1 Interrupt (3.43)</li> <li>• Category 2 Interrupt (3.44)</li> <li>• Code Generator (3.50)</li> <li>• Coordinate (3.63)</li> <li>• E2E Profile (3.75)</li> <li>• Error Detection Rate (3.85)</li> <li>• Failure Rate (3.95)</li> <li>• ICC1 (Implementation Conformance Class 1) (3.139)</li> <li>• ICC2 (Implementation Conformance Class 2) (3.140)</li> <li>• ICC3 (Implementation Conformance Class 3) (3.141)</li> <li>• Interrupt Frames (3.148)</li> <li>• Interrupt Handler (3.149)</li> <li>• Interrupt Logic (3.150)</li> <li>• Meta Model (3.159)</li> <li>• Mode (3.164)</li> <li>• Model (3.165)</li> <li>• Network Interface (NWI) (3.169)</li> <li>• NM Coordination Cluster (3.170)</li> <li>• NM Coordinator (3.171)</li> <li>• Rate Conversion (3.198)</li> <li>• Residual Error Rate (3.205)</li> <li>• SAE J1939 (3.213)</li> <li>• Safety Protocol (3.215)</li> <li>• Software Component Interface (SW-CI) (3.230)</li> <li>• Synchronize (3.243)</li> <li>• Variability (3.256)</li> <li>• Variant (3.257)</li> <li>• Variation Binding (3.259)</li> <li>• Variation Binding Time (3.260)</li> </ul>



<b>Document Change History</b>			
<b>Date</b>	<b>Release</b>	<b>Changed by</b>	<b>Change Description</b>
			<ul style="list-style-type: none"> <li>• Variation Definition Time (3.261)</li> <li>• Variation Point (3.262)</li> <li>• Formal adaptations</li> <li>• Legal disclaimer revised</li> </ul>
2008-08-13	3.1.1	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Legal disclaimer revised</li> </ul>
2007-12-21	3.0.1	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Following terms added:</li> <li>• Debugging (3.69)</li> <li>• Implementation Conformance Statement (3.142)</li> <li>• Document meta information extended</li> <li>• Small layout adaptations made</li> </ul>
2007-01-24	2.1.15	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• “Advice for users” revised</li> <li>• “Revision Information” added</li> </ul>
2007-11-28	2.1.14	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Following terms added:</li> <li>• FlexRay (3.104)</li> <li>• Vendor ID (3.263)</li> <li>• Callback (3.40)</li> <li>• Interrupt frames (3.148)</li> <li>• Interrupt vector table(3.152)</li> <li>• Accreditation (3.1)</li> <li>• Accreditation Body (3.2)</li> <li>• Conformance Test Agency (3.59)</li> <li>• Assessment (3.10)</li> <li>• Surveillance (3.242)</li> <li>• Attestation (3.14)</li> <li>• (Conformance) Declaration (3.70)</li> <li>• First party and (3.101)</li> <li>• Third party (3.252)</li> <li>• Safety (3.214)</li> <li>• ECU Configuration (3.78)</li> <li>• ECU Configuration Description (3.79)</li> <li>• Legal disclaimer revised</li> </ul>
2006-05-16	2.0	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• removed and added some terms</li> <li>• rework of several descriptions</li> <li>• and some formal changes</li> </ul>

<b>Document Change History</b>			
<b>Date</b>	<b>Release</b>	<b>Changed by</b>	<b>Change Description</b>
2005-05-31	1.0	AUTOSAR Administration	<ul style="list-style-type: none"><li>• Initial Release</li></ul>

## Disclaimer

This work (specification and/or software implementation) and the material contained in it, as released by AUTOSAR, is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the work.

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

This work may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only. For any other purpose, no part of the work may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

## Table of Contents

1	Abbreviations.....	19
2	Introduction.....	23
3	How to read this document .....	24
3.1	<Term> Template .....	24
4	Definitions.....	25
4.1	Abstract Platform .....	25
4.2	Acceptance Test Suite.....	25
4.3	Access Control Decision.....	25
4.4	Access Control Policy .....	25
4.5	Adaptability .....	26
4.6	Adaptive Application .....	26
4.7	Adaptive Platform Foundation .....	26
4.8	Adaptive Platform Services .....	27
4.9	Application .....	27
4.10	Application Interface .....	27
4.11	Application Programming Interface .....	27
4.12	Application Software Component.....	28
4.13	Architecture.....	28
4.14	Artifact.....	28
4.15	ASIL Decomposition .....	28
4.16	Asserted Property .....	29
4.17	Assessment .....	29
4.18	Asset .....	29
4.19	Asynchronous Communication.....	29
4.20	Asynchronous Function .....	30
4.21	Atomic Software Component.....	30
4.22	Audit.....	30
4.23	Automotive Safety Integrity Levels .....	30
4.24	AUTOSAR Adaptive Platform.....	30
4.25	AUTOSAR Application Interface.....	31
4.26	AUTOSAR Authoring Tool.....	31
4.27	AUTOSAR Blueprint .....	31
4.28	AUTOSAR Converter Tool .....	32
4.29	AUTOSAR Definition .....	32
4.30	AUTOSAR Interface .....	32
4.31	AUTOSAR Metamodel.....	32
4.32	AUTOSAR Model.....	33
4.33	AUTOSAR Partial Model .....	33
4.34	AUTOSAR Processor Tool .....	33
4.35	AUTOSAR Runtime for Adaptive Applications.....	33
4.36	AUTOSAR Run-Time Interface .....	34
4.37	AUTOSAR Service .....	34
4.38	AUTOSAR Tool.....	34
4.39	AUTOSAR XML description .....	34
4.40	AUTOSAR XML Schema.....	35

4.41	Availability .....	35
4.42	Basic Software .....	36
4.43	Basic Software Module .....	36
4.44	Bit Position .....	36
4.45	Blueprint .....	36
4.46	Bulk Data .....	37
4.47	Bus Mirroring.....	37
4.48	Bus Wake-Up.....	37
4.49	Bypassing .....	38
4.50	Calibration.....	38
4.51	Call Point.....	38
4.52	Callback .....	38
4.53	Callout.....	39
4.54	Cascaded Switch .....	39
4.55	Cascading Failure.....	40
4.56	Category 1 Interrupt.....	40
4.57	Category 2 Interrupt.....	40
4.58	Causality of Transmission .....	40
4.59	Classic Platform .....	41
4.60	Client .....	41
4.61	Client-Server Communication.....	41
4.62	Client-Server Interface.....	42
4.63	Cluster Signal.....	42
4.64	Code Generator .....	42
4.65	Code Variant Coding .....	42
4.66	Common Cause Failure.....	43
4.67	Communication Attribute .....	43
4.68	Complex Driver .....	43
4.69	Composition .....	43
4.70	Compositionality.....	44
4.71	Conditioned Signal.....	44
4.72	Configuration.....	44
4.73	Confirmation.....	45
4.74	Connector .....	45
4.75	Control Flow.....	45
4.76	Coordinate .....	45
4.77	Data.....	46
4.78	Data Element .....	46
4.79	Data Flow .....	46
4.80	Data Variant Coding .....	46
4.81	Deadline .....	47
4.82	Debugging.....	47
4.83	Dependability .....	47
4.84	Dependent Failure .....	47
4.85	Diagnostic Coverage .....	48
4.86	Diagnostic Event.....	48
4.87	Diversity .....	48
4.88	Dynamic PDU .....	48
4.89	Dynamic Routing.....	49
4.90	E2E Profile .....	49

4.91	E2E protection alive counter.....	49
4.92	E2E protection sequence counter .....	49
4.93	ECU Abstraction Layer .....	50
4.94	ECU Configuration .....	50
4.95	ECU Configuration Description.....	50
4.96	Electrical Signal .....	51
4.97	Electronic Control Unit .....	51
4.98	Empty Function .....	51
4.99	Entry Point .....	51
4.100	Error.....	52
4.101	Error Detection Rate .....	52
4.102	Ethernet Switch Port Groups .....	52
4.103	Event .....	52
4.104	Event Message (SOME/IP) .....	53
4.105	Executable.....	53
4.106	Executable Entity Cluster .....	53
4.107	Execution Order Constraint.....	54
4.108	Execution Time .....	54
4.109	Exit Point .....	54
4.110	External Port.....	55
4.111	Fail-operational .....	55
4.112	Fail-safe.....	55
4.113	Fail-silent .....	55
4.114	Failure Mode .....	56
4.115	Failure.....	56
4.116	Failure Rate.....	56
4.117	Fault.....	56
4.118	Fault Detection .....	57
4.119	Fault Reaction .....	57
4.120	Fault Reaction Time .....	57
4.121	Fault Tolerance .....	58
4.122	Fault Tolerant Time Interval .....	58
4.123	Feature .....	58
4.124	Flag.....	58
4.125	FlexRay .....	58
4.126	Foundation .....	59
4.127	Frame .....	59
4.128	Frame PDU .....	59
4.129	Freedom from Interference .....	60
4.130	Function.....	60
4.131	Functional Cluster .....	60
4.132	Functional Network .....	60
4.133	Functional Safety Concept.....	61
4.134	Functional Safety Requirement.....	61
4.135	Functional Unit .....	61
4.136	Functionality .....	61
4.137	Gateway .....	62
4.138	Gateway ECU.....	62
4.139	Graceful Degradation .....	62
4.140	Hardware Connection .....	62

4.141	Hardware Element .....	63
4.142	Hardware Interrupt .....	63
4.143	Hardware Port .....	63
4.144	Health Indicator .....	63
4.145	Hook .....	64
4.146	Host ECU .....	64
4.147	Host Port .....	64
4.148	Hypervisor .....	64
4.149	Identity and Access Management (IAM) .....	65
4.150	Identity Information .....	65
4.151	Implementation Conformance Class 1 (ICC1) .....	65
4.152	Implementation Conformance Class 2 (ICC2) .....	66
4.153	Implementation Conformance Class 3 (ICC3) .....	67
4.154	Independence .....	67
4.155	Independent Failures .....	68
4.156	Indication .....	68
4.157	Integration .....	68
4.158	Integration Code .....	68
4.159	Interface .....	68
4.160	Internal Port .....	69
4.161	Interrupt Frames .....	69
4.162	Interrupt Handler .....	69
4.163	Interrupt Service Routine .....	70
4.164	Interrupt Vector Table .....	70
4.165	Interrupt .....	70
4.166	Intrusion Detection System .....	70
4.167	Invalid Flag .....	71
4.168	Invalid Value of Signal .....	71
4.169	I-PDU .....	71
4.170	Life Cycle .....	71
4.171	LIN Bus Idle .....	72
4.172	Link State Accumulation .....	72
4.173	Link Time Configuration .....	72
4.174	Logical Execution Time .....	72
4.175	Log and Trace .....	73
4.176	Machine .....	73
4.177	Manifest .....	74
4.178	Mappable Element .....	74
4.179	Mapping .....	74
4.180	Master Switch .....	74
4.181	MCAL Signal .....	75
4.182	Metadata .....	75
4.183	MetaDataltem .....	75
4.184	Microcontroller .....	76
4.185	Microcontroller Abstraction Layer .....	76
4.186	Mistake .....	76
4.187	Mode .....	77
4.188	Multimedia Stream .....	77
4.189	Multiplexed PDU .....	77
4.190	Network Interface .....	78

4.191	NM Coordination Cluster.....	78
4.192	NM Coordinator .....	79
4.193	Notification.....	79
4.194	Onboard Security Event.....	79
4.195	OS Application.....	79
4.196	OS Event .....	80
4.197	Partitioning .....	80
4.198	Protocol Control Information .....	80
4.199	Protocol Data Unit .....	81
4.200	PDU MetaData .....	81
4.201	PDU Timeout.....	82
4.202	Performance.....	82
4.203	Peripheral Hardware .....	82
4.204	Personalization.....	82
4.205	Plausibility .....	83
4.206	Policy Decision Point (PDP).....	83
4.207	Policy Enforcement Point (PEP) .....	83
4.208	Port.....	83
4.209	Port Interface.....	84
4.210	Post-build Time Configuration.....	85
4.211	Post-build Hooking .....	85
4.212	Pre-build Hooking.....	85
4.213	Pre-Compile Time Configuration.....	85
4.214	Predictability .....	86
4.215	Pretended Networking.....	86
4.216	Private Interface .....	86
4.217	Probability of Failure .....	86
4.218	Procedure Call.....	87
4.219	Process .....	87
4.220	Processed Manifest.....	87
4.221	Proven In Use Argument.....	87
4.222	Provide Port.....	88
4.223	Rapid Prototyping.....	88
4.224	Rapid Prototyping Memory Interface .....	88
4.225	Rapid Prototyping Tool.....	88
4.226	Rate Conversion .....	89
4.227	Raw Data Stream.....	89
4.228	Recovery .....	89
4.229	Redundancy .....	89
4.230	Reentrancy .....	90
4.231	Reliability .....	90
4.232	Relocatability .....	90
4.233	Require Port .....	91
4.234	Required property .....	91
4.235	Residual Error Rate.....	91
4.236	Resource .....	91
4.237	Resource-Management.....	92
4.238	Response Time .....	92
4.239	Risk.....	92
4.240	Robustness .....	92



4.241	RTE Event .....	93
4.242	Runnable Entity .....	93
4.243	SAE J1939 .....	93
4.244	Safe State .....	94
4.245	Safety .....	94
4.246	Safety Case .....	94
4.247	Safety Goal .....	94
4.248	Safety Measure .....	95
4.249	Safety Mechanism .....	95
4.250	Safety Protocol .....	95
4.251	Sample Application .....	95
4.252	Scalability .....	96
4.253	Scheduler .....	96
4.254	Service Data Unit .....	96
4.255	Security .....	96
4.256	Secure Channel .....	97
4.257	Security Event .....	97
4.258	Sender-Receiver Communication .....	97
4.259	Sender-Receiver Interface .....	97
4.260	Sensor/Actuator SW-Component .....	98
4.261	Server .....	98
4.262	Service .....	98
4.263	Service Discovery .....	98
4.264	Service Instance .....	99
4.265	Service Interface .....	99
4.266	Service Oriented Communication .....	99
4.267	Service Port .....	100
4.268	Service Proxy .....	100
4.269	Service Skeleton .....	100
4.270	Services Layer .....	101
4.271	Signal Service Translation .....	101
4.272	Slave Switch .....	101
4.273	Software Cluster .....	102
4.274	Software Component .....	102
4.275	Software Component Interface .....	102
4.276	Software Configuration .....	103
4.277	Software Interrupt .....	103
4.278	Software Package .....	103
4.279	Software Signal .....	103
4.280	Software Unit .....	104
4.281	Special Periphery Access .....	104
4.282	Standard Periphery Access .....	104
4.283	Standard Software .....	104
4.284	Standardized AUTOSAR Blueprint .....	104
4.285	Standardized AUTOSAR Interface .....	105
4.286	Standardized Blueprint .....	105
4.287	Standardized Interface .....	105
4.288	Static Configuration .....	106
4.289	Synchronization Points .....	106
4.290	Synchronize .....	107

4.291	Synchronous Communication .....	107
4.292	Synchronous Function .....	107
4.293	System .....	107
4.294	System Constraint.....	108
4.295	System Health Monitor.....	108
4.296	System Signal .....	108
4.297	Systematic Fault.....	108
4.298	Task.....	109
4.299	Technical Signal.....	109
4.300	Timed Communication .....	109
4.301	Timeout .....	110
4.302	Uplink Port.....	110
4.303	Use Case.....	110
4.304	Validation.....	110
4.305	Variability.....	111
4.306	Variant .....	111
4.307	Variant Coding.....	111
4.308	Variation Binding .....	112
4.309	Variation Binding Time .....	112
4.310	Variation Point.....	112
4.311	Vehicle State Manager.....	112
4.312	Vendor ID .....	113
4.313	Verification.....	113
4.314	VFB View.....	113
4.315	Virtual Functional Bus .....	114
4.316	Virtual Integration .....	114
4.317	Virtualization.....	114
4.318	Wake-up and sleep on dataline .....	114
4.319	Worst Case Execution Time .....	115
4.320	Worst Case Response Time .....	115
Annex 1: Literature .....		116

## 1 Abbreviations

<b><i>Abbreviation</i></b>	<b><i>Description</i></b>
<b>AA</b>	Adaptive Application
<b>ADC</b>	Analog Digital Converter
<b>AMM</b>	Application Mode Management
<b>AP</b>	AUTOSAR Adaptive Platform
<b>API</b>	Application Programming Interface
<b>ARA</b>	AUTOSAR Runtime for Adaptive Applications
<b>ARP</b>	Address Resolution Protocol
<b>ARTI</b>	AUTOSAR Run-Time Interface
<b>ASAM</b>	Association for Standardization of Automation and Measuring systems
<b>ASD</b>	Abstract System Description
<b>ASIL</b>	Automotive Safety Integrity Levels
<b>ASW</b>	Application SoftWare
<b>ATS</b>	Acceptance Test Suite
<b>AUTOSAR</b>	AUTomotive Open System Architecture
<b>BFx</b>	Bitfield functions for fixed point
<b>BSW</b>	Basic Software
<b>BSWM</b>	Basic SoftWare Mode manager
<b>BSWMD</b>	Basic SoftWare Module Description
<b>CAN</b>	Controller Area Network
<b>CCF</b>	Common Cause Failure
<b>CDD</b>	Complex Driver
<b>CP</b>	Classic Platform
<b>COM</b>	Communication
<b>CPU</b>	Central Processing Unit
<b>CRC</b>	Cyclic Redundancy Check
<b>DAC</b>	Digital to Analog Converter
<b>DEM</b>	Diagnostic Event Manager
<b>DET</b>	Development Error Tracer
<b>DHCP</b>	Dynamic Host Configuration Protocol
<b>DIO</b>	Digital Input/Output

<b>DLC</b>	Data Length Code
<b>DoIP</b>	Diagnostics over Internet Protocol
<b>DTD</b>	Document Type Definition
<b>E2E</b>	End to End
<b>ECU</b>	Electronic Control Unit
<b>EE</b>	Executable Entities
<b>EEC</b>	Executable Entity Cluster
<b>EEPROM</b>	Electrically Erasable Programmable Read-Only Memory
<b>EOC</b>	Execution Order Constraint
<b>EOCEERG</b>	Execution Order Constraint Executable Entity Reference Group
<b>FIFO</b>	First In First Out
<b>FPU</b>	Floating Point Unit
<b>FW</b>	Fire Wire
<b>GPT</b>	General Purpose Timer
<b>GSM</b>	Global System for Mobile Communication
<b>HW</b>	Hardware
<b>I-PDU</b>	Interaction Layer Protocol Data Unit
<b>ICC</b>	Implementation Conformance Class
<b>ICMP</b>	Internet Control Message Protocol
<b>ICOM</b>	Intelligent COMMunication controller
<b>ICU</b>	Input Capture Unit
<b>IDL</b>	Interface Description Language
<b>IEC</b>	International Electrotechnical Commission
<b>IFI</b>	Interpolation Floating point
<b>IFx</b>	Interpolation Fixed point
<b>IO</b>	Input/ Output
<b>ISR</b>	Interrupt Service Routine
<b>L-PDU</b>	Protocol Data Unit of the data Link layer
<b>L-SDU</b>	SDU of the data Link layer
<b>LET</b>	Logical Execution Time
<b>LIFO</b>	Last In First Out
<b>LIN</b>	Local Interconnected Network
<b>LT</b>	Log and Trace
<b>LSB</b>	Least Significant Bit

<b>μC</b>	MicroController
<b>MCAL</b>	Microcontroller Abstraction Layer
<b>MCU</b>	Micro Controller Unit
<b>ME</b>	Mappable Element
<b>MFI</b>	Mathematical Floating point
<b>MFx</b>	Math – Fixed Point
<b>MIPS</b>	Million Instructions Per Second
<b>MMU</b>	Memory Management Unit
<b>MMI</b>	Man Machine Interface
<b>MOST</b>	Media Oriented Systems Transport
<b>μP</b>	MicroProcessor
<b>MPU</b>	Memory Protection Unit
<b>MSB</b>	Most Significant Bit
<b>N-PDU</b>	Protocol Data Unit of the Network layer (transport protocols)
<b>N-SDU</b>	SDU of the Network layer (transport protocols)
<b>NVRAM</b>	Non-Volatile Random Access Memory
<b>OEM</b>	Original Equipment Manufacturer
<b>OIL</b>	ISO 17356-6 (OSEK/VDX Implementation Language)
<b>OS</b>	Operating System
<b>OSEK</b>	Open Systems and the Corresponding Interfaces for Automotive Electronics
<b>PCI</b>	Protocol Control Information
<b>PDU</b>	Protocol Data Unit
<b>PS</b>	Product Supplier
<b>PWM</b>	Pulse Width Modulation
<b>RAM</b>	Random Access Memory
<b>RDG</b>	Runnable Dependency Graph
<b>RfC</b>	Request for Change
<b>RP</b>	Rapid Prototyping
<b>RTE</b>	Runtime Environment
<b>SAE</b>	Society of Automotive Engineers
<b>SDU</b>	Service Data Unit
<b>SIL</b>	Safety Integrity Level
<b>SOA</b>	Service Oriented Architecture

<b>SP</b>	Synchronization Point
<b>SPI</b>	Serial Peripheral Interface
<b>SW</b>	Software
<b>SW-C</b>	Software Component
<b>SWS</b>	Software Specification
<b>TC</b>	Timed Communication
<b>TCP</b>	Transmission Control Protocol
<b>TP</b>	Transport Protocol
<b>TTCAN</b>	Time Triggered CAN
<b>TTP</b>	Time Triggered Protocol
<b>UDP</b>	User (Universal) Datagram Protocol
<b>UdpNm</b>	UDP Network Management
<b>USB</b>	Universal Serial Bus
<b>VFB</b>	Virtual Functional Bus
<b>VSM</b>	Vehicle State Manager
<b>VISS</b>	Vehicle Information Service Specification
<b>VMM</b>	Vehicle Mode Management
<b>WCET</b>	Worst Case Execution Time
<b>WCRT</b>	Worst Case Response time
<b>W3C</b>	World Wide Web Consortium
<b>XCP</b>	Universal Calibration Protocol
<b>XML</b>	Extensible Markup Language
<b>XSD</b>	XML Schema Definition

## 2 Introduction

This document is the overall glossary of AUTOSAR. It contains definitions of all major terms and notions used within AUTOSAR. It does not claim to be complete and please keep in mind that some WPs have more specific terms defined within their domain specific glossary.

### 3 How to read this document

The title of the subchapters is identical to the term to be defined.

#### 3.1 <Term> Template

<b>Definition</b>	<i>&lt;term to be defined&gt;</i>
<b>Initiator</b>	<i>&lt;functional cluster which responsible for the term&gt;</i>
<b>Further Explanations</b>	<i>&lt;further explanation of the definition&gt;</i>
<b>Comment</b>	<i>&lt;comment or hints&gt;</i>
<b>Example</b>	<i>&lt;example of the term&gt;</i>
<b>Reference</b>	<i>&lt;reference of definition&gt;</i>



## 4 Definitions

### 4.1 Abstract Platform

<b>Definition</b>	A Software Platform defined by AUTOSAR for the definition of functional level communications independent of Classic / Adaptive / Offboard deployments.
<b>Initiator</b>	System Design
<b>Further Explanations</b>	An Abstract Platform exists only in the methodological and modeling sense and is not a physical deployable platform. In the context of AUTOSAR Classic and AUTOSAR Adaptive functional level communications means VFB level communications.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

### 4.2 Acceptance Test Suite

<b>Definition</b>	A test case description used in the context of Acceptance Testing
<b>Initiator</b>	Acceptance Testing
<b>Further Explanations</b>	ISO 9646 distinguishes between Abstract Test Suites and Executable Test Suites. For AUTOSAR the earlier relates to the Acceptance Test Specifications, whereas the latter to the test implementations or Acceptance Test Suites.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	ISO 9646, Parts 1,2 and 4

### 4.3 Access Control Decision

<b>Definition</b>	The Access Control Decision is a Boolean value indicating if the requested operation is permitted or not. It is based on the identity of the caller and the Access Control Policy.
<b>Initiator</b>	Security
<b>Further Explanations</b>	In the case of IAM, the 'caller' is an Adaptive Application
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

### 4.4 Access Control Policy

<b>Definition</b>	Access Control Policies are bound to the targets of calls (i.e. Service interfaces) and are used to express what Identity Information are necessary to access those interfaces.
<b>Initiator</b>	Security
<b>Further Explanations</b>	Policies can be provided through configuration / modeling or by statically pre-programming them into the PDP.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.5 Adaptability

<b>Definition</b>	Adaptability is the ability of a system to adjust itself to changed circumstances in its environment in order to continue to provide the intended functionality.
<b>Initiator</b>	Safety
<b>Further Explanations</b>	One should distinguish between changes in the environment of the system/vehicle ("run-time adaptability") and changes in the development environment where software architecture (like AUTOSAR) is used ("design-time predictability").
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	Antonio Carlos Schneider Beck, Carlos Arthur Lang Lisboa, Luigi Carro (eds.), Adaptable Embedded Systems, Springer Science & Business Media, 27 Nov 2012 Twan Basten, Roelof Hamberg, Frans Reckers, Jacques Verriet, Model-Based Design of Adaptive Embedded Systems, Springer Science & Business Media, 15 Mar 2013

## 4.6 Adaptive Application

<b>Definition</b>	Software that follows the Adaptive AUTOSAR specifications and therefore can be deployed onto an Adaptive Platform instance. It consists of its implementation, operational data (e.g. map data) and its metadata given by the Application Design Model. An Adaptive Application contains at least one executable. In order to be deployable on different Adaptive Platforms, it only uses ARA programming interfaces.
<b>Initiator</b>	Execution Management
<b>Further Explanations</b>	Adaptive Applications are generally more coarse grain than SW-Cs of the Classic Platform. They use exclusively Adaptive Platform APIs, and may offer and use services. They are implemented by one or several executables, byte code or libraries with defined entry points and may comprise multiple parts (e.g. libraries, data files).
<b>Comment</b>	The goal of Adaptive Platform is to achieve portability of Adaptive Applications among different implementations of the Adaptive Platform at least on source-code level, potentially also on object-code level.
<b>Example</b>	--
<b>Reference</b>	--

## 4.7 Adaptive Platform Foundation

<b>Definition</b>	Part of an Adaptive Platform implementation, which provides standardized platform functionality to Applications via software interfaces (APIs).
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	The Adaptive Platform Foundation includes of core system functionalities such as OS, Execution Manager, Communication Management and Persistency.
<b>Comment</b>	The goal of Adaptive Platform is to achieve portability of Adaptive Applications among different implementations of the Adaptive Platform at least on source-code level, potentially also on object-code level.
<b>Example</b>	--
<b>Reference</b>	--

## 4.8 Adaptive Platform Services

<b>Definition</b>	Standard platform services that is provided by an application which is part of AUTOSAR platform implementation.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.9 Application

<b>Definition</b>	A software (or program) that is specified to the solution of a problem of an end user requiring information processing for its solution. The Software Configuration of a software entity.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	To 1. of Definition: In AUTOSAR Application software is located above the AUTOSAR RTE (RunTimeEnvironment).
<b>Comment</b>	Definition 1 is the “by default” meaning for application in AUTOSAR. When definition 2 is meant, it has to be explicitly mentioned.
<b>Example</b>	--
<b>Reference</b>	[ISO 2382-20]

## 4.10 Application Interface

<b>Definition</b>	A Port Interface used by a Software Component as specified in the software component template.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	[TPS_SoftwareComponentTemplate]

## 4.11 Application Programming Interface

<b>Definition</b>	An Application Programming Interface (API) is the prescribed method of a specific software part by which a programmer writing a program can make requests to that software part.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	ISO 17356-3 (OSEK/VDX Operating System)
<b>Reference</b>	--

## 4.12 Application Software Component

<b>Definition</b>	An Application Software Component is a specific Software Component which realizes a defined functionality on application level and runs on the AUTOSAR infrastructure. It communicates only through the AUTOSAR Runtime Environment.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	Application Software Components are located "above" the AUTOSAR Runtime Environment.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.13 Architecture

<b>Definition</b>	The fundamental organization of a system embodied in its components, their static and dynamic relationships to each other, and to the environment, and the principles guiding its design and evolution.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	"Static and dynamic" added to EAST definition.
<b>Example</b>	--
<b>Reference</b>	[IEEE 1471], [EAST-Glossary]

## 4.14 Artifact

<b>Definition</b>	This is a Work Product Definition that provides a description and definition for tangible work product types. Artifacts may be composed of other artifacts ([14]). At a high level, an artifact is represented as a single conceptual file.
<b>Initiator</b>	Methodology and Templates
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.15 ASIL Decomposition

<b>Definition</b>	See ISO 26262, Part 1, ID 1.7
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	ISO 26262, Part 1, ID 1.7

## 4.16 Asserted Property

<b>Definition</b>	A property or quality of a design entity (e.g. SW component or system) is asserted, if the design entity guarantees that this property or quality is fulfilled.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	A property or quality of a design unit can be asserted by the design unit itself or in combination with another design unit.
<b>Comment</b>	--
<b>Example</b>	If the worst case execution time of a task (w.r.t. a certain CPU etc.) is asserted to be 3 ms, the execution time of this task will under any circumstances be less than or equal to 3 ms.
<b>Reference</b>	Compare Required property

## 4.17 Assessment

<b>Definition</b>	See ISO 26262, Part 1, ID 1.4
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	ISO 26262, Part 1, ID 1.4

## 4.18 Asset

<b>Definition</b>	An item that has been designed for use in multiple contexts.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	An asset can be design, specifications, source code, documentation, test suits, manual procedures, etc.. From a security perspective anything that has a value to any of the stakeholders such as critical data (information, software) and critical functions, that could potentially be subject to attacks and possibly, but not necessarily, motivates countermeasures.
<b>Reference</b>	[IEEE 1517], [EAST-Glossary]

## 4.19 Asynchronous Communication

<b>Definition</b>	Asynchronous communication does not block the sending software entity. The sending software entity continues its operation without getting a response from the communication partner(s).
<b>Initiator</b>	Communication
<b>Further Explanations</b>	There could be an acknowledgement by the communication system about the sending of the information. A later response to the sending software entity is possible.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.20 Asynchronous Function

<b>Definition</b>	A Function is called asynchronous if the described functionality is not guaranteed to be completed the moment the function returns to the caller.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.21 Atomic Software Component

<b>Definition</b>	Non-composed Software-Component.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	An Atomic Software Component might access HW or not, therefore not all Atomic SW-Cs are relocatable.
<b>Comment</b>	--
<b>Example</b>	Application Software-Component, Complex Driver
<b>Reference</b>	--

## 4.22 Audit

<b>Definition</b>	See ISO 26262, Part 1, ID 1.5
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	ISO 26262, Part 1, ID 1.5

## 4.23 Automotive Safety Integrity Levels

<b>Definition</b>	See ISO 26262, Part 1, ID 1.7
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	ISO 26262, Part 1, ID 1.7

## 4.24 AUTOSAR Adaptive Platform

<b>Definition</b>	An adaptive computing platform standardized by AUTOSAR. In a narrow term, it
-------------------	--

	refers to its specification. In a broad term, it may refer to an instance of Adaptive Platform implementation.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.25 AUTOSAR Application Interface

<b>Definition</b>	A set of Blueprints which are standardized by AUTOSAR and which can be used for creating AUTOSAR Interfaces of an Application. AUTOSAR interfaces that are derived from Standardized Blueprints are Standardized AUTOSAR Interfaces.
<b>Initiator</b>	Application Interfaces
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	[EXP_AIUserGuide]

## 4.26 AUTOSAR Authoring Tool

<b>Definition</b>	An AUTOSAR Tool used to create and modify AUTOSAR XML descriptions.
<b>Initiator</b>	Methodology and Templates
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	System Description Editor
<b>Reference</b>	--

## 4.27 AUTOSAR Blueprint

<b>Definition</b>	An AUTOSAR Blueprint is a Blueprint for an AUTOSAR element. It also includes that it is specified within the AUTOSAR project which attributes are mandatory to be specified for the blueprint of a specific class of AUTOSAR element types as well as how to derive an AUTOSAR object from that blueprint.
<b>Initiator</b>	Application Interfaces
<b>Further Explanations</b>	The AUTOSAR meta-model supports the pre-definition of model elements taken as the basis for further modeling. These pre-definitions are called blueprints. [TPS_STDT_00002]
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	[TPS Standardization Template]

## 4.28 AUTOSAR Converter Tool

<b>Definition</b>	An AUTOSAR Tool used to create AUTOSAR XML files by converting information from other AUTOSAR XML files.
<b>Initiator</b>	Methodology and Templates
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	ECU Flattener
<b>Reference</b>	--

## 4.29 AUTOSAR Definition

<b>Definition</b>	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.
<b>Initiator</b>	Methodology and Templates
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.30 AUTOSAR Interface

<b>Definition</b>	The AUTOSAR Interface of a software component refers to the collection of all Ports of that component through which it interacts with other components.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	Note that an AUTOSAR Interface is different from a Port Interface. The latter characterizes one specific port of a component.
<b>Example</b>	--
<b>Reference</b>	[AUTOSAR Specification of Virtual Functional Bus], Chapter “ Modeling of Communication, Graphical Notation”

## 4.31 AUTOSAR Metamodel

<b>Definition</b>	The AUTOSAR metamodel is a UML2.0 model that defines the language for describing AUTOSAR systems and related artifacts.
<b>Initiator</b>	Methodology and Templates
<b>Further Explanations</b>	--
<b>Comment</b>	The AUTOSAR XML Schema is derived from the AUTOSAR metamodel.
<b>Example</b>	--
<b>Reference</b>	[UML 2.0]



### 4.32 AUTOSAR Model

<b>Definition</b>	This is an instance of the AUTOSAR Metamodel. The AUTOSAR Model represents aspects suitable to the intended use according to the AUTOSAR methodology.
<b>Initiator</b>	Methodology and Templates
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

### 4.33 AUTOSAR Partial Model

<b>Definition</b>	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.
<b>Initiator</b>	Methodology and Templates
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

### 4.34 AUTOSAR Processor Tool

<b>Definition</b>	An AUTOSAR Tool used to create non-AUTOSAR files by processing information from AUTOSAR XML files.
<b>Initiator</b>	Methodology and Templates
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	RTE Generator
<b>Reference</b>	--

### 4.35 AUTOSAR Runtime for Adaptive Applications

<b>Definition</b>	A set of standard application interfaces provided by Functional Clusters, which belong to either Adaptive Platform Foundation or Adaptive Platform Services.
<b>Initiator</b>	General
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

### 4.36 AUTOSAR Run-Time Interface

<b>Definition</b>	The AUTOSAR Run-Time Interface shall define an interface between build tools and debugging/tracing tools.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	The interface shall ease and speed up the debugging, tracing and verification of system behavior as well as round-trip engineering.
<b>Comment</b>	
<b>Example</b>	--
<b>Reference</b>	[AUTOSAR Specification of ARTI]

### 4.37 AUTOSAR Service

<b>Definition</b>	The term service is used in the layered software architecture to denote the highest layer of the AUTOSAR software architecture that interacts with the application. The term service is used in the meaning defined by the service-oriented architecture. This meaning has the strongest relation to the usage of the term service on the AUTOSAR adaptive platform. The term service is also used to describe the handling of diagnostic services, e.g.UDS service ReadDataByIdentifier, for the communication between a diagnostic tester and a diagnostic stack on an (AUTOSAR) ECU.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

### 4.38 AUTOSAR Tool

<b>Definition</b>	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 AUTOSAR Authoring Tool, an AUTOSAR Converter Tool, an AUTOSAR Processor Tool or as a combination of those.
<b>Initiator</b>	Methodology and Templates
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

### 4.39 AUTOSAR XML description

<b>Definition</b>	In AUTOSAR this is a serialized AUTOSAR model. 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 must validate successfully against the AUTOSAR XML Schema.
<b>Initiator</b>	Methodology and Templates
<b>Further</b>	--

<b>Explanations</b>	
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.40 AUTOSAR XML Schema

<b>Definition</b>	The AUTOSAR XML Schema is an XML language definition for exchanging AUTOSAR Models and descriptions.
<b>Initiator</b>	Methodology and Templates
<b>Further Explanations</b>	The AUTOSAR XML Schema is a W3C XML schema that defines the language for exchanging AUTOSAR models. This Schema is derived from the AUTOSAR Metamodel. The AUTOSAR XML Schema defines the AUTOSAR data exchange format.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.41 Availability

<b>Definition</b>	<ol style="list-style-type: none"> <li>1. Availability is the ability of the system to perform a function A completely according to its specification.</li> <li>2. The ratio of the total time the system is performing a function A (according to 1) during a given interval to the length of the interval. Alternative: The probability that the system is performing the function A at a specified time t</li> <li>3. In a degraded mode the system has the ability to perform a subset B of A if full A is not available. In this case, the functionality B is available.</li> </ol>
<b>Initiator</b>	Safety
<b>Further Explanations</b>	<p><b>Full functionality (A)</b> System is available w.r.t functionality A</p> <p><b>Degraded functionality (B)</b> <math>B &lt; A</math> and <math>B \cap A \equiv B</math> System is available w.r.t functionality B (but not available w.r.t. functionality A)</p> <p><b>Alternative / Safe State functionality (C)</b> <math>C \cap A \equiv \emptyset</math> <math>C \cap B \equiv \emptyset</math> System is not available w.r.t any intended functionality</p> <p>Failure <math>F_1</math>: Due to <math>F_1</math> - transition to degraded functionality (1)</p> <p>Failure <math>F_2</math>: Due to <math>F_2</math> - transition directly to safe / alternative functionality (2)</p> <p>Failure <math>F_3</math>: Due to <math>F_3</math> - (occurring during degraded mode) transition to safe / alternative state (3)</p> <p>Failure <math>F_4</math>: Due to <math>F_4</math> - transition to the same functionality (e.g. due to redundancy) (4)</p> <p>Repair transitions: Repair</p> <p>NOTES: 1. This diagram assumes the system is already in full intended functionality mode and neglects the start-up, shut-down, reset, etc. transitions which lead to this state. 2. The repair transitions are conceptual transitions for illustration purposes.</p>
<b>Comment</b>	1. Degraded modes are covered by this definition (see example)
<b>Example</b>	1. Power Steering: if the support function fails it is not available while the steering

	as a base function has full availability. 2. From a security perspective availability is an attribute that ensures correct and timely access upon demand by an authorized entity.
<b>Reference</b>	See ISO 26262, Part 1, ID 1.8

## 4.42 Basic Software

<b>Definition</b>	The Basic Software (BSW) provides the infrastructural (schematic dependent and schematic independent) functionalities of an Electronic Control Unit. It consists of Integration Code and Standard Software.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	MCAL, AUTOSAR services, Communication Layer
<b>Reference</b>	--

## 4.43 Basic Software Module

<b>Definition</b>	A collection of software files (code and description) that define a certain basic software functionality present on an Electronic Control Unit.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	Standard Software may be composed of several software modules that are developed independently. A software module may consist of Integration Code, and/or Standard Software.
<b>Comment</b>	--
<b>Example</b>	A Digital IO Driver, Complex Driver, OS are examples of basic software modules.
<b>Reference</b>	--

## 4.44 Bit Position

<b>Definition</b>	In AUTOSAR the bit position N within an I-PDU denotes the bit I, with $I = N \text{ modulo } 8$ , within the byte J, with $J = N / 8$ . The byte J and bit position I is interpreted in accordance to the definition in ISO 17356-4 (OSEK/VDX Communication).
<b>Initiator</b>	Communication
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.45 Blueprint

<b>Definition</b>	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.
<b>Initiator</b>	Methodology and Templates

<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	Standardized Blueprint and AUTOSAR Blueprint.
<b>Reference</b>	--

## 4.46 Bulk Data

<b>Definition</b>	“Bulk Data” is a set of data such big in size, that standard mechanisms used to handle smaller data sets become inconvenient. This implies that bulk data in a software system are modeled, stored, accessed and transported by different mechanisms than smaller data sets.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	Bulk data are typically handled by adding a level of abstraction (e.g. files) which separates the containment of the data from the internal structure.
<b>Comment</b>	The critical size, above which data must be regarded as bulk data depends on the technical infrastructure (e.g. bus system) and the considered use case (transport, storage etc.).
<b>Example</b>	Data on a persistent medium which has a capacity of a few kBytes (e.g. EEPROM) can be directly accessed via memory addresses, address offsets can be mapped to symbols of a programming language: No bulk data mechanisms are needed. For media with bigger capacity this becomes inconvenient or even impossible, so that a file system is used: The data are treated as bulk data.
<b>Reference</b>	--

## 4.47 Bus Mirroring

<b>Definition</b>	Forwarding information from an internal vehicle bus to an external bus, e.g. the diagnostic connector.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	Bus Mirroring is used to make internal buses accessible to external testers such that internal buses can be debugged in case of errors.
<b>Comment</b>	Because the external (or intermediary) buses typically do not have sufficient bandwidth to transport all information from an internal bus, filters have to be applied to select the frames that are relevant to the analysis.
<b>Example</b>	An ECU does not go to sleep in a vehicle. The engineer wants to check the NM traffic to check for irregular behavior, e.g. concerning partial networking information.
<b>Reference</b>	--

## 4.48 Bus Wake-Up

<b>Definition</b>	A bus wake-up is caused by a specific wake pulse on the bus defined within the specification of the dedicated communication standard (e.g. CAN, LIN, FR). A bus wake-up initiates that the transceiver and controller leave their energy saving mode and enter normal mode to start bus communication again.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--

<b>Reference</b>	--
------------------	----

## 4.49 Bypassing

<b>Definition</b>	The experimental incorporation of new functionality within an ECU image.
<b>Initiator</b>	Runtime Environment
<b>Further Explanations</b>	Bypassing involves the incorporation of new functionality or to replace existing functionality to an existing ECU image without requiring that the image be rebuilt.
<b>Comment</b>	Bypassing can be either "internal" where the new/ replacement functionality is present on the ECU image or "external" where an Rapid Prototyping Tool provides the functionality out with the ECU.
<b>Example</b>	An RP tool intercepts the output of a bypassed RunnableEntity via the RP Memory Interface and replaces the value with the bypass result. Subsequent RunnableEntitys then process the bypass value rather than the original result.
<b>Reference</b>	--

## 4.50 Calibration

<b>Definition</b>	Calibration is the adjustment of parameters of SW-Components realizing the control functionality (namely parameters of AUTOSAR SW-Cs, ECU abstraction or Complex Driver).
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	Only those software modules can be calibrated, which are above RTE and ECU Abstraction and CDD. Calibration is always done at post-build time. Used techniques to set calibration data include end-of-line programming, garage programming and adaptive calibration (e.g. in the case of anti-pinch protection for power window).
<b>Comment</b>	--
<b>Example</b>	The calibration of the engine control will take into account the production differences of the individual motor this system will control.
<b>Reference</b>	--

## 4.51 Call Point

<b>Definition</b>	A point in a Software Component where the SW-C enforce an execution entity (Entry Point) in another SW-C.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	Request Service Send Information
<b>Reference</b>	--

## 4.52 Callback

<b>Definition</b>	Functionality that is defined by an AUTOSAR module so that lower-level modules (i.e. lower in the Layered Software Architecture) can provide notification as
-------------------	--

	required (e.g. when certain events occur or asynchronous processing completes).
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	<p>In AUTOSAR, modules usually provide a register mechanism for callback functions which is set through configuration.</p> <p>A module provides callbacks so that other modules can initiate its processing while the module calls <b>Callouts</b> to execute functionality that could not be specified by AUTOSAR, i.e. <b>Integration Code</b></p>
<b>Comment</b>	--
<b>Example</b>	<p>(from the viewpoint of a particular SWS):</p> <p>The module being specified (Msws) should be informed about an <b>Event</b> in another module (Mexternal). In this example, Msws calls Mexternal to perform some processing and can only resume when Mexternal completes. Upon completion, Mexternal calls Msws's callback function. That is, the called module (Mexternal) <b>CALLS</b> the calling module (Msws) <b>BACK</b> when complete ==&gt; a callback.</p>
<b>Reference</b>	--

## 4.53 Callout

<b>Definition</b>	Function stubs that the system designer can replace with code to add functionality to a module which could not be specified by AUTOSAR.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	<p>A module calls callouts to execute functionality that could not be specified by AUTOSAR, i.e. integration code while the module provides <b>Callbacks</b> so that other modules can initiate its processing.</p> <p>Callouts can be separated into two classes:</p> <ol style="list-style-type: none"> <li>1) callouts that provide mandatory functionality and thus serve as a hardware abstraction layer</li> <li>2) callouts that provide optional functionality</li> </ol>
<b>Comment</b>	--
<b>Example</b>	<p>In the EcuM:</p> <p>For class 1): EcuM_EnableWakeupSources</p> <p>For class 2): The Init Lists (EcuM_AL_DriverInitZero)</p>
<b>Reference</b>	--

## 4.54 Cascaded Switch

<b>Definition</b>	A Cascaded Switch is an Ethernet switch that exists of at least two Ethernet switches: a master switch and a slave switch. The master switch and the slave switch are connected by uplink ports.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	<p>Request Service</p> <p>Send Information</p>
<b>Reference</b>	--

## 4.55 Cascading Failure

<b>Definition</b>	See ISO 26262, Part 1, ID 1.13
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	ISO 26262, Part 1, ID 1.13

## 4.56 Category 1 Interrupt

<b>Definition</b>	Category 1 (Cat1) Interrupts are supported by the OS but their code is only allowed to call a very small subset of OS functions. Furthermore they can bypass the OS. The code of Category 1 Interrupts depends (normally) on the used compiler and microcontroller. Category 1 Interrupts are not allowed to use the ISR() macro. Category 1 Interrupts need to implement/establish their own Interrupt Frame. Nevertheless they have to be configured in order to be included in the Interrupt Vector Table.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.57 Category 2 Interrupt

<b>Definition</b>	Category 2 (Cat2) Interrupts are supported by the OS and their code can call a subset of OS functions. The definition of the Cat2 Interrupt must use the ISR() macro in order to be recognized by the OS. The Interrupt Frame of a Category 2 Interrupt is managed by the OS.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	<pre>ISR(timer1) {     /* here is the code which handles timer1 interrupts */     ... }</pre>
<b>Reference</b>	--

## 4.58 Causality of Transmission

<b>Definition</b>	Transmit order of PDUs with the same identifier (instances of PDUs) from a source network is preserved in the destination network.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	Transmission of Protocol Data Units with the same identifier has a particular temporal order in a given source network. After routing over a gateway the



	temporal order of transmission of PDUs in a destination network may be changed. Only in case that the temporal order is the same, causality is given. Otherwise causality is violated. Causality can be in contradiction to prioritization of PDUs.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.59 Classic Platform

<b>Definition</b>	Software Platform defined by AUTOSAR for deeply embedded systems and Application Software with high demands regarding predictability, safety and responsiveness.
<b>Initiator</b>	General
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.60 Client

<b>Definition</b>	Software entity which uses services of a Server.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	The client and the server might be located on one Electronic Control Unit or distributed on different calculation units (e.g. ECU, external diagnostic tester).
<b>Comment</b>	Adapted from Balzert.
<b>Example</b>	--
<b>Reference</b>	[Balzert99]

## 4.61 Client-Server Communication

<b>Definition</b>	A specific form of communication in a possibly distributed system in which software entities act as Clients, Servers or both, where 1...n clients are requesting services via a specific protocol from typically one server.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	<p>Client-server communication can be realized by synchronous or asynchronous communication.</p> <ul style="list-style-type: none"> <li>• Client takes initiative: requesting that the server performs a service, e.g. client triggers action within server (server does not start action on its own)</li> <li>• Client is after service request blocked / non-blocked</li> <li>• Client expects response from server: data flow (+ control flow, if blocked)</li> </ul> <p>One example for 1 client to n server communication (currently not supported) is a functional request by diagnosis. This has to be treated as a specific exception.</p>
<b>Comment</b>	Adapted from Hyper Dictionary
<b>Example</b>	Internet (TCP/IP)
<b>Reference</b>	[Hyper Dictionary]

## 4.62 Client-Server Interface

<b>Definition</b>	The client-server interface is a special kind of Port Interface used for the case of Client-Server Communication. The client-server interface defines the operations that are provided by the Server and that can be used by the Client.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	[AUTOSAR Specification of Virtual Functional Bus]

## 4.63 Cluster Signal

<b>Definition</b>	A cluster signal represents the aggregating system signal on one specific communication cluster. Cluster signals can be defined independently of frames. This allows a development methodology where the signals are defined first, and are assigned to frames in a later stage.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.64 Code Generator

<b>Definition</b>	The Code Generator consumes complete and correctly formed XML for a BSW module and generates code and data that configures the module.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	[AUTOSAR_InterruptHandling_Explanation.doc]

## 4.65 Code Variant Coding

<b>Definition</b>	Adaptation of SW by selection of functional alternatives according to external requirements
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	Code Variant Coding might influences RTE (RuntimeEnvironment) and Basic Software Modules, not only the application software modules. Code Variant Coding is always done at pre-compile time or at link time. Code Variant Coding also includes vehicle-specific (not user-specific) SW adaptation due to end-customer wishes (e.g. deactivation of speed dependent automatic locking).
<b>Comment</b>	In case of the C language the #if or #ifdef directive can be used for creating code variants. Code Variant Coding is a design time concept.
<b>Example</b>	The same window lifter ECU is used for cars with 2 and 4 doors, however different code segments have to be used in both cases.

<b>Reference</b>	
------------------	--

## 4.66 Common Cause Failure

<b>Definition</b>	See ISO 26262, Part 1, ID 1.14
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	ISO 26262, Part 1, ID 1.14

## 4.67 Communication Attribute

<b>Definition</b>	Communication attributes define, according to the development phase, behavioral as well as implementation aspects of the AUTOSAR communication patterns.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	The exact characteristics of the communication patterns provided by AUTOSAR (client-server and sender-receiver) can be specified more precisely by communication attributes.
<b>Comment</b>	See chapter 4.1.6 in Specification of the Virtual Functional Bus
<b>Example</b>	--
<b>Reference</b>	[AUTOSAR Specification of Virtual Functional Bus]

## 4.68 Complex Driver

<b>Definition</b>	A software entity not standardized by AUTOSAR that can access or be accessed via AUTOSAR Interface and/ or Basic Software Module APIs.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	CDD used to be the acronym for Complex Device Driver, but is not limited to drivers.
<b>Comment</b>	--
<b>Example</b>	<ul style="list-style-type: none"> <li>• Communication stack CDD to support the communication on a bus not supported by AUTOSAR</li> <li>• Reuse of legacy SW</li> <li>• Integration of software with high HW interaction requirements within a standardized AUTOSAR Architecture</li> </ul>
<b>Reference</b>	--

## 4.69 Composition

<b>Definition</b>	<p>An AUTOSAR Composition encapsulates a collaboration of Software Components, thereby hiding detail and allowing the creation of higher abstraction levels.</p> <p>Through Delegation Connectors a Composition explicitly specifies, which Ports of the internal components are visible from the outside.</p> <p>AUTOSAR Compositions are a type of Components, e.g. they can be part of further compositions.</p>
-------------------	---

<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	See Virtual Functional Bus Specification, Chapter "VFB View, Meta-Model"
<b>Example</b>	--
<b>Reference</b>	[AUTOSAR Specification of Virtual Functional Bus]

## 4.70 Compositionality

<b>Definition</b>	Compositionality is given when the behavior of a software component or subsystem of a system is independent of the overall system load and configuration.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	Compositionality is an important property of deterministic systems. This property leads to a complete decoupling of systems. Smooth subsystem integration without backlashes is then easily achievable.
<b>Comment</b>	--
<b>Example</b>	A new component or a subsystem can be added to a system without changing the behavior of the original components.
<b>Reference</b>	--

## 4.71 Conditioned Signal

<b>Definition</b>	The conditioned signal is the internal electrical representation of the electrical signal within the ECU. It is delivered to the processor and represented in voltage and time (or, in case of logical signals, by high or low level).
<b>Initiator</b>	General
<b>Further Explanations</b>	The Electrical Signal usually can not be processed by the peripherals directly, but has to be adopted. This includes amplification and limitation, conversion from a current into a voltage and so on. This conversion is performed by some electronical devices in the ECU and the result of the conversion is called the Conditioned Signal. The description means for the Conditioned Signal can also be the same as for Technical Signals and Electrical Signals, but limited to electrical voltage
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.72 Configuration

<b>Definition</b>	The arrangement of hardware and/or software elements in a system.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	A configuration in general takes place before runtime.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	[AST-Glossary], [ISO 61511-1]

## 4.73 Confirmation

<b>Definition</b>	Service primitive defined in the ISO/OSI Reference model (ISO 7498). With the 'confirmation' service primitive a service provider informs a service user about the result of a preceding service request of the service user.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	A confirmation is e.g. a specific notification generated by the underlying layer to inform about a Message Transmission Error.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	[SEK BD]

## 4.74 Connector

<b>Definition</b>	A connector connects Ports of Software Component and represents the flow of information between those ports.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	For more information see AUTOSAR Specification of VFB
<b>Example</b>	AssemblyConnector, DelegationConnector
<b>Reference</b>	[AUTOSAR Specification of Virtual Function Bus]

## 4.75 Control Flow

<b>Definition</b>	The directed transmission of information between multiple entities, directly resulting in a state change of the receiving entity.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	A state change could result in an activation of a schedulable entity.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.76 Coordinate

<b>Definition</b>	To control and harmonize two or more events or operations to act in an organized and predictable way.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	Two NM Channels can be coordinated to synchronize different stages of network sleep.
<b>Reference</b>	AUTOSAR Generic NM Interface

## 4.77 Data

<b>Definition</b>	A reinterpretable representation of information in a formalized manner suitable for communication, interpretation or processing.
<b>Initiator</b>	General
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	Flag, Notification, etc.
<b>Reference</b>	[ISO 2382-1]

## 4.78 Data Element

<b>Definition</b>	Data elements are declared within the context of a Sender-Receiver Interface. They serve as the data units that are exchanged between sender and receiver.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	[AUTOSAR SoftwareComponentTemplate]

## 4.79 Data Flow

<b>Definition</b>	The directed transmission of Data between multiple entities. The transmitted data are not directly related to a state change at the receiver side.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	Asynchronous communication.
<b>Reference</b>	--

## 4.80 Data Variant Coding

<b>Definition</b>	Adaptation of SW by setup of certain characteristic data according to external requirements.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	Data Variant Coding might influences RTE (RunTimeEnvironment) and Basic Software Modules not only the application software modules (Multiple configuration parameter sets are needed). Variant Coding also includes vehicle-specific (not user-specific) SW adaptation due to end-customer wishes (e.g. deactivation of speed dependent automatic locking). Used techniques to select variants include end-of-line programming and garage programming.
<b>Comment</b>	The major difference with calibration is that this later doesn't aim to adapt the SW functionality itself but only aims to adjust the characteristic data of the SW to the HW/SW environment. Characteristic data in the source code of a software function have a significant impact on the functionality of the software.
<b>Example</b>	<ul style="list-style-type: none"> <li>- Steering wheel controller adaptation to the left or right side can be done with Variant Coding. (Selection of the configuration.)</li> <li>- Country related adaptation of MMI with respect to speed and/or temperature unit</li> </ul>

	(km/h vs. mph, °C vs. F).
<b>Reference</b>	

## 4.81 Deadline

<b>Definition</b>	The point in time when an execution of an entity must be finished.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	A deadline is calculated dependent on its local reference system.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	[OS Specification]

## 4.82 Debugging

<b>Definition</b>	Debugging is the process of gathering information in case of a software problem. The information is used to analyze the software problem.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	To analyze and later fix a software problem, in many cases more information than the one provided by the software API is necessary. This can be for example the state of internal variables of the software or a trace of the communication. The information can be collected by different means, e.g. an emulator or a tracing tool for the communication bus.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.83 Dependability

<b>Definition</b>	Dependability is defined as the trustworthiness of a computer system such that reliance can justifiably be placed on the service it delivers.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	[EAST-Glossary]

## 4.84 Dependent Failure

<b>Definition</b>	See ISO 26262, Part 1, ID 1.22
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	ISO 26262, Part 1, ID 1.22

## 4.85 Diagnostic Coverage

<b>Definition</b>	See ISO 26262, Part 1, ID 1.25
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	ISO 26262, Part 1, ID 1.25

## 4.86 Diagnostic Event

<b>Definition</b>	A diagnostic event defines the atomic unit that can be handled by the DEM module. The status of a diagnostic event represents the result of a monitor. The DEM receives the result of a monitor from SW-C via the RTE or other BSW modules.
<b>Initiator</b>	Diagnostics
<b>Further Explanations</b>	--
<b>Comment</b>	For definition of 'monitor' see chapter "Diagnostic monitor definition" in Specification of DEM
<b>Example</b>	--
<b>Reference</b>	[AUTOSAR Specification of Diagnostic Event Manager]

## 4.87 Diversity

<b>Definition</b>	See ISO 26262, Part 1, ID 1.28
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	ISO 26262, Part 1, ID 1.28

## 4.88 Dynamic PDU

<b>Definition</b>	Protocol Data Unit with dynamic identifier.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	Dynamic PDUs are PDUs where the <bus> identifier (e.g. CAN ID) is dynamically assigned (transmission) or evaluated (reception) at run time.
<b>Comment</b>	AUTOSAR supports two types of dynamic PDUs in CanIf: CanIf_SetDynamicTxId (only transmission), and PDUs with MetaData (reception and transmission).
<b>Example</b>	PDU with variable source address, encoded in the CAN ID, e.g. ISO15765 NormalFixed.
<b>Reference</b>	--



## 4.89 Dynamic Routing

<b>Definition</b>	The routing of signals or Protocol Data Units in a gateway can be changed throughout operation without change of the operation mode of the gateway.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	Dynamic routing requires the change of routing tables during operation. It is not intended to use dynamic routing in the gateway.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	[EAST-Glossary]

## 4.90 E2E Profile

<b>Definition</b>	A functional and complete description of a specific communication stack in terms of data structures, services, behavioral state-machines, error handling. E2E Profiles are defined in AUTOSAR E2E Library. An E2E Profile is configurable by runtime parameters. A specific set of runtime parameters is called E2E profile variant. In order to reach interoperability, the application developers should use the E2E profile variants defined in the E2E library.
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.91 E2E protection alive counter

<b>Definition</b>	An Alive Counter is a counter which is incremented in every transmission request. The counter value is checked at the receiver side, whether it changes at all, but not if the counter was incremented correctly.
<b>Initiator</b>	Safety
<b>Further Explanations</b>	With an alive counter, the receiver monitors if new values arrive, this means it checks if the sender is still alive
<b>Comment</b>	--
<b>Example</b>	ISO 26262-6:2018 Annex D.2.4 EXAMPLE 3: Communication protocols can contain information such as identifiers for communication objects, keep-alive messages, alive counters, sequence numbers, error detection codes and error-correcting codes.
<b>Reference</b>	--

## 4.92 E2E protection sequence counter

<b>Definition</b>	A Sequence Counter is a counter which is incremented in every transmission request. The value of the counter value is checked at the receiver side for each message, whether it was incremented correctly.
<b>Initiator</b>	Safety
<b>Further Explanations</b>	A Sequence Counter is used to check whether messages got lost or a message is repeated.
<b>Comment</b>	--
<b>Example</b>	ISO 26262-6:2018 Annex D.2.4 EXAMPLE 3: Communication protocols can

	contain information such as identifiers for communication objects, keep-alive messages, alive counters, sequence numbers, error detection codes and error-correcting codes.
<b>Reference</b>	--

## 4.93 ECU Abstraction Layer

<b>Definition</b>	The ECU Abstraction Layer is located above the Microcontroller Abstraction Layer and abstracts from the ECU schematic. It is implemented for a specific ECU and offers an API for access to peripherals and devices regardless of their location (onchip/offchip) and their connection to the microcontroller (port pins, type of interface). Task: Make higher software layers independent of the ECU hardware layout.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	The ECU Abstraction Layer consists of the following parts: <ul style="list-style-type: none"> <li>• I/O Hardware Abstraction</li> <li>• Communication Hardware Abstraction</li> <li>• Memory Hardware Abstraction</li> <li>• Crypto Hardware Abstraction</li> <li>• Onboard Device Abstraction</li> </ul> Properties: <ul style="list-style-type: none"> <li>• Implementation: <math>\mu</math>C independent, ECU hardware dependent</li> <li>• Upper Interface (API): <math>\mu</math>C and ECU hardware independent, dependent on signal type</li> </ul>
<b>Comment</b>	--
<b>Example</b>	See Layered Software Architecture
<b>Reference</b>	[AUTOSAR SoftwareArchitecture]

## 4.94 ECU Configuration

<b>Definition</b>	Activity of integrating and configuring one ECU's software.
<b>Initiator</b>	Methodology and Templates
<b>Further Explanations</b>	Further Explanations: ECU Configuration denotes the activity when one ECU's software is set up for a specific usage inside the ECU. In AUTOSAR the ECU Configuration activity is divided into "Pre-compile time", "Link time" and "Post-build time" configuration.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	ECU Configuration Description, Pre-Compile Time Configuration, Link Time Configuration, Post-build Time Configuration.

## 4.95 ECU Configuration Description

<b>Definition</b>	Output of the ECU Configuration activity containing the values of configuration parameters and references.
<b>Initiator</b>	Methodology and Templates
<b>Further Explanations</b>	ECU Configuration Description holds the configuration parameter values and references to other module's configurations which have been defined in the ECU Configuration activity.
<b>Comment</b>	ECU Configuration Description may contain the whole ECU Configuration information or only the parts relevant for a specific configuration step (e.g. Pre-

	compile time).
<b>Example</b>	--
<b>Reference</b>	ECU Configuration Description, Pre-Compile Time Configuration, Link Time Configuration, Post-build Time Configuration.

## 4.96 Electrical Signal

<b>Definition</b>	The electrical signal is the electrical representation of Technical Signals. Electrical signals can only be represented in voltage, current and time
<b>Initiator</b>	General
<b>Further Explanations</b>	When a sensor processes the Technical Signal it is converted into an Electrical Signal. The information can be provided in the current, the voltage or in the timely change of the signal (e.g. a pulse width modulation).
<b>Comment</b>	To describe the Electrical Signal the same means as for the Technical Signal can be used, limited to electrical current and voltage.
<b>Example</b>	--
<b>Reference</b>	--

## 4.97 Electronic Control Unit

<b>Definition</b>	Embedded computer system consisting of at least one CPU and corresponding peripherals which are placed in one housing.
<b>Initiator</b>	General
<b>Further Explanations</b>	An ECU is typified by a connection to one or more in-vehicle networks, sensors and actuators.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.98 Empty Function

<b>Definition</b>	Any C function defined by an AUTOSAR specification which does not implement or alter behavior required to accomplish the assigned functional responsibility.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	As such an empty function in the context of AUTOSAR can still have code but this code shall not impact the state machine other than error reporting. Auxiliary code like validating arguments to report to the DET does not constitute functional behavior because without the code and proper calling this code would still fulfill its architectural responsibility.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.99 Entry Point

<b>Definition</b>	A point in a Software Component where an execution entity of the SW-C begins.
<b>Initiator</b>	Software and Architecture
<b>Further</b>	--

<b>Explanations</b>	
<b>Comment</b>	--
<b>Example</b>	<ul style="list-style-type: none"> <li>• Service of the Server in Client/Server Communication</li> <li>• Reaction after receive Information (Notification)</li> </ul>
<b>Reference</b>	--

## 4.100 Error

<b>Definition</b>	See ISO 26262, Part 1, ID 1.36
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	ISO 26262, Part 1, ID 1.36

## 4.101 Error Detection Rate

<b>Definition</b>	Ratio between detected lost/faulty words/symbols/blocks, divided by the total number of symbols/words/blocks sent.
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.102 Ethernet Switch Port Groups

<b>Definition</b>	Ethernet Switch Port groups are Ethernet switch ports of an Ethernet switch which are grouped to so called port groups. Ethernet Switch Port groups are only relevant for the host ECU. Ethernet Switch Port Groups are derived from the model per VLAN and per PNC. The host port is participating in all port groups. A Ethernet Switch Port Group could be a mix of internal and external ports.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.103 Event

<b>Definition</b>	State change of a hardware and/or software entity.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	See OS Event, RTE Event, Diagnostic Event and Event Message (SOME/IP)
<b>Comment</b>	--

<b>Example</b>	--
<b>Reference</b>	--

#### 4.104 Event Message (SOME/IP)

<b>Definition</b>	Event – a message sent by an ECU implementing a service instance to an ECU using this service instance (Publish/Subscribe).
<b>Initiator</b>	Communication
<b>Further Explanations</b>	Eventgroup – a logical grouping of 1 or more events. An eventgroup is part of a service.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.105 Executable

<b>Definition</b>	Part of an application which consists of either a file containing executable code with a defined entry point and suitable for the platform instance as the target of deployment (deployment time) or software code which is ready to be integrated for a specific platform.
<b>Initiator</b>	Execution Management
<b>Further Explanations</b>	In POSIX systems, an executable is typically running within a single Process. Therefore, intra-executable communication is different from inter-executable communication and should therefore be considered during design time of an executable.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.106 Executable Entity Cluster

<b>Definition</b>	A set of Executable Entities (EEs) and a reference to a set of Execution Order Constraints (EOCs) between these EEs. The Executable Entity Cluster is formed for the purpose of mapping EEs to LET intervals and to tasks. Several EECs may be mapped to the same LET interval.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	An Executable Entity Cluster (EEC) groups EEs from any Software Component. EEs with different triggers/periods can be part of the same EEC, if the triggers/periods are harmonic (i.e. that all periods are integer multiples of the smallest period). The EEC can reference a LET interval specification.

<b>Comment</b>	The set of EOCs is cycle-free.
<b>Example</b>	--
<b>Reference</b>	--

### 4.107 Execution Order Constraint

<b>Definition</b>	Defines the execution order between instances of Executable Entities (EEs) within the same LET interval.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	An EE 2 is marked as successor/directSuccessor of an EE 1, if the execution order of the instances of the respective Executable Entities is 1→2 (2 runs after 1) within the LET interval. 
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	Specification of Timing Extensions

### 4.108 Execution Time

<b>Definition</b>	The time during which a program is actually executing, or more precisely during which a certain thread of execution is active.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	The execution time of software is the time during which the CPU is executing its instructions. The time the CPU spends on task switches or on the execution of other pieces of software is not considered here. See also: response time, worst case execution time, worstcase response time.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

### 4.109 Exit Point

<b>Definition</b>	A point in a Software Component where an execution entity of the SW-C ends.
<b>Initiator</b>	Software and Architecture
<b>Further</b>	--

<b>Explanations</b>	
<b>Comment</b>	--
<b>Example</b>	Return point.
<b>Reference</b>	--

#### 4.110 External Port

<b>Definition</b>	External Ports are ports of an automotive Ethernet switch used to communicate over an Ethernet physical connection with other ECUs (e.g. 100BASE-TX, 100BASE-T1).
<b>Initiator</b>	Communication
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.111 Fail-operational

<b>Definition</b>	Property of a system or functional unit. Describes the ability of a system or functional unit to continue normal operation at its output interfaces despite the presence of hardware or software faults.
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	<ol style="list-style-type: none"> <li>1. Typically, a fail-operational system or functional unit has no safe state.</li> <li>2. Safety means are not regarded as a part of the normal functionality respectively operation.</li> </ol>
<b>Example</b>	Braking system
<b>Reference</b>	--

#### 4.112 Fail-safe

<b>Definition</b>	Property of a system or functional unit. In case of a fault the system or functional unit transits to a safe state.
<b>Initiator</b>	Safety
<b>Further Explanations</b>	Fail safe systems needs to have a safe state. Note: not all the systems have a safe state.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	See also note of ISO 26262, Part 1, ID 1.137

#### 4.113 Fail-silent

<b>Definition</b>	Fail-silent is a property of a system in which no output is produced in the presence of a fault. In automotive domain, fail-silent systems are usually only used if the next hierarchical system level provides a safe-state.
-------------------	--

<b>Initiator</b>	Safety
<b>Further Explanations</b>	Fail-silent is a special case of the fail-safe property.
<b>Comment</b>	--
<b>Example</b>	The fail-silent property can be used to avoid that “babbling idiots” disturb the overall communication.
<b>Reference</b>	--

#### 4.114 Failure Mode

<b>Definition</b>	See ISO 26262, Part 1, ID 1.40
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	ISO 26262, Part 1, ID 1.40

#### 4.115 Failure

<b>Definition</b>	See ISO 26262, Part 1, ID 1.39
<b>Initiator</b>	Safety
<b>Further Explanations</b>	Termination is a reduction in, or loss of, ability of an element or an item to perform a function as required. There is a difference between “to perform a function as required” (stronger definition, use-oriented) and “to perform a function as specified”, so a failure can result from an incorrect specification.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	ISO 26262, Part 1, ID 1.39

#### 4.116 Failure Rate

<b>Definition</b>	See ISO 26262, Part 1, ID 1.41
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	ISO 26262, Part 1, ID 1.41

#### 4.117 Fault

<b>Definition</b>	See ISO 26262, Part 1, ID 1.42
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--



<b>Example</b>	--
<b>Reference</b>	ISO 26262, Part 1, ID 1.42

#### 4.118 Fault Detection

<b>Definition</b>	The action of monitoring errors and setting fault states to specific values is called fault detection.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	<p>The different states are called "not detected"/ "present"/ "intermittent or maturing"/...</p> <p>The names of the fault states are following the ISO/SAE norms; however there is a coordination step in between the states of the DTCs (Diagnostic Trouble Code → see definition in ISO 15765/ ISO14229) and the states of the faults.</p> <p>The SW-C's Fault Detection is executed decentralized, e.g. each SW-C sets the state of a fault according to the defined fault qualification (SW-C Template). Therefore the Fault Detection is implemented in the SW-C (SW-C could be either Application SW Component or Basic SW Component). There are exceptions; these will be pointed out individually for each fault. The SW-C's developer will define the conditions (=fault qualification), when these conditions are fulfilled the SW-C notifies a fault to the Diagnostic Memory Management.</p>
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	[ISO 15765], [ISO14229] [AUTOSAR Specification of Virtual Functional Bus]

#### 4.119 Fault Reaction

<b>Definition</b>	In case of a Failure of a SW-C there is a specific action to be carried out. This action is called "Fault Reaction".
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	Fault Reactions can be implemented decentralized in the SW-C. There might also be the need of coordinating the fault reactions since there are reactions excluding each other. This will be done by a central fault reaction manager.
<b>Reference</b>	--

#### 4.120 Fault Reaction Time

<b>Definition</b>	See ISO 26262, Part 1, ID 1.44
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	See ISO 26262, Part 1, ID 1.44

### 4.121 Fault Tolerance

<b>Definition</b>	Ability to deliver the specified functionality in the presence of one or more specified faults.
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

### 4.122 Fault Tolerant Time Interval

<b>Definition</b>	See ISO 26262, Part 1, ID 1.45
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	ISO 26262, Part 1, ID 1.45

### 4.123 Feature

<b>Definition</b>	A Feature is a notable characteristic of a system.
<b>Initiator</b>	General
<b>Further Explanations</b>	AUTOSAR defines and interacts with many entities where the term Feature can be applied (e.g. the AUTOSAR standard itself, its implementations, ECUs built with AUTOSAR, AUTOSAR Authoring Tools, AUTOSAR Feature Model). For each usage the term Feature may be used in a refined way - which is then defined for that specific usage (e.g. [TPS_FeatureModelExchangeFormat]).
<b>Comment</b>	--
<b>Example</b>	CAN FD support, Automatic windshield wiper, Editing of the FlexRay schedule
<b>Reference</b>	[EAST-Glossary], [TPS_FeatureModelExchangeFormat]

### 4.124 Flag

<b>Definition</b>	A piece of data that can take on one of two values indicating whether a logical condition is true or false.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	Notification flag
<b>Reference</b>	--

### 4.125 FlexRay

<b>Definition</b>	Automotive time-triggered and fault-tolerant network communication protocol that
-------------------	--

	is standardized in the ISO 17458 and provides options for deterministic data that arrives in a predictable time frame as well as dynamic event-driven data.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	FlexRay provides a communication cycle with a pre-defined space for static and dynamic data.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	For FlexRay specifications please see ISO 17458. For more details about FlexRay in Classic AUTOSAR please see AUTOSAR_SWS_FlexRayDriver, AUTOSAR_SWS_FlexRayInterface, AUTOSAR_TPS_SystemTemplate.

## 4.126 Foundation

<b>Definition</b>	The AUTOSAR Foundation contains the standards artifacts that are cross platform.
<b>Initiator</b>	General
<b>Further Explanations</b>	<p>The role of the Foundation in AUTOSAR is two fold:</p> <p>1) To group those artifacts which are agnostic of, or outside of, the immediate scope of the Classic and Adaptive Platform. <b>Example:</b> Protocol specification (PRS), AUTOSAR Abstract Platform or Non-AUTOSAR platform artifacts.</p> <p>2) To provide a repository for those artifacts, which have content applicable to both Classic and Adaptive Platform. <b>Example:</b> Requirement Specification (RS), Model (MMOD or MOD), Template Specification (TPS), Adaptive Software Specification (ASWS) artifacts. This depth of this commonality between Classic and Adaptive Platform is typically detailed further in the respective artifact.</p>
<b>Comment</b>	--
<b>Example</b>	
<b>Reference</b>	--

## 4.127 Frame

<b>Definition</b>	Data unit according to the data link protocol specifying the arrangement and meaning of bits or bit fields in the sequence of transfer across the transfer medium.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	A CAN frame consists of up to 8 bytes of payload data and additional protocol specific bits / bit fields (e.g. CAN-Identifier).
<b>Reference</b>	[ISO 17356, Glossary]

## 4.128 Frame PDU

<b>Definition</b>	A PDU that fits into 1 frame instance e.g. it does not need to be fragmented across more than 1 frame for transmission over a network.
<b>Initiator</b>	Communication
<b>Further</b>	--

<b>Explanations</b>	
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.129 Freedom from Interference

<b>Definition</b>	See ISO 26262, Part 1, ID 1.49
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	ISO 26262, Part 1, ID 1.49

#### 4.130 Function

<b>Definition</b>	<ol style="list-style-type: none"> <li>1. A task, action or activity that must be accomplished to achieve a desired outcome.</li> <li>2. A part of programming code that is invoked by other parts of the program to fulfill a desired purpose.</li> <li>3. In mathematics, a function is an association between two sets of values in which each element of one set has one assigned element in the other set so that any element selected becomes the independent variable and its associated element is the dependent variable.</li> </ol>
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	Due to the different meanings in texts using the term application the appropriate meaning should be explained in detail or referenced.
<b>Example</b>	<ol style="list-style-type: none"> <li>2. C-Code Function</li> <li>3. <math>Y=f(x)</math></li> </ol>
<b>Reference</b>	[IEEE12331], [EAST-Glossary]

#### 4.131 Functional Cluster

<b>Definition</b>	Set of requirements grouped by the aspect they refer to.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.132 Functional Network

<b>Definition</b>	A logical structure of interconnections between defined functional parts of Features.
-------------------	---

<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.133 Functional Safety Concept

<b>Definition</b>	See ISO 26262, Part 1, ID 1.52
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	ISO 26262, Part 1, ID 1.52

#### 4.134 Functional Safety Requirement

<b>Definition</b>	See ISO 26262, Part 1, ID 1.53
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	ISO 26262, Part 1, ID 1.53

#### 4.135 Functional Unit

<b>Definition</b>	An entity of software or hardware, or both, capable of accomplishing a specified purpose.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	ECU, Software Component, ...
<b>Reference</b>	[ISO 2382-1]

#### 4.136 Functionality

<b>Definition</b>	Functionality comprises User-visible and User-non-visible functional aspects of a system.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	EAST glossary not applicable, due to use of function.
<b>Example</b>	Functionality of a communication system is a user-non-visible aspect.
<b>Reference</b>	--

### 4.137 Gateway

<b>Definition</b>	A gateway is functionality within an ECU that performs a frame or signal mapping function between two communication systems. Communication system in this context means e.g. a CAN system or one channel of a FlexRay system.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	Gateway ECU 0

### 4.138 Gateway ECU

<b>Definition</b>	A gateway ECU is an Electronic Control Unit that is connected to two or more communication channels, and performs gateway functionality.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	Gateway 4.137

### 4.139 Graceful Degradation

<b>Definition</b>	Graceful Degradation: The system continues to operate in the presence of errors, accepting a partial degradation of functionality or performance during recovery or repair. Found in the literature also as "fail soft".
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	Safety means are not regarded as a part of the normal functionality respectively operation. Also known as: Fail-reduced, Fail-soft
<b>Example</b>	"Limp home" functionality for ECU (reduce torque to assure an arrival at home or service station)
<b>Reference</b>	See also: ISO 26262:DIS Part 1: 3.181 - warning and degradation strategy.

### 4.140 Hardware Connection

<b>Definition</b>	HW Connections are used to describe the connection of Hardware Elements among each other. It defines/characterizes the interrelationship among HW Elements (for abstract modelling). The Hardware Port of the HW Elements serve as connection points for this purpose.
<b>Initiator</b>	General
<b>Further Explanations</b>	In AUTOSAR are 2 kinds of HW Connections defined: Assembly HW Connection Delegation HW Connection

<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	[AUTOSAR Specification of ECU Resource Template]

#### 4.141 Hardware Element

<b>Definition</b>	The HW Element is the main describing element of an Electronic Control Unit. It provides Hardware Port for being interconnected among each others. A generic HW Element specifies definitions valid for all specific HW Elements.
<b>Initiator</b>	General
<b>Further Explanations</b>	A HW Element is the piece or a part of the piece to be described with the ECU Resource Template. It uses other elements as primitive: This means HW elements can be nested (through HW Containers, a hierarchical structure of HW Elements). At the lowest level a HW Element only uses primitives
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	[AUTOSAR Specification of ECU Resource Template]

#### 4.142 Hardware Interrupt

<b>Definition</b>	Interrupt triggered by HW event
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	2 sorts of HW events <ul style="list-style-type: none"> <li>• Processor-intern: events as for example division by zero, arithmetical overflow, non-implemented instruction</li> <li>• Processor-extern: events as for example response of peripheral device (e.g. PWM), memory error, timer</li> </ul>
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	Translation/Adaptation from [VDI Lexikon]

#### 4.143 Hardware Port

<b>Definition</b>	The HW port exposes functionality to the exterior of the Hardware Element. HW elements can be connected via Hardware Connection. It defines a connection Endpoint for the HW Element.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	HW elements provide HW ports for being interconnected among each others. Each HW port has a name which is unique within the HW element it is located in.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	[AUTOSAR Specification of ECU Resource Template]

#### 4.144 Health Indicator

<b>Definition</b>	Health Indicators (HIs) are an evaluation metric of current system performance with regard to safety requirement
<b>Initiator</b>	Safety
<b>Further</b>	Health Indicator format: <HI_ID, Performance, Reliability, SubsystemState>

<b>Explanations</b>	
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	ASWS_HealthMonitoring

#### 4.145Hook

<b>Definition</b>	An intervention point within ECU software for the exchange of data.
<b>Initiator</b>	Runtime Environment
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	Hooks used to read ECU variables and/ or overwrite ECU variables with values generated by Rapid Prototyping algorithm.
<b>Reference</b>	--

#### 4.146Host ECU

<b>Definition</b>	A Host ECU is a ECU that controls one or more automotive Ethernet switches (e.g. switch on / off the Ethernet switch and its ports, read and write the Ethernet switch configuration). For this purpose the host ECU is connected to the Ethernet switch over a common control interface (e.g. SPI, MDIO). The host ecu also take part in the network communication. For this purpose the host ecu is connected by a data interface (e.g. MII) to a specific Ethernet switch port (host port).It transmits and receives Ethernet frames.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.147Host Port

<b>Definition</b>	A host port is a port of an automotive Ethernet switch where the data interface (e.g. MII) of the Host ECU is connected to. The host port could either be an internal port or an external port. The host port has a special role from the perspective of the software. (see link accumulation and port groups)
<b>Initiator</b>	Communication
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.148Hypervisor

<b>Definition</b>	Low-level software that provides and manages several virtual machines in one
-------------------	--



	physical machine. Maybe an independent software or contained as an OS functionality.
<b>Initiator</b>	Execution Management
<b>Further Explanations</b>	Shared physical resources are either exclusively assigned to single virtual machine, or accessed through virtual device which is managed by Hypervisor. Various hardware and software mechanisms can support the efficient implementation of virtual devices.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.149 Identity and Access Management (IAM)

<b>Definition</b>	IAM is about managing access rights of Adaptive Applications to interfaces of the Adaptive Platform Foundation and Services.
<b>Initiator</b>	Security
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.150 Identity Information

<b>Definition</b>	The access control is decided / enforced upon the identity information which represents properties of the Adaptive Applications.
<b>Initiator</b>	Security
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	An example for identity information are Capabilities.
<b>Reference</b>	--

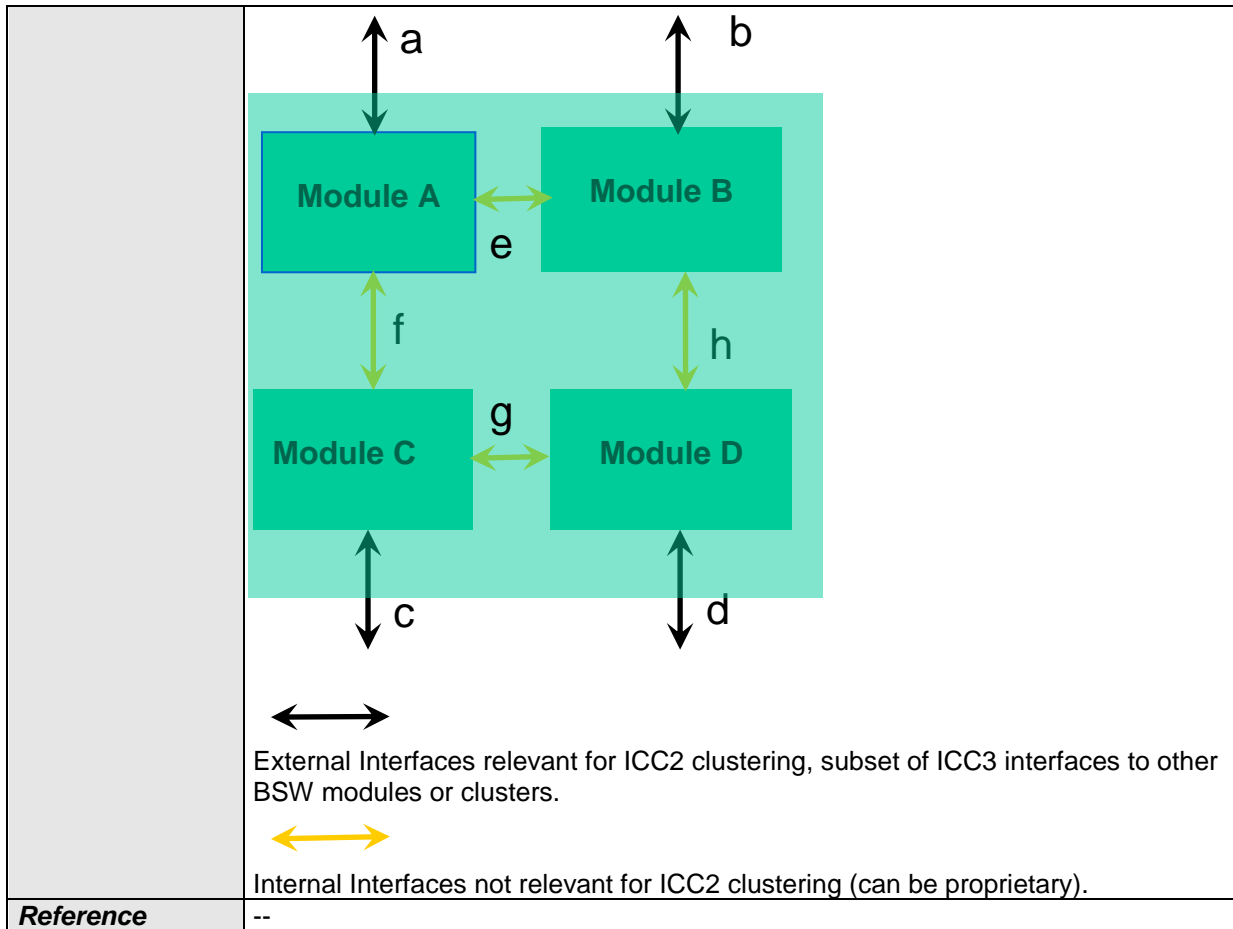
#### 4.151 Implementation Conformance Class 1 (ICC1)

<b>Definition</b>	An ICC1 cluster offers a Software Component Interface and/or Network Interface. The Software Component Interface and Network Interface of an ICC1 cluster provide the functional behavior as specified in the AUTOSAR specifications on ICC3 level.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	In an ICC1 cluster the basic software is regarded as a black box. It allows legacy platforms to migrate to AUTOSAR: <ul style="list-style-type: none"> <li>- to be integrated into an AUTOSAR network</li> <li>- to support SW-Cs.</li> </ul> <p>The features of an ICC1 cluster can be a subset of the ICC3 features (e.g. FlexRay not used). This has to be indicated in the Implementation Conformance Statement (ICS).</p> <p>The functionality represented in AUTOSAR by the RTE must be a part of any ICC1 cluster that provides an SW-CI.</p>

	<p>Typically an ICC1 cluster</p> <ul style="list-style-type: none"> <li>- is not structured into Basic Software (BSW) modules (ICC3) or BSW module clusters (ICC2)</li> <li>- has a proprietary internal structure and might consist of legacy/proprietary or highly optimized code.</li> </ul> <p>An ICC1 cluster shall provide an interface to the boot loader.</p> <p>ICC1 shall support SW-C compatible configuration for SW-CI and AUTOSAR Network compatible Configuration for NWI.</p>
<b>Comment</b>	Up to Release 4.0 the boot loader architecture is not standardized in AUTOSAR. Therefore the term ICC1 is not applicable to the boot loader architecture itself.
<b>Example</b>	--
<b>Reference</b>	--

#### 4.152 Implementation Conformance Class 2 (ICC2)

<b>Definition</b>	<p>ICC2 clusters logically related ICC3 Basic Software (BSW) modules (2... N modules).</p> <p>The number of Cluster Features in an ICC2 cluster is a subset of the union of the number of features of the clustered ICC3 modules.</p>
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	<p>Each ICC2 cluster presents a subset of the clustered ICC3 module's interfaces. ICC2 cluster provides the functional behavior as specified in the AUTOSAR specifications on ICC3 level.</p> <p>ICC2 cluster have a proprietary internal structure and might consist of proprietary or highly optimized code.</p> <p>ICC2 shall support AUTOSAR ECU Configuration description as an input for the Cluster Configuration It shall be possible to combine ICC2 Clusters and ICC3 Modules in a BSW Architecture.</p> <p>Application interface Conformance (above RTE, Software Component Interface) and Bus Conformance (Network Interface) must be testable for a BSW which contain one or more ICC2 clusters.</p>
<b>Comment</b>	--
<b>Example</b>	<p>Example of a ICC2 Cluster</p> $\text{ICC2 Cluster Y} \subseteq (\text{ICC3 Module A} \cup \text{ICC3 Module B} \cup \text{ICC3 Module C} \cup \text{ICC3 Module D})$



### 4.153 Implementation Conformance Class 3 (ICC3)

<b>Definition</b>	For ICC3 the AUTOSAR BSW consists of BSW modules as defined in the Basic Software Module List, including the RTE. ICC3 is the highest level of granularity.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	All Basic Software modules as defined in the BSW module list including the RTE, must comply with the defined interfaces and functionality as specified in their respective Software specification document (SWS).
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

### 4.154 Independence

<b>Definition</b>	See ISO 26262, Part 1, ID 1.61
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	ISO 26262, Part 1, ID 1.61

#### 4.155 Independent Failures

<b>Definition</b>	See ISO 26262, Part 1, ID 1.62
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	ISO 26262, Part 1, ID 1.62

#### 4.156 Indication

<b>Definition</b>	Service primitive defined in the ISO/OSI Reference Model (ISO 7498). With the service primitive 'indication' a service provider informs a service user about the occurrence of either an internal event or a service request issued by another service user.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	An indication is e.g. a specific notification generated by the underlying layer to inform about a Message Reception Error.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.157 Integration

<b>Definition</b>	The progressive assembling of system components into the whole system.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	[ISO 2382-20]

#### 4.158 Integration Code

<b>Definition</b>	Code that the Integrator needs to add to an AUTOSAR System, to adapt non-standardized functionalities. Examples are Callouts of the ECU State Manager and Callbacks of various other Basic Software Modules.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.159 Interface

<b>Definition</b>	A shared boundary between two Functional Units defined by various
-------------------	---

	characteristics pertaining to the functions, physical interconnections, signal exchanges, and other characteristics, as appropriate.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	In AUTOSAR the interface has specific meanings: See Standardized AUTOSAR Interface and Standardized Interface.
<b>Comment</b>	--
<b>Example</b>	Diagnosis Service
<b>Reference</b>	[ISO 2382-1]

#### 4.160 Internal Port

<b>Definition</b>	Internal Ports are ports of an automotive Ethernet switch used for local communication (Host ECU or cascaded switch)
<b>Initiator</b>	Communication
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.161 Interrupt Frames

<b>Definition</b>	An interrupt frame is the code which handles the entering/leaving of (C written) interrupt service routines. This code is microcontroller specific and often written in assembly language. Interrupt frames are typically generated by the OS generation tool.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	[AUTOSAR_InterruptHandling_Explanation.doc]

#### 4.162 Interrupt Handler

<b>Definition</b>	In the case of a Category 2 interrupt, the ISR is synonymous with Interrupt Handler. In the case of Category 1 interrupt the Interrupt Handler is the function called by the hardware interrupt vector. In both cases the Interrupt handler is the user code that is normally a part of the BSW module. So the Interrupt Handler is a user level piece of code.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	[AUTOSAR_InterruptHandling_Explanation.doc]

#### 4.163 Interrupt Service Routine

<b>Definition</b>	A software routine called in case of an Interrupt.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	ISRs have normally higher priority than normal processes and can only be suspended by another ISR which presents a higher priority than the one running.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	[VDI Lexikon]

#### 4.164 Interrupt Vector Table

<b>Definition</b>	An interrupt vector table is a table of interrupt vectors that associates the Interrupt Service Routine with the corresponding interrupt request (typically by an array of jumps or similar mechanisms).
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.165 Interrupt

<b>Definition</b>	Event that enforces the processor to change its state. This interruption causes the normal sequence of instructions to be stopped. Once an interrupt occurred, the running software entity is suspended and an Interrupt Service Routine (the one dedicated to this interrupt) is called.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	Two sorts of interrupts exists: Hardware Interruptd and Software Interrupts.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	Translation/Adaptation from [VDI Lexikon]

#### 4.166 Intrusion Detection System

<b>Definition</b>	A system that is responsible for detecting, recording and reporting security events.
<b>Initiator</b>	Security
<b>Further Explanations</b>	An AUTOSAR compliant IDS consist of security sensors, Intrusion Detection System Manager (IdSM) and Intrusion Detection System Reporter (Idsr).
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.167 Invalid Flag

<b>Definition</b>	For a signal in a PDU an optional invalid flag can be added to the PDU payload layout. This flag indicates the validity of other signals in the payload. In case the invalid flag of a signal is set to true in a PDU instance, the respective signal in the payload of the PDU instance does not contain a valid signal value.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	This mechanism may be used in gateways to indicate that parts of a PDU do not contain valid data.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.168 Invalid Value of Signal

<b>Definition</b>	For a signal in a PDU an optional invalid value can be defined.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	The invalid value is element of the signal value range that can be represented and transported by the signal. The invalid value is the value that is used in all situations where the receiver should be notified that the value in a signal is not valid.
<b>Comment</b>	--
<b>Example</b>	In case a PDU for a destination network of a gateway is composed from two PDUs of two different source networks, the failure to receive one PDU can be indicated as invalid values in the respective signals of the transmitted PDU in the destination network.
<b>Reference</b>	--

#### 4.169 I-PDU

<b>Definition</b>	Interaction Layer Protocol Data Unit Collection of messages for transfer between nodes in a network. At the sending node the Interaction Layer ( <i>IL</i> ) is responsible for packing messages into an I-PDU and then sending it to the Data Link Layer ( <i>DLL</i> ) for transmission. At the receiving node the DLL passes each I-PDU to the IL which then unpacks the messages sending their contents to the application.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	ISO 17356-4 (OSEK/VDX Communication) specifies an Interaction Layer and works on I-PDUs
<b>Reference</b>	[ISO 17356, Glossary]

#### 4.170 Life Cycle

<b>Definition</b>	The course of development/evolutionary stages of a model element during its life time.
<b>Initiator</b>	Methodology and Templates
<b>Further Explanations</b>	A life cycle consists of a set of life cycle states. A life cycle state can be attached to an element in parallel to its version information.

	A typical life cycle is {valid, obsolete} and means that a valid element is up to date when first introduced but is substituted later by a new one and therefore gets the life cycle state "obsolete".
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.171 LIN Bus Idle

<b>Definition</b>	Bus Idle is defined as no transition between recessive and dominant bit values on the LIN bus.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	LIN slave nodes observe the LIN line for bus idle state. After a specific duration of bus idle (bus idle timeout), slave nodes enter bus sleep mode.
<b>Comment</b>	Synonym "LIN Bus Inactivity"
<b>Example</b>	--
<b>Reference</b>	ISO17987-2 specification

#### 4.172 Link State Accumulation

<b>Definition</b>	The link state of a certain switch port group is accumulated by embracing the link state of each port that is part of the port group. The rule how to embracing the link state is specified in the Ethernet Interface.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

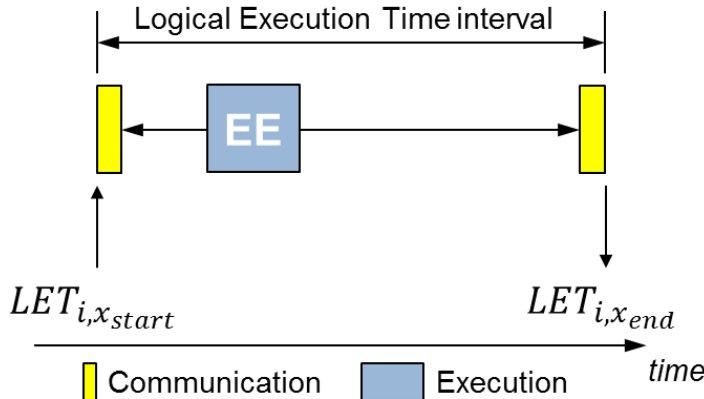
#### 4.173 Link Time Configuration

<b>Definition</b>	The configuration of the SW module is done until link time.
<b>Initiator</b>	Methodology and Templates
<b>Further Explanations</b>	The object code of the SW modules receives parts of its configuration from another object code file or it is defined by linker options.
<b>Comment</b>	--
<b>Example</b>	Initial value of a signal.
<b>Reference</b>	--

#### 4.174 Logical Execution Time

<b>Definition</b>	Is a fixed time interval. Input data is read at the beginning of this interval and output data is written at the end of the interval. Processing of the data is limited within the time interval.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	The logical execution time (LET) is a real-time programming abstraction. Conceptually, a LET program execution has three steps: read the program input



	<p>(in zero time), then execute, and finally write the output (in zero time) exactly when the time has elapsed since reading input. Hence, communication logically happens instantaneously at fixed points in time and the program executes within a time window with a deadline represented by the LET.</p>  <p>When the execution of a program finishes before the deadline, writing the output is (logically) delayed until the deadline. This makes the deadline a logical upper and lower bound for the execution. Thus, using a faster processor does not result in lower response time, but in decreased core utilization. Therefore, the behaviour of the application is the same on any platform able to execute the program within the LET interval.</p>
<b>Comment</b>	Practical implementation of the LET may use buffers to avoid write bursts at the end of a LET interval. Instead, only buffers are switched.
<b>Example</b>	--
<b>Reference</b>	Henzinger, T.A. et al: Giotto: A Time-Triggered Language for Embedded Programming. In: Proceedings of the IEEE, vol.91, 2003. Pp. 94-99

### 4.175 Log and Trace

<b>Definition</b>	Log and Trace provides interfaces for applications to forward logging and tracing information onto the communication bus, the console, or to the file system.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

### 4.176 Machine

<b>Definition</b>	A Machine consists of a set of computing resources – such as CPU cores, memory or peripheral (e.g. communication) devices – and has the ability to execute software applications.
<b>Initiator</b>	Execution Management
<b>Further Explanations</b>	Computing resources can exist either physically or virtually. A Machine may have physical access to its resources or may run in a virtualized environment.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.177 Manifest

<b>Definition</b>	A Manifest represents a piece of AUTOSAR model description that is created to support the configuration of an AUTOSAR Adaptive Platform product and which is uploaded to the AUTOSAR Adaptive Platform product, potentially in combination with other artifacts (like binary files) that contain executable code to which the Manifest applies.
<b>Initiator</b>	Methodology and Templates
<b>Further Explanations</b>	Manifests are often used to denote a piece of configuration content that ships along a given piece of software and is used to deploy the software in the field. Three examples of manifest are: - Execution Manifest - Service Instance Manifest - Machine Manifest
<b>Comment</b>	The Manifest may contain platform implementation dependent data, as well as generic data derived from Application System Description.
<b>Example</b>	--
<b>Reference</b>	--

#### 4.178 Mappable Element

<b>Definition</b>	A mappable element is a part of a MCAL module which can be assigned to a partition via a reference parameter in the Base Software Module Description of the module. Mappable elements allow the formal description on how MCAL modules are available respectively- distributed on partitions (and thus cores).
<b>Initiator</b>	General
<b>Further Explanations</b>	The type- and scope of the functionality represented by a mappable element strongly depends on the MCAL module. A mappable element may closely represent the hardware (e.g. a channel) but can also represent subsets of HW units as well as groups of HW units.
<b>Comment</b>	Single partition MCAL driver define the complete driver as mappable element. Thus indicating that advanced multi-partition use-cases are not supported.
<b>Example</b>	Adc channel, CAN controller
<b>Reference</b>	--

#### 4.179 Mapping

<b>Definition</b>	Mapping designates the distribution of elements in the logical view to elements in the physical view.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	In general several entities may be allocated to one container but an entity may be allocated to only one container.
<b>Comment</b>	--
<b>Example</b>	a) Mapping of AUTOSAR Signals onto Frames (for inter-ECU communication). b) Mapping of SW-C onto ECUs (Distribution of the SW-Components to the ECUs).
<b>Reference</b>	--

#### 4.180 Master Switch

<b>Definition</b>	A Master Switch is an Ethernet switch where the host port is located.
-------------------	---

<b>Initiator</b>	Communication
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.181 MCAL Signal

<b>Definition</b>	The MCAL signal is the software representation of the Conditioned Signal. It is provided by the microcontroller abstraction layer (MCAL) and is further processed by the ECU abstraction.
<b>Initiator</b>	General
<b>Further Explanations</b>	The processing unit is accessing the Conditioned Signal through some peripheral device that typically digitises the Conditioned Signal into a software representation. The transformation from the Conditioned Signal to the MCAL Signal has to take the digitalization error into account in order to provide information about the quality loss between the Technical Signal and the MCAL Signal.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.182 Metadata

<b>Definition</b>	Metadata is data about data
<b>Initiator</b>	Methodology and Templates
<b>Further Explanations</b>	Metadata includes pertinent information about data, including information about the authorship, versioning, access-rights, timestamps etc..
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.183 MetaDataItem

<b>Definition</b>	Defined item of MetaData for a Protocol Data Unit e.g. a diagnostic address, a MAC address, a CAN ID, or a J1939 node address.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	An ordered list of MetaDataItems defines the layout of PDU MetaData. Each MetaDataItem has a fixed type and length, and enables the accessing modules to parse the PDU MetaData, and to access items of the types that are relevant for the module.
<b>Comment</b>	MetaData was revised with AUTOSAR 4.3.
<b>Example</b>	A PDU exchanged between CanIf and PduR can carry the CAN ID as MetaDataItem to enable range routing of CAN messages.
<b>Reference</b>	--

#### 4.184 Microcontroller

<b>Definition</b>	Hardware element that integrates computing and communication resources as well as peripheral circuits in a single chip, including memories.
<b>Initiator</b>	General
<b>Further Explanations</b>	Microcontrollers are normally designed for small embedded systems and allow hardware designs with minimal amount of external parts. Microcontroller designs are normally optimized for silicon area and often support hard real-time and high-integrity demands.
<b>Comment</b>	Classic AUTOSAR is intended for Microcontroller based embedded systems.
<b>Example</b>	--
<b>Reference</b>	--

#### 4.185 Microcontroller Abstraction Layer

<b>Definition</b>	Software layer containing drivers to enable the access of onchip peripheral devices of a microcontroller and offchip memory mapped peripheral devices by a defined Application Programming Interface. Task: make higher software layers independent of the microcontroller.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	The Microcontroller Abstraction Layer is the lowest software layer of the Basic Software. The Microcontroller Abstraction Layer consists of the following parts: <ul style="list-style-type: none"> <li>• I/O Drivers</li> <li>• Communication Drivers</li> <li>• Memory Drivers</li> <li>• Crypto Drivers</li> <li>• Microcontroller Drivers</li> </ul> Properties: <ul style="list-style-type: none"> <li>• Implementation: <math>\mu</math>C dependent</li> <li>• Upper Interface (API): standardizable and <math>\mu</math>C independent</li> </ul>
<b>Comment</b>	--
<b>Example</b>	Examples of drivers located in the Microcontroller Abstraction Layer are: <ul style="list-style-type: none"> <li>• onchip eeprom driver</li> <li>• onchip adc driver</li> <li>• offchip flash driver</li> </ul>
<b>Reference</b>	[AUTOSAR Software Architecture]

#### 4.186 Mistake

<b>Definition</b>	Human error
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	[DIN 40041]

#### 4.187 Mode

<b>Definition</b>	A Mode is a certain set of states of the various state machines that are running in the vehicle that are relevant to a particular entity, e.g. a SW-C, a BSW module, an application or a whole vehicle. In its lifetime, an entity changes between a set of mutually exclusive Modes. These changes are triggered by environmental data, e.g. signal reception, operation invocation.
<b>Initiator</b>	Runtime Environment
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.188 Multimedia Stream

<b>Definition</b>	A consistent sequence of digital data versus time which is suited as input for devices which transfer these data into a continuous visible or audible impression to humans. When transferred over a physical link, multimedia stream data typically are produced at the same rate (by the data source), as they are consumed (by the data sinks).
<b>Initiator</b>	General
<b>Further Explanations</b>	A multimedia stream usually follows a certain standard (e.g. MPEG-x). When transferred over a physical link, a multimedia stream needs a certain minimum bandwidth (in terms of bits/second) in order to allow continuous impressions. A multimedia stream in a car typically exists for several seconds (a warning signal, a navigation hint) up to several hours (a video film, a phone call, playing a radio program). Resources (e.g. bus system channels) needed by the stream have to be allocated continuously over this lifetime (this is a difference to e.g. file transfer, which may be split into several chunks of data). The source of a multimedia stream typically is a specialized device and/or software program (a tuner, a microphone, a text-to-speech engine, etc.). The same holds for the sinks (an audio amplifier or mixer, a voice recognition software, an MPEG decoder, etc.).
<b>Comment</b>	The term "visible or audible impression to humans" should not be taken too literally, because streams can also be used to transfer machine readable data (e.g. modem, encrypted signals). But it is this condition, which defines the standards and technology used in multimedia streams.
<b>Example</b>	Audio stream as output of or input to a telephone (mono, low bandwidth) Audio stream as output of a radio tuner (stereo, high bandwidth) Video stream as output of a television tuner An example for the physical implementation on a multimedia bus is the Firewire isochronous stream. see reference
<b>Reference</b>	[IEEE 1394]

#### 4.189 Multiplexed PDU

<b>Definition</b>	A multiplexed PDU is a PDU with a configurable number of different payload layouts.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	Each instance of a multiplexed PDU has a distinct layout. The set of possible layouts is statically defined. A selector signal defines which layout is used in a

	PDU instance. The selector signal must reside at the same position in all layouts. Each layout is identified by a unique selector value. The length of each instance of a multiplexed PDU is fixed.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.190 Network Interface

<b>Definition</b>	A Network Interface is the sum of all interfaces offered by the Basic Software towards its connected network.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	<p>The interface that the Basic Software shares via the communication lines with other systems that behave like AUTOSAR ECUs in order to</p> <ul style="list-style-type: none"> <li>- allow distributed Software Components to exchange inter-ECU signals and to</li> <li>- operate the communication lines (the network)</li> </ul> <p>is called Network Interface.</p> <p>A Network Interface (NWI) denotes the interface between the Basic Software and the physical network (OSI Layer 0) to which the ECU executing the Basic Software is connected to (e.g. CAN, LIN, FlexRay). The NWI therefore transports network data packets between the Basic Software and the physical network.</p> <p>The interfaces included within the term NWI are:</p> <ul style="list-style-type: none"> <li>- Logical interfaces, including <ul style="list-style-type: none"> <li>o Network Management</li> <li>o Data Management</li> <li>o Data transmission/reception</li> </ul> </li> </ul> <p>The interfaces excluded from the term NWI are:</p> <ul style="list-style-type: none"> <li>- The physical network interface (CAN, FlexRay etc).</li> </ul> <p>Note that, attention must be given to the physical form of the network, since it is not formally specified by AUTOSAR.</p> <p>The NWI provided by a given ECU supports the transfer of data to and from the ECU, and management of the network.</p> <p>For the purposes of this definition, the Basic Software can be designed according to ICC1, ICC2 or ICC3.</p>
<b>Comment</b>	The term has been introduced as a short-hand to aid in discussion of the conformance of the content of ICC1 / 2 and to define the backward compatibility between releases and revisions. However, since from the network perspective, the clustering of the Basic Software is invisible, the Network Interface is applicable to all potential Basic Software conformance classes (ICC1, ICC2, ICC3) in the same way.
<b>Example</b>	--
<b>Reference</b>	Software Component Interface (SW-CI)

## 4.191 NM Coordination Cluster

<b>Definition</b>	A discrete set of NM Channels on which shutdown is coordinated.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	The NM Coordinator will keep all presently awake NM Channels of an NM Coordination Cluster awake until it is possible to coordinate network sleep on all the awake channels.

<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	AUTOSAR Generic NM Interface

## 4.192NM Coordinator

<b>Definition</b>	A functionality of the Generic NM Interface which allows coordination of network sleep for multiple NM Channels.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	Depending on configuration, different level of synchronous network sleep can be achieved. The NM Coordinator is using a generic coordination algorithm which, by means of individually configured timeout and synchronization indications can coordinate a synchronized shutdown of multiple NM Channels.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	AUTOSAR Generic NM Interface

## 4.193Notification

<b>Definition</b>	Informing a software entity about a state change of a hardware and/or software entity which has occurred.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	The informing about a state change can be done by an activation of a software part or by setting a Flag.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.194Onboard Security Event

<b>Definition</b>	Analog to ISO/IEC 27000:2018 an onboard security event (SEv) is the identified occurrence of an onboard system, service or network state indicating a possible breach of information security policy or failure of controls, or a previously unknown situation that can be security relevant.
<b>Initiator</b>	Security
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	The failed verification of a single secured PDU is typically an onboard security event.
<b>Reference</b>	

## 4.195OS Application

<b>Definition</b>	A block of software including tasks, interrupts, hooks and user services that form a cohesive functional unit.
<b>Initiator</b>	Software and Architecture

<b>Further Explanations</b>	<p>Trusted: An OS-Application that may be executed in privileged mode and may have unrestricted access to the API and hardware resources. Only trusted applications can provide trusted functions.</p> <p>Non-trusted: An OS-Application that is executed in non-privileged mode has restricted access to the API and hardware resources.</p>
<b>Comment</b>	The trusted / non-trusted attribute of an OS-Application is not related to ASIL/non-ASIL.
<b>Example</b>	--
<b>Reference</b>	[AUTOSAR Specification of OS]

## 4.196OS Event

<b>Definition</b>	<p>The event mechanism</p> <ul style="list-style-type: none"> <li>• is a means of synchronization</li> <li>• is only provided for extended tasks</li> <li>• initiates state transitions of tasks to and from the waiting state.</li> </ul>
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	[ISO 17356-3 (OSEK/VDX Operating System)]

## 4.197Partitioning

<b>Definition</b>	See ISO 26262, Part 1, ID 1.85
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	ISO 26262, Part 1, ID 1.85

## 4.198Protocol Control Information

<b>Definition</b>	Information which is needed to pass a Service Data Unit from one instance of a specific protocol layer to another instance. E.g. it contains source and target information.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	The PCI is added by a protocol layer on the transmission side and is removed again on the receiving side.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--



**4.199 Protocol Data Unit**

<b>Definition</b>	The Protocol Data Unit (PDU) contains Service Data Unit and Protocol Control Information.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	<p>On the transmission side the PDU is passed from the upper layer to the lower layer, which interprets this PDU as its SDU.</p> <p>The diagram illustrates the transmission of a PDU through multiple layers. It starts with Layer N+1, which contains a 'data structure' and is labeled as a 'PDU'. An arrow labeled 'LayerN_Tx (*PDU);' points down to Layer N. Layer N contains a 'data structure' and a 'PCI' box, labeled as an 'SDU'. A larger box labeled 'PDU' encompasses the 'PCI' and 'data structure' of Layer N. An arrow labeled 'void LayerN_Tx (*SDU);' points down to Layer N-1. Layer N-1 contains a 'data structure' and a 'PCI' box, labeled as an 'SDU'. An arrow labeled 'LayerN+1_Tx (*PDU);' points down to the TP layer. The TP layer contains a 'data structure' and a 'PCI' box, labeled as an 'SDU'. A larger box labeled 'PDU' encompasses the 'PCI' and 'data structure' of the TP layer. An arrow points down to the CAN layer. The CAN layer contains a 'data structure' and a 'PCI' box, labeled as an 'SDU'. A larger box labeled 'PDU' encompasses the 'PCI' and 'data structure' of the CAN layer.</p>
<b>Comment</b>	--
<b>Example</b>	ISO 17356-4 (OSEK/VDX Communication)
<b>Reference</b>	[ISO 17356-4 (OSEK/VDX Communication)]

**4.200 PDU MetaData**

<b>Definition</b>	Additional data of a Protocol Data Unit, which is not part of the payload.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	Metadata is placed alongside the PDU payload in a separate buffer. The layout of the MetaData is determined by an ordered list of MetaDataItems.
<b>Comment</b>	MetaData was introduced to transport parts of the CAN ID or addressing information alongside the data of a PDU.
<b>Example</b>	Diagnostics according to ISO 15765/14229, J1939 parameter group handling.
<b>Reference</b>	--

## 4.201 PDU Timeout

<b>Definition</b>	Maximum time between the receptions of two instances of one PDU is exceeded.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	This timeout indicates that the last reception of a PDU instance is too long in the past. As a consequence it can be concluded that the data in the last PDU instance is outdated.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.202 Performance

<b>Definition</b>	Performance is a set of measurable characteristics (e.g. time, memory, resources usage, power consumption, etc.) which may be used to compare different system, SW element, algorithm, etc. implementations.
<b>Initiator</b>	Safety
<b>Further Explanations</b>	Scalability refers to the characteristic of a system to increase performance by adding additional resources. If the software performance requirements change (e.g more functions that impact the response time), scalability comes into play. Scalability is the ability of a system to continue to meet its response time or throughput objectives as the demand for the software functions increases.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.203 Peripheral Hardware

<b>Definition</b>	Hardware devices integrated in micro-controller architecture to interact with the environment.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	Memory, CAN-Controller, ADC, DIO, etc.
<b>Reference</b>	--

## 4.204 Personalization

<b>Definition</b>	User-specific and memorized adjustment of SW data or selection of functional alternatives.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	Seat parameters (position, activation status of drive-dynamic seat) can be stored in correlation to a user ID. For a given user ID the seat can be adjusted according to the stored position parameters and the drive-dynamic seat can be activated or deactivated.

<b>Reference</b>	--
------------------	----

## 4.205 Plausibility

<b>Definition</b>	Runtime Plausibility check is a method to verify during runtime if inputs for a computation/algorithm or results of a computation/algorithm are reasonable against corresponding values of a simplified reference model.
<b>Initiator</b>	Safety
<b>Further Explanations</b>	Range checks are a subset of plausibility checks. The additional knowledge can be taken from various sources, e.g. the physical domain of the value or from a model representing the computation/algorithm more roughly and calculating in parallel to the actual computation/algorithm.
<b>Comment</b>	--
<b>Example</b>	<ul style="list-style-type: none"> <li>▪ Range Check: for determination that a value for a car velocity is plausible, the knowledge that a normal vehicle cannot be faster than 400km/h.</li> <li>▪ Plausibility: for determination that that a value for a car velocity is plausible, the history of the values can be used and the knowledge that a certain acceleration for a car cannot be exceeded. E.g. velocity was 10 km/h and increases within 1 Sec to 100 km/h.</li> </ul>
<b>Reference</b>	--

## 4.206 Policy Decision Point (PDP)

<b>Definition</b>	The PDP represents the logic in which the access control decision is made. It determines if the application is allowed to perform the requested task.
<b>Initiator</b>	Security
<b>Further Explanations</b>	The PDP provides an Access Control Decision.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.207 Policy Enforcement Point (PEP)

<b>Definition</b>	The PEP represents the logic in which the Access Control Decision is enforced. It communicates directly with the corresponding Policy Decision Point (PDP) to receive the Access Control Decision.
<b>Initiator</b>	Security
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.208 Port

<b>Definition</b>	A port belongs to a Software Component and is the interaction point between the component and other components. The interaction between specific ports of specific components is modeled using Connectors. A port can either be a Provide Port or an Require Port.
-------------------	--

<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	For more information see AUTOSAR Specification of VFB
<b>Example</b>	--
<b>Reference</b>	[AUTOSAR Specification of Virtual Functional Bus]

## 4.209Port Interface

<b>Definition</b>	A Port Interface characterizes the information provided or required by a Port of a Software Component.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	A Port Interface is either a Client-Server Interface in case Client-Server Communication is chosen or a Sender-Receiver Interface in case Sender-Receiver Communication is used.
<b>Comment</b>	For more information see: AUTOSAR Specification of VFB
<b>Example</b>	--
<b>Reference</b>	[AUTOSAR Specification of Virtual Functional Bus]

## 4.210 Post-build Time Configuration

<b>Definition</b>	The configuration of the SW module is possible after building the SW module.
<b>Initiator</b>	Methodology and Templates
<b>Further Explanations</b>	The SW may receive its configuration file that can be downloaded to the ECU separately, avoiding a re-compilation and re-build of the ECU SW modules. In order to make the post-build time re-configuration possible, the re-configurable elements shall be stored at a known position in the ECU storage area
<b>Comment</b>	--
<b>Example</b>	Identifiers of the CAN frames
<b>Reference</b>	--

## 4.211 Post-build Hooking

<b>Definition</b>	The insertion of Hooks to facilitate Rapid Prototyping support into a (complete) ECU hex image.
<b>Initiator</b>	Runtime Environment
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	Detection of reads and/or writes of ECU variables by analysis of the instruction stream.
<b>Reference</b>	--

## 4.212 Pre-build Hooking

<b>Definition</b>	The insertion of Hooks to facilitate Rapid Prototyping support into software source prior to creating an ECU hex image.
<b>Initiator</b>	Runtime Environment
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.213 Pre-Compile Time Configuration

<b>Definition</b>	The configuration of the SW module is done at source code level and will be effective after compile time.
<b>Initiator</b>	Methodology and Templates
<b>Further Explanations</b>	The source code contains all the ECU configuration data and when compiled together, it produces the given SW.
<b>Comment</b>	--
<b>Example</b>	Preprocessor switch for enabling the development error detection and reporting
<b>Reference</b>	--

#### 4.214 Predictability

<b>Definition</b>	Predictability is the degree to which a correct prediction or forecast of a system's state / behavior can be made either qualitatively or quantitatively.
<b>Initiator</b>	Safety
<b>Further Explanations</b>	Important type of predictability occurs in the design of systems that are subject to real-time requirements. A good overview of predictability criteria and how to achieve them can be found in
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	John A. Stankovic, Krithi Ramamritham, What is predictability for real-time systems?, Journal of Real-Time Systems, Volume 2 Issue 4, Nov. 1990, 247-254

#### 4.215 Pretended Networking

<b>Definition</b>	Method to reduce energy consumption in an existing active network without changing network infrastructure.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.216 Private Interface

<b>Definition</b>	A private interface is an interface within the Basic Software of AUTOSAR which is neither standardized nor defined within AUTOSAR.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	The goal of the private interface is to enable a more efficient implementation of basic software modules. Basic software modules sharing a private interface have to be distributed as one package. This package has to behave exactly the same as separate modules would. It must provide the same standardized interfaces to the rest of the basic software and/or RTE as separate modules would. It has to be configured exactly the same as separate modules would be configured.
<b>Comment</b>	Private interfaces contradict the goal of exchangeability of standard software modules and should be avoided.
<b>Example</b>	--
<b>Reference</b>	--

#### 4.217 Probability of Failure

<b>Definition</b>	Probability of the occurrence of a failure in a system or functional unit.
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.218 Procedure Call

<b>Definition</b>	A simple statement that provides the actual parameters for and invokes the execution of a procedure (software function).
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	A synchronous communication mechanism can be implemented by a procedure call.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	[ISO 2382-15]

## 4.219 Process

<b>Definition</b>	An executable unit managed by an operating system scheduler that has its own name space and resources (including memory) protected against use from other processes.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	A process consists of n Task (n>=1)
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.220 Processed Manifest

<b>Definition</b>	A Processed Manifest is a Manifest that is stored in the implementation specific format on the AUTOSAR Adaptive Platform product. Usually this is done in combination with other artifacts (like binary files) that contain executable code to which the Manifest applies.
<b>Initiator</b>	Execution Management
<b>Further Explanations</b>	Manifests are often used to denote a piece of configuration content that ships along a given piece of software and is used to deploy the software in the field.  There are several kinds of manifest: <ul style="list-style-type: none"> <li>▪ Execution Manifest</li> <li>▪ Machine Manifest</li> </ul>
<b>Comment</b>	The Manifest may contain platform implementation dependent data, as well as generic data derived from Application System Description.
<b>Example</b>	--
<b>Reference</b>	--

## 4.221 Proven In Use Argument

<b>Definition</b>	See ISO 26262, Part 1, ID 1.90
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	ISO 26262, Part 1, ID 1.90

#### 4.222 Provide Port

<b>Definition</b>	Specific Port providing Data or providing a service of a Server.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	The Provide Port is sometimes abbreviated as PPort or P-Port.
<b>Comment</b>	--
<b>Example</b>	<ul style="list-style-type: none"> <li>• Server Port</li> <li>• Sender Port</li> </ul>
<b>Reference</b>	--

#### 4.223 Rapid Prototyping

<b>Definition</b>	The experimental incorporation of new functionality.
<b>Initiator</b>	Runtime Environment
<b>Further Explanations</b>	Rapid Prototyping (RP) permits a user to quickly perform experiments to add new functionality, or to replace/bypass existing functionality, without requiring an ECU image to be built.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.224 Rapid Prototyping Memory Interface

<b>Definition</b>	The memory access pattern necessary for Rapid Prototyping Tool.
<b>Initiator</b>	Runtime Environment
<b>Further Explanations</b>	The RP memory interface provides the well-defined memory access pattern required by RP tool to ensure consistent and complete access to bypass values.
<b>Comment</b>	--
<b>Example</b>	A mandated “write-read” cycle within RTE APIs provides the RP tool with an opportunity to bypass (i.e. substitute with value generated from an alternative algorithm) the written value before it is read and then subsequently used within the generated code.
<b>Reference</b>	--

#### 4.225 Rapid Prototyping Tool

<b>Definition</b>	Software and/ or hardware tools to support Rapid Prototyping.
<b>Initiator</b>	Runtime Environment
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	Dedicated prototyping interfaces on ECUs accessed by PC-based software tools.
<b>Reference</b>	--



#### 4.226 Rate Conversion

<b>Definition</b>	Operation to change the timing between two transmissions of the same Pduld on one physical Network.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.227 Raw Data Stream

<b>Definition</b>	A Raw Data Stream is a collection of data that is unprocessed, basically a series of bytes without any information on how to interpret it.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	Raw Data Streams are used in communication links, mainly over ethernet reading streaming data from sensors. An API for Raw Data Streams is defined for Adaptive AUTOSAR applications in SWS Communication Management.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.228 Recovery

<b>Definition</b>	Returning to intended functionality after fault detection without violating the safety goals.
<b>Initiator</b>	Safety
<b>Further Explanations</b>	<ul style="list-style-type: none"> <li>▪ Restart mode: Restart Operation from the initial state of operation</li> <li>▪ Continue mode: Restart Operation from the last known state of operation</li> <li>▪ Recovery by repetition: repeat until timeout to cope i.e. random transmission errors.</li> <li>▪ Forward error recovery: relies on continue from an erroneous state by making selective corrections to the system state. This includes making the controlled environment safe, which may be damaged because of the failure</li> <li>▪ Backward error recovery: relies on restoring the system to a previous safe state and executing an alternative section of the program. This has the same functionality but uses a different algorithm (c.f. N-Version Programming)</li> <li>▪ Recovery Point: The point to which a process is restored is called a recovery point and the act of establishing it is termed check-pointing (saving appropriate system state)</li> <li>▪ Recovery testing is the forced failure of the software in a variety of ways to verify that recovery is properly performed.</li> </ul>
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.229 Redundancy

<b>Definition</b>	Existence of means in addition to the means that would be sufficient for an element to perform a required function or to represent information.
-------------------	---

	Hardware element redundancy includes replicated or additional hardware means added to the system to support fault tolerance. Software element redundancy includes the additional SW units and/or data used to support fault tolerance.
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	See ISO 26262, Part 1, ID 1.94

## 4.230 Reentrancy

<b>Definition</b>	In AUTOSAR a Function is called reentrant if it can be interrupted in the middle of its execution and then safely called again ("re-entered") before its previous invocations complete execution. AUTOSAR differs between <ul style="list-style-type: none"> <li>• (full) reentrancy</li> <li>• non reentrancy</li> </ul> and <ul style="list-style-type: none"> <li>• conditional reentrancy.</li> </ul>
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	Reentrancy is always considered from the viewpoint of the caller. A Function which is conditional reentrant has to document the conditions for the reentrancy. Typical cases for conditional reentrancy are functions which are reentrant as long as a function parameter is different to (possible) ongoing calls.
<b>Comment</b>	An implementation of a Function might be reentrant for one system but only conditional reentrant (or even non reentrant) for another one. It always depend how the reentrancy was realized (e.g. locks). As an example, just consider a function which uses interrupt locks to realize full reentrancy on a single core system. If this implementation is used in a multi core system its reentrancy will only be conditional reentrant for calls from the same core.
<b>Example</b>	--
<b>Reference</b>	--

## 4.231 Reliability

<b>Definition</b>	Probability of a system or functional unit to perform as expected under specified conditions within a time interval.
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.232 Relocatability

<b>Definition</b>	Capability of a software part being executed on different hardware environments without changing the code of the software part.
<b>Initiator</b>	Software and Architecture
<b>Further</b>	--

<b>Explanations</b>	
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.233 Require Port

<b>Definition</b>	Specific Port requiring Data or requiring a service of a server.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	The Require Port is sometimes abbreviated as RPort or R-Port.
<b>Comment</b>	--
<b>Example</b>	<ul style="list-style-type: none"> <li>• Client Port</li> <li>• Receiver Port</li> </ul>
<b>Reference</b>	--

#### 4.234 Required property

<b>Definition</b>	A <i>required</i> property or quality of a design entity (e.g. SW component or system) is a property or quality which has to be fulfilled by the environment of this design entity.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	A property or quality can be required by a stakeholder (e.g. customer) or another design entity.
<b>Comment</b>	--
<b>Example</b>	<p>1) In order to meet its functionality, a SW component A requires a minimum temporal resolution of a signal (information on a required port) which has to be fulfilled by SW component B.</p> <p>2) SW component requires to be activated by the runtime environment every 100ms with a jitter of 10ms.</p>
<b>Reference</b>	Compare term Asserted Property

#### 4.235 Residual Error Rate

<b>Definition</b>	The ratio of the number of bits, unit elements, or blocks incorrectly received and undetected, to the total number of bits, unit elements, characters, or blocks sent.
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.236 Resource

<b>Definition</b>	A resource is a required but limited hardware entity of an Electronic Control Unit, which in general can be accessed concurrently, but not simultaneously, by multiple software entities.
-------------------	---

<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	The definition from ISO 17356-3 (OSEK/VDX Operating System) cannot be used, due to the specific usage in ISO 17356.
<b>Example</b>	CPU-load, interrupts (mechanism itself and the resulting CPU-load), memory, peripheral hardware, communication, ...
<b>Reference</b>	--

## 4.237 Resource-Management

<b>Definition</b>	Entity which controls the use of Resources.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	The main functionality of resource management is the control of simultaneous use of a single resource by several entities, e.g. scheduling of requests, multiple access protection.
<b>Comment</b>	--
<b>Example</b>	OS-scheduler (CPU-load management)
<b>Reference</b>	--

## 4.238 Response Time

<b>Definition</b>	Time between receiving a stimulus and delivering an appropriate response or reaction.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	The response time describes the time between a stimulus like e.g. the state change of hardware or software entity and the expected reaction of the system (e.g. response, actuator activation). Synonym: reaction time See also: execution time, worst case execution time and worst case response time.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.239 Risk

<b>Definition</b>	See ISO 26262, Part 1, ID 1.99
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	ISO 26262, Part 1, ID 1.99

## 4.240 Robustness

<b>Definition</b>	Ability of a system or functional unit to perform as expected also under
-------------------	--

	unexpected conditions.
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.241 RTE Event

<b>Definition</b>	An RTE Event encompasses all possible situations that can trigger execution of a Runnable Entity by the RTE. Thus they can address timing, data sending and receiving, invoking operations, call server returning, mode switching, or external events. RTE Events can either activate a runnable entity or wakeup a runnable entity at its waitpoints.
<b>Initiator</b>	Runtime Environment
<b>Further Explanations</b>	Note 'event' in this context is not necessarily synonymous with 'RTEEvent' as defined in the VFB specification. In particular, RTE Events that result from communication are handled by communication-triggered runnable entities.
<b>Comment</b>	Events can have a variety of sources including time.
<b>Example</b>	Scheduling of runnable entities from angular position, e.g. a crankshaft, that are used to trigger an interrupt and hence an RTE notification. A software component needs to perform a regular interval, e.g. flash an LED, reset a watchdog, etc.
<b>Reference</b>	--

#### 4.242 Runnable Entity

<b>Definition</b>	A Runnable Entity is a part of an Atomic Software Component which can be executed and scheduled independently from the other Runnable Entities of this Atomic Software-Component. It is described by a sequence of instructions that can be started by the RTE. Each runnable entity is associated with exactly one Entry Point.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	A Runnable Entity contains at least two points for the Scheduler: 1 Entry Point and 1 Exit Point. Due to the reason that an Atomic Software Component is not dividable, all its Runnable Entities are executed on the same ECU.
<b>Comment</b>	In general a task in the runtime system consists out of n Runnable Entities of m Atomic Software-Components.
<b>Example</b>	Server function of a Software Component.
<b>Reference</b>	--

#### 4.243 SAE J1939

<b>Definition</b>	SAE J1939 is a vehicle bus standard created by the SAE (Society of Automotive Engineers, a USA standards body) for car and heavy duty truck industries.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	The J1939 standard encompasses the following areas: - bus physics (J1939/11, J1939/15) - CAN message layout (J1939/21)

	<ul style="list-style-type: none"> <li>- request/response and multi packet transport protocols (J1939/21)</li> <li>- network management used to assign a unique address to each node (J1939/81)</li> <li>- diagnostics layer comparable to UDS in complexity (J1939/73)</li> <li>- standardized application signals and messages (J1939/71)</li> </ul>
<b>Comment</b>	The J1939 standard is used by most truck manufacturers worldwide and is prescribed for OBD in some states of the USA. It is also used as a base for other standards for maritime (NMEA 2000), agricultural (ISO 11783), and military (MiICAN A) applications.
<b>Example</b>	--
<b>Reference</b>	<a href="http://www.sae.org/">http://www.sae.org/</a>

#### 4.244 Safe State

<b>Definition</b>	See ISO 26262, Part 1, ID 1.102
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	ISO 26262, Part 1, ID 1.102

#### 4.245 Safety

<b>Definition</b>	See ISO 26262, Part 1, ID 1.103
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	ISO 26262, Part 1, ID 1.103

#### 4.246 Safety Case

<b>Definition</b>	See ISO 26262, Part 1, ID 1.106
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	ISO 26262, Part 1, ID 1.106

#### 4.247 Safety Goal

<b>Definition</b>	See ISO 26262, Part 1, ID 1.108
<b>Initiator</b>	Safety
<b>Further</b>	--

<b>Explanations</b>	
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	ISO 26262, Part 1, ID 1.108

#### 4.248 Safety Measure

<b>Definition</b>	See ISO 26262, Part 1, ID 1.110
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	ISO 26262, Part 1, ID 1.110

#### 4.249 Safety Mechanism

<b>Definition</b>	See ISO 26262, Part 1, ID 1.111
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	ISO 26262, Part 1, ID 1.111

#### 4.250 Safety Protocol

<b>Definition</b>	A communication protocol defining the necessary mechanisms to ensure the integrity of transmitted data and to detect any communication related error.
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.251 Sample Application

<b>Definition</b>	Defined system used for evaluation purposes.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	The application may be simplified for better understanding within the evaluation phase.
<b>Comment</b>	--
<b>Example</b>	Diagnosis Application Exterior Light Management

<b>Reference</b>	--
------------------	----

## 4.252 Scalability

<b>Definition</b>	The degree to which assets can be adapted to specific target environments for various defined measures.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	Target environment introduced compared to EAST-Glossary.
<b>Example</b>	--
<b>Reference</b>	[EAST-Glossary]

## 4.253 Scheduler

<b>Definition</b>	The scheduler handles the scheduling of the tasks/runnable entities (definition→ 4.298 / 4.242) according to the priority and scheduling policy (pre-defined or configurable). It has the responsibility to decide during run-time when which task can run on on the CPU of the ECU.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	There are many strategies (priority-based, time-triggered, round-robin, ...) a scheduler can use, depending of the selected and/or implemented algorithms
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	[AUTOSAR Specification of Virtual Functional Bus]

## 4.254 Service Data Unit

<b>Definition</b>	Service Data Unit is the data passed by an upper layer, with the request to transmit the data. It is as well the data, which is extracted after reception by the lower layer and passed to the upper layer.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	A SDU is part of a Protocol Data Unit.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.255 Security

<b>Definition</b>	Protection of data, software entities or resources from accidental or malicious acts.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	Slightly adapted norm.
<b>Example</b>	--
<b>Reference</b>	[ISO 2382-8]



## 4.256 Secure Channel

<b>Definition</b>	A secure channel is a communication channel between two parties. A secure channel shall at least provide integrity and authenticity of the communication. It shall provide means to authenticate at least one participant of the communication. It may provide means to mutually authenticate both participants of the communication. A secure channel may additionally provide the confidentiality of the communication.
<b>Initiator</b>	Security
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	A TLS or IPsec channel for secure SOME/IP communication between two Adaptive Platform instances.
<b>Reference</b>	--

## 4.257 Security Event

<b>Definition</b>	An event which is related to the security of the ECU and should be reported for later analysis.
<b>Initiator</b>	Security
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	Failed negotiation of a shared secret, failed verification of a download signature, successful update of a certificate
<b>Reference</b>	--

## 4.258 Sender-Receiver Communication

<b>Definition</b>	A communication pattern which offers asynchronous distribution of information where a sender communicates information to one or more receivers, or a receiver receives information from one or several senders.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	The process of sending data does not block the sender and the sender usually gets no response from the receivers
<b>Comment</b>	Often used for data or event distribution
<b>Example</b>	--
<b>Reference</b>	[AUTOSAR Specification of Virtual Functional Bus]

## 4.259 Sender-Receiver Interface

<b>Definition</b>	A sender-receiver interface is a special kind of Port Interface for the case of Sender-Receiver Communication. The sender-receiver interface defines the data-elements which are sent by a sending component (which has a p-port providing the sender-receiver interface) or received by a receiving component (which has an r-port requiring the sender-
-------------------	--

	receiver interface).
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	A special kind of Port-Interface
<b>Example</b>	--
<b>Reference</b>	[AUTOSAR Specification of Virtual Functional Bus]

#### 4.260 Sensor/Actuator SW-Component

<b>Definition</b>	Software Component dedicated to the control of a sensor or actuator.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	There will be several Sensor/ Actuator SW-Cs in each ECU. In general there will be one Sensor/Actuator SW-C for each sensor and one for each actuator (=> number of Sensor/Actuator SW-C = number of sensors + number of actuators).
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.261 Server

<b>Definition</b>	Software entity which provides services for Clients
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	The Server and the clients using its service might be located on one ECU or distributed on different calculation units (e.g. ECU).
<b>Comment</b>	Adapted from Balzert.
<b>Example</b>	--
<b>Reference</b>	[Balzert99]

#### 4.262 Service

<b>Definition</b>	A service is a type of operation that has a published specification of interface and behavior, involving a contract between the provider of the capability and the potential clients.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	Diagnosis service, ...
<b>Reference</b>	[EAST-Glossary]

#### 4.263 Service Discovery

<b>Definition</b>	Generic functionality provided by the Communication Management to Applications that allows Applications at runtime to find locally or remotely available Service Instances providing the requested service.
<b>Initiator</b>	Communication
<b>Further</b>	Based on Application query, the Communication Management provides list of

<b>Explanations</b>	compatible Service Instances. Compatibility is defined by compatibility rules and may consider version or QoS attributes.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.264 Service Instance

<b>Definition</b>	The properties of a service instance are described by a specific service interface. A service instance has a unique identity.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	It is accessible by other Applications by using a Service Requester Proxy at runtime and is typed by a specific Service Interface.  It is addressable within the vehicle network by its Service Instance ID, an abstraction from of the physical location.  Optionally, authentication data is associated with a Service Instance for authentication at runtime.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.265 Service Interface

<b>Definition</b>	A service interface is a special kind of port interface (see Port Interface) used in the Adaptive platform. It defines both data elements for event based communication and operations that are provided by the service provider and that can be used by the service requester.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.266 Service Oriented Communication

<b>Definition</b>	Communication, for which communication partners are generally not defined during design time, but dynamically discovered and bound during runtime.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	Communication partners are generally not defined during design time, but dynamically discovered and bound during runtime. Adaptive AUTOSAR Applications are therefore developed agnostic to the concrete context, assuming model of loosely-coupled components.
<b>Comment</b>	This is the standard communication paradigm for communication between AUTOSAR Adaptive Applications.
<b>Example</b>	--
<b>Reference</b>	--

## 4.267 Service Port

<b>Definition</b>	
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	The interface of a Service Port has to be a Standardized AUTOSAR Interface. A Service Port does not need to be connected to another Port in the VFB View.
<b>Comment</b>	If a service is provided by the ECU where a specific Atomic Software Component is located the VFB View is sufficient. If a service is provided by another ECU the connection of the service call to the service has to be done explicitly during the mapping step.
<b>Example</b>	Write data to non volatile memory.
<b>Reference</b>	--

## 4.268 Service Proxy

<b>Definition</b>	A facade that represents a specific service on code level from the perspective of the service consumer providing methods for all functionalities offered by the represented service.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	The service consumer side Application code interacts with this local facade, which then knows how to propagate these calls to the real service implementation and back. The Service Proxy is typically an instance of a service proxy class which itself is potentially generated from ServiceInterface according to standardized patterns and implemented by platform-specific Communication Management.  The Service Proxy provides placeholder for Service Instance ID, which is set at runtime by requesting Application implementation using Service Discovery or statically based on Planned Dynamics.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.269 Service Skeleton

<b>Definition</b>	A representation of a specific service on code level from the perspective of the service implementation, which provides functionalities according to the service definition and allows to connect the service implementation to the Communication Management transport layer, so that the service implementation can be contacted by service consumers.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	The Service Skeleton is typically an instance of a service skeleton class which itself is potentially generated from a ServiceInterface according to standardized patterns and implemented by platform-specific Communication Management.  The skeleton provides placeholder for the Service Instance ID, which is set by platform implementation, e.g. based on Application Description at design time, or by the Vehicle Software Configuration Manager at setup.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.270 Services Layer

<b>Definition</b>	The Services Layer is the highest layer of the Basic Software which also applies for its relevance for the application software: while access to I/O signals is covered by the Hardware Abstraction Layer, the Services Layer offers Operating system services Vehicle network communication and management services Memory services (NVRAM management) Diagnosis Services (including KWP2000 interface and error memory) ECU state management  Task: Provide basic services for application and basic software modules
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	The Services Layer consists of the following parts: Communication Services Memory Services System Services
<b>Comment</b>	--
<b>Example</b>	Network Management, NVRAM Manager, ECU State Manager
<b>Reference</b>	[AUTOSAR Software Architecture]

## 4.271 Signal Service Translation

<b>Definition</b>	Signal Service Translation is the standardized way to map the definition of signal-based communication to service-oriented communication, and vice versa.
<b>Initiator</b>	Manifests & Templates
<b>Further Explanations</b>	Adaptive Platform restricts communication paradigm to service-oriented communication, a major part of the vehicle however still uses signal-based communication means - therefore a translation of these two approaches has to be performed.
<b>Comment</b>	One goal of AUTOSAR is to support the development of a whole vehicle in a seamless way. The translation of communication paradigms is essential for a holistic design. The AUTOSAR Methodology supports the translation activity on either a Classic platform gateway ECU or on an Adaptive platform Machine.
<b>Example</b>	--
<b>Reference</b>	--

## 4.272 Slave Switch

<b>Definition</b>	A Slave Switch is an Ethernet switch which is connected to a master switch by uplink ports
<b>Initiator</b>	Communication
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.273 Software Cluster

<b>Definition</b>	A Software Cluster groups all AUTOSAR artifacts which are relevant to deploy software on a machine. This includes the definition of applications, i.e. their executables, execution manifests, communication and diagnostics. In the context of diagnostics a Software Cluster might be addressed individually by its own set of diagnostic addresses.
<b>Initiator</b>	Diagnostics
<b>Further Explanations</b>	--
<b>Comment</b>	A Software Cluster is used to partition Applications running on a Machine into individually updateable clusters.
<b>Example</b>	--
<b>Reference</b>	SRS_Diag_04216

#### 4.274 Software Component

<b>Definition</b>	Software-Components are architectural elements that provide and/or require interfaces and are connected to each other through the Virtual Function Bus to fulfill architectural responsibilities.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	A Software Component has a formal description defined by the software component template. Software Components can be abbreviated as SW-Cs. SW-Cs may be atomic components, parameter components or compositions. Also the software modules providing the Software Component Interface of a Basic Software Module are called Software Components.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.275 Software Component Interface

<b>Definition</b>	A Software-Component Interface (SW-CI) is the sum of all interfaces offered by the Basic Software towards the Software Component.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	A SW-CI denotes the interface between an SW-C and the underlying Basic Software cluster including the RTE. The SW-CI therefore comprises all Application Programming Interface Functions and Callbacks that the SW-C requires from and provides to the Basic Software (generally by means of RTE mechanisms). It includes also the mechanisms allowing SW-Cs sharing the SW-CI to communicate with one another. For the purposes of this definition, the Basic Software clustered on an ECU can be designed according to ICC1, 2 and 3.
<b>Comment</b>	The term has been introduced as a short-hand to aid in discussion of the conformance of the content of Basic Software clusters of conformance class ICC1 / 2 and to define the backward compatibility between releases and revisions. However, since from the SW-C perspective, the clustering of the Basic Software is invisible, the Component Interface is applicable to all potential Basic Software conformance classes (ICC1, ICC2, ICC3) in the same way.
<b>Example</b>	--
<b>Reference</b>	Network Interface (NWI)

## 4.276 Software Configuration

<b>Definition</b>	The arrangement of software elements in a SW system.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	A software element is a clearly definable software part. A software configuration is a selection version of software modules, software components, parameters and generator configurations. Calibration and Variant Coding can be regarded as subset of Software Configuration.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	[EAST-Glossary]

## 4.277 Software Interrupt

<b>Definition</b>	Interrupt triggered by SW event.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	SW events are for example calling an operating system service, starting a process with higher priority.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	Translation/Adaptation from [VDI Lexikon]

## 4.278 Software Package

<b>Definition</b>	Unit for deployment of software onto Adaptive AUTOSAR Platform instances containing zero or more executables and the metadata to install and execute it on the Machine.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	Typically, a software package contains one or more executables however it is permitted to have no executables to enable update of configuration metadata.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	Translation/Adaptation from [VDI Lexikon]

## 4.279 Software Signal

<b>Definition</b>	A Software Signal is an asynchronous event transmitted between one process and another.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	A SW Signal is the software implementation of an (control-) information. Additionally it may have attributes (e.g. freshness, data type, ...). It is exchanged between SW-Components.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.280 Software Unit

<b>Definition</b>	See ISO 26262, Part 1, ID 1.125
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	ISO 26262, Part 1, ID 1.125

#### 4.281 Special Periphery Access

<b>Definition</b>	Special functions to standard peripheral devices or special peripherals.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	Is only used when, because of technical issues, no standard periphery access can be used
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.282 Standard Periphery Access

<b>Definition</b>	Standard functions to typical standard peripheral devices that are available on an ECU (most microcontroller integrated) used in automotive embedded applications.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	Digital Input/Output, Analog/Digital Converter, Pulse Width (De)Modulator, EEPROM, FLASH, Capture Compare Unit, Watchdog Timer
<b>Reference</b>	--

#### 4.283 Standard Software

<b>Definition</b>	Standard Software is software which provides schematic independent infrastructural functionalities on an ECU. It contains only Standardized Interfaces, Standardized AUTOSAR Interfaces and/or Private Interfaces.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	ISO 17356, MCAL, Services
<b>Reference</b>	ISO 17356, Road vehicles -- Open interface for embedded automotive applications

#### 4.284 Standardized AUTOSAR Blueprint

<b>Definition</b>	A Standardized AUTOSAR Blueprint is an AUTOSAR Blueprint standardized
-------------------	---



	within the AUTOSAR project. Its derived objects are considered as being standardized within the AUTOSAR project as well.
<b>Initiator</b>	Application Interfaces
<b>Further Explanations</b>	Blueprints were introduced within the AUTOSAR projects to enable standardization of ports without standardizing the static view of the architecture (i.e. the software components providing or requesting the ports). Sometimes it is not possible to standardize all attributes of an AUTOSAR element because the values of some attributes are project specific. Nevertheless it enables better collaboration if some of the attributes are standardized. Additionally blueprints enable adding descriptions and long names in different languages.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	[MOD AI Specification]

#### 4.285 Standardized AUTOSAR Interface

<b>Definition</b>	This is an AUTOSAR Interface which is standardized within the AUTOSAR project.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	AUTOSAR Services interact with other components through a Standardized AUTOSAR Interface. AUTOSAR Interfaces can be derived from AUTOSAR Application Interfaces.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.286 Standardized Blueprint

<b>Definition</b>	A Blueprint is called a Standardized Blueprint if the derived objects are considered as being standardized as well. It also includes that additionally concrete standardized rules exist how to specify the blueprint as well as how to derive an object from that blueprint. This is typically not done for a specific blueprint but for all blueprints of the same class.
<b>Initiator</b>	Application Interfaces
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

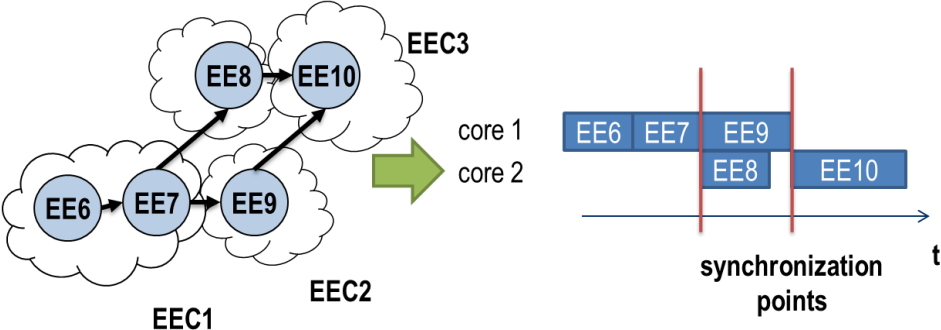
#### 4.287 Standardized Interface

<b>Definition</b>	A software interface is called Standardized Interface if a concrete standardized API exists.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	Modules in the Basic Software interact which each other through Standardized Interfaces.
<b>Comment</b>	--
<b>Example</b>	ISO 17356-4: COM Interface
<b>Reference</b>	--

### 4.288 Static Configuration

<b>Definition</b>	A setup where the routing configuration cannot be changed during normal operation of the gateway.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	Static configuration doesn't allow reconfiguration of the routing during normal operation e.g. during driving. Static configuration does not restrict the update of the configuration in specific maintenance operation modes (e.g. programming mode).
<b>Comment</b>	--
<b>Example</b>	A software update may change a routing configuration such that a PDU is routed into two instead of one destination networks.
<b>Reference</b>	--

### 4.289 Synchronization Points

<b>Definition</b>	Used to synchronize the execution of EEs of different tasks that execute within the same LET interval. A synchronization point realizes the execution order of EEs that belong to different EECs.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	This pattern is used when EEs on multiple cores are synchronized, e.g. to ensure Execution Order Constraints.   <p>Two kinds of synchronization are in focus here. The execution of Executable Entity Cluster may be synchronized by actively waiting at a barrier or by advancing the execution until to relative point in time after LET interval start.</p>
<b>Comment</b>	Synchronization points can be derived directly from the EOCs and the mapping of EEs to tasks. The synchronization can be implemented by time (advance) or as a barrier. A synchronization point between EECs that are mapped to the same task can already be fulfilled by the EE positions in the task.  It is important to know that direct access before the synchronization point does not lead to interference, i.e. no other access to the same memory location can take place. The use of advance requires confident knowledge of the EE's WCET.
<b>Example</b>	Execution Order Constraints between Executable Entities imposed by a data dependency. A Runnable Entity C waiting for input from Runnable Entity A and B. A synchronization point before C guarantees that A and B have produced the input for C when it starts execution.

<b>Reference</b>	--
------------------	----

## 4.290 Synchronize

<b>Definition</b>	To make two or more events or operations to occur at the same predefined moment in time.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	Two NM Channels can enter Bus Sleep Mode at the same time ("synchronized network sleep") or they can be ordered to go to sleep at the same time ("synchronized shutdown initiation").
<b>Reference</b>	AUTOSAR Generic NM Interface

## 4.291 Synchronous Communication

<b>Definition</b>	A communication is synchronous when the calling software entity is blocked until the called operation is evaluated. The calling software entity continues its operation by getting the result.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	Synchronous communication between distributed functional units has to be implemented as remote procedure call.
<b>Comment</b>	Are further mechanisms possible?
<b>Example</b>	--
<b>Reference</b>	--

## 4.292 Synchronous Function

<b>Definition</b>	A function is called synchronous if the described functionality is guaranteed to be completed the moment the function returns to the caller.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.293 System

<b>Definition</b>	An integrated composite that consists of one or more of the processes, hardware, software, facilities and people, that provides a capability to satisfy a stated need or objective.
<b>Initiator</b>	General
<b>Further Explanations</b>	--
<b>Comment</b>	ITEA EAST uses IEEE 14407 standard. Here not applicable because of problem with the definition of function.

	One correct interpretation is: - it might be a composition of one or more ECUs
<b>Example</b>	Braking system
<b>Reference</b>	[ISO 12207]

#### 4.294 System Constraint

<b>Definition</b>	Boundary conditions that restrict the Design-Freedom of the (cars E/E-) System.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	The design of ECU Networks and the distribution of functionalities to ECUs are limited by several constrains. These constraints result mostly by the communication matrix and safety requirements
<b>Comment</b>	--
<b>Example</b>	An existing communication matrix that restricts the distribution of signals to frames is a system constraint. Another system constraint is a safety requirement that does not allow to map a specified Software component to specific ECU.
<b>Reference</b>	--

#### 4.295 System Health Monitor

<b>Definition</b>	System Health Monitor (SHM) analyzes the health of subsystems.
<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	ASWS_HealthMonitoring

#### 4.296 System Signal

<b>Definition</b>	The system signal represents the communication system's view of data exchanged between SW components which reside on different ECUs. The system signals allow to represent this communication in a flattened structure, with (at least) one system signal defined for each data element sent or received by a SW component instance. If data has to be sent over gateways, there is still only one system signal representing this data. The representation of the data on the individual communication systems is done by the cluster signals.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

#### 4.297 Systematic Fault

<b>Definition</b>	See ISO 26262, Part 1, ID 1.131
-------------------	---------------------------------

<b>Initiator</b>	Safety
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	ISO 26262, Part 1, ID 1.131

## 4.298 Task

<b>Definition</b>	A Task is the smallest schedulable unit managed by the OS. The OS decides when which task can run on the CPU of the ECU.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	A Runnable Entity of a software component runs in the context of a task. Also the Basic Software Modules runs in the context of a task.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	[AUTOSAR Specification of Virtual Functional Bus]

## 4.299 Technical Signal

<b>Definition</b>	The technical signal is the physical value of an external event coupled to an AUTOSAR system. Technical signals are represented in SI units (e.g. pressure in PA).
<b>Initiator</b>	General
<b>Further Explanations</b>	The term Technical Signal is used when we are referring to the "real world" signal that is under consideration. So typical Technical Signals are temperature, velocity, torque, force, electrical current and voltage, etc.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

## 4.300 Timed Communication

<b>Definition</b>	Implicit communication with logical publication at the end of a LET interval and logical read at the beginning of a LET interval. Access to the timed communication buffer is available for all EEs that reference the LET interval through EECs.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	Data consistency is guaranteed between EECs which reference the same LET interval (regardless of their task and core mapping). <div style="text-align: center;"> <p>The diagram illustrates the execution of three tasks with different priorities: 5 ms task (highest), 10 ms task (middle), and 20 ms task (lowest). The vertical axis represents task priority, and the horizontal axis represents time (t). Each task's execution is shown as a series of blue blocks. Communication events are marked with yellow vertical bars. A red line indicates the sequence of communication events, showing that higher-priority tasks can interrupt lower-priority tasks. A red arrow points to a specific communication event labeled 'LET time interval'.</p> </div>

	Implicit communication crossing LET interval boundaries is redefined to timed communication.
<b>Comment</b>	One OS task can have several implicit communication buffers for several LET intervals. Implicit communication between EEs referencing the same LET interval remains unchanged.
<b>Example</b>	--
<b>Reference</b>	--

### 4.301 Timeout

<b>Definition</b>	Notification with respect to deadline violation of an event or task (e.g. while working on/with information: receiving, sending, processing, etc.).
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

### 4.302 Uplink Port

<b>Definition</b>	A Uplink Port is a port of an automotive Ethernet switch which is connected to another Ethernet automotive switch (cascaded switch). A Uplink Port could either be an internal port or an external port. One Uplink Port is connected to another Uplink Port. The Uplink Port has a special role from the perspective of the software.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

### 4.303 Use Case

<b>Definition</b>	A model of the usage by the user of a system in order to realize a certain functional feature of the system.
<b>Initiator</b>	General
<b>Further Explanations</b>	--
<b>Comment</b>	Added certain compared to EAST-glossary.
<b>Example</b>	--
<b>Reference</b>	[EAST-Glossary]

### 4.304 Validation

<b>Definition</b>	Confirmation by examination and provisions of objective evidence that the
-------------------	---

	particular requirements for a specific intended use are fulfilled.
<b>Initiator</b>	General
<b>Further Explanations</b>	In design and development, validation concerns the process of examining a product to determine conformity with user needs. Validation is normally performed on the final product under defined operating conditions. It may be necessary in earlier stages. "Validated" is used to designate the corresponding status. Multiple validations may be carried out if there are different intended uses. [ISO 8402: 1994]
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	[IEEE 1012:1998]

### 4.305 Variability

<b>Definition</b>	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.
<b>Initiator</b>	Methodology and Templates
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	As an example, such a system property selection manifests itself in a particular "receive port" for a connection.
<b>Reference</b>	--

### 4.306 Variant

<b>Definition</b>	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.
<b>Initiator</b>	Methodology and Templates
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

### 4.307 Variant Coding

<b>Definition</b>	Adaptation of SW by selection of functional alternatives according to external requirements (e.g. country-dependent or legal restrictions).
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	The major difference with calibration is that this later doesn't aim to adapt the SW functionality itself but only aims to adjust the SW to the HW/SW environment, e.g. the calibration of engine control SW that is adjusted to the physical parameters of every engine. Variant Coding also includes vehicle-specific (not user-specific) SW adaptation due to end-customer wishes (e.g. deactivation of speed-dependent automatic locking). Variant Coding is always done after compile time. Used techniques to select variants include end-of-line programming and garage programming.
<b>Comment</b>	--

<b>Example</b>	Country related adaptation of MMI with respect to speed and/or temperature unit (km/h vs. mph, °C vs. F).
<b>Reference</b>	--

### 4.308 Variation Binding

<b>Definition</b>	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.
<b>Initiator</b>	Methodology and Templates
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

### 4.309 Variation Binding Time

<b>Definition</b>	The variation binding time determines the step in the methodology at which the variability given by a set of variable properties is resolved.
<b>Initiator</b>	Methodology and Templates
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

### 4.310 Variation Point

<b>Definition</b>	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.
<b>Initiator</b>	Methodology and Templates
<b>Further Explanations</b>	--
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

### 4.311 Vehicle State Manager

<b>Definition</b>	An OEM specific application that controls vehicle level state and interacts with various entities in the vehicle system.
<b>Initiator</b>	Safety and UCM
<b>Further Explanations</b>	--
<b>Comment</b>	The OEMs may just have one such application and send the vehicle state on the communication bus or may have one Master Vehicle State Manager and Client Vehicle State Manager applications on each ECU.



<b>Example</b>	--
<b>Reference</b>	SWS_UpdateAndConfigurationManagement

### 4.312 Vendor ID

<b>Definition</b>	A vendor ID is a unique identification of the vendor of a software component. All basic software modules (→definition 4.43) conformant to the AUTOSAR standard shall provide a readable vendor ID.
<b>Initiator</b>	General
<b>Further Explanations</b>	AUTOSAR Vendor IDs are used to determine vendors of basic software modules before and during runtime. The mechanism is used to improve bug handling. AUTOSAR currently only provides Vendor IDs to members of the AUTOSAR partnership.
<b>Comment</b>	To apply for an AUTOSAR vendor ID the possible member has to send an E-Mail to request@autosar.org. Within the request name of the company, company address and contact person should be listed.
<b>Example</b>	Vendor ID for EEPROM driver is called: EEP_VENDOR_ID
<b>Reference</b>	SRS_BSW_00374

### 4.313 Verification

<b>Definition</b>	Confirmation by examination and provisions of objective evidence that specified requirements have been fulfilled.
<b>Initiator</b>	General
<b>Further Explanations</b>	In design and development, verification concerns the process of examining the result of a given activity to determine conformity with the stated requirement for that activity. “Verified” is used to designate the corresponding status. [ISO 8402: 1994]
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	[IEEE 1012:1998]

### 4.314 VFB View

<b>Definition</b>	The VFB View describes systems or subsystems in the car independently of these resources; in other words, independently of: <ul style="list-style-type: none"> <li>• what kind of and how many ECUs are present in the car</li> <li>• on what ECUs the entities in the VFB-View run</li> <li>• how the ECUs are interconnected: what kind of network technology (CAN, LIN,...) and what kind of topology (presence of gateways) is used</li> </ul>
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	In the VFB-View, the system or subsystem under consideration is a Composition which consists out of Connectors and Components.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	[AUTOSAR Specification of Virtual Functional Bus]

### 4.315 Virtual Functional Bus

<b>Definition</b>	The Virtual Functional Bus is an abstraction of the communication between Atomic Software Components and AUTOSAR Services. This abstraction is such that specification of the communication mechanisms is independent from the concrete technology chosen to realize the communication.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	After compilation and linking of software for a dedicated Electronic Control Unit the Virtual Functional Bus interfaces are realized by the AUTOSAR Runtime Environment.
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

### 4.316 Virtual Integration

<b>Definition</b>	The simulated, modeled and/or calculated (not real) combination of software entities forming a System.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	By virtual integration several constraints and/or requirements are checked without the need of real hardware units, like needed CPU load, needed memory, completeness of interfaces, fulfillment of timing requirements etc.).
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

### 4.317 Virtualization

<b>Definition</b>	Virtualization is a mechanism which hides the physical characteristics of a computing platform from the users, presenting instead another abstract computing platform. It can be used to fulfill functional safety requirements like availability, partitioning, resource conflict management, recovery etc.
<b>Initiator</b>	Safety
<b>Further Explanations</b>	<p>Different types of hardware virtualization include:</p> <ul style="list-style-type: none"> <li>▪ Full virtualization – almost complete simulation of the actual hardware to allow software, which typically consists of a guest operating system, to run unmodified.</li> <li>▪ Partial virtualization – some but not all of the target environment attributes are simulated. As a result, some guest programs may need modifications to run in such virtual environments.</li> <li>▪ Paravirtualization – a hardware environment is not simulated; however, the guest programs are executed in their own isolated domains, as if they are running on a separate system. Guest programs need to be specifically modified to run in this environment.</li> </ul>
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	--

### 4.318 Wake-up and sleep on dataline

<b>Definition</b>	Wake-up request or sleep request which is transmitted over a dataline of an
-------------------	---

	Automotive Ethernet switched network.
<b>Initiator</b>	Communication
<b>Further Explanations</b>	Wake-up on dataline is used for Ethernet switched network and is defined for 100Base-T1. A wake-up on dataline without an established link between the communication partner is called a wake-up pulse (WUP). A wakeup on dataline with an established link between the communication partners is called a wake-up request (WUR). A sleep request is transmitted as a burst of continues LPS (low power signal)
<b>Comment</b>	--
<b>Example</b>	--
<b>Reference</b>	OPEN ALLIANCE Sleep/Wake-up specification for Automotive Ethernet

### 4.319 Worst Case Execution Time

<b>Definition</b>	Maximum possible time during which a program is actually executing
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	The worst case execution time of a piece of software is the maximum possible time during which the CPU is executing instructions which belong to this piece. The worst case execution time is often identified by analytical methods. It is required to determine if a schedule meets the overall timing requirements. Abbreviation: WCET See also: response time, execution time, worst case response time
<b>Comment</b>	This definition has been extended by WP COM
<b>Example</b>	--
<b>Reference</b>	--

### 4.320 Worst Case Response Time

<b>Definition</b>	Maximum possible time between receiving a stimulus and delivering an appropriate response or reaction.
<b>Initiator</b>	Software and Architecture
<b>Further Explanations</b>	The worst case response time describes the maximum possible time between a stimulus like e.g. the state change of hardware or software entity and the expected reaction of the system (e.g. response, actuator activation). Typically: worst-case execution-time + infrastructure-overhead + scheduling-policy = worst-case reaction time Synonym: worst case reaction time See also: response time, execution time, worst case execution time
<b>Comment</b>	Worst case reaction time was renamed to worst case response time because response time is the more common terminology. This definition has been extended by WP COM.
<b>Example</b>	--
<b>Reference</b>	--

## Annex 1: Literature

**[Balzert99]**, Balzert, H. “Lehrbuch Grundlagen der Informatik”  
Spektrum Verlag, Heidelberg, 1999

**[DCE-IDL]**, Remote Procedure Call  
<http://www.opengroup.org/onlinepubs/9629399/chap1.htm>

**[EAST-Glossary]**, ITEA Project 00009 EAST-EEA Embedded Electronic Architecture  
“Glossary”, Version 6.1, 2003

**[Hyper Dictionary]**, Hyper Dictionary, 2003  
[www.HyperDictionary.com](http://www.HyperDictionary.com)

**[IEEE 1471]**, Institute of Electrical and Electronics Engineers, Inc. “IEEE 1471-2000:  
IEEE Recommended Practice for Architectural Description for Software- Intensive  
Systems”, 2001

**[IEEE 1517]**, Institute of Electrical and Electronics Engineers, Inc. “IEEE 1517-1999:  
IEEE Standard for Information Technology – Software Life Cycle Processes – Reuse  
Processes”, 2000

**[ISO 12207]**, International Standardization Organization “ISO/IEC 12207 Information  
technology – Software life cycle process”, first edition, Geneva, 1995

**[ISO 2382-1]**, International Standardization Organization “ISO/IEC 2382 Part 1  
Information technology – Vocabulary – Fundamental Terms”, Third Edition, Geneva,  
1993

**[ISO 2382-14]**, International Standardization Organization “ISO/IEC 2382 Part 14  
Information technology – Vocabulary – Reliability, maintainability and availability”,  
Second Edition, Geneva, 1997

**[ISO 2382-15]**, International Standardization Organization “ISO/IEC 2382 Part 15  
Information technology – Vocabulary – Programming Languages”, First Edition,  
Geneva, 1999

**[ISO 2382-20]**, International Standardization Organization “ISO/IEC 2382 Part 20  
Information technology – Vocabulary – System Development”, First Edition, Geneva,  
1990

**[ISO 2382-8]**, International Standardization Organization “ISO/IEC 2382 Part 1  
Information technology – Vocabulary – Security”, Second Edition, Geneva, 1998

**[ISO 26262-6]**, International Standardization Organization “ISO 26262 Part 6  
Road vehicles – Functional safety – Part 6: Product development at the software  
level, Geneva, 2018

**[ISO 61511-1]**, International Standardization Organization “ISO/IEC 61511 Part 1 Information technology – Software life cycle process”, First Edition, Geneva, 1995

**[ISO DIS 26262, Part 1]**, International Standardization Organization “ISO/IEC 26262 Part 1 Road vehicles – Functional safety: Vocabulary”

**[OMG-IDL]**,

[http://www.omg.org/technology/documents/formal/corba\\_2.htm](http://www.omg.org/technology/documents/formal/corba_2.htm)

**[ISO 17356]**, ISO versions of the OSEK standards

**[ISO 7498]**, Information processing systems -- Open Systems Interconnection -- Basic Reference Model

**[ISO/IEC 27000:2018]**, Information technology - Security techniques - Information security management systems – Overview and vocabulary

**[DIN 40041]**, DIN 40041 Ausgabe:1990-12 Zuverlässigkeit; Begriffe  
Deutsche Industrie Norm

**[VDI Lexikon]**, Translation/Adaptation from VDI Lexikon Informatik und Kommunikationstechnik,  
Springer Verlag, Berlin 1999,

**[IEEE.610.12-1990]**, IEEE Standard Glossary of Software Engineering Terminology;  
ISBN 1-55937-067-X, SH13748

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

**[IEEE 1394]** Firewire, see “isochronous stream”  
<http://www.1394ta.org/Technology/Specifications/specifications.htm>

**[ISO 17458]** Road vehicles – FlexRay communication system