Document Title	Modeling Guidelines of Basic Software UML Model
Document Owner	AUTOSAR GbR
Document Responsibility	AUTOSAR GbR
Document Version	1.1.1
Document Status	Final
Part of Release	2.1
Revision	0014

Document Change History			
Date	**Version**	**Changed by**	**Change Description**
24.01.2007	1.1.1	AUTOSAR Administration	• "Advice for users" revised • "Revision Information" added
05.12.2006	1.1.0	AUTOSAR Administration	• Usage of packages clarified • Sequence diagram modelling clarified • Legal disclaimer revised
27.06.2006	1.0.0	AUTOSAR Administration	Initial release

Page left intentionally blank

Disclaimer

Any use of these specifications requires membership within the AUTOSAR Development Partnership or an agreement with the AUTOSAR Development Partnership. The AUTOSAR Development Partnership will not be liable for any use of these specifications.

Following the completion of the development of the AUTOSAR specifications commercial exploitation licenses will be made available to end users by way of written License Agreement only.

Advice to users of AUTOSAR Specification Documents:

AUTOSAR Specification Documents may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).
Any such exemplary items are contained in the Specification Documents for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such Specification Documents, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

Table of Contents

1 Scope of this Document

This Modeling Guideline contains guidelines for the usage of the Enterprise Architect UML Modeling Tool (EA) that is used for the detailed architecture design of the AUTOSAR Basic Software.

Each guideline has its unique identifier starting with the prefix “BSW_UMLGuide” (BSW = Basic Software). For any review annotations, remarks or questions please refer to this unique ID rather than chapter or page numbers!

2 Related Documentation

2.1 Deliverables of AUTOSAR work packages

[1] List of Basic Software Modules
https://svn.autosar.org/repos/10Releases/
AUTOSAR_BasicSoftwareModules.pdf

[2] General Requirements on Basic Software Modules
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SRS_General.pdf

[3] Layered Software Architecture
https://svn.autosar.org/repos/10Releases/
AUTOSAR_LayeredSoftwareArchitecture.pdf

2.2 Related standards and norms

[4] UML 2.0, Unified Modeling Language: Superstructure, Version 2.0, OMG document formal/05-07-04."

3 Terms and abbreviations

Terms	***Definitions***
BSW Module Component	Each BSW module is modeled using one “UML Component” and several “Interface Classes” within the BSW UML model. The “UML Component” represents the internal behavior or C-file(s) of the BSW module. It is called “BSW Module Component”
UML Component	Model element defined by [4].
Interface class	UML 2.0 class with stereotype “interface”.
BSW Module Interface	Each BSW module is modeled using one “UML Component” and several “Interface Classes” within the BSW UML model. The “Interface classes” represent the header files of a specific BSW module. They are called “BSW Module Interfaces”
Tree view	The “project view” window within the Enterprise Architect is called “Tree view”.

4 Requirements on the modeling of the Basic Software

4.1 General

4.1.1 [BSW_UMLGuide_00017] UML 2.0

Initiator:	WP1.1.2
Date:	14.10.2004
Short Description:	UML 2.0 shall be used for modeling the BSW UML model.
Type:	Changed according to bug #7277
Importance:	high
Description:	The UML specification 2.0 shall be used for modeling the AUTOSAR Basic Software with the tool Enterprise Architect (EA).
Rationale:	Not defining new modeling techniques when there are techniques already available and standardized.
Use Case:	Modeling the Basic Software of AUTOSAR.
Dependencies:	--
Conflicts:	--
Supporting Material:	--

4.1.2 [BSW_UMLGuide_00001] Allowed elements

Initiator:	WP1.1.2
Date:	07.10.2004
Short Description:	Allowed elements
Type:	Changed due to bug #10724
Importance:	high
Description:	The following elements of UML are allowed to use within the Basic software overall UML model: - package - class - component - interface - lifeline - fragment - note - node (with stereotype peripheral or cluster) - state (including initial, final, fork/join, choice, exit) - action - decision - activity - boundary Other elements shall not be used.
Rationale:	Restriction of different modeling techniques.
Use Case:	Modeling of the complete communication stack
Dependencies:	[BSW_UMLGuide_00053] Allowed diagrams
Conflicts:	--
Supporting Material:	--

4.1.3 [BSW_UMLGuide_00002] Allowed relationships

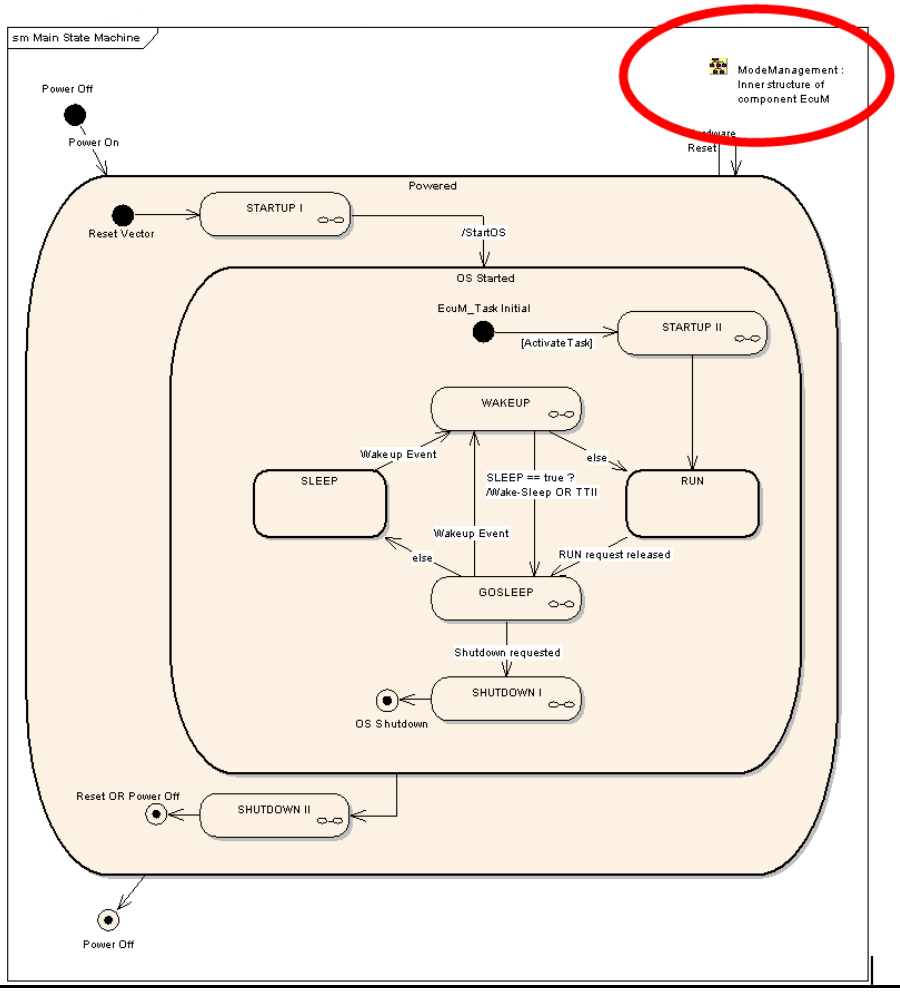
Initiator:	WP1.1.2
Date:	07.10.2004
Short Description:	Allowed relationships
Type:	Changed due to bug #6794
Importance:	high
Description:	The following relationships are allowed to use within the Basic software overall UML model: - realize - nesting - dependency ('import') : for type includes - dependency ('use') : for accessing functional APIs - message - Self-message - Call - transition - activity edge Other relationships shall not be used.
Rationale:	Restriction of different modeling techniques.
Use Case:	Modeling of the complete communication stack
Dependencies:	[BSW_UMLGuide_00053] Allowed diagrams
Conflicts:	--
Supporting Material:	--

4.1.4 [BSW_UMLGuide_00053] Allowed set of diagrams

Initiator:	WP1.1.2
Date:	07.10.2004
Short Description:	Allowed set of diagrams
Type:	Changed according to bug #10724
Importance:	high
Description:	Only a reduced set of diagrams shall be used. Allowed structural diagrams are : - Package diagrams and - Component diagrams. Allowed sequence diagrams are: - Sequence diagrams - Activity diagrams and - State machine diagrams.
Rationale:	Restriction of different modeling techniques.
Use Case:	Modeling of the complete communication stack
Dependencies:	[BSW_UMLGuide_00001] Allowed elements [BSW_UMLGuide_00002] Allowed relationships
Conflicts:	--
Supporting Material:	--

4.1.5 [BSW_UMLGuide_00047] Links between diagrams shall be hyperlinks

Initiator:	4.2.2.1.9 (CZ)

Date:	14.02.2005
Short Description:	Links between diagrams shall be hyperlinks
Type:	Changed (11.05.2005, WP 1.1.2)
Importance:	High
Description:	If the relationship between two diagrams shall be visualized, then the link shall be modeled as a hyperlink.
Rationale:	Easier navigation between diagrams.
Use Case:	
Dependencies:	--
Conflicts:	--
Supporting Material:	--

4.2 Structural Design

4.2.1 [BSW_UMLGuide_00054] Use of Packages

Initiator:	Technical Office
Date:	31.07.2006
Short Description:	Use of Packages
Type:	New
Importance:	High
Description:	Packages may be used in three ways: (1) To group (only) sub-packages, (2) to represent one BSW module, grouping the BSW module component and its interfaces or (3) placed below a package of the second type to group

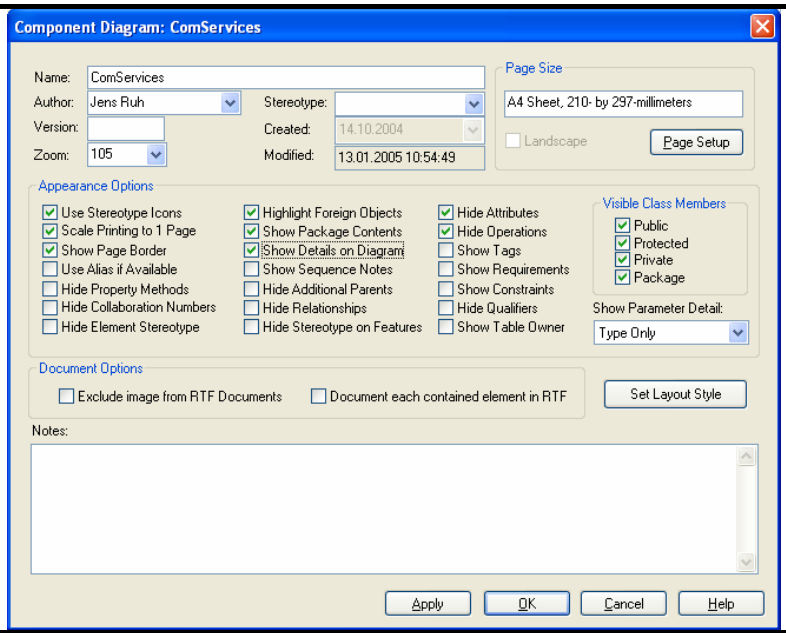
	additional elements detailing a BSW module. Packages of the first or second type shall only be added by the technical office.
Rationale:	Clear structure of the BSW UML model.
Use Case:	Modeling of the complete software architecture
Dependencies:	[BSW_UMLGuide_00003] Diagrams usage [BSW_UMLGuide_00009] Version numbers of software modules [BSW_UMLGuide_00035] Sub elements of BSW modules
Conflicts:	--
Supporting Material:	--

4.2.2 [BSW_UMLGuide_00003] Diagrams usage

Initiator:	WP1.1.2
Date:	07.10.2004
Short Description:	Diagram usage
Type:	Changed according to bug #10724.
Importance:	high
Description:	Each package containing only sub-packages shall at least have one structural “Component” diagram which shows the contents and, if possible, the relationships of the packages which it contains. Each package representing a BSW module shall at least have one structural “Component” diagram which shows at least the "realize" and the "use" relationships of the BSW module component which it contains. The name of this diagram shall be equal to the BSW module component name. This diagram shall be placed below the appropriate diagram within the tree view.
Rationale:	Have for each structural element a diagram showing the elements containing it.

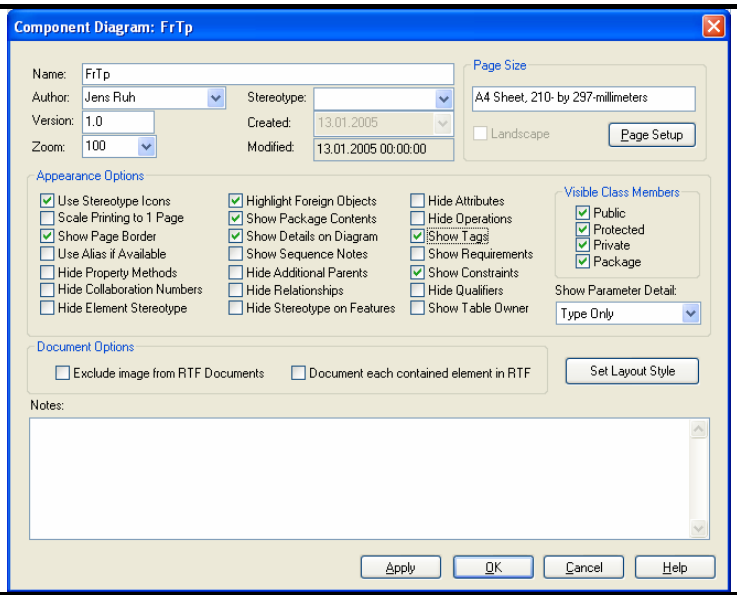
4.2.3 [BSW_UMLGuide_00038] Component diagram appearance options

Initiator:	WP1.1.2
Date:	08.02.2005
Short Description:	Component diagram appearance options
Type:	New
Importance:	high
Description:	In general only the following appearance options of component diagrams shall be set: - Use stereotype icons - yes - Scale printing to 1 page - optional - Show page border - yes - Highlight foreign objects - yes - Show package contents - optional - Show details on diagram - yes - Hide operations - yes - Hide attributes - yes If a user requires more options, he shall ask the technical office for clearance: technical.office@autosar.org.
Rationale:	Harmonization of diagram appearance.

Use Case:	
Dependencies:	--
Conflicts:	--
Supporting Material:	--

4.2.4 [BSW_UMLGuide_00039] Component diagram appearance of BSW module diagrams

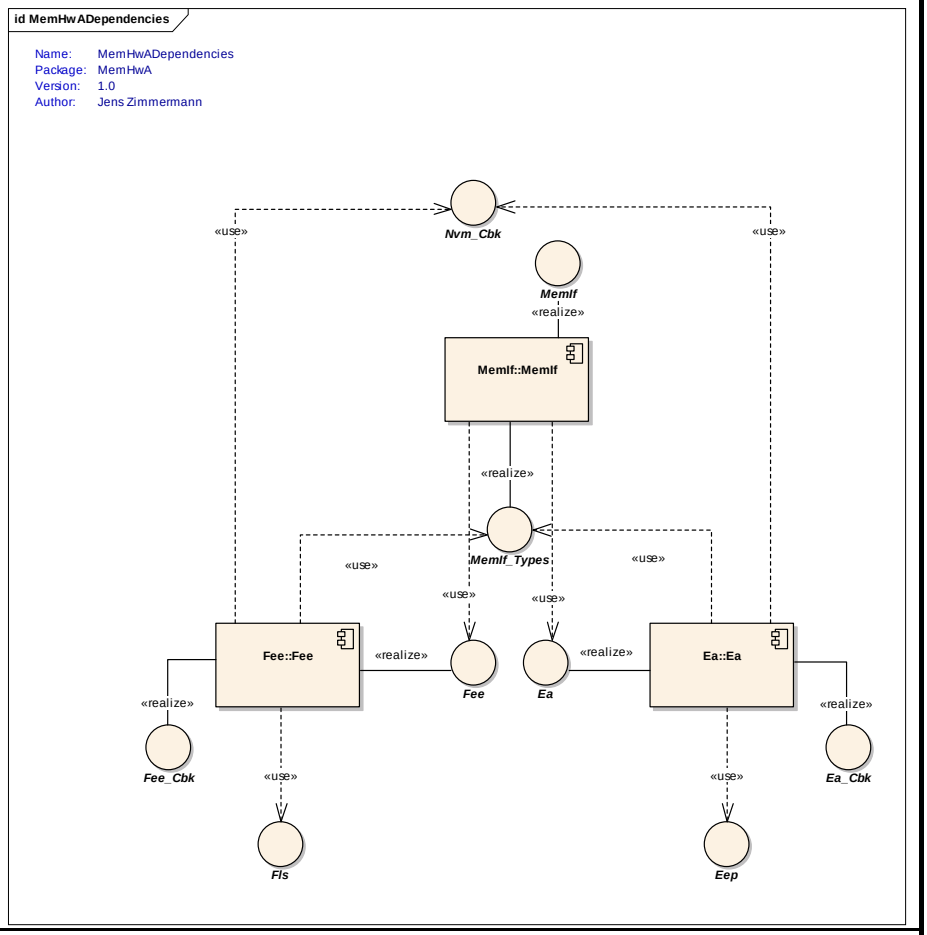
Initiator:	WP1.1.2
Date:	13.01.2005
Short Description:	Component diagram appearance of BSW module diagrams
Type:	new
Importance:	high
Description:	Diagrams placed below the BSW module package shall apply the following changes to the appearance options compared to BSW_UMLGuide_00038: - Hide attributes - No - Hide operations - No - Show constraints - yes - Show Tags - yes
Rationale:	Visualizing all attributes, operations and constraints of a BSW module component.

Use Case:	
Dependencies:	BSW_UMLGuide_00038
Conflicts:	--
Supporting Material:	--

4.2.5 [BSW_UMLGuide_00004] Header File Modeling

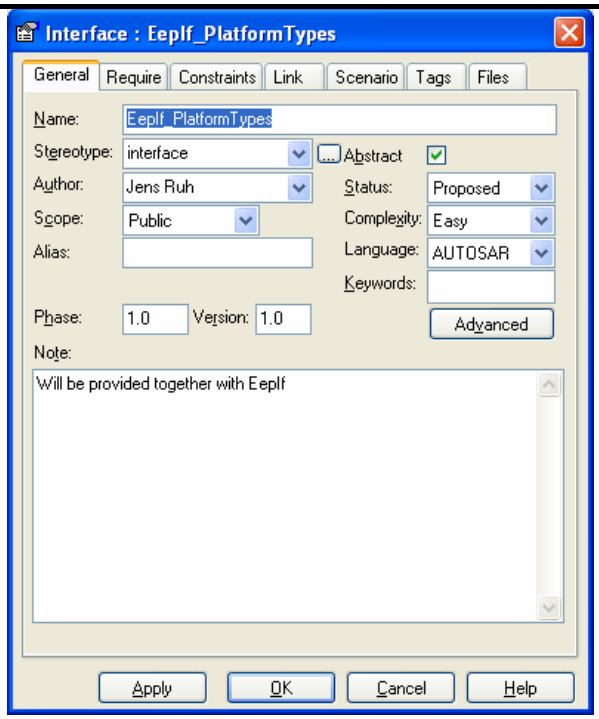
Initiator:	WP1.1.2
Date:	07.10.2004
Short Description:	Header File Modeling
Type:	Changed due to bug #6747 and bug #9820
Importance:	High

Description:	Each header file providing external interfaces of a basic software module shall be modeled as an own interface class (a “BSW Module Interface”). The element representing the basic software module shall have a “realize” relationship to all interface classes specifying its external interfaces.
	If the realize relation is drawn operations/interfaces of the component representing the BSW module shall neither be overwritten nor implemented (select “Cancel” in “Override Operations/Interfaces” dialog).
	
	The names of the interface classes shall follow the following naming convention:
	
	`<module prefix>_<name of the interface as specified within the SWS>`
	
	`<name of the interface as specified within the SWS>`: Partly defined by general SRS (e.g. BSW00370)
	
	If no interface definition is available the default name is:
	`<module prefix>_MissingSWS`
	When the interface specification is available the UML element representing it shall only be RENAMED (not deleted).
	
	If an interface specification is available, but the file structure has not been refined the name of the interface is:
	`<module prefix>`

	Note: UML 2 does not require showing the explicit <<realize>> stereotype. But because the Enterprise architect automatically adds the stereotype this should be done in a harmonized way throughout the model.
Rationale:	Enabling the showing of “include” relationships between modules.
Use Case:	Modeling of the sequences of API calls during a Lin frame transmission 
Dependencies:	BSW00370, BSW00346, BSW00353, BSW00361
Conflicts:	--
Supporting Material:	--

4.2.6 [BSW_UMLGuide_00029] Header File ownership

Initiator:	WP1.1.2
Date:	12.01.2005
Short Description:	Header File ownership
Type:	new
Importance:	high
Description:	Interfaces shall have only one realize relationship to another element. The interfaces are owned by those components realizing them. The ownership of interfaces having no ‘realize’ relationship shall be documented as note in the interface itself.
Rationale:	Enabling the showing of implemented packages.

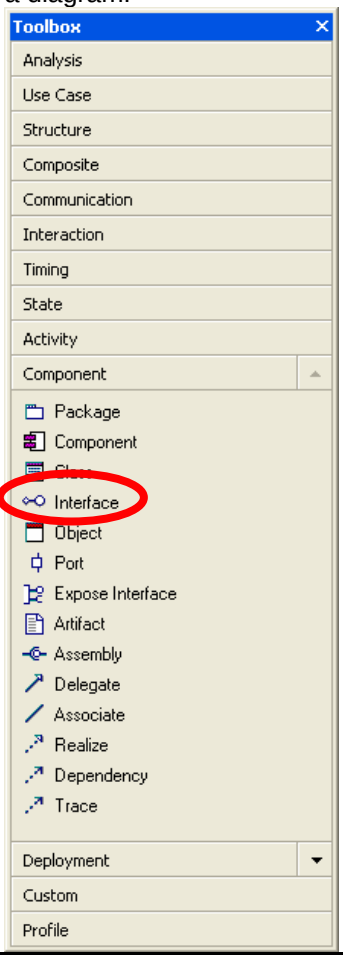
Use Case:	
Dependencies:	--
Conflicts:	--
Supporting Material:	--

4.2.7 [BSW_UMLGuide_00005] Basic Software Module Modeling

Initiator:	WP1.1.2
Date:	12.01.2005
Short Description:	Basic Software Module Modeling
Type:	new
Importance:	high
Description:	Each basic software module source code file(s) shall be modeled as an UML package containing one component (the “BSW Module Component”) and the appropriate interfaces (the “BSW Module Interfaces”). The name of the package and component shall be the Prefix of the basic software module (specified within the basic software list).
Rationale:	Restriction of different modeling techniques.
Use Case:	Modeling of ECU State Manager. Name of this component: EcuM
Dependencies:	--
Conflicts:	--
Supporting Material:	--

4.2.8 [BSW_UMLGuide_00052] Interface creation

Initiator:	WP1.1.2
Date:	07.11.2005
Short Description:	Interface creation
Type:	new

Importance:	high
Description:	Interfaces shall only be created by dragging “Interface” from the toolbox into a diagram. 
Rationale:	There are two ways in EA to create an interface class: (a) Create a regular class and afterwards add the stereotype <<interface>>. (b) Directly drag a new interface into a diagram (e.g. from the toolbox window). Only (b) leads to a real interface in the EA sense: o correct icon in the project view o class and all operations are enforced to be abstract o ...?
Use Case:	Modeling of the complete communication stack
Dependencies:	--
Conflicts:	--
Supporting Material:	--

4.2.9 [BSW_UMLGuide_00006] Interface Modeling

Initiator:	WP1.1.2
Date:	07.10.2004
Short Description:	Interface Modeling
Type:	new

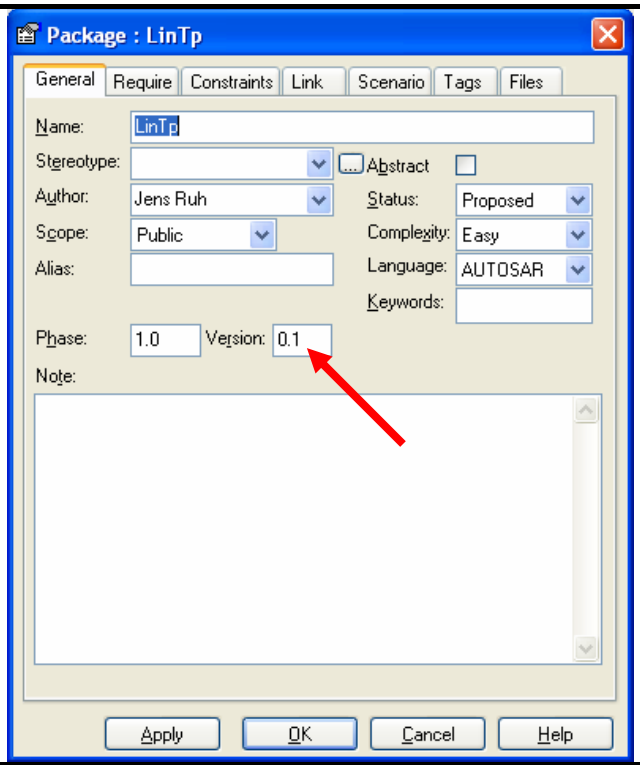
Importance:	high
Description:	Each external interface class shall be modeled in “circle notation” within the “component diagram” of a package containing the basic software module components and/or classes.
Rationale:	Restriction of different modeling techniques.
Use Case:	Modeling of ECU State manager.
Dependencies:	--
Conflicts:	--
Supporting Material:	--

4.2.10 [BSW_UMLGuide_00008] Interface Location

Initiator:	WP1.1.2
Date:	07.10.2004
Short Description:	Interface Location
Type:	Renamed due to bug #6749.
Importance:	high
Description:	Each external interface class of a basic software module shall be placed in the same hierarchy (within the tree view) as the element realizing it.
Rationale:	Model everything in the same style so that it is easier to understand
Use Case:	Modeling of EEP Interfaces:
Dependencies:	BSW_UMLGuide_00006
Conflicts:	--
Supporting Material:	--

4.2.11 [BSW_UMLGuide_00009] Version numbers of software modules

Initiator:	WP1.1.2
Date:	07.10.2004
Short Description:	Version numbers of software modules
Type:	Changed according to bug #10724
Importance:	high
Description:	The version number of each UML element related to only one BSW module shall be equal to the version number of the SWS which it is derived from, i.e. the version numbers of the package representing a BSW module and the contained component, interfaces and diagrams. If no SWS is available the version number 0.0 shall be taken.
Rationale:	Model everything in the same style so that it is easier to understand
Use Case:	Modeling of Fls Interfaces:

	
Dependencies:	--
Conflicts:	--
Supporting Material:	--

4.2.12 [BSW_UMLGuide_00010] Component Definition

Initiator:	WP1.1.2
Date:	08.10.2004
Short Description:	Component Definition
Type:	new
Importance:	high
Description:	Each “BSW Module Component” in the cluster diagram shall be marked as “Composite Element”.
Rationale:	Easier navigation within the model
Use Case:	Look into the details of one specific component.
Dependencies:	--
Conflicts:	--
Supporting Material:	--

4.2.13 [BSW_UMLGuide_00011] Accessing interfaces of other components

Initiator:	WP1.1.2
Date:	08.10.2004
Short Description:	Accessing interfaces of other components
Type:	Changed due to bug #6779.
Importance:	high
Description:	If a basic software module requires the access of another module this relation shall be modeled as a “use” dependency between the component of

	the basic software module requiring the access and the appropriate Interface class of the other basic software module. If the interface to be accessed is not in the same package a link to the interface shall be copied into the appropriate diagram. The link shall be modeled in circle notation. If it is possible that multiple instances of the accessed module exist a Multiplicity (“*”, “1..*” or “0..*”) shall be added to the “Target Role” within the dependency properties of the “Use” relation between the component of the basic software module requiring the access and the appropriate Interface class of the other basic software module.
Rationale:	Restriction of modeling techniques
Use Case:	Eep Interface “uses” interface of Eep driver.
Dependencies:	--
Conflicts:	--
Supporting Material:	--

4.2.14 [BSW_UMLGuide_00055] Use of parameter kind

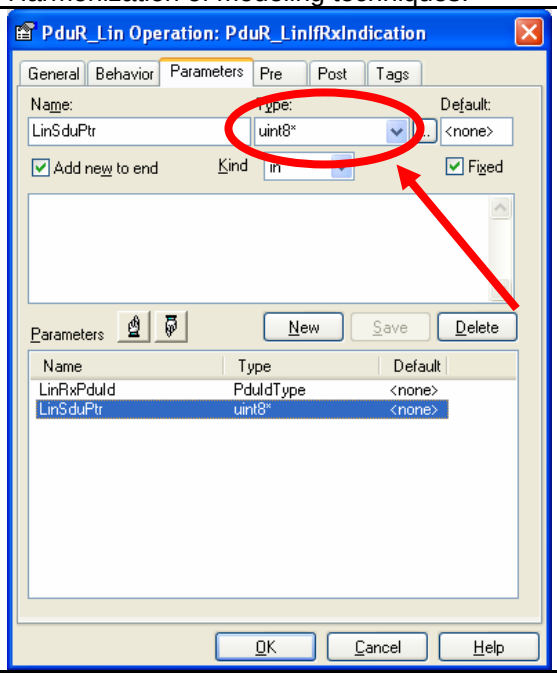
Initiator:	Technical Office
Date:	31.07.2006
Short Description:	Use of parameter kind
Type:	New
Importance:	High
Description:	The drop down list for the 'kind' of function parameters should be set to the appropriate value (in/out, as given in the respective SWS). In the case that EA adds a '*' to the parameter type (for 'out') although it is already a pointer (by a typedef), this should be annotated in the field 'notes' of the respective function.
Rationale:	Use the in/out feature and work around the EA bug.
Use Case:	Modeling of the com stack types
Dependencies:	[BSW_UMLGuide_00037] Definition of pointer types
Conflicts:	--
Supporting Material:	--

4.2.15 [BSW_UMLGuide_00056] Allowed stereotypes for type definitions

Initiator:	Technical Office
Date:	31.07.2006
Short Description:	Allowed stereotypes for type definitions
Type:	New
Importance:	High
Description:	Only the following stereotypes shall be used for the element 'class' in a type declaration: enumeration, struct and type (*), where (*) is a primitve type, a list of primitive types or 'hardware dependent'.
Rationale:	Unified type declarations.
Use Case:	Modeling of the com stack types
Dependencies:	[BSW_UMLGuide_00037] Definition of pointer types [BSW_UMLGuide_00027] Definition of structures
Conflicts:	--
Supporting Material:	Allowed primitive types are defined in SRS General ([GeneralSRS])

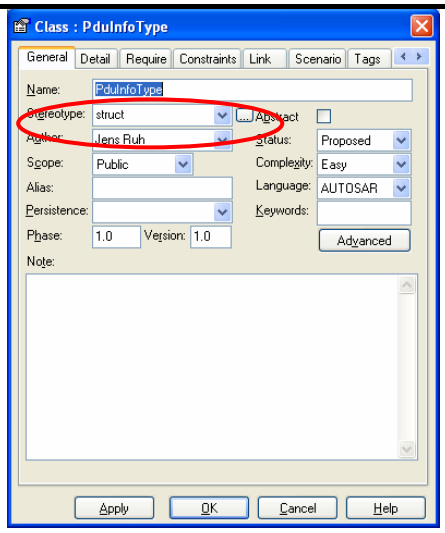
	BSW00304 and maintained in Language Data Types AUTOSAR within the model.

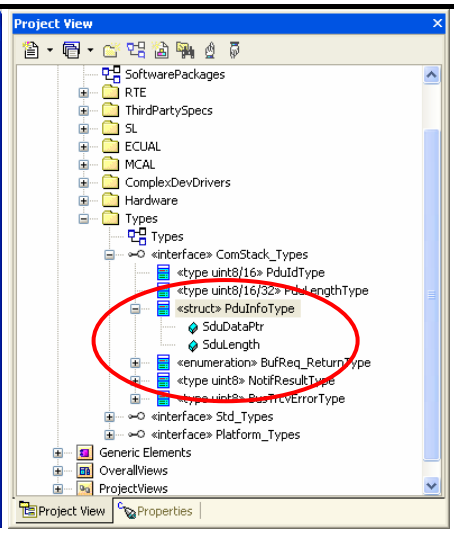
4.2.16 [BSW_UMLGuide_00037] Definition of pointer types

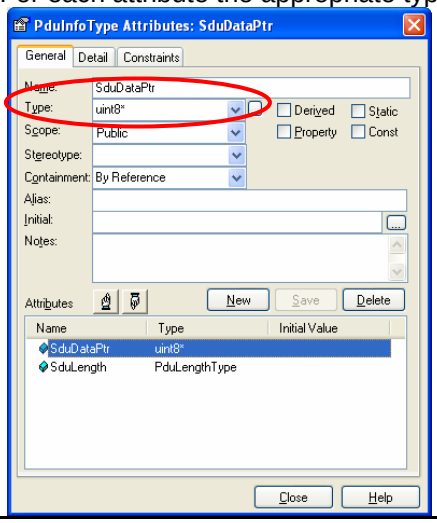
Initiator:	WP1.1.2
Date:	13.01.2005
Short Description:	Definition of pointer types
Type:	new
Importance:	high
Description:	If a parameter or a return value of a module interface represents a pointer the asterix(es) shall be placed directly after the original type.
Rationale:	Readability of the module (specific views will otherwise filter out that information). Harmonization of modeling techniques.
Use Case:	
Dependencies:	--
Conflicts:	--
Supporting Material:	--

4.2.17 [BSW_UMLGuide_00027] Definition of structures

Initiator:	WP1.1.2
Date:	26.11.2004
Short Description:	Definition of structures
Type:	Changed due to bug #9820
Importance:	high
Description:	Each type definition which represents a structure declaration shall be modeled as a class with the stereotype ‘struct’. All possible entries shall be defined as attributes of that class. The attributes shall have the scope “public”.

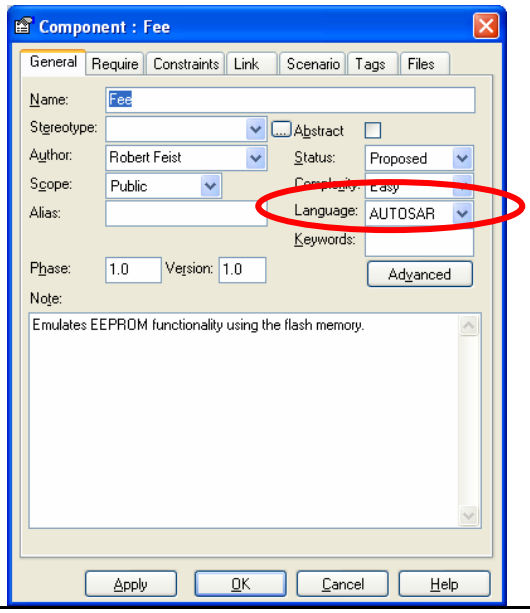
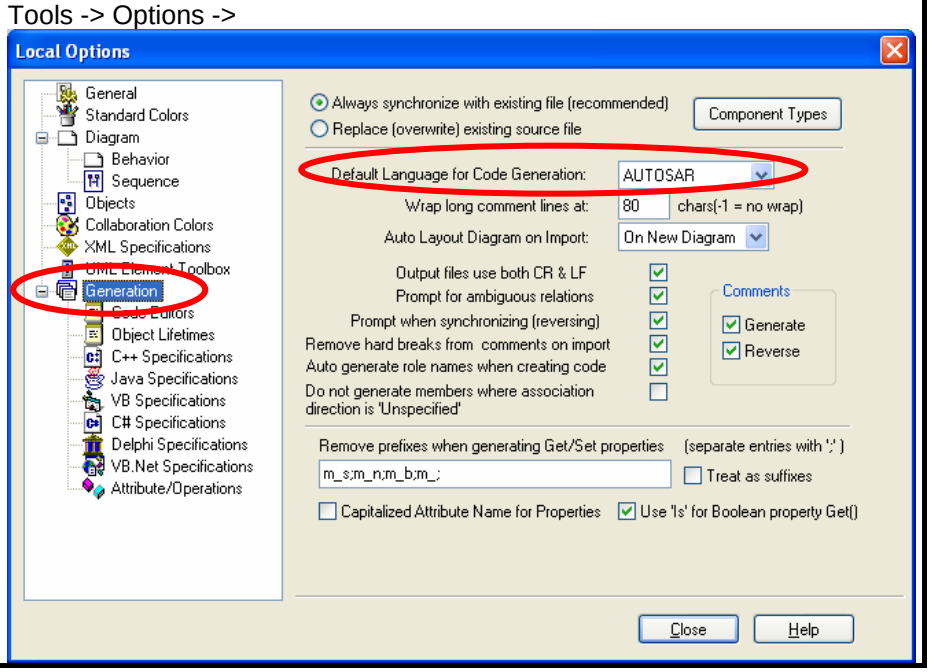
<table>
<tr><td></td><td> 



For each attribute the appropriate type shall be specified:

</td></tr>
<tr><td>Rationale:</td><td>All types have to be defined in the same way, so that there are no inconsistencies within the model.</td></tr>
<tr><td>Use Case:</td><td>The example shown in the description represents the following Std_Types enumeration:
<pre>typedef struct PduInfoType
{
      uint8* SduDataPtr,
      uint16 Length
};</pre></td></tr>
<tr><td>Dependencies:</td><td>Definition of simple types [BSW_UMLGuide_00028]</td></tr>
<tr><td>Conflicts:</td><td>--</td></tr>
<tr><td>Supporting Material:</td><td>--</td></tr>
</table>

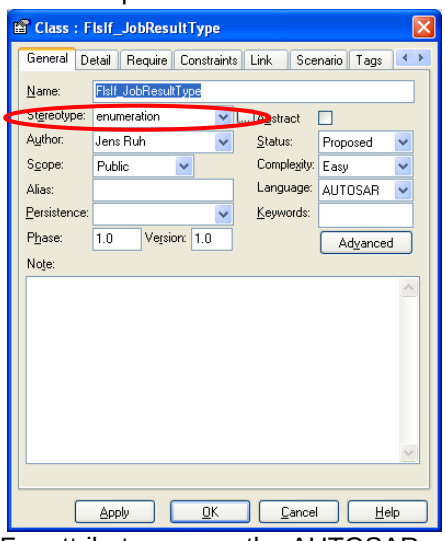
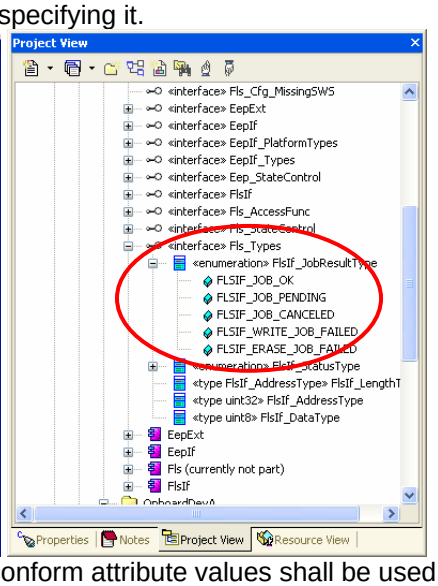
4.2.18 [BSW_UMLGuide_00025] ‘Language’ definition of Components

Initiator:	WP1.1.2
Date:	26.11.2004
Short Description:	‘Language’ definition of Components
Type:	Changed (supporting material added)
Importance:	high
Description:	The ‘Language’ Attribute of at least each Component which represents a

	Software module shall be set to AUTOSAR. 
Rationale:	The AUTOSAR language defines a set of basic types which shall be used within AUTOSAR.
Use Case:	Using AUTOSAR type uint8 as return value.
Dependencies:	--
Conflicts:	--
Supporting Material:	If you set in the options menu AUTOSAR to the default language the correct language will be set automatically: Tools -> Options ->\ 

4.2.19 [BSW_UMLGuide_00026] Definition of enumerations

Initiator:	WP1.1.2
Date:	26.11.2004
Short Description:	Definition of enumerations

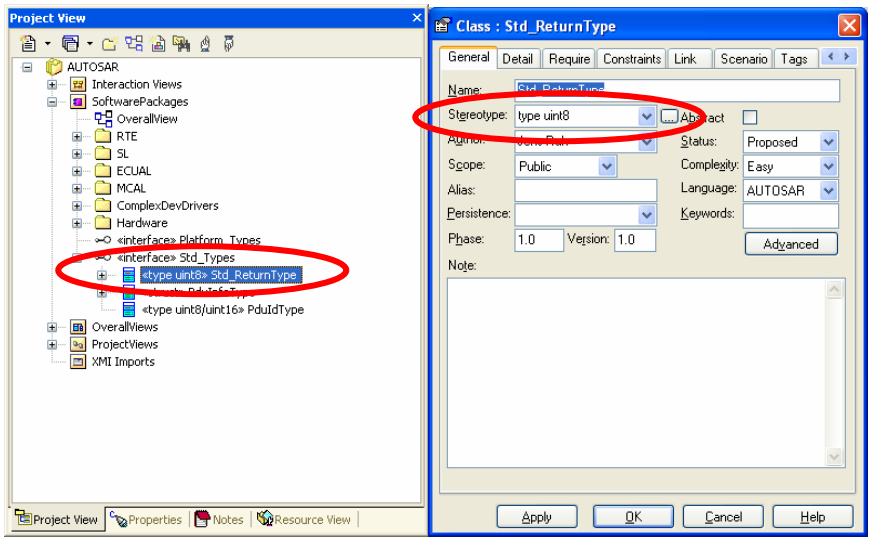
Type:	Changed due to bug #6711
Importance:	high
Description:	Each type definition representing an enumeration shall be modeled as a class with the stereotype ‘enumeration’. All possible entries have to be defined as attributes of this class. The order of the attributes from top to bottom shall represent the order of the enumeration specified. The Attributes shall have no type and the ‘scope’ has to be set to ‘public’. It shall be placed below the interface specifying it.   For attribute names, the AUTOSAR-conform attribute values shall be used. If for this attribute value a code number exist, this code number shall be placed in the "initial” value entry of a type's detailed properties form in EA.
Rationale:	All types have to be defined in the same way, so that there are no inconsistencies within the model.

Use Case:	The example shown in the description represents the following Std_Types enumeration:
	`typedef enum FlsIf_JobResultType`
	`{`
	`  FLSIF_JOB_OK,`
	`  FLSIF_JOB_PENDING,`
	`  …`
	`};`
	
	Need to represent DCM's Neg Resp Codes (NRCs), for which an enum type is defined in SWS DCM v1.1.6, section 8.1.2.8. Place the NRC "DEM_E_xxx" in [attribute]"name" and the 0x value in "initial” value.

Dependencies:	Definition of simple types [BSW_UMLGuide_00028]
Conflicts:	--
Supporting Material:	--

4.2.20 [BSW_UMLGuide_00028] Definition of simple types

Initiator:	WP1.1.2
Date:	26.11.2004
Short Description:	Definition of simple types
Type:	Changed due to bug #6784
Importance:	high
Description:	Each type definition shall be modeled as a separate class. If the type to be

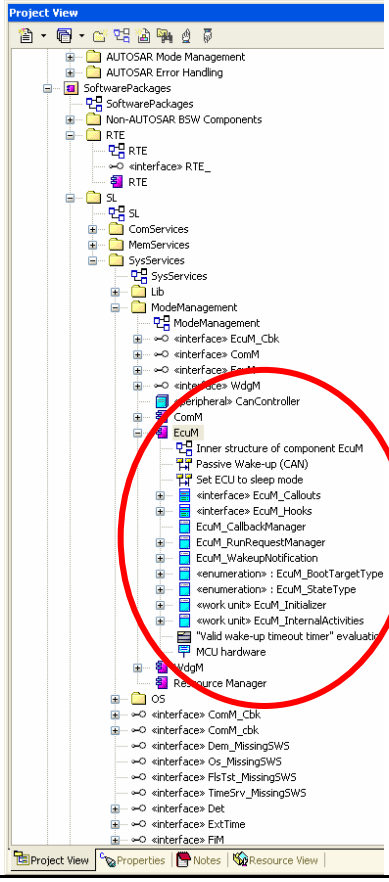
	specified is of an integer type (e.g. uint8) the to be selected stereotype shall be modeled as: Type <integer type>  If the simple type can take only dedicated values, these dedicated values shall be specified as attributes of the class. Each dedicated value shall have a symbolic name. The values (if known) shall be listed as “initial” in the attributes. If the type of the integer type is hardware dependent the to be chosen stereotype is: type hardware dependent
Rationale:	All types have to be defined in the same way, so that there are no inconsistencies within the model.
Use Case:	--
Dependencies:	Definition of structures [BSW_UMLGuide_00028], Definition of enumerations [BSW_UMLGuide_00026]
Conflicts:	--
Supporting Material:	Currently the enterprise architect c header import functionality does not support this kind of type definitions.

4.2.21 [BSW_UMLGuide_00024] Update of the interface of a modified component

Initiator:	WP1.1.2
Date:	22.11.2004
Short Description:	Update of the interface of a modified component
Type:	Changed to apply the new realize handling
Importance:	high
Description:	After each update of the interface definition in a component the 'realize' relationship between the component and the interface class shall not be redrawn. If the relationship is redrawn, operations/interfaces of the component representing the BSW module shall neither be overwritten nor implemented (select “Cancel” in “Override Operations/Interfaces” dialog).
Rationale:	The reason is that otherwise operations might be doubled within the model and the consistency is difficult maintain.
Use Case:	--
Dependencies:	--

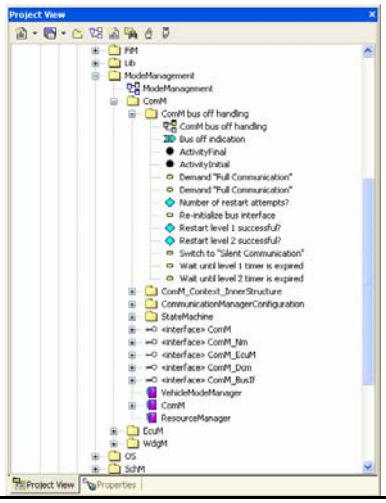
Conflicts:	--
Supporting Material:	--

4.2.22 [BSW_UMLGuide_00034] Refinement of BSW modules

Initiator:	WP1.1.2
Date:	08.02.2005
Short Description:	Refinement of BSW modules
Type:	Changed according to bug #6787
Importance:	high
Description:	Each BSW module should be refined by a substructure placed (in the “Tree view”) below the “BSW Module Component” of the module.
Rationale:	In general each BSW module is defined by one “BSW Module Component” and several “BSW Module Interfaces”. For a detailed specification of a BSW module this is not sufficient. A BSW in general consists out of several small parts. These small parts should be modeled within the UML model.
Use Case:	Refinement of EcuM. 
Dependencies:	--
Conflicts:	--
Supporting Material:	--

4.2.23 [BSW_UMLGuide_00035] Sub elements of BSW modules

Initiator:	WP1.1.2
Date:	13.01.2005

Short Description:	Sub elements of BSW modules
Type:	Changed according to bug #10724
Importance:	high
Description:	The internal behaviour of BSW modules may be modeled in two ways: (1) A package may be added to the package representing the BSW module or (2) elements can be placed below the BSW module component. In case of optional functionality or scaled functionality the respective dependency in the respective diagram shall be commented by selecting 'Attach Note or Constraint'.
Rationale:	The Sub elements of a component could be of type: - separation of concerns / grouping of functionality - optionally / scalability
Use Case:	Refinement of ComM. 
Dependencies:	--
Conflicts:	--
Supporting Material:	--

4.3 Behavioral Design

4.3.1 General

4.3.1.1 [BSW_UMLGuide_00030] Usage of Sequence Diagrams

Initiator:	WP1.1.2
Date:	13.01.2005
Short Description:	Usage of Sequence Diagrams
Type:	new
Importance:	high
Description:	Only sequence diagrams shall be used for modeling interactions of different modules.
Rationale:	Restriction of different modeling techniques.
Use Case:	Modeling of the sequences of API calls during a LIN frame transmission.
Dependencies:	BSW_UMLGuide_00007
Conflicts:	--
Supporting Material:	--

4.3.1.2 [BSW_UMLGuide_00031] Usage of State Machine Diagrams

Initiator:	WP1.1.2
Date:	13.01.2005
Short Description:	Usage of State Machine Diagrams
Type:	new
Importance:	high
Description:	Only state machine diagrams shall be used for modeling state dependencies within and in between elements.
Rationale:	Restriction of different modeling techniques.
Use Case:	Modeling of ECU Manager state changes.
Dependencies:	BSW_UMLGuide_00007
Conflicts:	--
Supporting Material:	--

4.3.2 Sequence Diagrams

4.3.2.1 [BSW_UMLGuide_00012] Location of Sequence Diagrams

Initiator:	WP1.1.2
Date:	13.01.2005
Short Description:	Location of Sequence Diagrams
Type:	Renamed due to bug #6749.
Importance:	High
Description:	All sequence diagrams have to be placed within the “Interaction View Package”
Rationale:	Definition of similar model structures
Use Case:	Modeling of the AUTOSAR COM stack
Dependencies:	--
Conflicts:	--
Supporting Material:	--

4.3.2.2 [BSW_UMLGuide_00020] Packages to contain sequence diagrams

Initiator:	WP1.1.2
Date:	25.01.2005
Short Description:	Packages to contain sequence diagrams
Type:	Changed according to bug #6790
Importance:	High
Description:	A new sequence diagram shall be put into an appropriate package. If no such package is available, it shall be requested from the technical office: technical.office@autosar.org.
Rationale:	Guarantee of readability and correct placement of the sequence tree.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	--

4.3.2.3 [BSW_UMLGuide_00021] Commenting of Sequence Diagrams

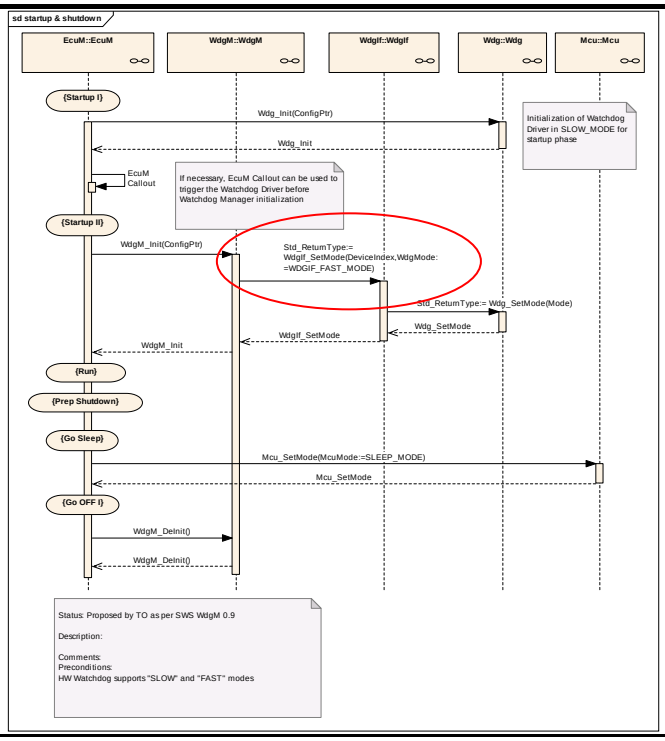
Initiator:	WP1.1.2
Date:	19.10.2004
Short Description:	Commenting of Sequence Diagrams
Type:	New
Importance:	High
Description:	Each Sequence diagram shall have a comment placed as 'note' within the diagram that contains the following items: • Status (open – proposed – approved – conflict – rejected) • Description • Comment If a sequence diagram is rejected or on conflict, the reason shall be described within the comment.
Rationale:	Give other people the chance to understand. Traceability

Use Case:	`Status: approved`
	
	`Description: A CAN frame is received and indicated to the upper layer in interrupt context.`
	
	`Comment: -none-`

Dependencies:	--
Conflicts:	--
Supporting Material:	--

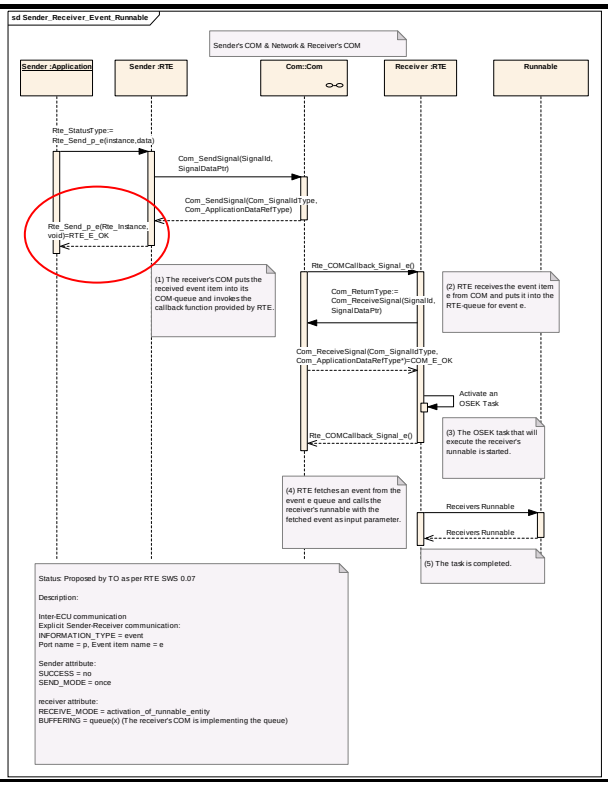
4.3.3 [BSW_UMLGuide_00057] Parameter values in sequence diagrams

Initiator:	Technical Office
Date:	31.07.2006
Short Description:	Parameter values in sequence diagrams
Type:	New
Importance:	High
Description:	If a function is called with a fixed value for one or more of its parameters in a sequence diagram, then this should be modeled by writing 'ParName:=value' in the field 'Parameters' of the respective message.
Rationale:	Unified message modeling.

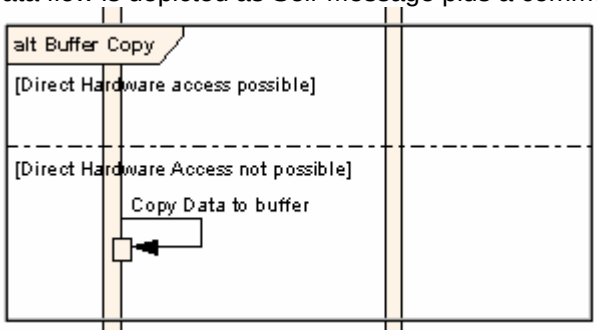
Use Case:	
Dependencies:	[BSW_UMLGuide_00019] Labeling returns
Conflicts:	--
Supporting Material:	--

4.3.3.1 [BSW_UMLGuide_00058] Return values in sequence diagrams

Initiator:	Technical Office
Date:	31.07.2006
Short Description:	Return values in sequence diagrams
Type:	New
Importance:	High
Description:	If the return of a function should be shown to give a specific value, then this should be modeled by writing 'FuncName=value' in the field 'Message' of the respective return-message.
Rationale:	Unified message modeling.

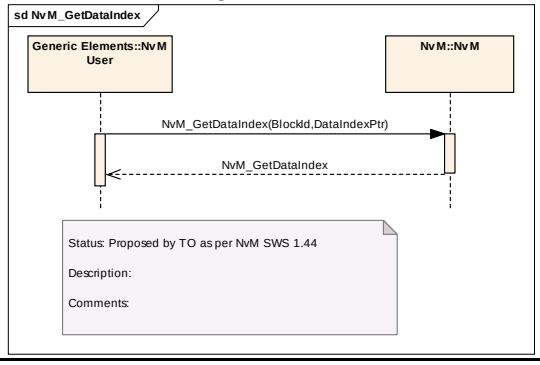
Use Case:	
Dependencies:	[BSW_UMLGuide_00019] Labeling returns
Conflicts:	--
Supporting Material:	--

4.3.3.2 [BSW_UMLGuide_00018] Modeling of data copying

Initiator:	WP1.1.2
Date:	18.10.2004
Short Description:	Modeling of data copying
Type:	new
Importance:	high
Description:	Within sequence diagrams, the following scheme shall be used for modeling data copied/stored/…: Data flow is depicted as Self-Message plus a comment field  In the message text is documented: • What data is copied • From source • To target
Rationale:	Definition of uniform data exchange modeling

Use Case:	Modeling of data exchange between buffers of different layers
Dependencies:	--
Conflicts:	--
Supporting Material:	--

4.3.3.3 [BSW_UMLGuide_00019] Labeling returns

Initiator:	WP1.1.2
Date:	07.10.2004
Short Description:	Labeling returns
Type:	Changed according to bug #10724
Importance:	High
Description:	Returns shall be labeled. The label shall be the name of the function it belongs to without parameter names or -types. Returns shall be marked as “Is Return”. It is announced that in future return labels can be automatically be selected within the Enterprise architect. If this is possible no hand-made adaptations shall be done after selection.
Rationale:	Allow easy identification to which function a return belongs
Use Case:	A UML message has the name “Transmit CAN PDU”. The return message has the same name. 
Dependencies:	--
Conflicts:	--
Supporting Material:	--

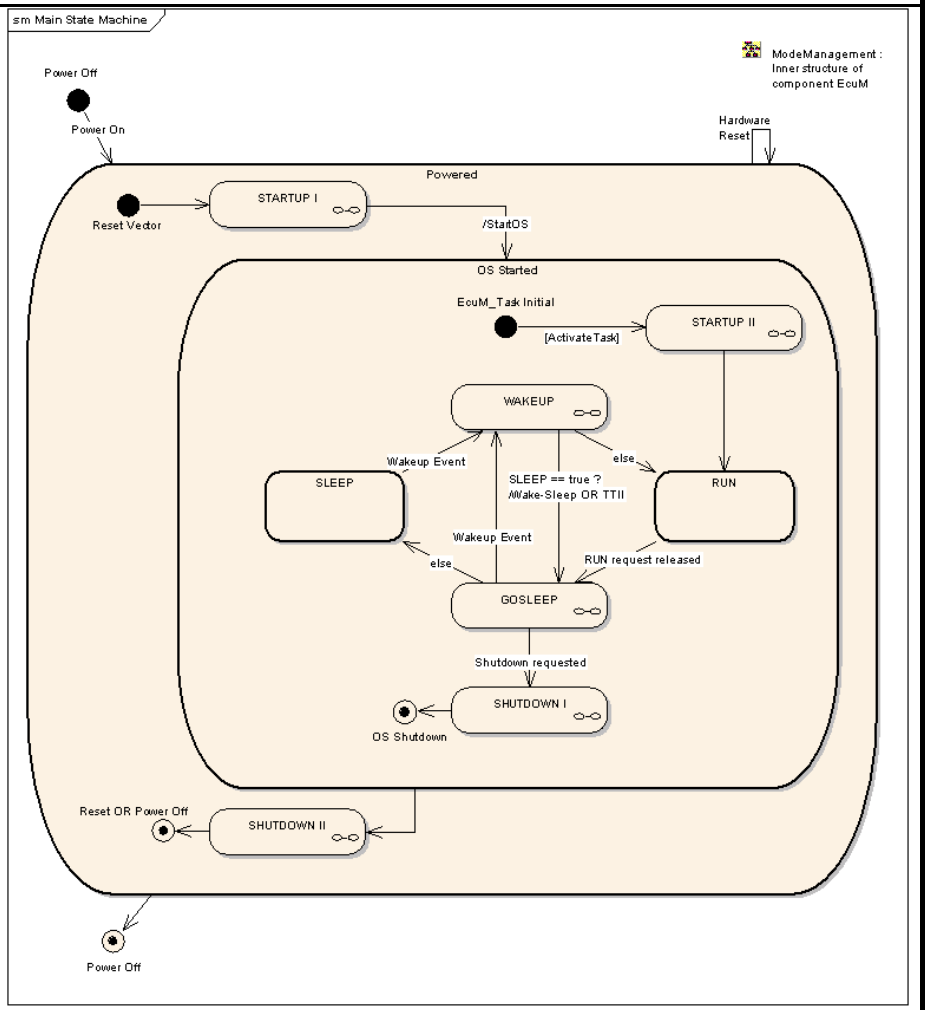
4.3.3.4 [BSW_UMLGuide_00036] Linking sequence diagrams

Initiator:	WP1.1.2
Date:	13.01.2005
Short Description:	Linking sequence diagrams
Type:	new
Importance:	high
Description:	Each package within the ‘Interaction Views’ package shall contain dedicated diagrams containing descriptions and links to sub diagrams. These diagrams will be generated by the model owner. The author of a sequence diagram shall therefore provide a short description of contents of generated packages and diagrams to the model owner.
Rationale:	Readability of overall module
Use Case:	--

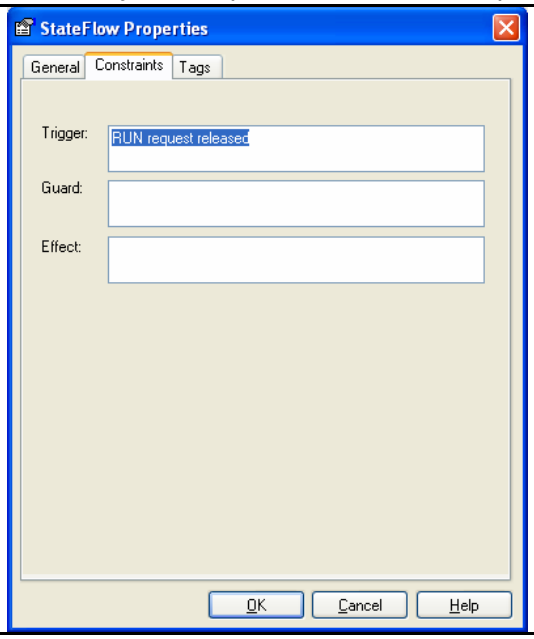
Dependencies:	--
Conflicts:	--
Supporting Material:	--

4.3.4 State Machine Diagrams

4.3.4.1 [BSW_UMLGuide_00041] States shall have thick lines

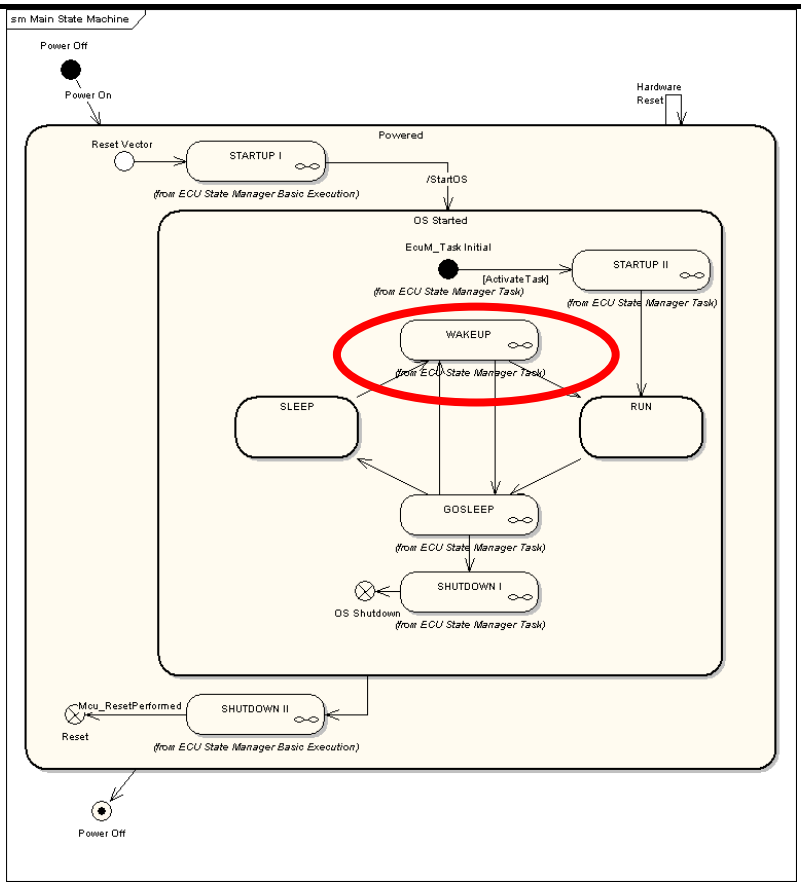
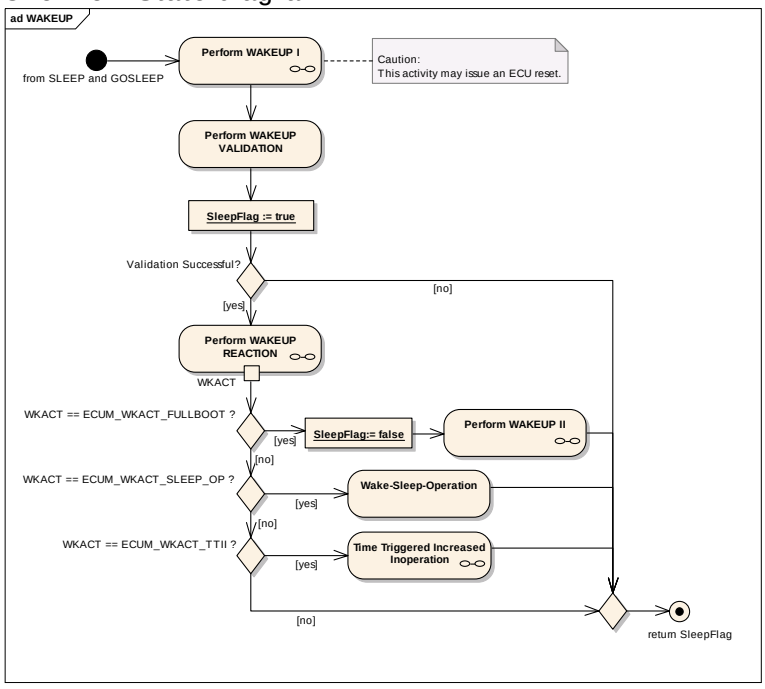
Initiator:	4.2.2.1.9 (CZ)
Date:	14.02.2005
Short Description:	States shall have thick lines
Type:	new
Importance:	high
Description:	The state boxes shall have an outline of 2 points
Rationale:	Allow easier distinction between states and activities
Use Case:	 (to see what is meant please zoom to 200%)
Dependencies:	BSW_UMLGuide_00048
Conflicts:	--
Supporting Material:	--

4.3.4.2 [BSW_UMLGuide_00042] A trigger condition shall be defined for each transition

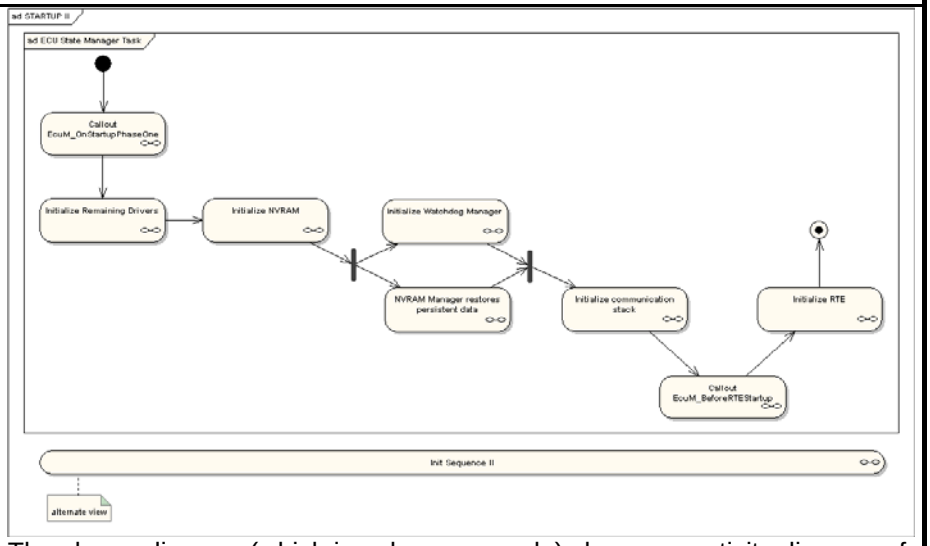
Initiator:	4.2.2.1.9 (CZ)
Date:	14.02.2005
Short Description:	A trigger condition shall be defined for each transition
Type:	Changed due to bug #6794
Importance:	High
Description:	In each transition between two states the trigger of the transition (the condition to make a transition) shall be defined in the 'Trigger' field of the 'State Flow Properties'.
Rationale:	Necessary for complete behavioral description
Use Case:	
Dependencies:	--
Conflicts:	--
Supporting Material:	--

4.3.4.3 [BSW_UMLGuide_00043] Transitions may be modeled with sub-activities

Initiator:	4.2.2.1.9 (CZ)
Date:	14.02.2005
Short Description:	Transitions may be modeled with sub-activities
Type:	Changed due to bug #6794
Importance:	High
Description:	To reduce complexity of diagrams Activities may be modeled in a hierarchical way by using sub-activities.
Rationale:	In some cases complex activities could trigger a transition. In these cases the diagrams become rather complex. These sub-activities may be visualized as sub activities to reduce complexity in one diagram.

Use Case:	 Overview State diagram.  Refined activity ‘WAKEUP’.
Dependencies:	--
Conflicts:	--
Supporting Material:	--

4.3.4.4 [BSW_UMLGuide_00046] Links to parent diagrams shall be drawn as hyperlink diagram references

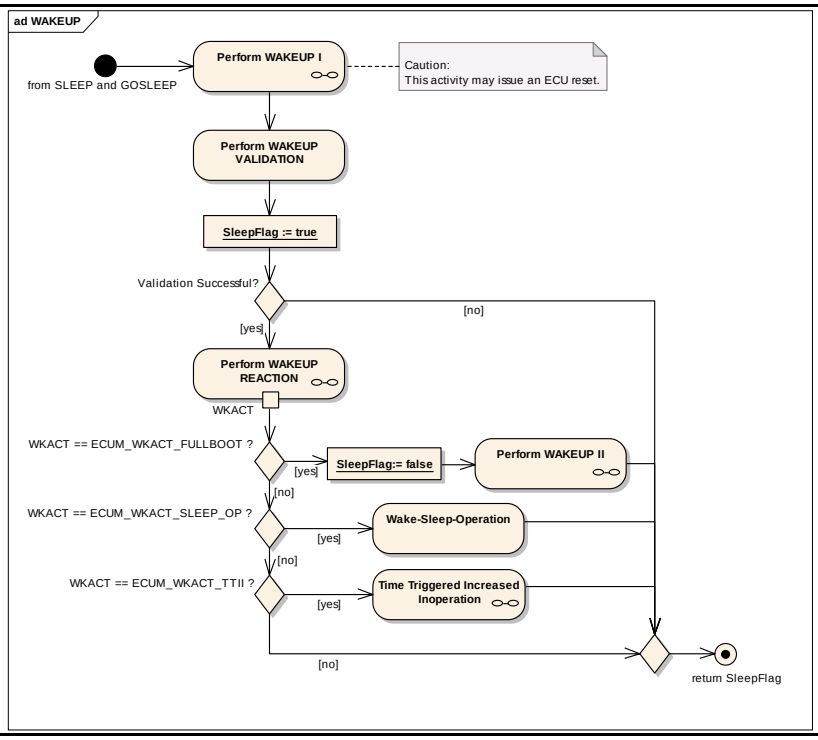
Initiator:	4.2.2.1.9 (CZ)
Date:	14.02.2005
Short Description:	Links to parent diagrams shall be drawn as diagram references
Type:	Changed due to bug #7379
Importance:	medium
Description:	 The shown diagram (which is only an example) shows an activity diagram of a sub-activity of a larger system. The frame around the diagram indicates the link to the parent diagram. Applies to state and activity diagrams similarly.
Rationale:	Simple forward/backward navigation
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	Diagram references do not work in the HTML-Report!

4.3.5 Activity Diagrams

4.3.5.1 [BSW_UMLGuide_00048] Activities shall have thin lines

Initiator:	4.2.2.1.9 (CZ)
Date:	14.02.2005
Short Description:	Activities shall have thin lines
Type:	new
Importance:	high
Description:	The activity boxes shall have an outline of 1 point
Rationale:	Allow easier distinction between states and activities
Use Case:	--
Dependencies:	BSW_UMLGuide_00041
Conflicts:	--
Supporting Material:	--

4.3.5.2 [BSW_UMLGuide_00049] Conditions to be defined for each branch

Initiator:	4.2.2.1.9 (CZ)
Date:	14.02.2005
Short Description:	Conditions to be defined for each branch
Type:	Changed according to bug #6795
Importance:	High
Description:	If a flow branches because of a condition, the 'Decision' element shall be used. All outgoing control flows must have set a 'Guard' constraint in 'Control Flow Properties'. If different flows shall be merged, also the "Decision" element shall be used.
Rationale:	--
Use Case:	
Dependencies:	--
Conflicts:	--
Supporting Material:	Similar rules apply for forks/joins. The idea is to have the control flow clearly defined. Object flow follows slightly different rules. Such as strict rules make UML clumsy for use with object flow elements.

4.3.5.3 [BSW_UMLGuide_00050] Activities to be re-used in sequence diagrams should also be drawn as sequence diagrams

Initiator:	4.2.2.1.9 (CZ)
Date:	14.02.2005
Short Description:	Activities to be re-used in sequence diagrams must also be drawn as sequence diagrams
Type:	new
Importance:	medium
Description:	If an activity is to be referenced within a sequence diagram the drawing messages will not look nice. Therefore this type of activities should also be modeled as sequence diagrams.
Rationale:	Nice modeling.

Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	--

4.4 Model synchronization

All guidelines related to the design master are not intended for 'normal' users, because the master of the BSW UML model will not be distributed. This chapter will be refined as soon as the process of collaborative work on the BSW UML model has been agreed.

4.4.1 [BSW_UMLGuide_00013] Creating a Design Master

Initiator:	WP1.1.2
Date:	14.10.2004
Short Description:	Creating a Design Master
Type:	Changed according to bug #6796
Importance:	High
Description:	Convert the base project to a Design Master using the Make Design Master option in the Tools (Manage .EAP File submenu.
Rationale:	--
Use Case:	This shall be done once for the project by AUTOSAR Technical Office only (already done – no more actions required).
Dependencies:	--
Conflicts:	--
Supporting Material:	--

4.4.2 [BSW_UMLGuide_00023] Design Master naming convention

Initiator:	WP1.1.2
Date:	14.10.2004
Short Description:	Design Master naming convention
Type:	New
Importance:	High
Description:	The file name of the Design Master shall have the following naming: Master_AR_BasicSWArchitecture.eap and to be managed by a version management tool.
Rationale:	--
Use Case:	Master_AR_BasicSWArchitecture.eap
Dependencies:	--
Conflicts:	--
Supporting Material:	--

4.4.3 [BSW_UMLGuide_00014] Creating replicas from the Design Master

Initiator:	WP1.1.2

Date:	14.10.2004
Short Description:	Creating replicas from the Design Master
Type:	new
Importance:	high
Description:	Create replicas from the design master using the Create New Replica option in the Tools (Manage .EAP File submenu. Further work must be done on the replica.
Rationale:	--
Use Case:	To create your own local working model. This has to be done only once.
Dependencies:	--
Conflicts:	--
Supporting Material:	--

4.4.4 [BSW_UMLGuide_00022] Replica naming convention

Initiator:	WP1.1.2
Date:	14.10.2004
Short Description:	Replica naming convention
Type:	new
Importance:	high
Description:	Replicas of the design master shall have the following naming convention: `Replica_AR_BasicSWArchitecture_<Version number>_<Author short name>.eap`
Rationale:	--
Use Case:	Replica_AR_BasicSWArchitecture_V24.1_CMA.eap
Dependencies:	--
Conflicts:	--
Supporting Material:	--

5 Administrative Info

Last used Requirements ID is [BSW_UMLGuide_00058]