MSR Sub-project MEDOC

Structure Principles

Scope: Behavior / Software

MSR Working Group MEDOC, Dipl.-Ing. Bernhard Weichel

06/09/2002 13:49:11 msrsw-sp-en.xml

Abstract

This document describes the principles for the structure of the MSR development documentation MEDOC within the scope of the software for control units. Notes are given on the the basic development process. The structure of the *MSRSW.DTD* is described in detail.

Table of Contents

Introduction

Companies

MSR Working Group MEDOC [MSR-MEDOC]

Name Roles	Departement	Address	Contact
Dipl.-Ing.(FH) Uwe B-less			
Dipl.-Inform. Helmut Gengenbach			
Dipl. Ing. Eckard Jakobi			
Dipl.-Ing. Herbert K-lein			
Dipl. Ing. Oliver Mar-cks			
Dipl.-Inform. Peter Rauleder			
Dipl.-Ing. Martin Trin-schek			
Dipl.-Ing. Bernhard Weichel			

Version Information

Document Part	Editor			
	Company	Version	State	Remarks
23.7.98	Dipl.-Ing. Bernhard Weichel			
For details refer to nr. 1, Page 127	MSR-MEDOC	2	cd	

1 Introduction

This document describes the principles for the structure of the MSR development documentation MEDOC within the scope of the software for control units.

It is pointed out here that MSR does not conduct any standardization of the systems or their features that are described with MEDOC. MEDOC supports the use of (inter)national standards and in-house norms, as well as non-standardized norms, for the description of systems of data relevant to the documentation of development processes.

2 Allgemeine Projektdaten

1 Notes on the software development process

1.1 Cooperation between manufacturer and supplier

The scenarios in Figure 1 Manufacturer-supplier scenarios p. 10 depicting the cooperation between manufacturer and supplier for the development of software have been taken into consideration in the definition for *MSRSW.DTD*.

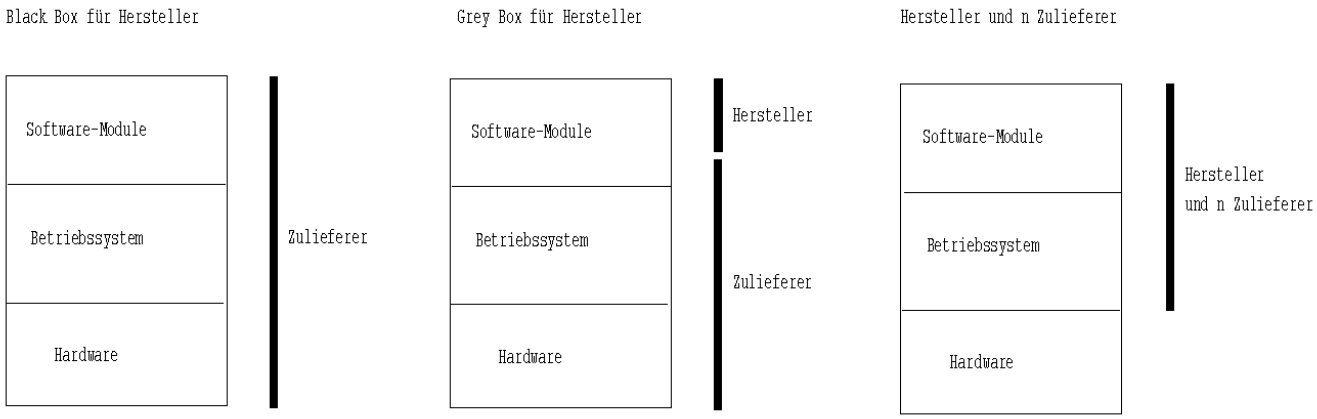

Figure 1: Manufacturer-supplier scenarios

Black Box All developments are performed by the supplier. The supplier prepares the complete documentation.

Grey Box Manufacturer and supplier each have a defined set of control unit functions for which they are responsible. The respective development results are summarized.

White Box Manufacturer and supplier work together on control unit functions. There is an ongoing exchange taking place, of individual components as well (e.g. definitions for variables etc.).

1.2 Phase model for software generation

A phase model is outlined in this chapter for the generation of software. This constitutes the basis for a common understanding amongst all MSR participants. The phases can be repeated in loops.

Those results to be exchanged between manufacturer and supplier are identified on the basis of this phase model.

1.2.1 System analysis

The system is described at a logic level in the system analysis. The functional logic model given can be simulated (e.g. in *Matrix-X* or *Ascet*). The same structures are used for the system analysis and the system design (and hence the same DTD as well).

Functional analysis

The functional analysis includes a description of the context, the functions (hierarchical, incl. diagnostic functions, safety functions, etc.) and the information flows (informal). A formal definition, an informal description, application notes and customer-servicing notes are documented for each function (compare Figure 18 Function variants p. 33).

Description of time dependencies

The time dependencies can be modeled using RT analysis, status diagrams, Petri networks, etc.

1.2.2 Analysis review

The contents for the analysis review can be defined according to company-internal guidelines.

Review protocol

The protocol can include establishing the completeness, the absence of contradictions, the testability, etc. of the required functions.[1]

1.2.3 System design

For the system design, the physical functional model is derived from the functional logic model.

Physical functional model

The functional model includes the description of the context, the implemented functions (hierarchical, incl. diagnostic functions, safety functions, etc.) and the information flows (informal).

An informal description, application notes and customer-servicing notes are documented for each function (compare Figure 18 Function variants p. 33).

Description of time dependencies

In the system design, functions for example, are compiled into tasks and assigned to computing levels (scheduling tables).

With the exception of variables, *MSRSW.DTD* 1.1.0 does not possess any explicit means for describing time dependencies. These therefore have to be described informally for the functional description (**<sw-function-desc>**) or in **<add-info>**.

Partitioning in hardware und software

Partitioning is carried out taking resources and other secondary conditions (e.g. costs, etc.) into consideration.

Software architecture

The software architecture can be described in the *MSRSW.DTD* in that a **<sw-function>** is applied with **<sw-function-class>** = *architecture*.

Hardware architecture

Described in *MSRSYS.DTD*.

Test specification

Described in *MSRSYS.DTD*.

1.2.4 Design review

The contents for the design review can be defined according to company-internal guidelines.

Review protocol

The protocol can include establishing the completeness, the absence of contradictions, the testability, etc. of the required functions. *MSRREP.DTD* can be used for this.

1

1.2.5 Coding

Function modules

The functions defined in the system design can be implemented within the scope of the coding. This implementation process can also be carried out by code generators as required.

Details of the data dictionary are developed in this phase (**<sw-data-dictionaries>**), such as for instance parameter structures (**<sw-params>**), variables **<sw-variables>** , conversion formulae (**<sw-compu-methods>**). Source texts, complier command files etc., are established as well.

1.2.6 Module test

Formal and functional test of the coded Definition Software module p. 94software module under laboratory conditions.

Test report

The test report describes the environment, the application as well as the results of the module test. *MSRREP.DTD* can be used for this.

1.2.7 Integration

Individual modules linked together in this phase. The integration is carried out in the test and/or target environment.

Data status

The overall result of integration is the Definition Program status p. 94program status.

1.2.8 Application (calibration)

The control device is adapted to a specific vehicle or a specific engine in the application. This is realized by specifying the pertinent parameter contents.

Data status

The overall result of the application is the Definition Data status p. 93data status.

1.2.9 System test

MSRREP.DTD can be used for preparation of the test report.

1.3 Language levels

A differentiation can be made for the software functions in control units between the different language levels (refer to Figure 2 Language levels p. 13). There is in particular, a standardization level where the software engineering standards for documentation, specifications for variables and parameters, C-headers, the C-source and for operating system instructions are to be found.

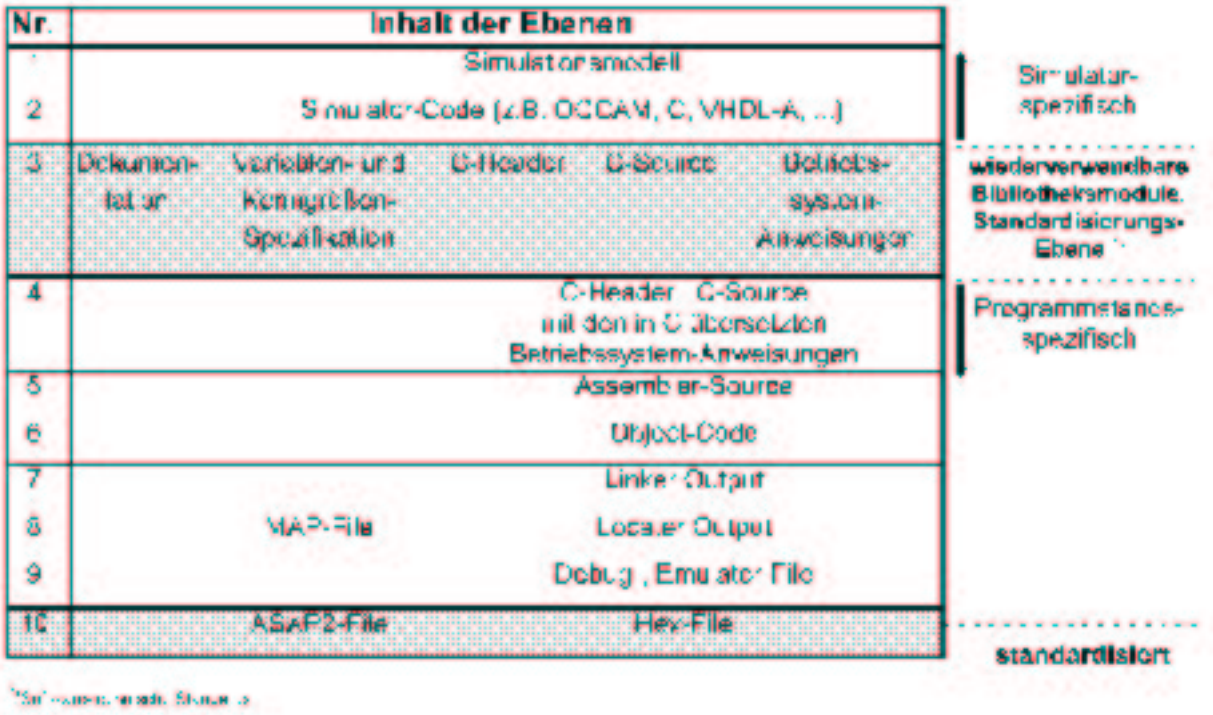

Figure 2: Language levels

The standardization at this level is practiced in many standardization projects, such as MSR-AG MEDOC, MSR-AG Code Generators and OSEK (refer to Figure 3 Standardization projects p. 13).

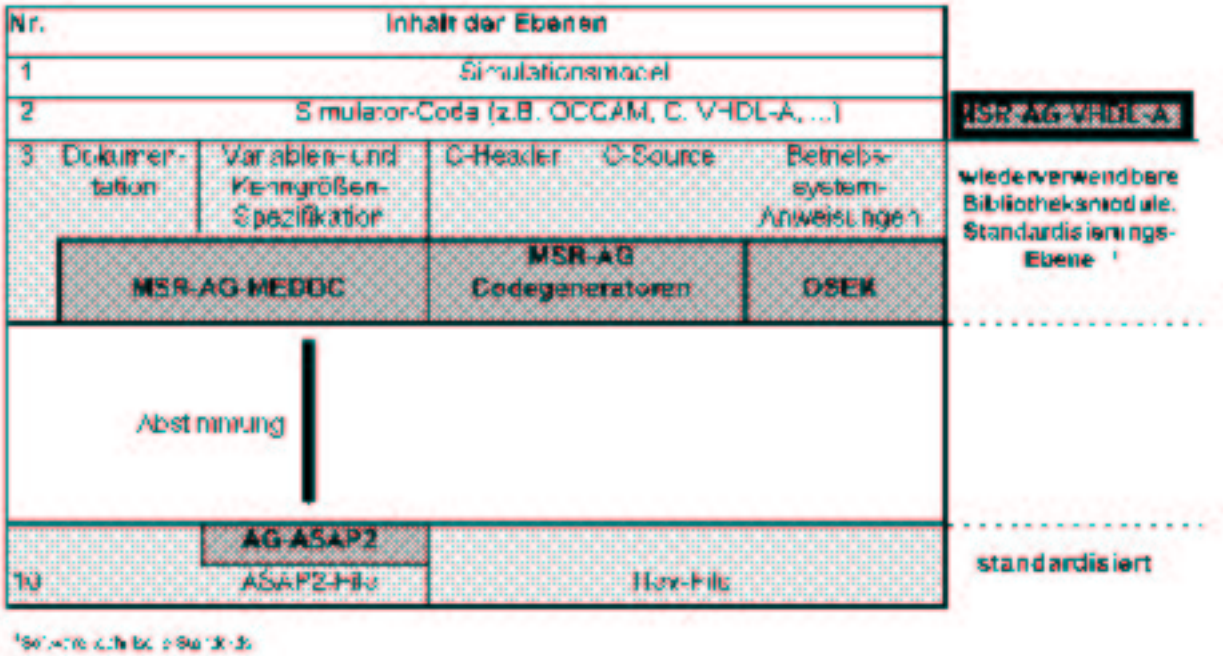

Figure 3: Standardization projects

A standardization has already taken place at the lowest level: ASAP2 File and Hex File.

2 Structuring

2.1 General

2.1.1 Link to MSRSYS

The behavior of a component can be described as a function hierarchy in the *MSRSYS.DTD* (**<behavior>**). This description is independent of the implementation in hardware or software.

Realization the software is supported by *MSRSW.DTD*. This is a design oriented on functions[2]. The relationship between functions in *MSRSYS.DTD* and *MSRSW.DTD* can be established by references (**<sw-fulfils>**).

2.1.2 Variant handling

The following scenarios are considered in variant handling:

Internal variants: These variants are controlled by the control unit without any large amount of reprogramming. The control unit program can switch between several variants by a code word being reprogrammed (in general by by conducting diagnostics at the end of the line) or even by the application of connector bridges. This type of variant is in effect a function of the software itself.

External variants: These variants are for units that are very similar that only however contain one variant. Variant exchange is by extensive reprogramming. This concerns as a rule, both the differently programmed variants as well as the unprogrammed control unit having different part numbers.

Variant descriptions are important in the area of functions (**<sw-function-spec>**) and the parameter contents (**<sw-param-contents>**).

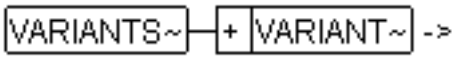

Figure 4: >Function variants

2.1.2.1 Variant-dependent parameters

The description of the **<sw-param-contents>** is possible in **<sw-function-spec-variant>** as well as in **<sw-param-contents-spec>**. The implicit variant handling **<sw-param-content>** is given at both points by the fact that **<sw-param>** is referenced in **<sw-param-content>**. **<sw-function-variant>** contains an explicit variant cross-reference. The arrangement of **<sw-param>** to the variants is given by the **<variant-def-ref>**s of the **<sw-function-variant>**s where these are referenced (implicit allocation).

The allocation of parameters to variants is thus exclusively by the use in function variants. An explicit arrangement of parameters to variants would be necessary (by **<variant-def-refs>** in **<sw-param>**) for an automated testing of contradiction-free variant descriptions for functions and parameters. This has been put back for the time being since it has not been clarified whether the thereby resulting complexity is necessary and justified[3].

2

3

2.1.2.2 Variant-dependent conversion formulae

Conversion formulae are influenced by the hardware (signal processing etc). It can thus happen that differing HW variants can also cause different conversion formulae. An example for this can be the replacement of a sensor that is not even installed in the control unit. This shall however be explicitly supported by our data/documentation model. Such a case is treated rather as follows:

- A separate **<sw-compu-method>** is applied for each variant as required. These must then differ as a minimum by the **<short-name>** as well as by the **<long-name>** used for each.
- The variant-specific **<sw-compu-method>**s are referenced in variant-specific **<sw-param>**s and **<sw-variable>**s. These must in turn differ in the **<short-name>** as well as in the **<long-name>** and can again be referenced in **<sw-function-variant>**s. Thus variant handling is not carried out concealed (i.e. by processing in the DTD), but rather is exclusively defined by the user.

2.1.3 Data dictionary storage

The data dictionary can be entered in the document, distributed both globally (in **<msrsw>**) as well as as functionally local (in **<sw-function-variant>**). It is however to be handled as a logical dictionary. This flexibility allows the use of the *MSRSW.DTD* in different process phases and process models. Furthermore, this flexibility can be used for adaptation to the possibilities available for the various tools and the export filters for these.

2.1.4 Parameter contents storage

Parameter storage (**<sw-param-contents>**, compare Topic 2.2.5 Specification of parameter contents p. 34) can be stored both locally within **<sw-function-variant>** (Topic 2.2.4.1.6 Function-related (local) parameter contents p. 34) as well as globally within **<msrsw>** Topic 2.2.5 Specification of parameter contents p. 34.

The following conventions apply for this:

1. The allocation of parameters to functions is exclusively by means of references in the function descriptions (**<sw-param-ref>** in **<sw-function-variant>**).
2. It is possible to give **<sw-param-content>**s in **<sw-function-variant>**. Only **<sw-param>**s may be referenced here that are referenced in **<sw-function-variant>** as well. The provision of parameter contents in functions must therefore be consistent with the assignment of parameters to functions.
3. If a content is specified for a parameter (**<sw-param>**) both in **<sw-function-variant>** as well as in **<sw-param-content-spec>**, then the latter takes precedence.
4. If contents are given in two different functions for the same parameter, then one entry must also be included in **<sw-param-content-spec>**. This ensures that the straightforward precedence rule given above can be applied.
5. By inserting a **<sw-param-contents>**, the author of a **<sw-function-variant>** has the possibility of controlling those parameter contents which shall be displayed locally when formatting.

Differing processes can be operated with the DTD since **<sw-param-contents>** is possible both for the functions as well as globally for the entire software (within **<msrsw>**). The following is recommended:

- Parameter contents are given as programming requirements within **<sw-function-variant>**. By this, the relation is retained even for fragmentation at the function level.
- Parameter contents as the result of the application phase are given within **<msrsw>** as closed and function-overriding. **<sw-param-contents>** within **<msrsw>** (compare Topic 2.2.5 Specification of parameter contents p. 34) thus assumes an application data status.

2.1.5 Name spaces

It generally applies that the **<short-name>** has identifying character, especially for references to the outside (e.g. ASAP). Thus it is given by this that the **<short-name>** must be unambiguous within given name (definition) spaces. These name spaces are given by the defined heirarchies in the DTD [4]. Details are to be found in *[External Document: Concepts of MSR-DTD / State: Pending / Date: TBD / URL: / Relevant Position:]*

2.1.6 References within the software

References are given at many points within the software. Redundancy can be avoided by using references.

4

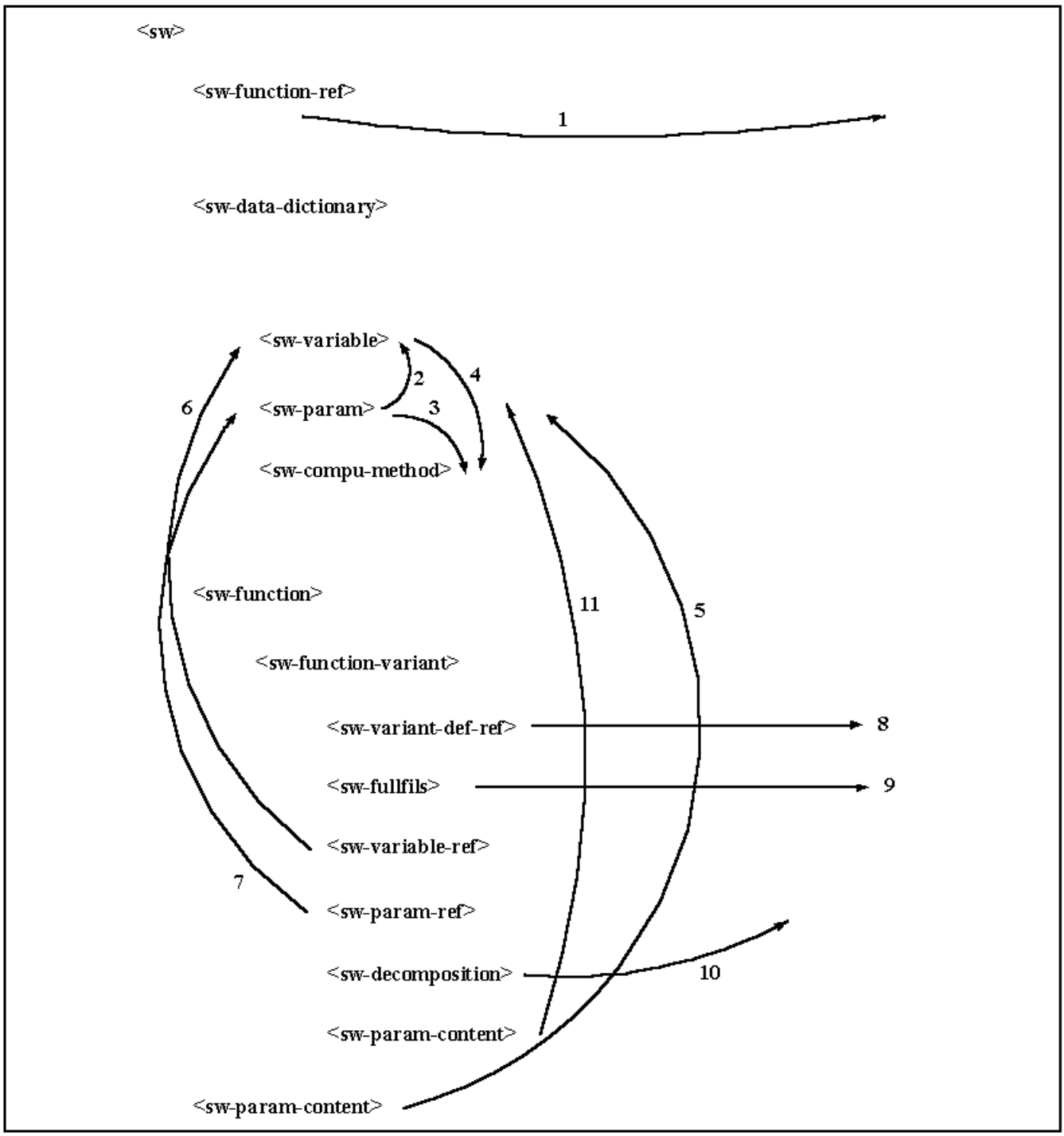

Figure 5: References within the software

Refer to chapter Topic 2.3 References p. 37.

Table 1: Reference types

No.	Designation	Source element	Target element	Meaning
1	**<sw-function-ref>**	**<msrsw>**	**<sw-function>**	Reference to a function used outside of **<MSRSW>** such as e.g. operating system or network function.
2	**<sw-variable-ref>**	**<sw-axis-individual>**	**<sw-variable>** in **<data-dictionary>**	Reference to the input variable of a parameter.
3	**<sw-compu-method-ref>**	**<sw-param-axis-values>**	**<sw-compu-method>**	Assigns a conversion formula to the axis values of a parameter.
4	**<sw-compu-method-ref>**	**<sw-variable>**	**<sw-compu-method>**	Assigns a conversion formual to a variable.
5	**<sw-param-ref>**	**<sw-param-content>**	**<sw-param>**	Allocates a parameter content to a parameter.
6	**<sw-variable-ref>**	**<sw-passing-variables>** , **<sw-local-variables>**	**<sw-variable>** in **<data-dictionary>**	Reference to a variable of a function that is used.
7	**<sw-param-ref>**	**<sw-function-variant>**	**<sw-param>**	Reference to a variable of a function variant that is used.
8	**<variant-def>**	**<variant-def>**	Variant in project data	Allocates one or more fictive variants to a function variant.
9	**<fulfills-ref>**	Part, SW function	Function	Allocates those (behavioral) functions to a part or a software function (none, one or several) that this realizes.
10	**<sw-function-ref>**	**<sw-data-dictionary>**	**<sw-function>**	Permits assignment of a data dictionary (fragment) to a certain function. Function-oriented data dictionaries can be written if **<sw-function>** is not filled out.
10	**<sw-function-ref>**	**<sw-param-contents>**	**<sw-function>**	Permits assignment of data contents to a certain function even if **<sw-function>** is not filled out.
10	**<sw-function-ref>**	**<sw-param-content>**	**<sw-function>**	Permist assignment of the content of a parameter to a certain function even if **<sw-function>** is not filled out.
10	**<sw-function-ref>**	**<sw-decomposition>**	**<sw-function>**	References to functions used are entered here.
11	**<sw-param-ref>**	**<sw-param-content>**	**<sw-param>**	Allocates a parameter content to a parameter.

2.2 Contents model software

The software comprises the following major elements (refer to Figure 6 Software structure, top level p. 19) (The software structure is shown in the following in the form of *Near&Far* graphics (compare Topic App. B Explanation of the Near&Far symbols p. 92)).

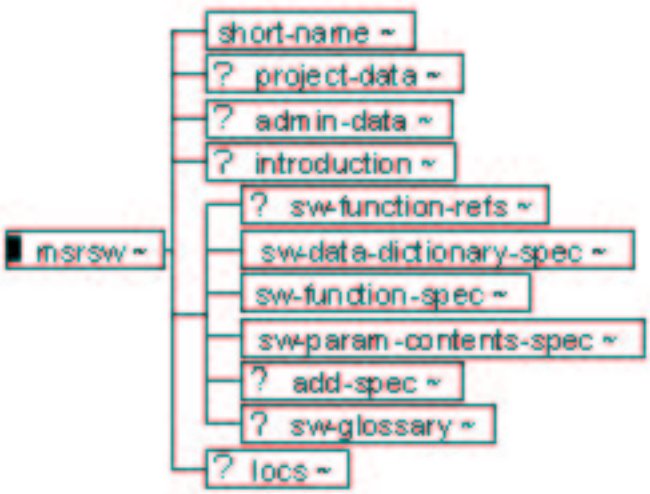

Figure 6: Software structure, top level

<short-name>	Short name for the entire document.
<project-data>	Project data (Topic 2.2.2 Project data p. 19)
<admin-data>	Administrative data (Topic App. E.1 Administrative data p. 96)
<introduction>	Short introduction (Topic App. E.7 Introduction p. 102)
<sw-function-refs>	Reference to external software (Topic 2.2.1 References to ”external” software p. 19)
<sw-data-dictionary-spec>	Global data dictionary (Topic 2.2.3 Data dictionary p. 20)
<sw-function-spec>	Detailed specification for the Definition Software function p. 94 software functions (Topic 2.2.4.1.1 Function definition p. 33)
<sw-param-contents-spec>	Global parameter contents (Topic 2.2.5 Specification of parameter contents p. 34)
<add-spec>	Additional, semi-formal specifications (Topic 2.2.7 Additional specifications p. 37)
<sw-glossary>	Permits the preparation of a glossary (Topic 2.2.6 Glossary for the document p. 37)
<locs>	Space for document-overriding references. For details see *[External Document: Concepts of MSR-DTD / State: Pending / Date: TBD / URL: / Relevant Position:]*

2.2.1 References to ”external” software

Functions from other control units can be referenced with the help of **<sw-function-refs>**. Examples for this are functions in multiple use such as operating system and network management functions.

2.2.2 Project data

Information pertaining to the project (**<project-data>**) can be entered here. Such entries include information as to company, employees and general project data. Details can be specified in the general project data on the following:

- Order justification
- Objectives
- Model
- Variant specification
- Limits to other projects
- Parallel developments
- Integration capability
- Acceptance conditions
- Schedule and plans
- Protocols, minutes of meeting

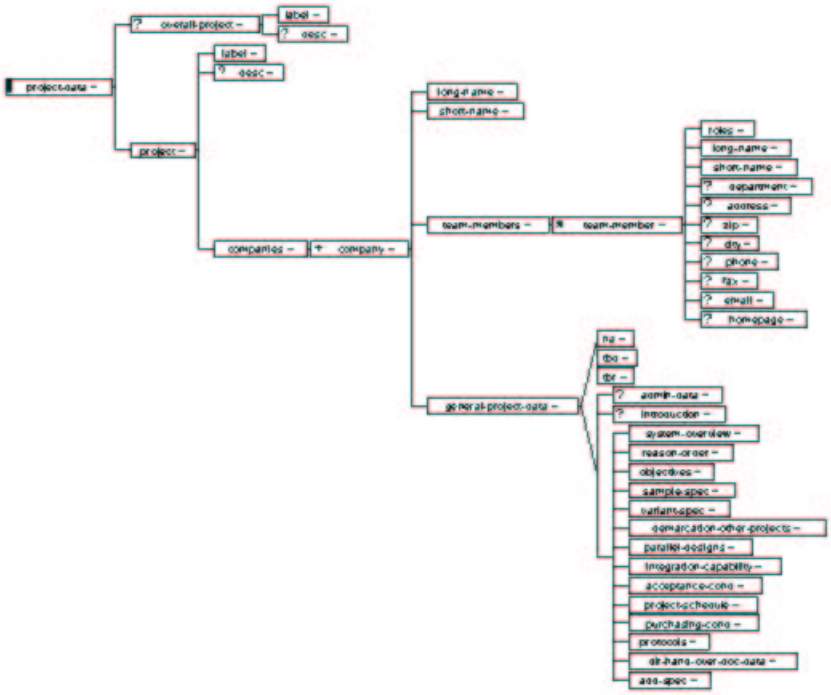

Figure 7: Project data

2.2.3 Data dictionary

The data structures for the control unit software as described, are specified in the data dictionary **<sw-datadictionary-spec>** .

A data dictionary (**<sw-datadictionary>**) comprises:

<desc>	Short description
<sw-data-dictionary-class>	This element can be used to identify the role of the present data dictionary in the process. By this for example, changed variables

	can be stored in a **<sw-datadictionary>** with the classification *changed*.
<sw-function-ref>	Permits allocation of a data dictionary to a certain function (Topic 2.1.6 References within the software p. 16).
<sw-units>	Units of measure that are used in the documentation (Topic 2.2.3.1 Units of measure p. 21).
<sw-physic-types>	Physical data types (Topic 2.2.3.2 Physical data types p. 22)
<sw-variables>	Variables (Topic 2.2.3.3 Variables p. 23)
<sw-params>	Structures for parameters (Topic 2.2.3.4 Parameter structures p. 24)
<sw-compu-methods>	Conversion formulae (Topic 2.2.3.5 Conversion formulae p. 28)
<sw-adressing-methods>	Addressing procedure for parameters and variables (Topic 3.2 Description of elements p. 40)
<sw-param-record-layouts>	Instructions for storing parameters in the control unit's memory device. These serve to control the access, from e.g. *Application systems*. (Topic 3.2 Description of elements p. 40)
<sw-code-syntaxes>	Instructions for generation the source code for the compiler for the control unit software (Topic 3.2 Description of elements p. 40)

Figure 8: Data dictionary

2.2.3.1 Units of measure

A description for the units of measure can be given both in **<sw-units>**. A unit of measure possesses a long and a short designation (**<long-name>** or **<short-name>**). The formatted output can be determined using the element **<sw-unit-display>**.

The element **<si-unit>** is used for mapping the **<sw-unit>** on a SI unit. This element possesses one attribute for each SI base unit and by which the associated exponent can be given[5]. This is needed when working with tools that can compute using SI units.

This mapping on SI units makes it possible to compare the units of measure that are used.

It is possible to establish a reference to other units of measure (preferably SI units) by means of **<sw-unit-ref>**. The procedure for converting to this unit is given in **<sw-unit-to-ref-method>**. The procedure for converting from this units is given in **<sw-unit-from-method>**. These methods are specified as 6 parameters in accordance with **<sw-asap-6-prm-method>**.

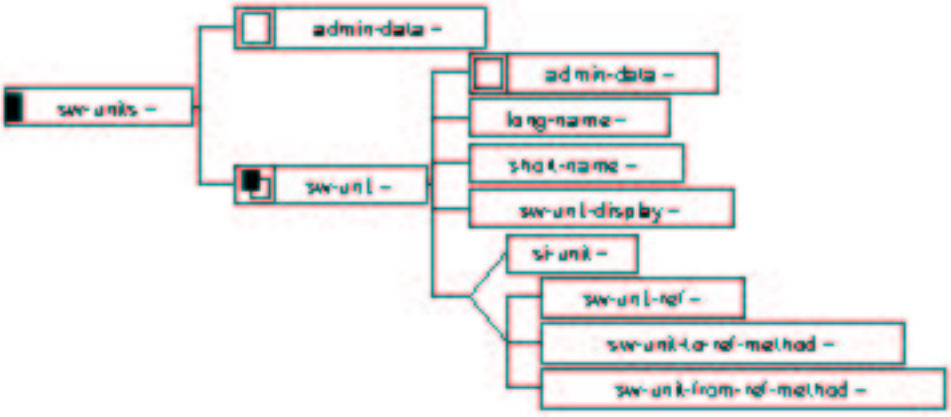

Figure 9: Units of measure

The following example highlights the relationships:

2.2.3.2 Physical data types

In the early phases in the development of control unit software, the work is carried out in physical parameters irrespective of the implementation. Global physical data types (**<sw-physic-types>**) can be defined in the data dictionary to support this phase.

Reference can be made for the definition of variables (**<sw-variable>**) to the physical data types that are already available by referencing (**<sw-physic-type-ref>**).

There is however also the possibility of directly defining the physical data type for the variable (**<sw-physic-type-1>**). For further details, refer to Figure 10 Physical data types p. 23 .

5

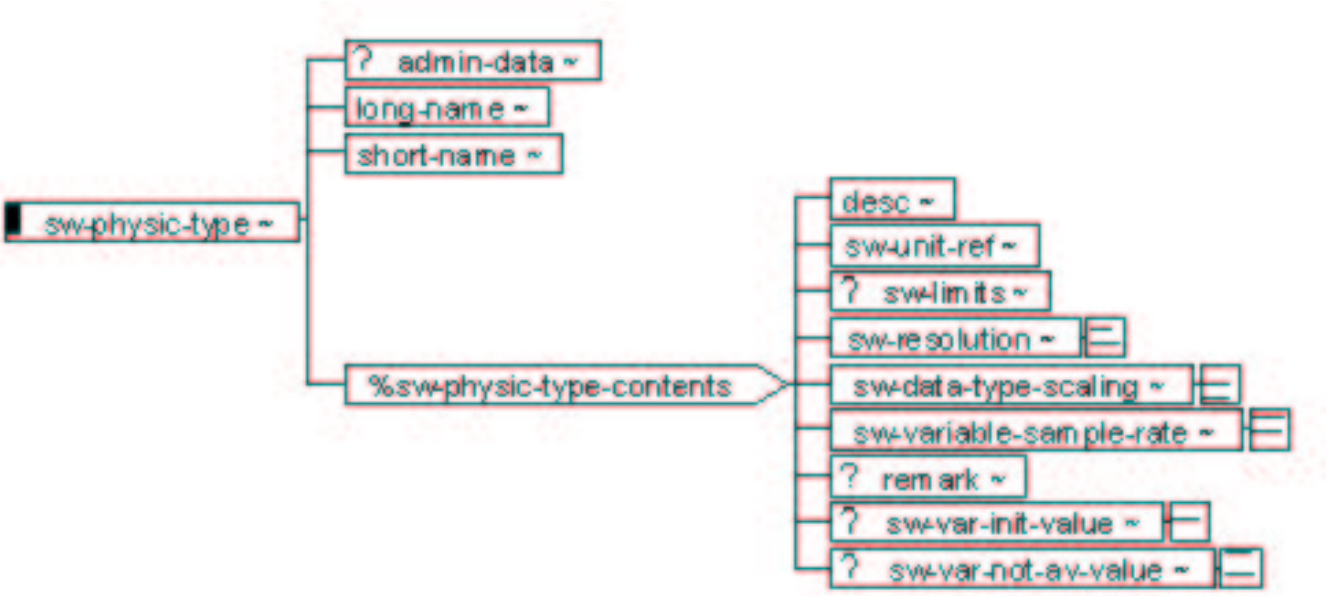

Figure 10: Physical data types

The following example demonstrates the definition of a physical data type:

2.2.3.3 Variables

Variables are defined in the data dictionary (**<sw-variables>**). They are principally to be treated globally. i.e. their **<short-name>** must be unique.

Functions use these variables for the exchange of information. These variables are referenced according to their order to do this (**<sw-function-variables>**).

Variables are however used as input parameters and as intermediate values as well.

Variables can be assigned with **<admin-data>** in order for instance, to support library management.

The structure for the description of software variables is recursive. This enables hierarchical structures for the variables to be built up, that e.g. can display a *STRUCT*-Konstrukt (in the programming language *C*).

The structure of**<sw-variable>** is shown in Figure 11 Variable p. 24. For a description of the elements, refer to Topic 3.2 Description of elements p. 40.

Figure 11: Variable

It shall be observed for the description of data types (variables and parameters) that data always possess an internal (coded) as well as an external (physical) display form (e.g. for an engine speed, internal 0..200: external 0..2000 RPM, internal 201..254: not defined and internal 255: engine speed not available). Conversion between these presentations is by means of the conversion formula referenced in **<sw-compu-method-ref>**.

Hence **<sw-limits>** can be given both in physical (**<phys>**)as well as in coded (**<phys>**) terms.

The internal value for *not available* is entered in **<sw-var-not-av-value>**. This is the value that the variable takes when the external is not available (unless not yet determined or there is an error present). There can be no external value given for *not available* since it is not available.

Not all variables are needed in the application system for the calibration. This fact is documented by the attribute **[calibration]**. It can take the values *calibration*, *no-calibration* and *not-in-memory* . This attribute is used exactly the way parameters are used.

2.2.3.4 Parameter structures

A control-unit program can be adapted both parameterized and quantitatively to the vehicle in question. The control-unit program is matched to the engine to be operated in its environment by appropriate values assigned to these parameters (e.g. body, exhaust-gas regulations). This adaptation process is termed application or calibration.

A differentiation shall be made between the structure of a parameter and its data contents. The parameter structures are described in the data dictionary in **<sw-params>** (compare Figure 12 Parameters p. 25). The parameter contents are included in **<sw-param-contents>**.

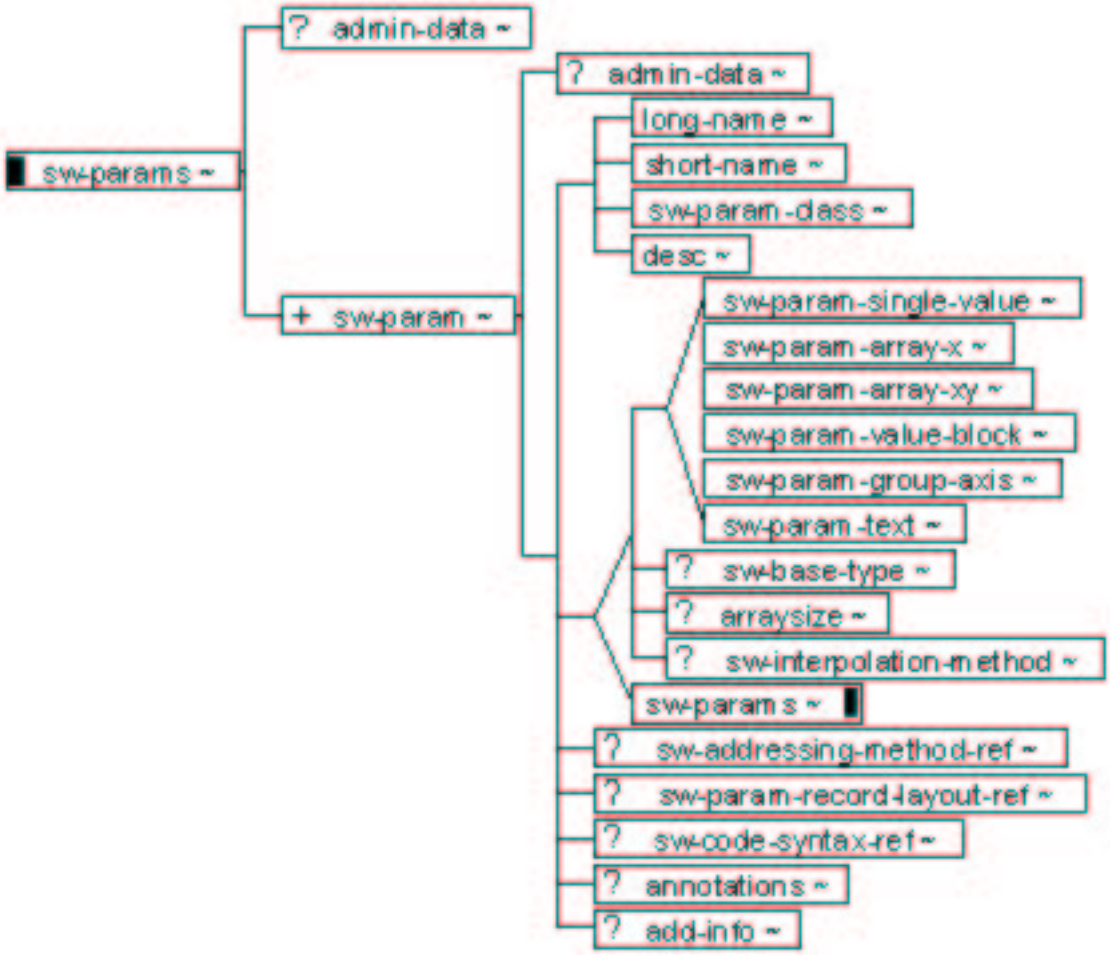

Figure 12: Parameters

The use of parameters in functions is by means of referencing (**<sw-param-ref>**). It is thereby indicated by the attribute [owns] with the possible value "no-owns"[6] of whether the parameter is to be allocated to this function, or it is defined in this function.

The following can be specified by means of an attribute **[calibration]**:

calibration The contents are matched for the parameters to the target vehicle in question during the application phase (compare Topic 1.2.8 Application (calibration) p. 12) (applied).

no-calibration The contents for the parameters cannot be changed by application syteme etc. It is only possible to read these.

not-in-memory This parameter is not saved in the EPROM of the control unit. Its contents are processed during software development (e.g. for configuration by header files). These can therefore neither be read nor changed.

There are different classes of parameters. These differ on the one hand by the assignment of **<sw-param-class>**, whereas on the other hand they are formed by appropriate characterization of the structures.

6

A differentiation is made between the following parameter classes:

Characteristic value A characteristic value (**<sw-param-single-value>**) is a parameter consisting of one single value. An engine-speed limit can be displayed by this for example.

The structure of the characteristic values is shown in Figure 13 Characteristic value p. 27.

System constant A system constant is a special characteristic value (**<sw-param-single-value>** characterized by **[calibration]**=*not-in-memory*) that is used for example, for adapting conversion formulae. This is not stored in the control unit.

Characteristic text A characteristic text (**<sw-param-text>**) is a parameter that consists of a one single text. Messages can be displayed by this, for the dashboard for example.

Characteristic In the case of a characteristic (**<sw-param-array-x>**), the output variable is computed by the control-unit program as a function of an input variable. Considered in mathematical terms, the characteristic is a function whereby it applies that: Output value = f(input value).

This function is thereby defined as a polyline in which the associated functional value (**<sw-param-axis-value>**) is given for a number (limited by **<max-count>**) of datapoints. Interpolation between two datapoints is usually linear in the control-unit program, extrapolation is constant outside of the program.

The structure of the characteristic is shown in Figure 14 Characteristic p. 28.

Fixed characteristic The datapoints cannot be arbitrarily defined in the case of fixed characteristics, but rather are computed in the control-unit programn by shift (**<shift>**) and offset (**<offset>**) (**<sw-axis-shift-offset>**). The fixed characteristic is displayed as **<sw-param-array-x>** as well, whereby **<sw-axis-shift-offset>** is used.

Map For a map (**<sw-param-array-xy>**), the output variable is computed by the control-unit program as a function of two input variables. The structure is otherwise that for a characteristic.

Fixed map As in the case of a fixed characteristic, the datapoints cannot be chosen arbitrarily either. The fixed map is also displayed as **<sw-param-array-xy>**, whereby **<sw-axis-shift-offset>** is used for both axes.

Group characteristic Group characteristics do not have datapoints of their own, but rather use the same datapoints (**<sw-param-group-axis>** referenced in **<sw-axis-grouped>**) together with other group characteristics or group maps. The group characteristic is displayed as **<sw-param-array-x>** as well, whereby **<sw-axis-grouped>** is used for the axis.

Group map Group maps do not have datapoints of their own, but rather use the same datapoints (group datapoints) together with other group characteristics or groups maps. The group map is displayed as **<sw-param-array-xy>** as well, whereby **<sw-axis-grouped>** is used for both axes (compare Topic 2.2.3.4 Example for a group map p. 28).

Characteristic values block A characteristic values block **<sw-param-value-block>** is a summary of a number (**<count>**) of identically structured characteristic values as a block. Hence for example, a characteristic value that is to be applied individually for each cylinder of the engine, could be displayed as a characteristic values block having 4 elements.

All characteristic values in the block possess the same properties. Each characteristic value receives however a separate identification (**<label>**), whereby a description for the use of the value can also be given. The characteristic values block is always handled as being closed. The components of the characteristic values block therefore cannot be individually referenced.

Characteristic values structure A characteristic values structure (**<sw-params>** in **<sw-param>**) is a summary of a number of differently structured parameters. The parameters for the

structure of e.g. a controller can be displayed by this (compare Topic 2.2.3.4 Example for a characteristic values structure: p. 28).

Group datapoints Group datapoints (**<sw-param-group-axis>**) define the datapoints that are jointly used by several group characteristics or group maps (**<sw-param-ref>** in **<sw-axis-grouped>**).

The parameter type concerned (e.g. characteristic) is defined by **<sw-param-class>**. The standardized designations from ASAP shall be used for this. These are[7]:

Table 2: Belegung von <sw-param-class>

Value[8]	Meaning	Remark
SYSTEM_CONSTANT	System constant	Equivalent in structure to a fixed value with **[calibration]**=*not-in-memory*
VALUE	Characteristic value	
CURVE_INDIVIDUAL	Characteristic	
MAP_INDIVIDUAL	Map	
CURVE_FIXED	Fixed characteristic	
MAP_FIXED	Fixed map	
CURVE_GROUPED	Group characteristic	
MAP_GROUPED	Group map	
ASCII	Characteristic text	
STRUCTURE	Parameter structure	
VALUE_BLOCK	Characteristic values block	
AXIS_VALUES	Group datapoints	

Example for a characteristic value

The structure of a characteristic value is shown in Figure 13 Characteristic value p. 27.

Figure 13: Characteristic value

The following example illustrates the use:

Example for a characteristic

The structure of the characteristic is shown in Figure 14 Characteristic p. 28.

7

8

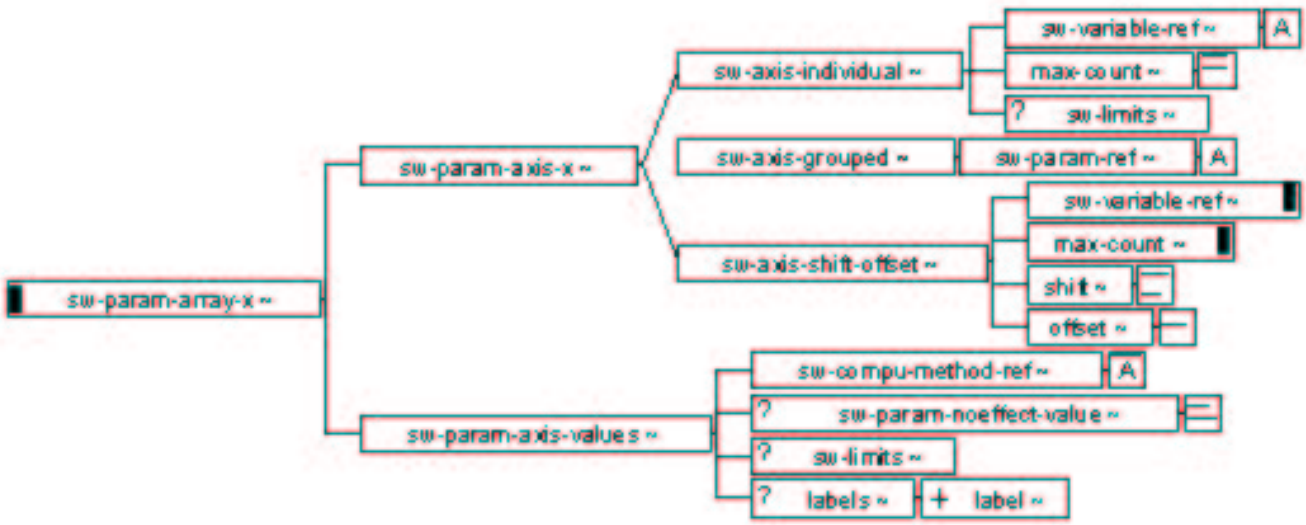

Figure 14: Characteristic

The following example illustrates the use:

Example for a group map

Example for a characteristic values structure:

2.2.3.5 Conversion formulae

The presentation of physical values as control-unit-internal values and vice-versa is defined in **<sw-compu-methods>** (compare Figure 15 Conversion formulae p. 29). Differentation is made among four different conversion methods:

Polynomial presentation
The conversion is specified by six parameters (**<sw-asap-6-prm-method>**) (compare Topic 2.2.3.5 Example for a polynomial definition p. 29). These are separated by blanks and represent the following formula:

$$int = (a \times x^2 + b \times x^1 + c) / (d \times x^2 + e \times x^1 + f)$$

Table-form definition
The conversion is defined in table-form in **<sw-compu-method-table>** (compare Topic 2.2.3.5 Example for a conversion formula in table form p. 29). The control-unit-internal value (**<cmt-int>**)[9] and the physical value (**<cmt-phys>**) are given for each pair of values (**<sw-compu-method-value-pair>**).

The type of interpolation is specified in **[interpolation-style]**.

Text-form presentation
This conversion (**<sw-compu-method-text>**) represents a control-unit-internal (**<cmt-int>**) value in a text (**<cmt-text>**) (e.g. a status identifier). One **<sw-compu-method-text-pair>** is applied for each pair of values (compare Topic 2.2.3.5 Example for a conversion in text form p. 29).

9

Program code — In **<prog-code>**, the conversion can be specified in the notation for a programming language. Programming language (**[prog-lang]**), programming language dialect (**[lang-subset]**) and the libraries used (**[used-libs]**) are given as attributes.

Mathematical presentation — The conversion is specified as a general mathematical expression (**<sw-compu-generic-math>**). The content is open, i.e. is subject to a bilateral agreement.[10]

Figure 15: Conversion formulae

Example for a polynomial definition

Example for a conversion formula in table form

Example for a conversion in text form

2.2.3.6 Addressing procedures

The parameters and variables can be addressed by different means in the control unit. The addressing procedures that are possible can be listed in **<sw-adressing-methods>** using a formal name and an informal description (**<sw-adressing-method-desc>**). Formal references (**<sw-adressing-method-ref>**) can be established in this way. For further details, refer to Topic 3 Description of elements and attributes p. 40

2.2.3.7 Memory layouts

The parameters and variables can be stored by differents means in the control unit. The storage schematics that are possible can be listed in **<sw-param-record-layout>** using a formal name and an informal description (**<sw-param-record-layout-desc>**). Formal references (**<sw-param-record-layout-ref>**) can be established in this way. For further details, refer to Topic 3 Description of elements and attributes p. 40.

2.2.3.8 Code syntaxes

The parameters and variables can be built into the source text for the compiler by code generators. Code syntaxes are used for control of these code generators. The addressing procedures that are possible can be listed in **<sw-code-syntaxes>** using a formal name and an informal description (**<sw-code-syntax-desc>**). Formal references (**<sw-adressing-method-ref>**) can be established in this way. For further details, refer to Topic 3 Description of elements and attributes p. 40

10

2.2.3.9 Agreement with ASAP

The scope of the data dictionary was determined to a great extent by matching this to the contents of *ASAP2* . This affects above all:

- **<sw-variable>**
- **<sw-param>**
- and the introduction of **<sw-compu-method>**.

The following table shows the terms used for *ASAP* and the MSR equivalents.

Table 3: Cross-reference of ASAP and MSR terms

ASAP	***MSR***
characteristics	Parameters (**<sw-param>**)
measurement	Variable (**<sw-variable>**)
compu-method	Conversion formula (**<sw-compu-method>**)

The element names are selected on the basis of the terms used in MSR so as to prevent overlapping with MSR element names already in existence (Characteristic, Measurement). Parameters (**<sw-param>**), variable (**<sw-variable>**). **<sw-compu-method>** is used for the conversion formula.

2.2.4 Functions

A function is a certain requirement or characteristic of the overall system that can be realized by hardware and/or software. A function is also however the solution to a control, regulation or display task. Such functions can also be termed application functions (compare Figure 16 Application functions p. 31).

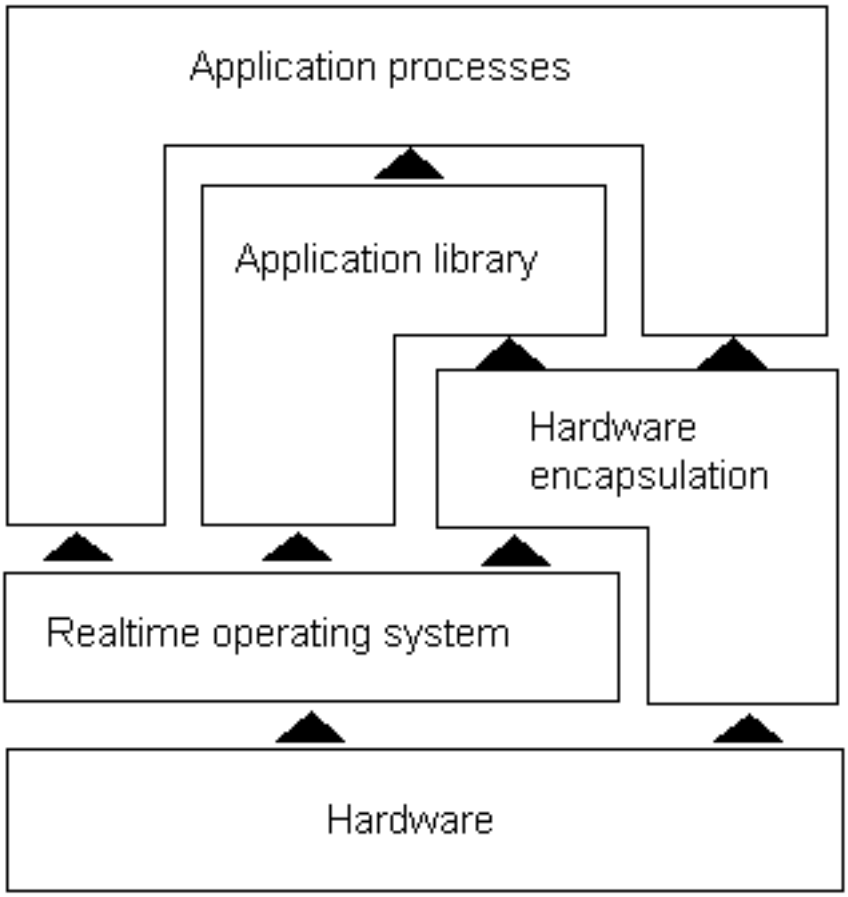

Figure 16: Application functions

The functions realized in the software are described in **<sw-function-spec>**. These functions are classified according to classes (**<sw-function-class>**). The description for the functions can be variant-specific. A minimum of one function variant must exist for each function (**<sw-function-variant>**).

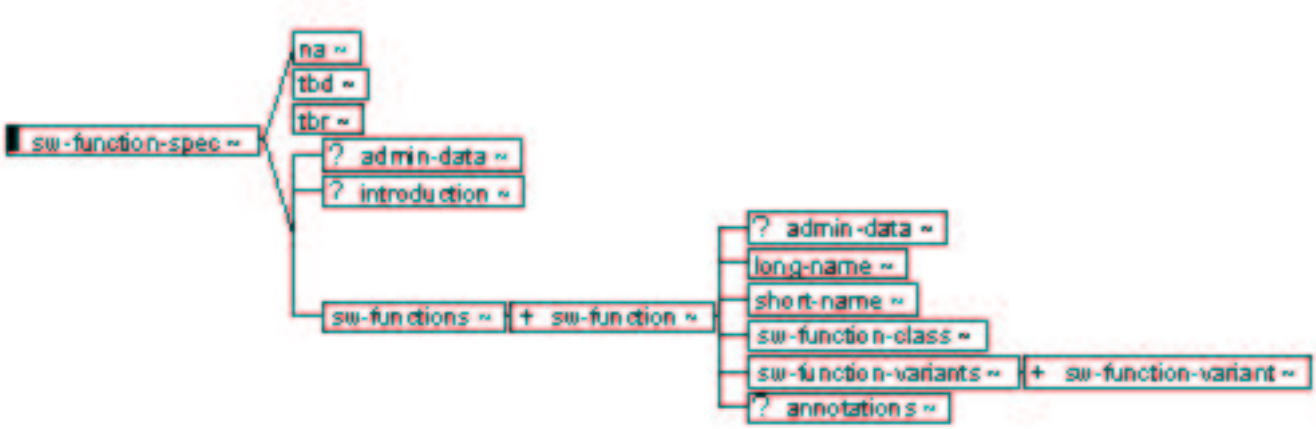

Figure 17: Function specification

2.2.4.1 Function variants

The function variant (**<sw-function-variant>**) is described in the following details (compare Figure 4 >Function variants p. 14):

Variant identifier References to the variants applicable for this description are given in **<variant-def-refs>** (compare Topic 2.1.2 Variant handling p. 14).

Function definition A semi-formal definition of the function is given in **<sw-function-def>**. This element is automatically loaded as a rule from the system design tools. The obligatory images are foreseen for accepting diagrams from the *system design tool* (compare Topic 2.2.4.1.1 Function definition p. 33).

Function description An informal description of the function is stored in **<sw-function-desc>** (compare Topic 2.2.4.1.2 Function description p. 33). This is prepared manually as a rule.

Requirements reference A reference to the system function in the *MSRSYS.DTD* is made (document-overriding) in **<sw-fulfils>** (compare Topic 2.2.4.1.3 Reference to component behavior p. 33).

Function variables The variables used in the function are listed in **<sw-function-variables>**. The type of application is also specified (compare Topic 2.2.4.1.4 Function variables p. 33).

Function parameters The parameters used in the function are listed in **<sw-param-refs>**. The type of application is specified as well by means of the attribute **[owns]** (compare Topic 2.2.4.1.5 Function parameters p. 34).

Local data dictionary A local functions data dictionary is installed in **<sw-data-dictionary-spec>** (compare Topic 2.2.3 Data dictionary p. 20).

Local parameter contents Local functions parameter contents are stored in **<sw-param-contens-spec>** (compare Topic 2.2.4.1.6 Function-related (local) parameter contents p. 34).

Test specifications Test specifications can be stored in **<sw-test-spec>** (compare Topic 2.2.4.1.7 Test specification p. 34).

Application notes Notes on calibration (application) of the present function can be given in **<sw-application-notes>** (compare Topic 2.2.4.1.8 Application notes p. 34).

Customer service notes Customer service notes with references to the present function can be given in **<sw-maintenance-notes>** (compare Topic 2.2.4.1.9 Customer servicing notes p. 34).

CARB documentation The documentation for *CARB* (*California Air Resource Board*) can be described in **<sw-carb-doc>** (compare Topic 2.2.4.1.10 CARB documentation p. 34).

Function structure The breakdown of the current function into sub-functions can be given in **<sw-decomposition>**. An overview of the entire function hierarchy can be generated from this (compare Topic 2.2.4.1.11 Function structure p. 34).

Notes Information can be forwarded in the process in **<annotations>** (vgl. Topic 3 Description of elements and attributes p. 40).

Additional information Information and descriptions for which there are no explicit structures in *MSRSW.DTD* can be saved in **<add-info>** (compare Topic App. E.4 Additional information p. 99).

Figure 18: Function variants

2.2.4.1.1 Function definition

A definition with formal components (e.g. a technical block diagram for a controller) can be established in **<sw-function-def>** for each software function. This element is loaded as a rule by the output of a *CASE tool* or a *system design tool*.

The report capabilities of the CASE tools must be used for this. The output can include both graphics as well as text.

In the interests of reproducible processes, care of this element should be either manually or not contain any manual entries at all.

2.2.4.1.2 Function description

A detailed description of the function can follow in **<sw-function-desc>**. As a rule, this is prepared manually and the automatically generated components should be stored in **<sw-function-def>**. Formal parameters can also be entered in **<prms>** and these can be tested automatically. This concerns secondary conditions, i.e. properties for a function that are not specified in **<sw-param>** because they are permanently coded in the program or are not given at all from the implementation technique.

2.2.4.1.3 Reference to component behavior

Behavior functions of the control unit that have been realized by the software function can be referenced in the element **<sw-fulfils>**. It is possible by this to decribe functions that are initially independent of the implementation (in a *MSRSYS.DTD* entity) and to establish the correspondence between behavior and software function within the scope of partitioning.

2.2.4.1.4 Function variables

Variables are listed in **<sw-function-variables>** that are read or manipulated by the software function. The actual variable description thereby is to be found in the data dictionary (**<sw-variables>**); references are given here. A distinction is made for the variables between the definition for variables and access to the variable.

If a function defines a variable, i.e. provides a variable or the value for this, then this shall be itemized in **<sw-function-export-variables>**.

Variables are listed in **<sw-function-import-variables>**) that are imported or defined by other variables. **<sw-function-local-variables>** establishes that this is not a transfer variable but rather that a local variable is concerned here.

Variables are listed in **<sw-function-modelonly-variables>** that exist only in the model (e.g. for simplification of the documentation). These do not appear during implementation in the control unit.

A distinction is made with regard to accessing variables between reading (**<sw-variable-read>**), writing (**<sw-variables-write>**) and reading-writing (**<sw-variables-readwrite>**).

The use of variable and parameters in functions in the following example.

2.2.4.1.5 Function parameters

The parameters associated with the software function are itemized in **<sw-param-refs>**. The arrangement of the parameters to the functions is by referencing these (**<sw-param-ref>**). It can be determined by the attribute **[no-own]** whether a function defines a parameter (i.e. the parameter belongs to the function) or whether the function imports a parameter (owns =”no-own”)[11].

2.2.4.1.6 Function-related (local) parameter contents

Parameter contents can be specified in **<sw-param-contents>** within **<sw-function-variant>**. These are then designated as local parameter contents (compare Topic 2.1.4 Parameter contents storage p. 15).

A detailed description of parameter contents is given Topic 2.2.5 Specification of parameter contents p. 34.

2.2.4.1.7 Test specification

The test procedures can be specified in **<sw-test-spec>** for each function variant.

2.2.4.1.8 Application notes

Notes on special features for application (calibration) of the function can be described in **<sw-application-notes>**.

2.2.4.1.9 Customer servicing notes

Information can be stored in **<sw-maintenance-notes>** that can be used later, e.g in customer servicing documents.

2.2.4.1.10 CARB documentation

Documentation required for *CARB* is entered in **<sw-carb-doc>**.

2.2.4.1.11 Function structure

The description of the software functions is in flat form. The breakdown (**<sw-decomposition>**) of the software function into details can be carried out by referencing other software functions. The multiple use of sub-functions is thus supported by this[12].

11

12

2.2.5 Specification of parameter contents

A global description of parameter contents (**<sw-param-contents>**) can be made in **<msrsw>** (compare Topic 2.1.4 Parameter contents storage p. 15). Parameter contents can be allocated to a function by **<sw-function-ref>**. A classification of parameter contents is possible by **<sw-param-contents-class>**, e.g.: ”*Test data*”, ”*Raw application data*”. Parameter contents can be given several times over. This is meaningful if for instance, many sets of test data shall be given for a parameter.

The content itself is allocated to a parameter by **<sw-param-ref>**. Parameter structure (in **<sw-param>**) and parameter content (in **<sw-param-contents>**) must agree in structure (that is to say, these carry the same **<sw-param-class>** for example).

The parameter contents are broken down into as many as three axes (**<sw-param-content-x>**, **<sw-param-content-y>**, **<sw-param-content-v>** and **<sw-param-content-text>**) in order to fill the structures in Topic 2.2.3.4 Parameter structures p. 24 with data. In the case of maps, all values are stored in **<sw-param-content-v>** on a line-by-line basis, whereby the index for the X-Achse is faster.

The data can be specified in differing formats for each axis:

<sw-param-values-phys>	Physical values, given as floating-point numbers as required
<sw-param-values-coded>	Control-unit-internal values, given as integers or a floating-point numbers
<sw-param-values-coded-hex>	Control-unit-internal values, given as hexadecimal numbers in ”*C*” format (e.g. ”*0x7f2a*”)
<sw-param-values-adr>	Address for the value in the control unit given as hexadecimal number in ”*C*” format (e.g. ”*0x7f2a*”). This address primarily serves documentation purposes (e.g. instructions for re-programming).
<sw-param-values-generic>	Additional formats can be stored here. The formats themselves are defined by the attribute **[type]**.

The parameter contents are designed such that these can also be used even without the data dictionary so as to be able to perform physical archiving for example. The elements **<sw-param-ref>**, **<sw-param-class>**, **<sw-unit>** are therefore available again as options. Any possible redundancy is thereby accepted since these elements are generated automatically. The reference **<sw-param-ref>** can no longer be made by means of ID/IDREF because the entity can only include parameter contents and not the parameter definitions. Thus only the natural addressing remains[13].

13

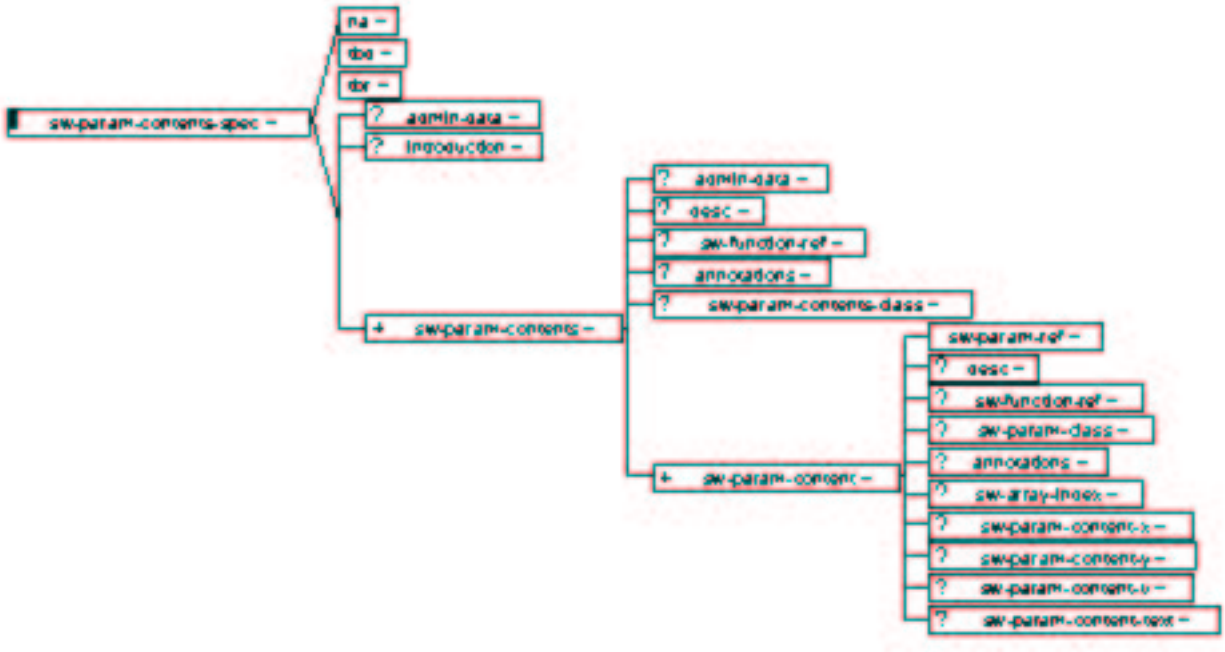

Figure 19: Parameter contents (overview)

Figure 20: Parameter contents (details)

The following examples illustrate the parameters content of a characteristic values and for a characteristic:

2.2.6 Glossary for the document

A glossary can be prepared for the document in question (**<sw-glossary>**).

2.2.7 Additional specifications

<add-spec> permits the inclusion of additional specifications that are not explicitly foreseen in the DTD (a register of abbreviations for example).

2.3 References

Additional relationships are introduced between the elements listed above. These are realized as references. The following table contains a listing of all types of references to be considered in this context and includes those already in existence.

Table 4: Types of references

No.	Designation	Source element	Target element	Meaning
1	**<system-ref>**	**<requirements>**, **<product-spec>**	**<part-type>**	Defines the top-level part type (exactly one) for each of the two views.
2	**<part-type-ref>**	**<part>**	**<part-type>**	Assigns the specific part type (exactly one) to a part.
3	**<sw-function-ref>**	**<software>**	**<sw-function>**	Allocates software functions to the software of a part type (none, one or several), that are described within the software of another part type, e.g. multiple use of such functions as operating system or network management functions.
4	**<sw-function-ref>**	**<sw-function>**	**<sw-function>**	Allocates software functions to a software function (none, one or several) within the same software description, that constitute the decomposition of the first.
5	**<fulfills-ref>**	Part, SW function	Function	Allocates those (behavioral) functions to a part or a software function (none, one or several) that it realizes.
6	**<function-view-ref>**	Function	Function	Allocates the (behavioral) function in the view requirement, to a specified (behavioral) function in the view consent, that is realized by the first. [14]

The following diagram illustrates the types of references as an example (no. 6 excepted).

[14]

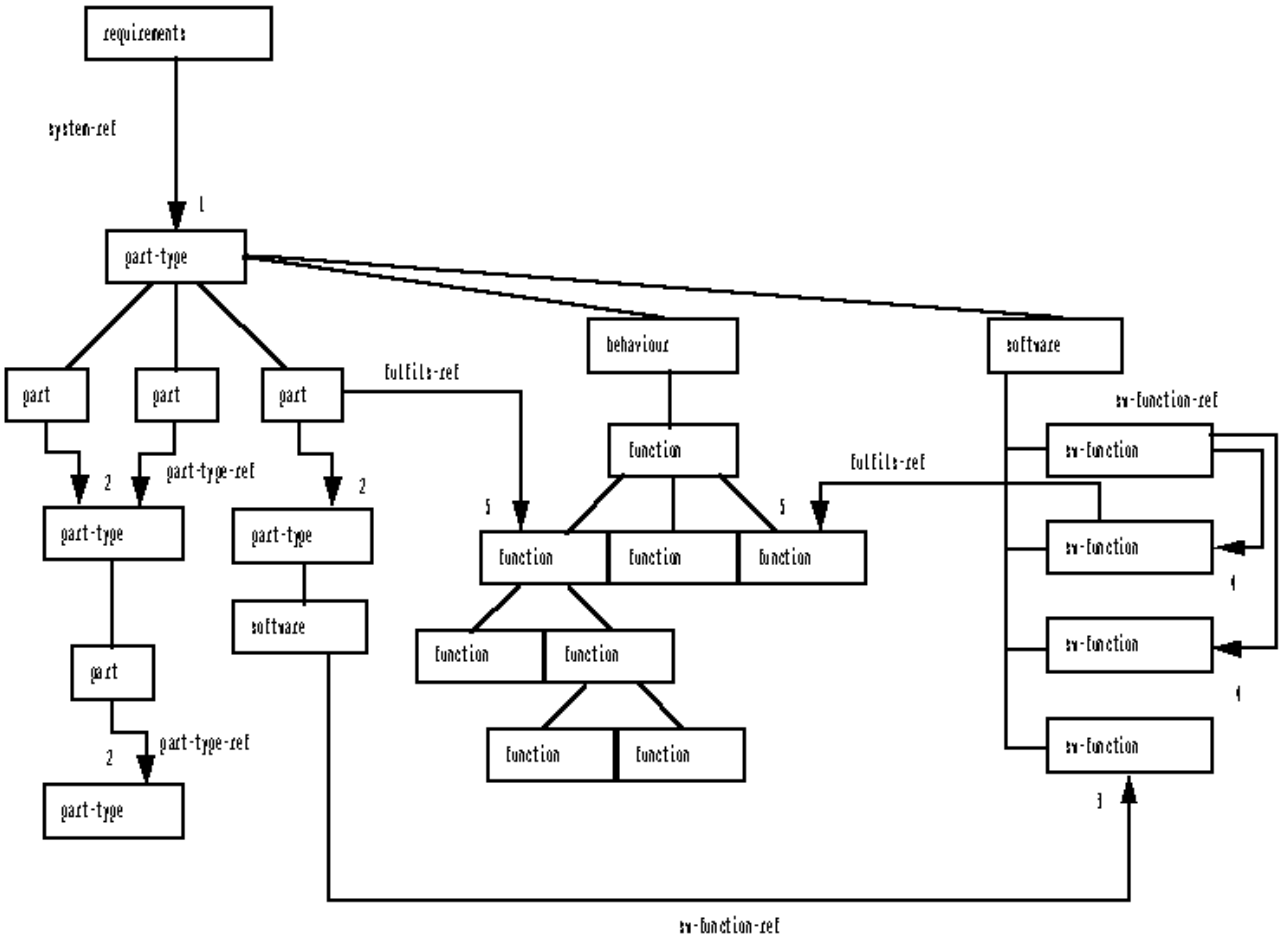

Figure 21: Extended structure with references

3 Description of elements and attributes

3.1 Classed elements

The following classes of elements are used in the software DTD:

- sw-addressing-method-class
- sw-data-dictionary-class
- sw-function-class
- sw-param-class
- sw-param-contents-class
- sw-param-record-layout-class
- sw-code-syntax-class

The element name is suffixed by ”-class” for all classed elements. The classes are not standardized and are not defined in the DTD. It is however recommended to use certain class designations, e.g. according to ASAP. Proposals are made to this end. Refer to **<sw-param-class>**, e.g. ”value”. The class designations are to be agreed between the project partners.

3.2 Description of elements

The elements of the DTD software are described in the following. The description of an element is made up of: Description of the element, details of attributes, details of the context, and an example is given for each.

These are elements of general applicability and are to be allocated to a user architecture. These element names are not prefixed by ”sw-”. These elements are used in all DTD’s that correspond to a certain user architecture. They are therefore not all listed again here. These are described in the document ”Concepts of the DTD”.

There is an attribute ”s” (Signature) that occurs in all elements and is therefore not listed again here.

Fixed attributes are also given in the examples for the sake of clarity (e.g. **[f-namespace]**). The fixed attributes no longer appear in an entity due to the software DTD, i.e the entity is only complete together with the DTD.

<abs>

Description	Absolute value for parameter characteristics. See parameter model (**<sw-prm>**).
Attribute	**[s]** Signature
Included in	**<prm-char>**
Example	

<add-info>

Description	Additional notes. The programming language used could be documented in a function using this element for example.
Attributes	-
Included in	**<sw-function-variant>**,**< sw-param>**, **<sw-variable-implementation>**

Example

<add-spec>

Description

Attributes -

Included in **<general-project-data>**

Example

<admin-data>

Description Information can be provided within the structure of this elements on the separate administration of sub-entities. It is for this reason that this element exists at all those points where a potential export and import is foreseen for separate administration purposes.

Attributes

[f-child-type]	For applications in order to make differing ”child-types” possible (type test). Example: ”language:selection”, i.e. the child element language is of the type selection, can thus be chosen from a selection list.
[s]	Signature

Included in All elements that are foreseen for a fragmentation.

Example

<annotation>

Description Information of different kinds can arise throughout the course of a process, (e.g. processing information, notes etc.) that shall be linked to the variables, parameters or function and be forwarded in the process. Such kinds of information are stored in **<annotation>**.

Attributes -

Included in **<annotations>**

Example

<arraysize>

Description In the case of variables, it can happen that these are present not as an element but rather in the form of an array. This value then gives the size of the array. Diemsions can be separated by blanks for multi-dimensional arrays.

Parameter fields and arrays can also be established.

Attributes

[S]	Signature

Included in **<sw-param>**, **<sw-variable-implementation>**

Example

<bit-base-name>

Description In**<sw-bit-representation>** in **<sw-variable>**. Name of the variable containing the bit.

Attributes -

Included in **<sw-bit-representation>**

Example

<bit-base-type>

Description In**<sw-bit-representation>** in **<sw-variable>**. Type (byte, word, long) of the variable containing the bit.

Attributes -

Included in **<sw-bit-representation>**

Example

<bit-count>

Description In**<sw-bit-representation>** in **<sw-variable>**. Number of associated bits.

Attributes -

Included in **<sw-bit-representation>**

Example

<bit-pos>

Description In**<sw-bit-representation>** in **<sw-variable>**. Postion of the bits in the basic size.

Attributes -

Included in **<sw-bit-representation>**

Example

<change>

Description Element in which a description for the type of change can be given.

Attributes -

Included in **<Modification>**

Example

<chapter>

Description A description of informal information not structured according to the application can be given in a chapter. Chapters can be structured according to a hierarchy. A chapter can contain paragraphs, tables, graphics, lists, etc.

Attributes -

Included in

Example

<cmt-int>

Description Internal value for a conversion formula

Attributes -

Included in **<sw-compu-method-text-pair>**, **<sw-compu-method-value-pair>**

Example

<cmt-phys>

Description Physical value for a conversion formula.

Attributes **[s]** Signature

Included in **<sw-compu-method-value-pair>**

Example

<cmt-text>

Description Value in text form for a conversion formula.

Attributes -

Included in **<sw-compu-method-text-pair>**

Example

<code>

Description

Attributes -

Included in **<variant-char>**, **<variant-char-value>**,**<variant-def>**

Example

<coded>

Description Coded/internal values for limits.

Attributes -

Included in **<sw-limits>**

Example

<coded-max>

Description Maximum coded limiting value

Attributes -

Included in **<coded>**

Example

<coded-min>

Description Minimum coded limiting value

Attributes -

Included in **<coded>**

Example

<companies>

Description Company-specific details, comprising 1 .. n companies.

Attributes -

Included in **<project>**

Example

<company>

Description Company-spefic details for a company participating in the project

Attributes

[f-child-type] For applications to make different ”child-types” possible

[f-id-class]

[f-name-space]

[id]

	[role]	Role of the company participating in the project. ”Manufacturer”, ”supplier”.

Included in **<companies>**

Example

<company-doc-info>

Description

Attributes -

Included in **<company-doc-infos>**

Example

<company-doc-infos>

Description Company-specific information for administrative data

Attributes -

Included in **<admin-data>**

Example

<comany-ref>

Description Reference to a company

Attributes -

Included in **<company-doc-info>**, **<company-revision-info>**

Example

<company-revision-info>

Description Company-specific information concerning a certain revision/variant.

Attributes **[f-child-type]**

Included in **<company-revision-infos>**

Example

<company-revision-infos>

Description Revision and variant information on an entity or fragment.

Attributes	**[s]**	Signature

Included in **<doc-revision>**

Example

<cond>

Description Condition for parameters.

Attributes -

Included in **<prm-char>**

Example

<count>

Description Number of elements for one parameter block.

Attributes -

Included in **<sw-param-value-block>**

Example

<date>

Description Date information, multiple-language possible

Attributes **[s]** Signature

Included in **<doc-revision>**, **<schedule>**

Example

<date-1>

Description Date information, multiple-language not possible

Attributes **[s]** Signature

Included in **<std>**, **<xdoc>**

example

<demarcation-other-project>

Description Delimitation to other projects

Attributes -

Included in **<general-project-data>**

Example

<department>

Description Department

Attributes -

Included in **<team-member>**

Example

<desc>

Description Description element

Attributes -

Included in **<figure>**, **<overall-project>**, **<prm>**, **<project>**, **<sw-data-dictionary>**, **<sw-param>**, **<sw-param-content>**, **<sw-param-contents>**, **<sw-physic-type-contents>**, **<sw-variable>**, **<tbd>**

Example

<doc-label>

Description

Attributes -

Included in **<company-doc-info>**

Example

<doc-revision>

Description

Attributes **[f-child-type]**

Included in **<doc-revisions>**

Example

<doc-revisions>

Description

Attributes -

Included in **<admin-data>**

Example

<entity-name>

Description

Attributes -

Included in **<company-doc-info>**

Example

<figure>

Description Graphics can be linked using this element.

Attributes -

Included in

Example

<file>

Description Information on a file name for referencing an external file, a standard

Attributes

[filename]	Name of the file.
[notation]	Type of file, e.g. ”wmf”, ”cgm”.
[tool]	Tool for displaying or editing the file
[tool-version]	Version of the tool with with the file was created

Included in **<std>**, **<xdoc>**, **<xfile>**

Example

<function-ref>

Description Reference to function or requirements-function.

Attributes -

Included in **<sw-fulfils>**

Example

<general-project-data>

Description General project data.

Attributes -

Included in **<company>**

Example

<generic-math>

Description Mathematical formula.

Attributes -

Included in **<formula>**

Example

<graphic>

Description Graphics file information

Attributes **[category]**

[filename]

[fit]

[height]

[notation]

[scale]

[width]

Included in **<ml-graphic>**

Example

<ie>

Description

Attributes **[type]**

Included in **<mixed-content-4>**, **<ml-data-1>**, **<ml-data2>**, **<ml-data-4>**

Example

<introduction>

Description Introductory section or introductory chapter

Attributes -

Included in **<company-revision-info>**, **<general-project-data>**, **<info>**, **<msrsw>**, **<sample-spec>**, **<sw-data-dictionary-spec>**, **<sw-glossary>**, **<sw-param-contents-spec>**, **<variant-spec>**

Example

<label>

Description A label is a designation for an object that must not, and cannot, be referenced, i.e. possesses no**< short-name>** and no **[id]**.

A **<label>** for **<sw-param-value-block>** is a long designation for the characteristic values of a characteristic values block.

Attributes -

Included in **<annotation>**, **<labels>**, **<overall-project>**, **<prms>**

Example

<labels>

Description Quantity of labels for axial values

Attributes -

Included in **<sw-param-axis-values>**

Example

<locs>

Description This element is used for document/entity-overriding referencing (HighTime referencing using namelocs)

Attributes -

Included in **<msrsw>**

Example

<language>

Description Identifies the Masterlanguage for **<admin-data>**

Attributes -

Included in **<admin-data>**

Example

<long-name>

Description Long designation, e.g. ”Engine temperature”.

Attributes -

Included in

Example

<max>

Description Maximum value for a parameter characteristic

Attributes -

Included in **<prm-char>**

Example

<max-count>

Description Maximum number of datapoints

Attributes -

Included in **<sw-axis-individual>**, **<sw-axis-shift-offset>**

Example

<min>

Description Minimum value for prm-char

Attributes -

Included in **<prm-char>**

Example

<modification>

Description

Attributes **[type]** Content-related, doc-related

Included in **<modifications>**

Example

<modifications>

Description

Attributes -

Included in **<doc-revision>**

Example

<msrsw>

Description Root elelement

Attributes **[f-namespace]**

[f-pubid] Fixed attribute: -//MSR//DTD MSR SOFTWARE DTD:V1.1.0:MSRSW.DTD//EN

[HyTime]

[pubid] -//MSR//DTD MSR SOFTWARE DTD:V1.1.0:MSRSW.DTD//EN

Included in -

Example

<na>

Description This element is used instead of sub-structures if certain statements are not relevant (”not applicable”).

Attributes -

Included in **<general-project-data>**, **<info>**, **<sample-spec>**, **<sw-data-dictionary-spec>**, **<sw-functions-spec>**, **<sw-glossary>**, **<sw-param-contents-spec>**

Example

<ncoi-1>

Description This is a general element that contains informal and non-software-specific structures (”none coded information”).

Attributes -

Included in **<info>**, **<sample>**, **<sw-function-def>**, **<sw-glossary>**

Example

<offset>

Description The datapoints are defined by an algorithm for fixed characteristics and maps. Example for an algorithm: Datapoint[i] = (... shift) * x + offset

Attributes -

Included in **<sw-axis-shift-offset>**

Example

<p>

Description Paragraph. A paragraph can comprise text and the elements tt (technical text), xref (cross reference), e (text attribute like bold), ft (footnote), sup (superscript), sub (subscript), ie (index entry), std (standard), xdoc (external document), xfile (external file) in any arbitrary, though not hierarchical, order.

Attributes **[help-entry]** Entry for a Help system

Included in add-info, annotation-text, chapter, cond, def, introduction, ncoi-1, p-level-elements, sw-addressing-method-desc, sw-application-notes, sw-carb-

doc, sw-code-syntax-desc, sw-function-desc, sw-maintenance-notes, sw-param-record-layout-desc, sw-test-spec

Example

<private-code>

Description

Attributes **<type>** -

Included in **<private-codes>**

Example

<private-codes>

Description

Attributes -

Included in **<company-doc-infos>**

Example

<prm>

Description A parameter model has been implemented in MSR. There are 2 possibilities given. A parameter can be specified either by defining **<abs>** ,**<tol>** or by defining **<min>**, **<typ>** and **<max>**.

Attribute **[f-id-class]**

[id]

Included in prms

Example

<prms>

Description List of parameters

Attribute -

Included in **<add-info>**, **<chapter>**, **<ncoi-1>**, **<sw-addressing-method-desc>**, **<sw-application-notes>**, **<sw-carb-doc>**, **<sw-function-desc>**, **<sw-maintenance-notes>**, **<sw-param-record-layout-desc>**, **<sw-test-spec>**, **<topic-1>**

Example

<prm-char>

Description Description of the parameter characteristics of a parameter

Attributes -

Included in **<prm>**

Example

<prog-code>

Description Description of conversion formulae in a programming language notation

Attributes **[lang-subset]**

[prog-lang] Programming language

[used-libs] Libraries used

Included in sw-compu-method

Example

<project>

Description Information regarding a project.

Attributes -

Included in **<project-data>**

Example

<project-data>

Description Project data

Attributes -

Included in **<msrsw>**

Example

<shift>

Description The datapoints are defined by an algorithm for fixed characteristics and fixed maps. Example for an algorithm: Datapoint[i] = (... shift) * x + offset

Attributes -

Included in **<sw-axis-shift-offset>**

Example

<short-name>

Description Short designation, e.g. ”TMOT”. The **<short-name>** has an identifying character, in particular for outside references (e.g. *ASAP*, *ASCET*).

Attributes -

Included in chapter, company, def-item, figure, formula, msrsw, namloc, prm, sample, std, sw-addressing-method, sw-code-syntax, sw-compu-method, sw-function, sw-param, sw-param-record-layout, sw-physic-type, sw-unit, sw-variable, team-member, topic-1, topic-2, variant-char, variant-def, xdoc, xfile

Example

<si-unit>

Description There are SI units of measure for the units of measure. STEP (ISO/DIS 10303-41. S96ff) is supported with regard to SI units by seven basic units (length, mass, time, electric_current, thermodynamic_temperature, amount_of_substance, luminous_intensity). These basic units are realized by means of attributes.

Attributes **[amount-ofsubstance-expo]**

[electric-current-expo]

[length-expo]

[luminous-intensity-expo]

[mass-expo]

[thermodynamic-temperature-expo]

[time-expo]

Included in **<sw-unit>**

Example

<reason>

Description

Attributes -

Included in **<modification>**

Example

<revision-label>

Description Revision

Attributes -

[]

Included in **<company-revision-info>**

Example

<state>

Description

Attributes -

Included in **<company-revision-info>**

Example

<std>

Description

Attributes **[f-child-type]** date1:date

[f-id-class] std

[id]

Included in

Example

<sw-addressing-method>

Description Describes the addressing scheme. A description of how a parameter or a RAM variable from the control unit is given. Examples for the differing types of addressing are the direct addressing and the indirect addressing by a vector. The addressing scheme can also be used to define the RAM range in which the a RAM variable lies (e.g. in the internal or external RAM). Addressing schemes can be used by parameters and RAM variables.

This element is a reference to an access method for a name that must be simulated in the following systems (e.g. *adjustment systems*). This name links the MSR entity and the following systems.

Addressing schemes are not standardized or pre-defined in the DTD. Neither are they fully modeled. They are only described in terms of text in **<sw-addressing-method-desc>**.

Attributes **[f-id-class]**

[id]

Included in **<sw-addressing-methods>**

Example

<sw-addressing-method-class>

Description Gives the class for the addressing scheme. Parameters or variables can only use addressing schemes of a certain class. This however cannot be assured by SGML means; the application must do this.

Addressing-scheme classes are not standardized or pre-defined in the DTD.

Attributes -

Included in **<sw-addressing-method>**

Example

<sw-addressing-method-desc>

Description Text-form description of the addressing schemes

Attributes -

Included in **<sw-addressing-method>**

Example

<sw-addressing-method-ref>

Description Reference to the addressing scheme, e.g. by giving the short name.

Attributes **[HyNames]**

[HYTIME]

[sw-addressing-method] idref

Included in **<sw-param>**, **<sw-variable-implementation>**

Example

<sw-addressing-methods>

Description All addressing schemes are defined in **<sw-data-dictionary>**

Attributes -

Included in **<sw-data-dictionary>**

Example

<sw-application-notes>

Description Application notes

Attributes -

Included in **<sw-function-variant>**

Example

<sw-array-index>

Description Reference to the sub-structure for parameter structures.

Attributes -

Included in **<sw-param-content>**

Example

<sw-asap-6-prm-method>

Description 6-parameter formula corresponding to ASAP (polynomial presentation. Presentation of the elements of the formula separated by white spaces. formula: int = $(a \times x^2 + b \times x^1 + c) / (d \times x^2 + e \times x^1 + f)$

Attributes -

Included in **<sw-compu-method>**

Example

<sw-axis-grouped>

Description Several characteristics have the same datapoints for running-time reasons that are referenced by the name.

Attributes -

Included in **<sw-param-axis-x>**, **<sw-param-axis-y>**

Example

<sw-axis-individual>

Description Description of an axis, the datapoints for which can be individually defined. Each characteristic or each map has its own private datapoint values (unlike a group characteristic).

Attributes -

Included in **<sw-param-axis-x>**, **<sw-param-axis-y>**, **<sw-param-group-axis>**

Example

<sw-axis-shift-offset>

Description Special form of datapoints used for reasons of optimizing the running time. The datapoints are computed according to the following algorithm: Datapoint[i] = ... + Offset.

Attributes **[]**

Included in **<sw-param-axis-x>**, **<sw-param-axis-y>**

Example

<sw-base-type>

Description Base type of SW variable (e.g. char, integer).

Attributes -

Included in **<sw-param>**, **<sw-variable-implementation>**

Example

<sw-bit-representation>

Description This optional element contains information or bit presentation in the form of elements (number of bits, bit position, bit base name, bit base type) for variables with the data type ”bit”.

Attributes -

Included in **<sw-variable-implementation>**

Example

<sw-carb-doc>

Description CARB documentation.

Attributes -

Included in **<sw-function-variant>**

Example

<sw-code-syntax>

Description Name reference to a code syntax. The presentation of RAM variables and parameters in the data source file is defined by the code syntax. The code syntaxes are described in the **<sw-data-dictionary>**. The code syntaxes described here can be used by variables and parameters by referencing.

The content of a code syntax is not fully modeled, i.e. the code syntax is only decribed in text-form in **<sw-code-syntax-desc>**. There are no standardized values or values pre-defined in the DTD for the code syntax.

Attributes **[f-id-class]**

[id]

Included in **<sw-code-syntaxes>**

Example

<sw-code-syntax-class>

Description Gives the class for the code syntax. Parameters or variables can only use the code syntax of a certain class. This however cannot be assured by SGML means; the application must do this. A code syntax is for example, suitable for a parameter or for a variable, or only for a characteristic. The combination of a certain code syntax with a certain saving scheme can also be expressed by the class.

There are no standardized values or values pre-defined in the DTD for the code syntax classes.

Attributes -

Included in **<sw-code-syntax>**

Example

<sw-code-syntax-desc>

Description Description of the code syntax

Attributes -

Included in **<sw-code-syntax>**

Example

<sw-code-syntax-ref>

Description A reference to a code syntax can be made optionally within a RAM variable or parameter.

Attributes **[HyNames]**

[HYTIME]

[sw-code-syntax]

Included in **<sw-param>**, **<sw-variable-implementation>**

Example

<sw-code-syntaxes>

Description List of code syntaxes.

Attributes -

Included in **<sw-data-dictionary>**

Example

<sw-compu-generic-math>

Description The conversion formula is described by a general mathematical expression.

Attributes -

Included in **<sw-compu-method>**

Example

<sw-compu-method>

Description Conversion formula. Computing procedure according to which a control-unit-internal variable can be converted into its physical value. The conversion formula must be reversible.

Attributes **[f-id-class]**

[fid]

Included in **<sw-compu-methods>**

Example

<sw-compu-method-ref>

Description Reference to a conversion formula

Attributes **[HyNames]**

[HYTIME]

[sw-compu-method]

Included in **<sw-param-axis-values>**

Example

<sw-compu-method-table>

Description With this form for the conversion formula, the relationship between internal and physical presentation is given using a table of for the internal-physical pairs of values. An attribute *interpolation-style* is given that can take the values *interpolation* (default) and *discrete*.

Attributes **[interpolation-style]** Possible values: Interpolation, no-interpolation, discrete. Default: Interpolation.

Included in **<sw-compu-method>**

Example

<sw-compu-method-text>

Description The conversion formula is described in text form. Unlike the other types of conversions, the physical side does not constitute a number but rather

a text. Applications are conceivable for switches (”on”/”off”) or for country identification (”D”, ”F”, ”US”).

Attributes -

Included in **<sw-compu-method>**

Example

<sw-compu-method-text-pair>

Description Includes an internal and a text-form value for text-type conversion formulae.

Attributes -

Included in **<sw-compu-method-text>**

Example

<sw-compu-method-value-pair>

Description Includes an internal and a physical value for tabular types of conversion formulae.

Attributes -

Included in **<sw-comput-method-table>**

Example

<sw-compu-methods>

Description List of conversion formulae.

Attributes -

Included in **<sw-data-dictionary>**

Example

<sw-data-dictionaries>

Description Size of the data dictionaries

Attributes -

Included in **<sw-data-dictionary-spec>**

Example

<sw-data-dictionary>

Description Data dictionary. Data types, variables, parameters, units of measure, addressing schemes, storage schemes, code syntaxes and conversion formulae can be described here.

Attributes -

Included in **<sw-data-dictionaries>**

Example

<sw-data-dictionary-spec>

Description Data-dictionary specification.

Attributes -

Included in **<msrsw>**

Example

<sw-data-dictionary-class>

Description Information can be provided in this element regarding the development and modifications made to data dictionaries, e.g. ”Data dictionary from company xy”.

Attributes -

Included in **<sw-data-dictionary>**

Example

<sw-data-type-scaling>

Description Quantization of a variable. Example: ”16 bit”.

Attributes -

Included in **<sw-physic-type-contents>**

Example

<sw-decomposition>

Description Formation of functional hierarchies by referencing included and used functions.

Attributes -

Included in **<sw-function-variant>**

Example

<sw-fullfils>

Description Allocates that or those (behavorial) function(s) to a software function that it realizes.

Attributes -

Included in **<sw-function-variant>**

Example

<sw-function>

Description Software function

Attributes **[f-id-class]**

[id]

Included in **<sw-functions>**

Example

<sw-function-class>

Description Function class. The functions can be broken down into different classes. There can be functions for example occurring only in the documentation/ analysis phase that can be grouped to a class.

Attributes -

Included in **<sw-function>**

Example

<sw-function-def>

Description A short decription for each function is given by a suitable means of presentation. A graphical and a short text are used as a rule. Example: ASCET functional model, status diagrams, SA/SD diagrams.

Attributes -

Included in **<sw-function-variant>**

Example

<sw-function-desc>

Description Extensive description in text form for the function.

Attributes -

Included in **<sw-function-variant>**

Example

<sw-function-export-variables>

Description List of exported variables. When a function defines a variable, i.e., provides a variable or the value, then it is to be listed here.

Attributes -

Included in **<sw-function-variables>**

Example

<sw-function-import-variables>

Description List of imported variables. Variables are listed here that are made available by other functions.

Attributes -

Included in **<sw-function-variables>**

Example

<sw-function-local-variables>

Description List of local variables, i.e. no transfer variable.

Attributes -

Included in **<sw-function-variables>**

Example

<sw-function-modelonly-variables>

Description List of variables that only exist in the model and no longer occur during implementation.

Attributes -

Included in **<sw-function-variables>**

Example

<sw-function-ref>

Description In **<msrsw>**, reference to functions used that are described outside of the SW. In **<sw-decomposition>**, reference to functions that are described within **<msrsw>**.

Attributes **[HyNames]**

[HYTIME]

[sw-function]

Included in **<sw-data-dictionary>**, **<sw-decomposition>**, **<sw-functions-refs>**, **<sw-param-content>**, **<sw-param-contents>**,

Example

<sw-function-refs>

Description References to functions used that are described outside of the document, e.g. operating system functions, network services.

Attributes -

Included in **<msrsw>**

Example

<sw-function-spec>

Description Functions specification. A functions specification can include several functions. The functions can be in hierarchical form by which reference to a function is made by **<sw-decomposition>**.

Attributes -

Included in **<msrsw>**

Example

<sw-function-variables>

Description List of variables used by the function

Attributes -

Included in **<sw-function-variant>**

Example

<sw-function-variant>

Description Function variant. If no function variants are used, then a minimum of one still has to be given. Each function variant must be completely described in the implementation V1.1.0, i.e. there is no re-use.

A features bar in **<project-data>** ... **<var-def>** (function switch) can be referenced by means of **< variant-def-ref>**.

Attributes -

Included in **<sw-function-variants>**

Example

<sw-function-variants>

Description Number of function variants.

Attributes -

Included in **<sw-function>**

Example

<sw-functions>

Description Number of functions in the functional specification/in the document. Envelope element.

Attributes -

Included in **<sw-function-spec>**

Example

<sw-glossary>

Description Glossary

Attributes -

Included in **<msrsw>**

Example

<sw-interpolation-method>

Description Interpolation method for parameters. The methods are not standardized or defined in the DTD. The methods are described in text form.

Attributes -

Included in **<sw-param>**

Example

<sw-limits>

Description Minimum and maximum values for the physical and the control-unit-internal values are given here.

Attributes -

Included in **<sw-axis-individual>**, **<sw-param-axis-values>**, **<sw-physic-type-contents>**, **<sw-variable-implementation>**

Example

<sw-maintenance-notes>

Description Customer servicing notes

Attributes -

Included in **<sw-function-variant>**

Example

<sw-param>

Description Software parameters. Software parameters are parameters (in ASAP characteristics): characteristics, maps and fixed values.

Parameters are as a rule, stored in the EPROM of the control unit. They are applicable, i.e. they can be read or modified using application tools within the scope of the plausibility and within the scope of set access protection attributes. Exceptions: e.g. cylinder number, not in the EPROM, not applicable. There is one attribute *calibration* for the possible attribute values *calibration* (default), *no-calibration* and *not-in-memory*.

Attributes **[calibration]** Possible values: Calibration (default), no-calibration, not-in-memory

[f-child-type]

Included in **<sw-params>**

Example

<sw-param-array-x>

Description Characteristic

Attributes -

Included in **<sw-param>**

Example

<sw-param-array-xy>

Description Map

Attributes -

Included in **<sw-param>**

Example

<sw-param-axis-values>

Description Value axis. Description of a characteristic value or the values of a characteristic or map

Attributes -

Included in **<sw-param-array-x>**, **<sw-param-array-xy>**, **<sw-param-single-value>**, **<sw-param-value-block>**

Example

<sw-param-axis-x>

Description Describes the x axis of a parameter (**<sw-param>**). This description consists of the input variable (as reference to a variable), min, max, max-count (max. number of datapoints).

Attributes -

Included in **<sw-param-array-x>**, **<sw-param-array-xy>**

Example

<sw-param-axis-y>

Description Describes the y axis of a parameter (**<sw-param>**). This description consists of the input variable (as reference to a variable), min, max, max-count (max. nzmver of datapoints).

Attributes -

Included in **<xs-param-array-xy>**

Example

<sw-param-class>

Description Gives the type of parameter (characteristic, map, characteristic value, fixed characteristic, group characteristic, fixed map).

Attributes -

Included in **<sw-param>**, **<sw-param-content>**

Example

<sw-param-content>

Description Parameter values

Attributes -

Included in **<sw-param-contents>**

Example

<sw-param-contents-class>

Description Gives the type of the data contents/data sets, e.g.: ”Test data”, ”Raw application data”.

The parameter classes are not standardized or defined in the DTD.

Attributes -

Included in **<sw-param-contents>**

Example

<sw-param-content-text>

Description The values of a test parameter are given here (sw-param-text).

Attributes -

Included in **<sw-param-content>**

Example

<sw-param-content-v>

Description An individual value is given here for a characteristic value. The values of the v axis are given here for a characteristic. The values of the v axis are given here for a map. The following convention thereby applies: Index of the x axis runs faster. Values are separated by blanks or line feeds.

Attributes -

Included in **<sw-param-content>**

Example

<sw-param-content-x>

Description The physical, internal, ... values for the x axis are given here.

Attributes -

Included in **<sw-param-content>**

Example

<sw-param-content-y>

Description The physical, internal, ... values for the y axis are given here.

Attributes -

Included in **<sw-param-content>**

Example

<sw-param-contents-spec>

Description Specification for parameter contents

Attributes -

Included in **<msrsw>**, **<sw-function-variant>**

Example

<sw-param-group-axis>

Description Group characteristic

Attributes -

Included in **<sw-param>**

Example

<sw-param-noeffect-value>

Description Value that cancels the action by a parameter (typically 0 or 1).

Attributes -

Included in **<sw-param-axis-values>**

Example

<sw-param-record-layout>

Description Storing scheme (in *ASAP record layout*). A storing scheme decribes the form for storing a parameter in the EEPROM. The storing scheme determines the sequence of parameters, components and their data types. Hence a storing scheme can define for a characteristic, that the address for the input variable shall be saved first stored as a byte, then the number of datapoints as an unsigned byte, then the datapoints as signed word and finally the values of the characteristic.

This element is the name for a reference to a storing scheme. This storing scheme designates the structure for the storage device in the memory of the control unit. The following systems (e.g. *adjustment systems*) must evaluate these storing schemes. The link to such types of systems is by their names.

Code syntax objects describe the presentation of an object in a programming language, the storing-scheme objects their storage in the EPROM. Code syntax objects and storing-scheme objects must be consistent. I.e. a parameter cannot use word storage for the storing scheme and byte storage for the code syntax.

The storing scheme is not fully modeled. The actual storing scheme can be described in text form in **<sw-param-record-layout-desc>**.

Attributes **[f-id-class]**

[id]

Included in **<sw-param-record-layouts>**

Example

<sw-param-record-layout-class>

Description Gives the class for the storing scheme. Parameters or variables can only use storing schemes of a certain class. This however cannot be assured by SGML means; the application must do this. The storage classes are not standardized and are not defined in the DTD.

Attributes -

Included in **<sw-param-record-layout>**

Example

<sw-param-record-layout-desc>

Description The storing scheme can be described here in text form.

Attributes -

Included in **<sw-param-record-layout>**

Example

<sw-param-ref>

Description Reference to **<sw-param>** parameters. If the attribute own is not defined for **<sw-param-ref>** in **<sw-function-variant>**, then this means that the parameter is defined in the function and is hence assigned to this.

Attributes **[HyNames]**

[HYTIME]

[owns]

[sw-param]

Included in **<sw-axis-grouped>**, **<sw-param-content>**, **<sw-param-refs>**

Example

<sw-param-single-value>

Description Applicable single paramater. A single parameter is treated as a characteristic with one element. The parameter is defined in**< sw-param>**, the value is given in **<sw-param-content>**. The allocation is by referencing **<sw-param-content>** to **<sw-param>** in**<sw-param-ref>**.

Attributes -

Included in **<sw-param>**

Example

<sw-param-target>

Description Reference to the name of a variable/RAM cell where the result of the interpolation shall be given.

Attributes -

Included in **<sw-param-group-axis>**

Example

<sw-param-text>

Description A description of parameters can be given here that have text as their values. Example: Message text for driver back-signals. The values are included in **<sw-param-content-text>**.

Attributes -

Included in **<sw-param>**

Example

<sw-param-value-block>

Description A parameters block comprises an array of characteristic values of the same type. The number is given in **<count>**.

Attributes -

Included in **<sw-param>**

Example

<sw-param-values-adr>

Description This element contains the addresses of parameters. The values are in the sub-element **<v>**, that is given once for a single characteristic value and several times for parameters.

Attributes -

Included in **<sw-param-content-v>**, **<sw-param-content-x>**, **<sw-param-content-y>**

Example

<sw-param-values-coded>

Description This element contains the values for parameters in coded form. The values are in the sub-element **<v>**, that is given once for a single characteristic value and several times for parameters.

Attributes -

Included in **<sw-param-content-v>**, **<sw-param-content-x>**, **<sw-param-content-y>**

Example

<sw-param-values-coded-hex>

Description This element contains the values for parameters in hexadecimal coded form. The values are in the sub-element **<v>**, that is given once for a single characteristic value einmal and several times for parameters.

Attributes -

Included in **<sw-param-content-v>**, **<sw-param-content-x>**, **<sw-param-content-y>**

Example

<sw-param-values-generic>

Description Values can be stored here in a format that is not directly supported. The format is given in the attribute **[type]**. The use of this element is process-specific.

Attributes **[type]**

Included in **<sw-param-content-v>**, **<sw-param-content-x>**, **<sw-param-content-y>**

Example

<sw-param-values-phys>

Description This element contains the values for parameters in physical form. The values are in the sub-element **<v>**, that is given once for a single characteristic value einmal and several times for parameters.

Attributes -

Included in **<sw-param-content-v>**, **<sw-param-content-x>**, **<sw-param-content-y>**

Example

<sw-params>

Description List of parameters (envelope element).

Attributes -

Included in **<sw-data-dictionary>**

Example

<sw-physic-type>

Description Definition of a physical, implementation-independent and re-usable data type. Physical data types are defined in the early phases of software development for the control unit.

Attributes **[f-id-class]**

[id]

Included in **<sw-physic-types>**

Example

<sw-physic-type-1>

Description Physical data types can be defined for **<sw-physic-types>** both in the data dictionary as well as directly for the varibles definition. The physical properties defined directly for variables have no **<long-name>** and no **<short-name>**, and cannot be referenced.

Attributes -

Included in **<sw-variable>**

Example

<sw-physic-type-contents>

Description Description of a physical data type.

Attributes -

Included in **<sw-physic-type>**, **<sw-physic-type-1>**

Example

<sw-physic-type-ref>

Description Reference/use of a physical data type by means of a variable

Attributes **[HyNames]**

[HYTIME]

[sw-physic-type]

Included in **<sw-variable>**

Example

<sw-physic-types>

Description Number of physical data types

Attributes -

Included in **<sw-data-dictionary>**

Example

<sw-resolution>

Description Resolution, e.g. ”100 Rev/m/s”. Whereby only ”100” is to be given, ”Rev/m/s” is the unit in **<sw-unit>**.

Attributes -

Included in **<sw-physic-type-contents>**

Example

<sw-unit>

Description Definition of a unit of measure

Attributes **[f-id-class]**

[id]

Included in **<sw-units>**

Example

<sw-unit-display>

Description Information as to how the unit of measure shall displayed in an output medium.

Attributes -

Included in **<sw-unit>**

Example

<sw-unit-ref>

Description Reference to a unit of measure already defined for use

Attributes **[HyNames]**

[HYTIME]

[sw-unit]

Included in **<sw-compu-method>**, **<sw-param-content-v>**, **<sw-param-conten-x>**, **<sw-param-content-y>**, **<sw-physic-type-contents>**

Example

<sw-var-init-value>

Description Initialization value for a variable

Attributes -

Included in **<sw-physic-type-contents>**, **<sw-variable-implementation>**

Example

<sw-var-not-av-value>

Description Internal value for an internal value not available cannot be given

Attributes -

Included in **<sw-physic-type-contents>**

Example

<sw-variable>

Description Variable (in ASAP ”*measurement*”). Variable parameter needed for the algorithm of a control-unit function. Variables are stored as a rule in the RAM. A differentiation is made between input, intermediate (local) and output variables.

Attributes **[calibration]**

[f-id-class]

[f-namespace]

[id]

Included in **<sw-variables>**

Example

<sw-variable-implementation>

Description Element for a SW variable for which implementation notes can be entered (not to be filled in manually)

Attributes -

Included in **<sw-variable>**

Example

<sw-variable-kind>

Description E.g. data flow, control flow

Attributes -

Included in **<sw-variable-implementation>**

Example

<sw-variable-ref>

Description Reference to **<sw-variable>** / RAM variable

Attributes **[HyNames]**

[HYTIME]

[sw-variable]

Included in **<sw-axis-individual>**, **<sw-axis-shift-offset>**, **<sw-function-modelonly-variables>**, **<sw-param-target>**, **<sw-variables-read>**, **<sw-variables-readwrite>**, **<sw-variables-write>**

Example

<sw-variable-sample-rate>

Description Sampling rate or resolution over time (e.g. crankshaft synchronization, every 10 ms).

Attributes -

Included in sw-physic-type-contents

Example

<sw-variables>

Description Number of variables for a data dictionary. The structure for describing the variables is recursive and hence hierarchical structures for the variable can be established by this (e.g. Struct-Construct in C).

Attributes -

Included in **<sw-data-dictionary>**, **<sw-variable-implemenation>**

Example

<sw-variables-read>

Description Variables to which reading access from a function is possible

Attributes -

Included in **<sw-function-export-variables>**, **<sw-function-import-variables>**

Example

<sw-variables-readwrite>

Description Variables to which reading and writing access from a function is possible

Attributes -

Included in **<sw-function-export-variables>**, **<sw-function-import-variables>**, **<sw-function-local-variables>**

Example

<sw-variables-write>

Description Variables to which writing access from a function is possible

Attributes -

Included in **<sw-function-export-variables>**, **<sw-function-import-variables>**

Example

<team-member-ref>

Description Reference to a team member

Attributes **[HyNames]**

[HYTIME]

[team-member]

Included in **<team-members>**

Example

<tbd>

Description ”to be defined”

Attributes -

Included in **<general-project-data>**, **<info>**, **<sample-spec>**, **<sw-data-dictionary-spec>**, **<sw-function-spec>**, **<sw-glossary>**, **<sw-param-contents-spec>**, **<variant-spec>**

Example

<tbr>

Description ”to be resolved”

Attributes -

Included in **<general-project-data>**, **<info>**, **<sample-spec>**, **<sw-data-dictionary-spec>**, **<sw-function-spec>**, **<sw-glossary>**, **<sw-param-contents-spec>**, **<variant-spec>**

Example

<topic-1>

Description Chapter structure. Unlike a chapter, this does not appear in the table of contents as a separate chapter.

Attributes **[f-id-class]** topic

[help-entry]

[id]

Included in **<add-info>**, **<chapter>**, **<ncoi-1>**, **<sw-addressing-method-desc>**, **<sw-application-notes>**, **<sw-carb-doc>**, **<sw-code-syntax-desc>**, **<sw-function-desc>**, **<sw-maintenance-notes>**, **<sw-param-record-layout-desc>**, **<sw-test-spec>**

Example

<topic-2>

Description Introductory chapter, chapter structure. Same as topic-1 except no prms. Unlike a chapter, this does not appear in the table of contents as a separate chapter.

Attributes **[f-id-class]** topic

[help-entry]

[id]

Included in **<introduction>**

Example

<used-languages>

Description List of the languages used for **<admin-data>**. The master language is given in **<language>**.

Attributes -

Included in **<admin-data>**

Example

<variant-char>

Description A description of variante features can be given here

Attributes **[id]**

[f-id-class]

[type] REQUIRED: new-part-number, no-new-part-number

Included in **<variant-chars>**

Example

<variant-def>

Description Description of variant definitions that can be referenced in **<sw-funtion-variant>**. The corresponding variant features (**<variant-def>**) are allocated here to a variant for a variants definition and the allocation of variant feature to feature value (**<variant-char-value>**, **<value>**) takes place.

Attributes -

Included in **<variant-defs>**

Example

<variant-def-ref>

Description

Attributes **[HyNames]**

[HYTIME]

[variant-def]

Included in **<variant-def-refs>**

Example

<variant-def-refs>

Description

Attributes -

Included in **<sw-function-variant>**

Example

3.3 Description of attributes

[amount-of-substance-exp]

Description Mole; quantity, compared to the number of atoms in 0.012 kilogram of carbon 12.

Possible values

Default IMPLIED

Included in si-unit

Example

[calibration]

Description Describes the calibration of a parameter (**<sw-param>**) or variable.

Possible values calibration, no-calibration, not-in-memory

Default calibration

Included in sw-param

Example

[category]

Description Graphics type

Possible values barcode, conceptual, engineering, flowchart, graph, logo, schematic, waveform

Default IMPLIED

Included in graphic

Example

[electric-current-expo]

Description Electric current (ampere).

Possible values

Default IMPLIED

Included in si-unit

example

[ext-id-class]

Description External ID class

Possible values

Default IMPLIED

Included in xref

Example

[f-child-type]

Description

Possible values

Default

Included in xref, std, xdoc, company, roles, sample, admin-data, doc-revision, company-revision-info

Example

[f-id-class]

Description All objects that carry a name/ have a **<short-name>** and hence have an **[id]** are identified by the attribute **[f-id-class]**. The value of **[f-id-class]** is chosen at the same time as the element name for the object.

Possible values chapter, company, def-item, figure, formula, prm, sample, std, sw-function, sw-unit, sw-physic-type, sw-variable, sw-param, sw-compu-method, sw-addressing-method, sw-record-layout, sw-code-syntax, table, team-member, topic, variant-char, variant-def, xdoc, xfile, external

Default REQUIRED

Included in xref

Example

[idref]

Description

Possible values

Default REQUIRED

Included in

Example

[f-namespace]

Description Name-space limits are introduced in order for example, to merge sub-trees without any conflict of names. These name-space limits are marked by the attribute **[f-namespace]**, the value for which agrees with the attribute **[f-id-class]** for the respective object type. All objects of one type must be named differently within this sub-tree.

Possible values

Default

Included in

Example

[f-pubid]

Description

Possible values

Default -//MSR//DTD MSR SOFTWARE DTD:V1.1.0:MSRSW.DTD//EN

Included in msrsw

Example

[id]

Description Identifier

Possible values

Default REQUIRED

Included in chapter, company, def-item, figure, formula, nameloc, prm, sample, std, sw-addressing-method, sw-function, sw-param, sw-param-record-layout, sw-physic-type, sw-unit, sw-variable, table, team-member, topic-1, topic-2, variant-char, variant-def, xdoc, xfile

Example

[interpolation-style]

Description Type of interpolation for a conversion table.

Possible values interpolation, no-interpolation, discrete

Default interpolation

Included in sw-compu-method-table

Example

[length-expo]

Description Length in meters

Possible values

Default IMPLIED

Included in si-unit

Example

[luminous-intensity-expo]

Description Luminous intensity in candela.

Possible values

Default IMPLIED

Included in si-unit

Example

[mass-expo]

Description Mass in grams

Possible values

Default IMPLIED

Included in si-unit

Example

[owns]

Description In **<sw-param-ref>** with possible attribute values ”no-own”.

Possible values no-own

Default IMPLIED

Included in sw-param-ref

Example

[s] *Description* Signature. That which has changed by substitution into an entity can be determined on the basis of the signature. The siganture is generated by a tool (e.g. timestamp).

Possible values

Default IMPLIED

Included in All elements

Example

[sw-compu-method]

Description In **<sw-compu-method-ref>** reference to conversion formula

Possible values

Default REQUIRED

Included in sw-compu-method-ref

Example

[sw-function]

Description In **<sw-function-ref>** reference to function

Possible values

Default REQUIRED

Included in sw-function-ref

Example

[sw-param]

Description In **<sw-param-ref>** reference to parameter

Possible values

Default REQUIRED

Included in sw-param-ref

Example

[termodynamic-temperature-expo]

Description Temperature in Kelvin

Possible values

Default IMPLIED

Included in sw-unit

Example

[time-expo]

Description Time in seconds

Possible values

Default IMPLIED

Included in si-unit

Example

4 Overview Changes

[SSW] MSR DOC Structure Principles for Software

4.1 Overview to Status following harmonization with ASAP

geplant für 26.04.96

4.2 Overview to Second revision

geplant für 29.04.96

4.3 Overview to Corrections

geplant für 03.06.96

Table 5: Requests zu Corrections

Name	Kind	Thema	State	Prio	Pd	geplant für	P.
owns	enh	sw-param-ref owns = ”owns”	**open**	A 2			p. 80
paramunit	enh	Parameter contents require units of measure	**in process** Concluded			Corrections	p. 80
paramhex	enh	Parameter contents at Integer/ HEX/Address level	**in process** Concluded			Corrections	p. 80
paramfunc	enh	Parameter contents must be assignable to functions	**in process** Concluded			Corrections	p. 80
kgrhier-achie		Support of parameter hierarchies	**in process** Concluded			Corrections	p. 81
kgarray	bug	Parameter contents for arrays	**in process** Concluded			Corrections	p. 81
sprmref	enh	SW-param-ref semantic in parameter contents	**passed** compare semantic reference in			Corrections	p. 82
annot	enh	Annotation	**in process** Concluded			Corrections	p. 82
ndim-array	enh	Support of multi-dimensional arrays	**passed** Concluded	SW 09			p. 82

Table 5 (Cont.): Requests zu Corrections

Name	Kind	Thema	State	Prio	Pd	geplant für	P.
unknown-SW	enh	Handling of software parameter forms not yet considered	**rejected**				p. 83
cleanup-sw-datatypes	enh	SW data types should be cleaned up	**passed**	10			p. 83
prog-code	enh	prog-code in sw-compu-method	**in process** Concluded	B1			p. 84
sw-gen-math	enh	sw-compu-generic-math	**open** H. Rauleder to track this.				p. 84
Osek	enh	OSEK aspects	**open** H. Bless maintains an element list				p. 85
diagnose	enh	Handling diagnostics/ diagnosis status	**open**				p. 85
literatur	doc	Complete bibliography	**open**				p. 85
security	enh	Handle safety relevance of SW functions	**open**				p. 86
glosssar	doc	Concur glossary	**open**	C1			p. 86
schedule	enh	Description of schedule dependencies	**open**			Corrections	p. 86
va-sample		Extend the implementation for variables	**in process** Concluded				p. 87
param-basic		Introduce basic type for parameters	**in process** Concluded				p. 87
msrsw	enh	Re-name root element sw in msrsw	**in process** Concluded	C1			p. 87
fktvars	enh	Optional functions variable	**in process** Concluded	C1			p. 87
label	enh	Introduce spacer for long name for param-content	**in process**	A1			p. 87
admin-in-pcont		Administrative data for parameter contents	**passed**				p. 88

Table 5 (Cont.): Requests zu Corrections

Name	Kind	Thema	State	Prio	Pd	geplant für	P.
fref-in-pcont		Function references in parameter contents	**open**				p. 88
list-adr		Lists of addressing schemes, storing schemes and code syntaxes.	**in process**				p. 88
glosopt		Glossary optional	**open**				p. 89
units		Units of measure	**open** open				p. 89
variables		Variable	**open** open				p. 89

Table 6: Lösungen in Corrections

Name	Kind	Thema	State	Prio	Pd	Occurred in	P.
paramunit	enh	Parameter contents require units of measure	**in process** Concluded			Corrections	p. 80
paramhex	enh	Parameter contents at Integer/ HEX/Address level	**in process** Concluded			Corrections	p. 80
paramfunc	enh	Parameter contents must be assignable to functions	**in process** Concluded			Corrections	p. 80
kgrhier-achie		Support of parameter hierarchies	**in process** Concluded			Corrections	p. 81
kgarray	bug	Parameter contents for arrays	**in process** Concluded			Corrections	p. 81
sprmref	enh	SW-param-ref semantic in parameter contents	**passed** compare semantic reference in			Corrections	p. 82
annot	enh	Annotation	**in process** Concluded			Corrections	p. 82
schedule	enh	Description of schedule dependencies	**open**			Corrections	p. 86

5 Changes

5.1 [owns] sw-param-ref owns = ”owns”

Kind	proposed by	State	Prio	days	Occurred in	Geplant für
enh	WI	**open**	A 2		[10] p. 77	

Subject The attribute **[owns]** in **<sw-param-ref>** must also permits the value ”owns”.

Begründung The attribute otherwise remain. This makes processing more difficult.

5.2 [paramunit] Parameter contents require units of measure

Kind	proposed by	State	Prio	days	Occurred in	Geplant für
enh	Hünerfeld	**in process** Concluded			[10] p. 77	[10] p. 77

Subject Parameter contents **<sw-param-contents>** require units of measure.

Begründung Parameter contents are used to document physical as well as for exchange with other systems. The entire data dictionary is not always available for such exchanges. It must therefore be possible to give an option for the units of measure.

Beschlossene Lösung

Introduce element **<sw-unit>**.

Release notes

5.3 [paramhex] Parameter contents at Integer/HEX/Address level

Kind	proposed by	State	Prio	days	Occurred in	Geplant für
enh	weichel	**in process** Concluded			[10] p. 77	[10] p. 77

Subject Parameter contents need not only be represented as integer variables.

Begründung It is meaningful for the documentation to store interger representations as well as address information.

Beschlossene Lösung

The children of **<sw-param-content>** (**<sw-param-content-x>** etc.) are given a separate element for each representation.

Release notes

5.4 [paramfunc] Parameter contents must be assignable to functions

Kind	proposed by	State	Prio	days	Occurred in	Geplant für
enh	hünerfeld	**in process** Concluded			[10] p. 77	[10] p. 77

Subject Parameter contents should be assignable to functions - at least as an option.

Begründung A function-oriented data management can be applied in autonomous operation.

Beschlossene Lösung

<sw-param-content> is given the possibility of allocating the contents to a function, i.e. to reference a function (**<sw-function-ref>**).

Release notes

5.5 [kgrhierachie] Support of parameter hierarchies

Kind	proposed by	State	Prio	days	Occurred in	Geplant für
	hünerfeld	**in process** Concluded			[10] p. 77	[10] p. 77

Subject It must be possible to support parameter structures as well.

Begründung If variable structures are formed, then this must also apply for parameters. Such structures are already in use.

Beschlossene Lösung

Recursive repeat of **<sw-params>** in **<sw-param>** can also represent parameter hierarchies.

Release notes

5.6 [kgarray] Parameter contents for arrays

Kind	proposed by	State	Prio	days	Occurred in	Geplant für
bug	Hünerfeld	**in process** Concluded			[10] p. 77	[10] p. 77

Subject Specifying parameter contents for arrays must also be possible.

Begründung Otherwise incomplete

Beschlossene Lösung

<arraysize> will be introduced in **<sw-param>**, as will **<sw-array-index>** in **<sw-param-content>** .

Release notes

5.7 [sprmref] SW-param-ref semantic in parameter contents

Kind	proposed by	State	Prio	days	Occurred in	Geplant für
enh	Hünerfeld Weichel	**passed** compare semantic reference in			[10] p. 77	[10] p. 77

Subject **<sw-param-ref>** in **<sw-param-contents>** must be semantic.

Begründung
- Support of purely parameter-contents files
- A read may not have the data dictionary available

Beschlossene Lösung

Will be treated generally within the scope of semantic referencing. This will be based on a hierarchy of **<short-names>**.

Release notes

5.8 [annot] Annotation

Kind	proposed by	State	Prio	days	Occurred in	Geplant für
enh	Hünerfeld	**in process** Concluded			[10] p. 77	[10] p. 77

Subject annotations should be capable of extraction from the data dictionary or from the function definitions for subsequent display in linked systems.

Begründung Displays can be controlled in connected systems from the specification. Such systems can also return information to the documentation. These annotations must be present in:
- Variables
- Parameter structures
- Functions
- Parameter contents (summaries of functions)
- Parameter contents (individual)

Beschlossene Lösung

The annotations can be applied in **<annotations>**.

Release notes

5.9 [ndim-array] Support of multi-dimensional arrays

Kind	proposed by	State	Prio	days	Occurred in	Geplant für
enh	??	**passed** Concluded	SW 09		[10] p. 77 [10] p. 77	

Subject Software variable should also be able to be a multi-dimensional array.

Begründung Because this can be the case.

Beschlossene Lösung

The contents model for *arraysize* is *#PCDATA*. The dimensions for multi-dimensional arrays are entered as interger numbers and separated by blanks.

5.10 [unknown-SW] Handling of software parameter forms not yet considered

Kind	proposed by	State	Prio	days	Occurred in	Geplant für
enh	??	**rejected**			[10] p. 77 [10] p. 77	

Subject Introduction of a ”back door” for software parameters structures not considered up to now.

Begründung It could happen that a software parameter form is encountered that does not match the forms in use to date.

Lösungsansatz 1

Introduction of a **<ncoi>** as a further option

pro Uses existing elements.

Provides the freedom required for the acquisition.

con The output of software parameter structures is not normally meaningful on paper, but these are rather processed automatically in the development process. This would no longer be possible with **<NCOI>**.

Beschlossene Lösung

This case will not be covered for the time being. The introduction of additional parameter forms has effects on the overall process chain. It would therefore not be sufficient to leave a back door open for this.

5.11 [cleanup-sw-datatypes] SW data types should be cleaned up

Kind	proposed by	State	Prio	days	Occurred in	Geplant für
enh	??	**passed**	10		[10] p. 77	

Subject **<sw-datatype-scaling> <sw-resolution>** should be cleaned up.

Begründung These elements appear redundant, in particular with respect to **<sw-compu-method>**

Lösungsansatz 1

The general endeavor shall be made to amalgamate **<sw-data-type>** and **<sw-compu-mehtod>** .

pro Additional overlaps could be handled by this.

con
- It could be that two identical variables (i.e. being from the same type) are implemented using different conversion formulae. There would be high redundancy in the conversion formulae in this case.
- Perhaps a general data type is needed at some point in time rather than one based only on computation formulae. A pseudo-conversion formula would have to be defined in such a case.
- There can be intermediate parameters for which there are no conversion formulae foreseen. A pseudo-conversion formula would have to be defined in such a case[15].

Beschlossene Lösung

The clean-up has been carried out. The elements have been consequently separated according to early phases (**<sw-physic-types>** and **<sw-varable-implementation>**)

5.12 [prog-code] prog-code in sw-compu-method

Kind	proposed by	State	Prio	days	Occurred in	Geplant für
enh	AG-SW	**in process** Concluded	B1		[10] p. 77	

Subject Description of conversion formulae in programming language notations.

Begründung More and more processing tools support the formulation of conversion formulae as a programmiing language fragment.

Beschlossene Lösung

<prog-code > will be introduced in parallel with **<sw-compu-method-text>**.

5.13 [sw-gen-math] sw-compu-generic-math

Kind	proposed by	State	Prio	days	Occurred in	Geplant für
enh	??	**open** H. Rauleder to track this.			[10] p. 77	

Subject A semantic mathematics model will be presenetd at the SGML Europe 96. This should be used in MSR-DOC.

15

Begründung Proprietory approaches could be overcome by this.

Lösungsansatz 1

5.14 [Osek] OSEK aspects

Kind	proposed by	State	Prio	days	Occurred in	Geplant für
enh	??	**open** H. Bless maintains an element list			[10] p. 77 [10] p. 77	

Subject OSEK needs shall also be included in the MSR-DTD and the structure principles for the software.

Begründung When *OSEK*-conform developed, then it is necessary to have a structured documentation. Important here are generation parameters etc. in particular.

Lösungsansatz 1

Formulation of the description language in SGML within the scope of Medoc.

pro Is certainly conform with MSR-DOC.

An entry environment could be realized using SGML tools.

con • SGML editors are not necessarily linked with the coding environments of the software development engineers. The OSEK instructions are normally to be found in the source text for the control-unit software (e.g. *C-Editor*).

Beschlossene Lösung

5.15 [diagnose] Handling diagnostics/diagnosis status

Kind	proposed by	State	Prio	days	Occurred in	Geplant für
enh	AG	**open**			[10] p. 77	

Subject Clarification is required for the extent to which the topics of diagnostics and diagnosis status are covered by the modeling to date.

Begründung Customer not intensively occupied with diagnostics up to now.

Beschlossene Lösung

5.16 [literatur] Complete bibliography

Kind	proposed by	State	Prio	days	Occurred in	Geplant für
doc	Rauleder	**open**			[10] p. 77	

Subject The bibliography must be complete.

Begründung

Beschlossene Lösung

5.17 [security] Handle safety relevance of SW functions

Kind	proposed by	State	Prio	days	Occurred in	Geplant für
enh	AG	**open**			[10] p. 77	

Subject It shall principally be clarified how aspects of safety and safety of relevance to software functions shall be generally displayed.

Begründung Use of *msrsw.dtd* for safety-relevant systems as well.

Beschlossene Lösung

5.18 [glosssar] Concur glossary

Kind	proposed by	State	Prio	days	Occurred in	Geplant für
doc	AG	**open**	C1		[10] p. 77	

Subject The glossary must be concurred.

Begründung In order that it is correct.

Beschlossene Lösung

5.19 [schedule] Description of schedule dependencies

Kind	proposed by	State	Prio	days	Occurred in	Geplant für
enh	AG	**open**			[10] p. 77	[10] p. 77

Subject The description tools for schedule dependencies and scheduling informationen must be analyzed and included.

Begründung System and design tools increasingly make allowance for this information.

See also **other:** Chapter 5.14 [Osek] OSEK aspects p. 85

Beschlossene Lösung

Release notes

5.20 [va-sample] Extend the implementation for variables

Kind	proposed by	State	Prio	days	Occurred in	Geplant für
	RB	**in process** Concluded			[10] p. 77	

Subject Include **<sw-variable-sample-rate>** for **<sw-variable-implementation>** as well.

Begründung A minimum requirement is given for physical data, the actual realization for the implementation.

5.21 [param-basic] Introduce basic type for parameters

Kind	proposed by	State	Prio	days	Occurred in	Geplant für
	RB	**in process** Concluded			[10] p. 77	

Subject **<sw-base-type>** (C data type) should be introduced for the value of the parameter as an option for **<sw-param>**.

5.22 [msrsw] Re-name root element sw in msrsw

Kind	proposed by	State	Prio	days	Occurred in	Geplant für
enh	RB	**in process** Concluded	C1		[10] p. 77	

Subject Re-name the root element **<sw>** in **<msrsw>**.

Begründung In accordance with the general nomenclature in DTD's.

5.23 [fktvars] Optional functions variable

Kind	proposed by	State	Prio	days	Occurred in	Geplant für
enh	RB	**in process** Concluded	C1		[10] p. 77	

Subject **<sw-function-variables>** optional in **<sw-function-variant>**.

Begründung Same handling as for parameters.

Beschlossene Lösung

<sw-function-variables> will be optional.

5.24 [label] Introduce spacer for long name for param-content

Kind	proposed by	State	Prio	days	Occurred in	Geplant für
enh	RB	**in process**	A1		[10] p. 77	

Subject Introduce an optional **< label>**l for **<sw-param-contents>** and for **<sw-param-content>**. The label corresponds to the long designation for the function.

Begründung For the calibration phase, parameter contents are exchanged without the data dictionary. A long designation for the function is desirable.

Beschlossene Lösung

The element **<label>** could lead to confusion for the software development engineers with the assembler labels. Elements constituting the display of a different name, e.g. another **<long-name>** or introduced only for layout purposes e.g. to define navigators in Panorama Pro, shall be expressed by a new element. This element is termed **<auto-text>** .

5.25 [admin-in-pcont] Administrative data for parameter contents

Kind	proposed by	State	Prio	days	Occurred in	Geplant für
	RB	**passed**				

Subject Introduce **<admin-data>** for parameter contents.

Begründung Different versions of parameter contents are created in the calibration phase.

5.26 [fref-in-pcont] Function references in parameter contents

Kind	proposed by	State	Prio	days	Occurred in	Geplant für
	RB	**open**				

Subject Introduction of **<sw-function-ref>** in**<sw-param-content>**.

Begründung Entities are exchanged with function-related parameter contents during the calibration phase. This exchange is project-overriding and must therefore be without data dictionary.

5.27 [list-adr] Lists of addressing schemes, storing schemes and code syntaxes.

Kind	proposed by	State	Prio	days	Occurred in	Geplant für
	RB	**in process**				

Subject Introduction of **<sw-addressing-methods>**, **<sw-code-syntaxes>** and **<sw-record-layouts>** in **<sw-data-dictionary>**.

Begründung Standardization and description of the objects, preparation for code generation.

5.28 [glosopt] Glossary optional

Kind	proposed by	State	Prio	days	Occurred in	Geplant für
	RB	**open**				

Subject .

Begründung

5.29 [units] Units of measure

Kind	proposed by	State	Prio	days	Occurred in	Geplant für
	RB	**open** open				

Subject Introduction of units of measure in the **< sw-data-dictionary>** formal referencing.

Begründung The units of measure are adirect part of the usability of physical parameters and must therefore be handled formally. Standardization of the units of measure.

Lösungsansatz 1

The inits of measure used in the data dictionary are listed using long and short designations. If this is not an SI unit, then the corresponding SI unit is referenced and the conversion to or from the SI unit is given. The reference and conversion to a SI unit is not applicable for a SI unit. The element **<sw-unit>** will be substituted by **<sw-unit-ref>**. Refer to the example givne in the following.

5.30 [variables] Variable

Kind	proposed by	State	Prio	days	Occurred in	Geplant für
	RB	**open** open				

Subject Modification of the contents model for variables **<sw-function-variables>**.

Begründung A distinction must be made for variables between the definition of a variable and the access to a variable.

Lösungsansatz 1

Variables that are defined by a function, are listed in **<sw-function-export-variables>**. If the variable is only used by the function itself in **<sw-function-local-variables>**. Variables that are used by the function, or are read or written yet not defined by the function, are listed in **<sw-function-import-variables>**.

A distinction is made with regard to the access to variables between reading (**<sw-variables-read>**), writing (**<sw-variables-write>**) and reading + writing (**<sw-variables-readwrite>**).

App. A Notes on processing

All software functions shall be listed for each software function that calls this up. This relationship is illustrated in the above structure by the software variables and their directung.

Minimum and maximum values are given in **<sw-limits>** for the coded (control-unit-internal) and the physical values. A semantics checker shall check the consistency of these values. The one can be computed from the other in a subsequent process if necessary provided this is capable of evaluating the conversion formulae.

App. B Explanation of the Near&Far symbols

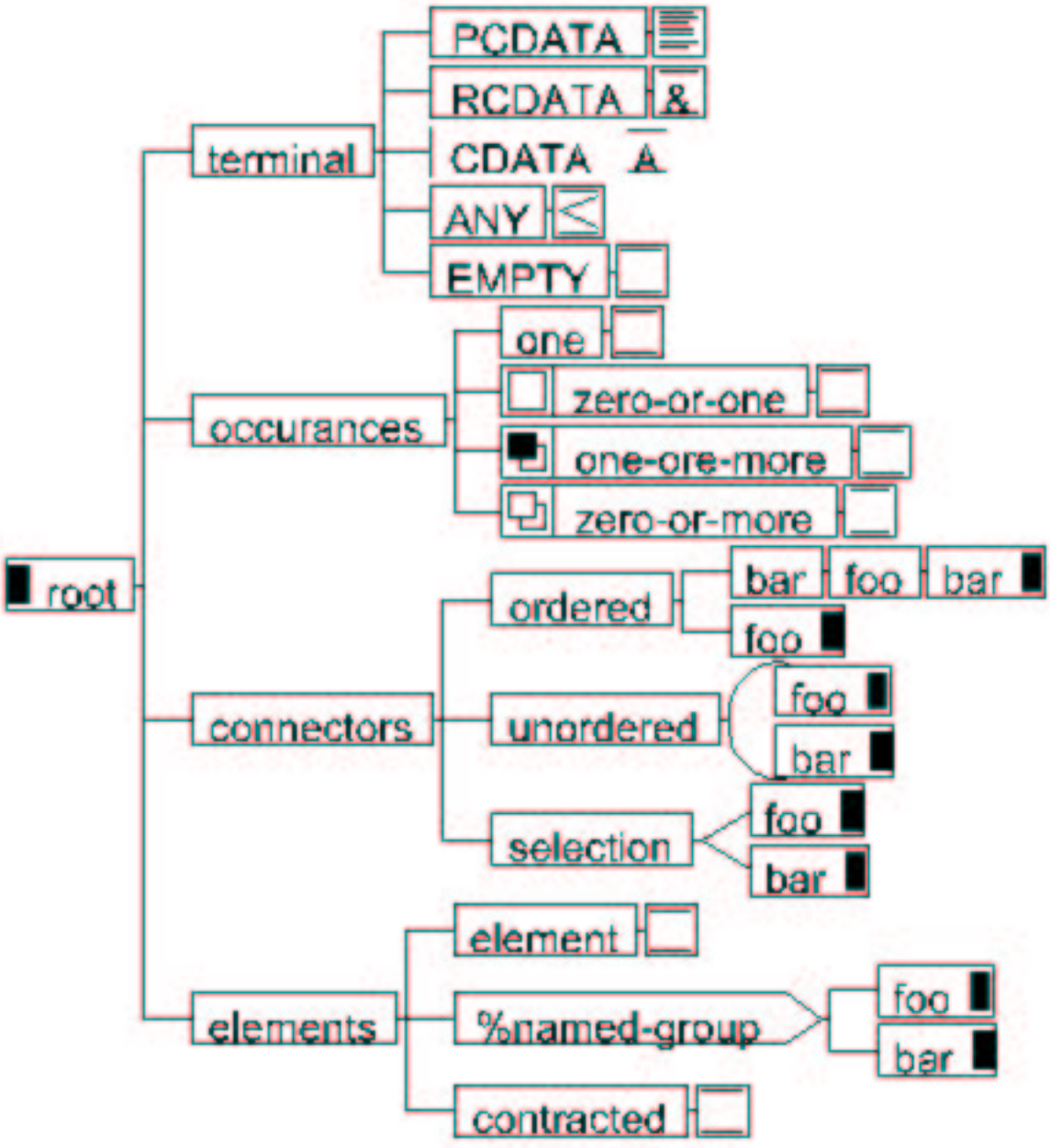

Figure 22: NearFar symbols

App. C Glossary

The glossary contains central terms. The explanations make reference in come cases to terminology used used within the scope of *MSRSYS.DTD*.

add-info, add-spec Elements within the *MSRSYS.DTD* that, as well as given topics/titles, enable a freely structured documentation of additional specifications (add-spec) and additional information (add-info) for objects as well as the prescribed description structure (add-info).

ASAP The ASAP interfaces have been agreed by the ”Study Group for Standardization of Application Systems (*ASAP*)”. Members of this study group are the German Automobile Manufacturers and companies from the supplier industry.

Data Contents of software parameters (parameter contents)

Data status Defined contents of the Software-Parameter (parameter contents) at a certain point in time. The data status is identified by a version number.

Decomposition Breakdown of a software function into sub-functions.

Function Function is understood as being a certain requirement or property of the overall system that can be realized either by hardware, software or a combination of both.

Function analysis The function analysis contains the description of the contexts, the functions (hierarchical, including diagnostic functions, safety functions etc.) and the information flows (informal). A formal definition, an informal description, as well as application notes and customer-servicing notes are documented for each function.

Hardware function Function within the behavior that has been realized in the hardware.

Hardware module Constituent of a hardware component (Part-Type); in certain circumstances the component itself.

Integration Phase in which the individual modules are joined. The integration can be carried out in the testing and/or target environment.

Logical function A function within the behavioral description.

MSRDOCDTD Exchange structure defined within the scope of the project in the standard SGML (ISO 8879).

OSEK OSEK stands for: ”Open Systems and the Corresponding Interfaces for Automotive Electronics”. OSEK was established in May 1993 by companies in the German automobile industry. Founding partners were: BMW, Bosch, Daimler-Benz, Opel, Siemens and VW. Project coordinator is the IIT at the University of Karlsruhe (Institue of Industrial Information Sytems). Peugeot and Renault joined the project with the French project VDX-approach (Vehicle Distribute eXecutive). The German and the French projects were merged into OSEK/VDX. The topics of operating systems, communication and net work management are covered in OSEK/VDX. Refer to *[External Document: Proceedings of the 1st International Workshop on Open Systems in Automotive Networks / Document number: ISBN 3-00-000259-6 / Date: 9.10.1995 / Publisher: IIIT (Institute for Industrial Information Technology), University of Karlsruhe / URL: / Relevant Position:]*

Part-type, part Central elements within the MSRDOCDTD for the description of the hardware-component structure.

Processing The processing describes (in the document ”Processing Guide for the MSR-DOCDTD”) the processing and layout definitions for the MSRDOCDTD. I.e. a description is given of how an SGML entity of this DTD is converted by a formatter into a technical layout for printing a document.

Program Part of the Definition Software p. 94software in the control unit that is capable of running. Data are handled separately, i.e. not as part of the program as these are developed in a separate process (compare Topic 1.2.8 Application (calibration) p. 12).

Programmability Characteristic of a component (e.g. a control unit) that describes the form in whhich and the extent to which the component can be programmed (end-of-line, off-line and field programmability).

Program status Executable code in the control unit without data (parameters etc.)

View The *MSRSYS.DTD* can take data in two views (views): ”requirements” and ”product-spec”.

Software Software capable of running, i.e. program and data in one control unit.

Software function A function realized in the software Definition Function p. 93in a control unit.

Software module A unit that can be combined by a compiler that can also comprise several software functions.

Software status Combination of Definition Program status p. 94program status and Definition Data status p. 93data status

System analysis In the system analysis, the system is described at a logic level. The logical function model thereby given can be simulated.

System design The physical function model is derived form the logical function model in the system design.

App. D Bibliography

Table :

Kurzbezeichnung	Bezeichnung	Stand	Index
OSEK/VDX, ISBN 3-00-000259-6	Proceedings of the 1st International Workshop on Open Systems in Automotive Networks	October 9, 1995	

1. *[External Document: Proceedings of the 1st International Workshop on Open Systems in Automotive Networks / Document number: ISBN 3-00-000259-6 / Date: October 9,1995 / URL: / Relevant Position:]*

App. E General structures

General structures that have been defined in the MSRSYS.DTD and that are also used in the software documentation or in that referenced by the software documentation are presented graphically in this annex.

Note that for historical reasons this chapter has some overlap with Topic App. F Basic Structures of the MSR Application Profile p. 104.

App. E.1 Administrative data

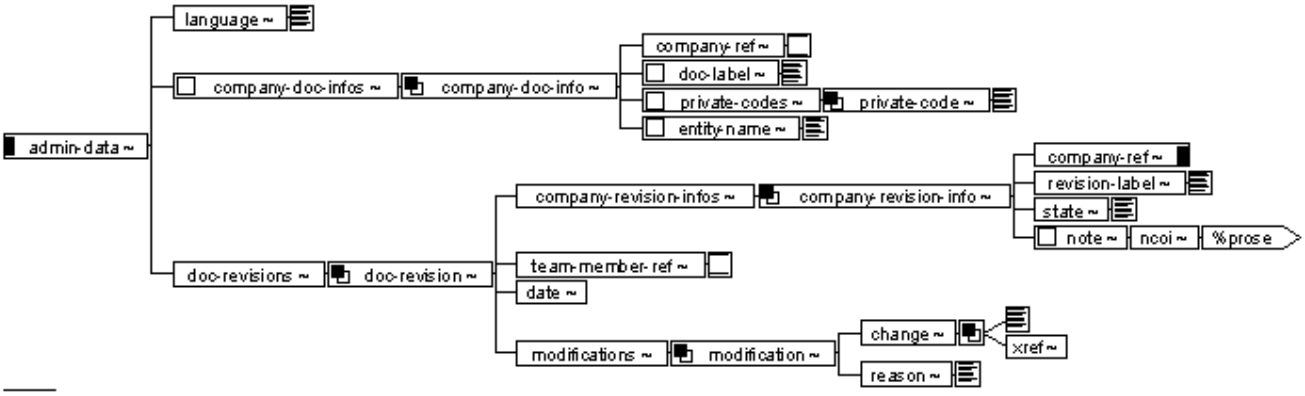

Figure 23: Administrative data

App. E.2 Parameter model

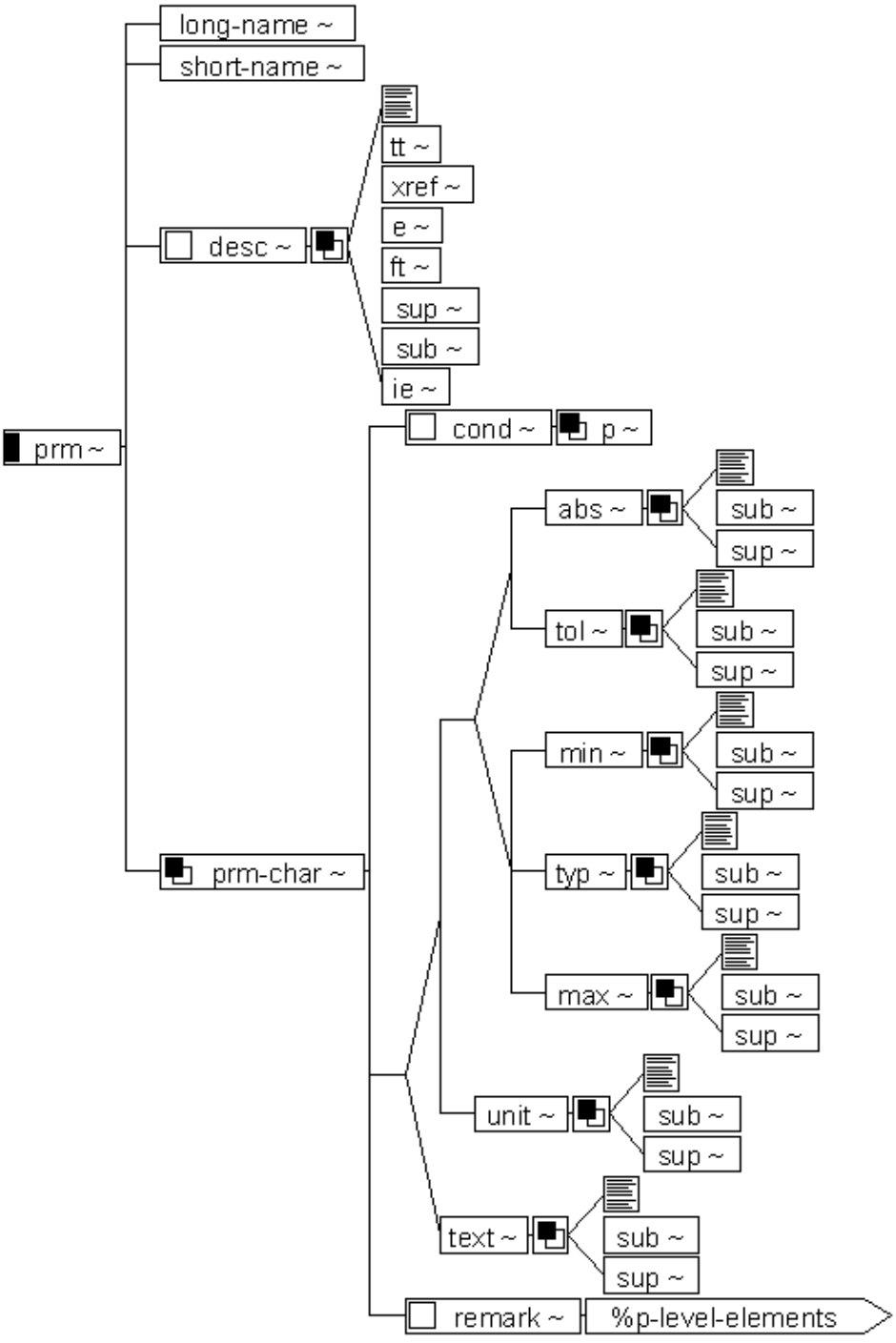

Figure 24: Parameter model

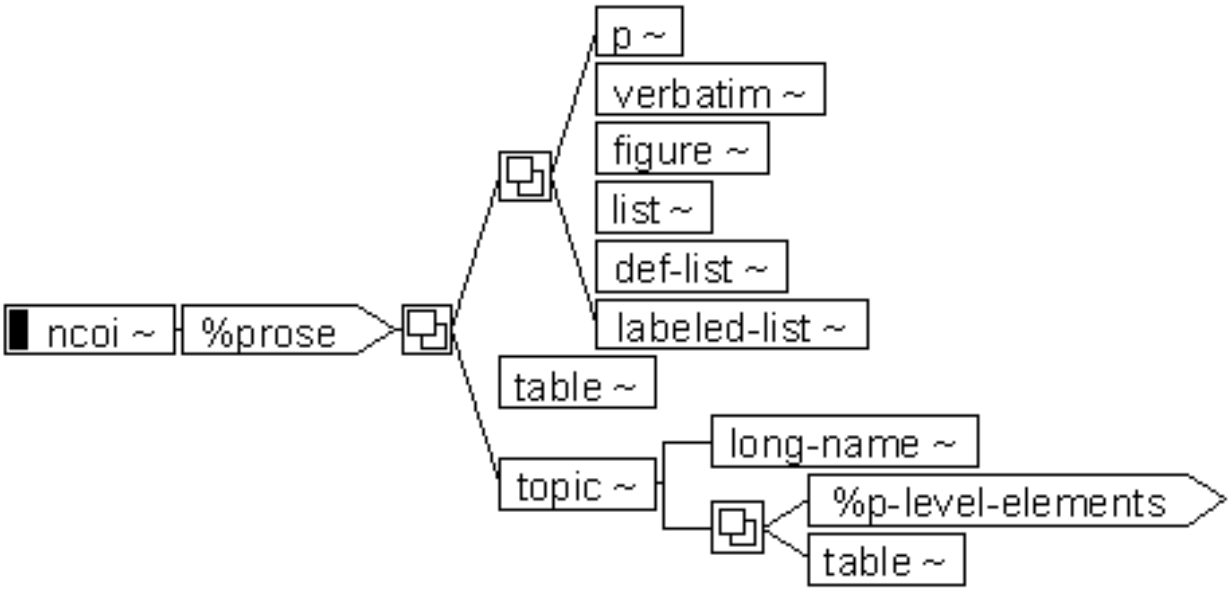

Figure 25: ncoi

App. E.3 Annotations

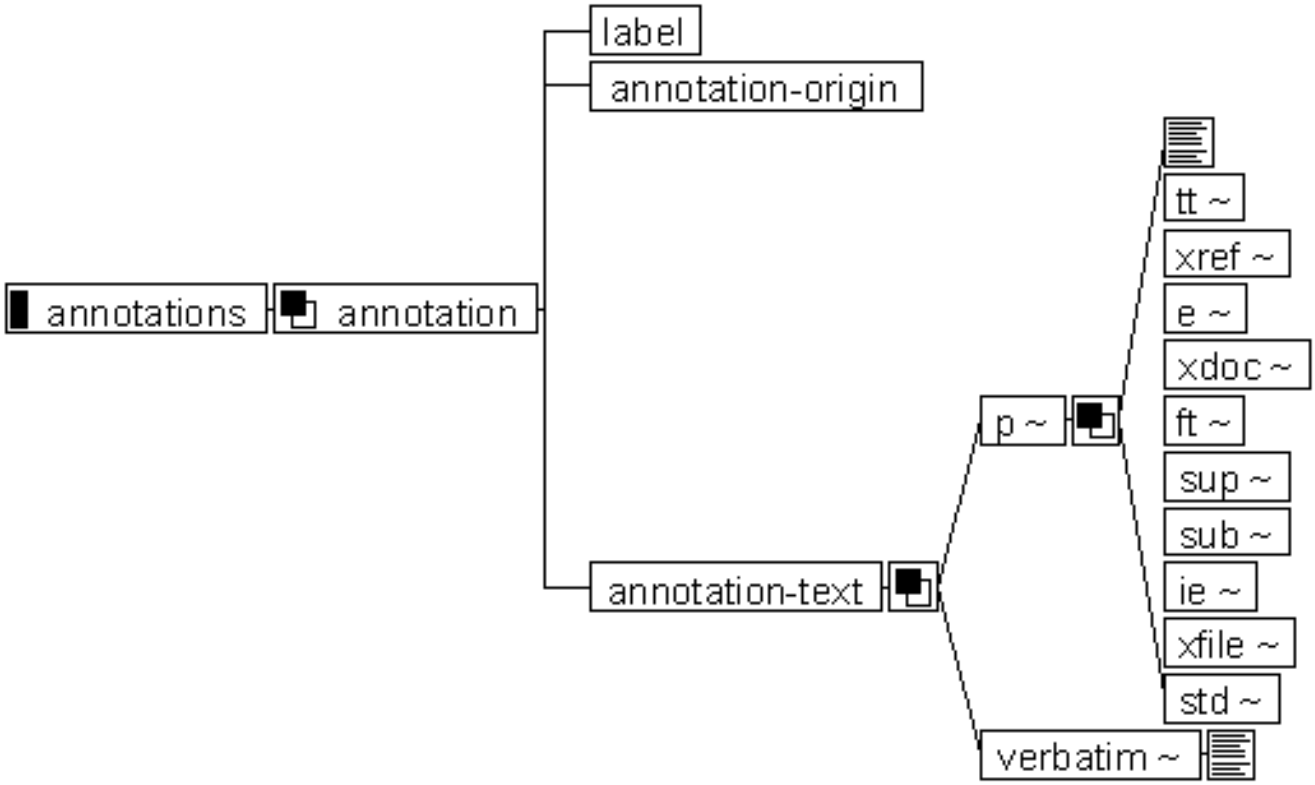

Figure 26: Annotations

App. E.4 Additional information

Permits the acquistion of additional information beyond the detailed information foreseen in the DTD.

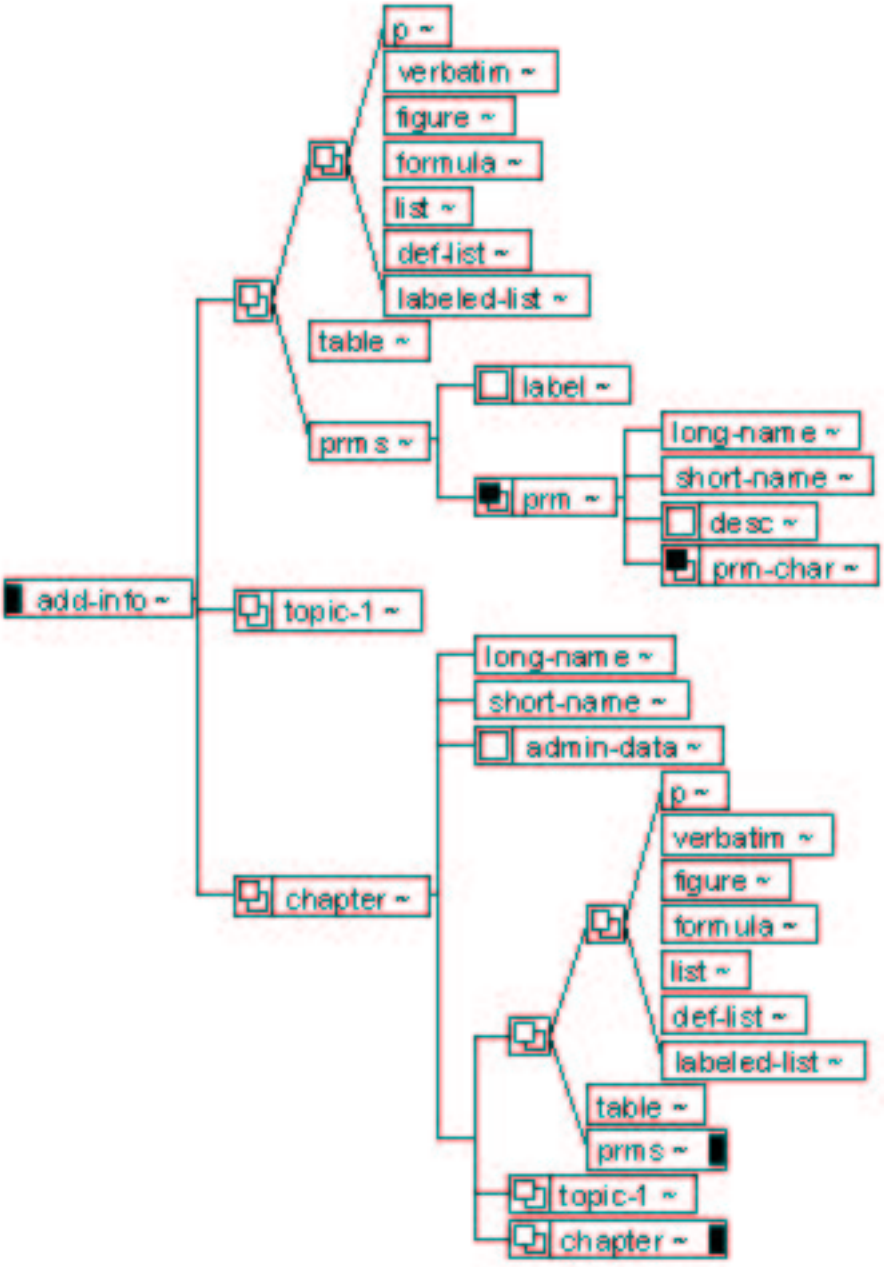

Figure 27: add-info

App. E.5 Topics

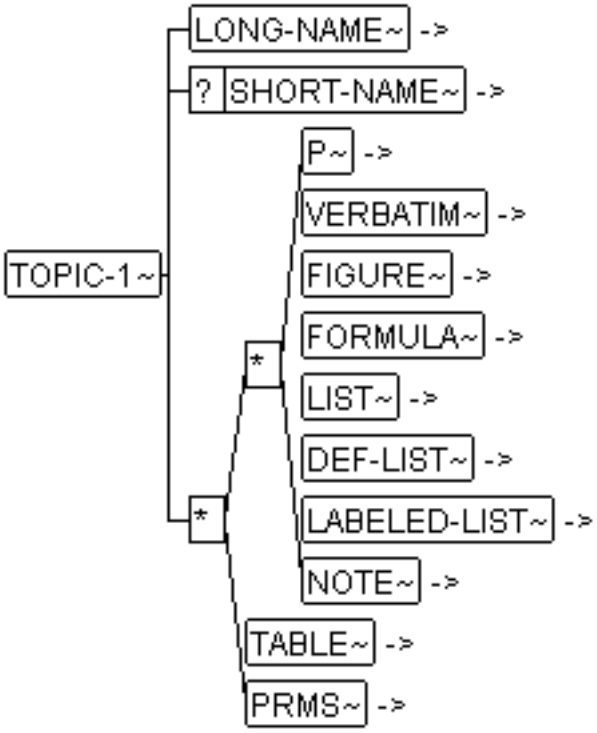

Figure 28: topic-1

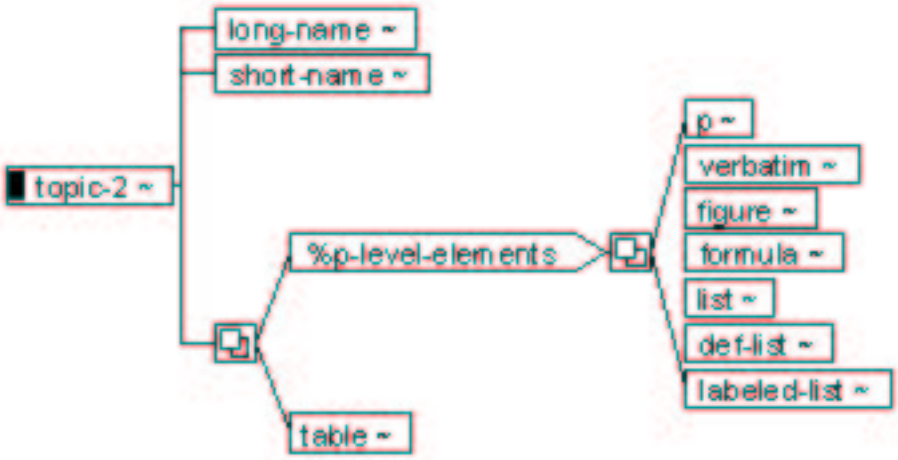

Figure 29: topic-2

App. E.6 Names list

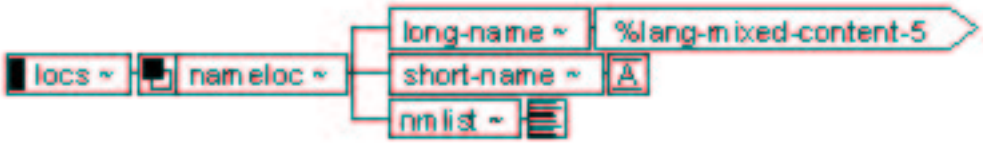

Figure 30: locs

App. E.7 Introduction

A short introduction for the object in question can be given in the **<introduction>**. Detailed descriptions should be included in **<add-spec>**.

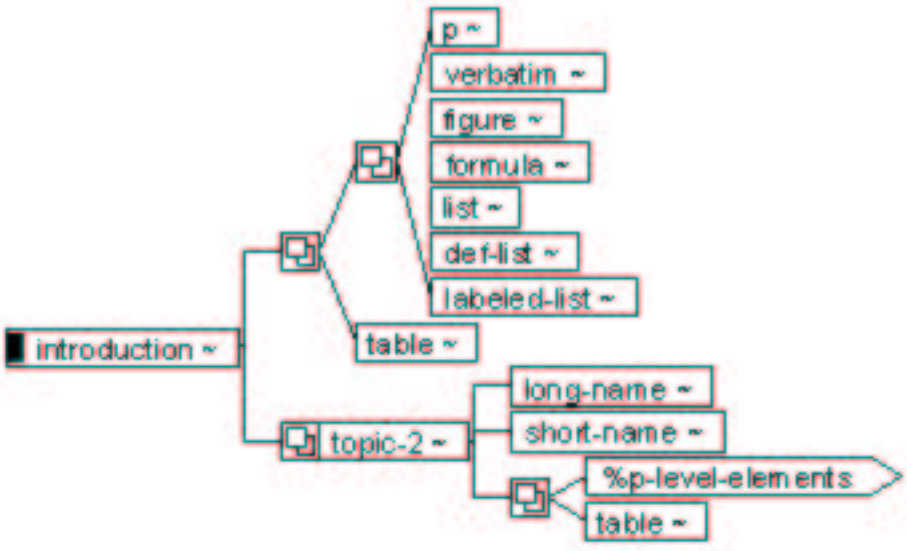

Figure 31: intro

App. E.8 SI units

STEP (ISO/DIS 10303-41, S96 ff) is supported with regard to SU units by the following seven basic units:

- length (meter),
- mass (gram),
- time (second),
- electric_current (ampere),
- thermodynamic_temperature (kelvin),
- amount_of_substance (mole; quantity; compared to the number of atoms in 0.012 kilogram of carbon 12),
- luminous_intensity (candela).

App. E.9 Multiple-language documents

The software DTD can be used either in one language or in several languages by means of a configuration file. The multiple language feature has not been introduced for all elements with PCDATA content. Elements that are used for referencing must however be unambiguous beyond all language barriers. Example **<short-name>**. An example for the multiple language feature is the **<long-name>**, refer to the following diagram and the following example.

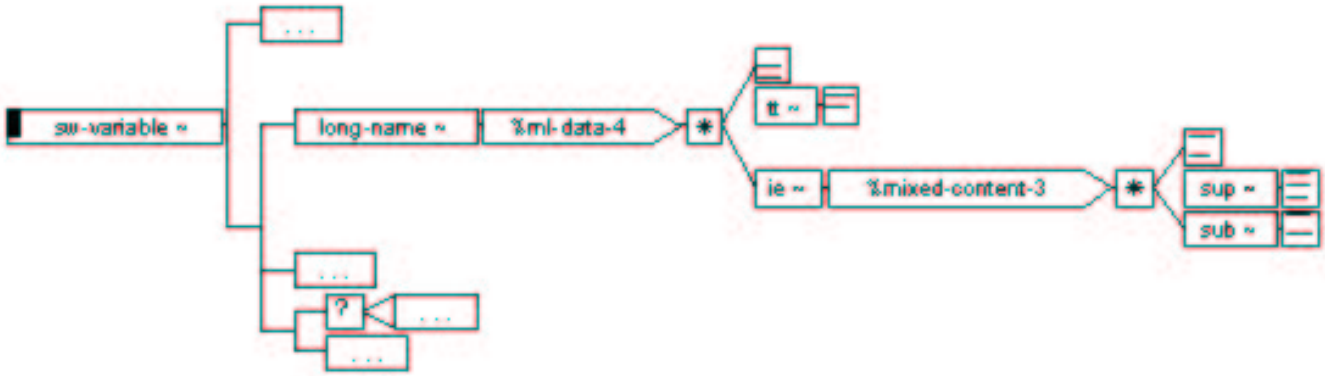

Figure 32: example long-name

App. F Basic Structures of the MSR Application Profile

All MSR DTDs are using some common data structures. These operating models are described in this chapter.

App. F.1 Not Content Orientated Information (ncoi)

<ncoi-1> contains all basic descriptive elements. There are also elements like **<chapter>** or **<fail-save-concept>** in the *MSRSYS DTD* which have the same content model as **<ncoi-1>**.

The figure below illustrates the structure of **<ncoi-1>**.

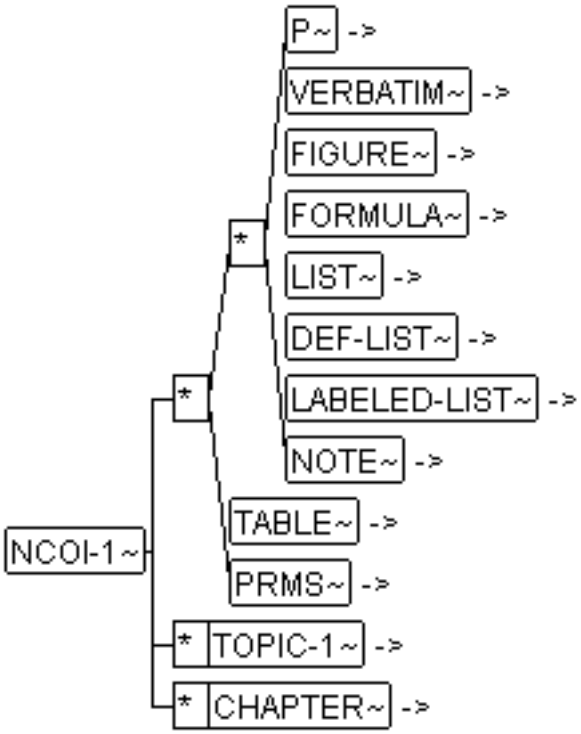

Figure 33: Structure of <ncoi-1>

There also are two weaker ncoi models (*ncoi-2* and *ncoi-3*) with lesser elements than **<ncoi-1>** . *ncoi-2* has no **<chapters>**. *ncoi-3* has also no chapters and futhermore another ”topic” model without **<prms>**.

The components of ncoi [17] are interchangeable between all MSR DTDs[18] without any changes.

App. F.1.1 Chapter

<chapter> is a sequence of paragraph level elements mixed with **<chapter>**. **<chapter>**s can be nested as deeply as required. It is up to the author to make sure, that the nesting of the chapters can be handled by the processing system[19].

17

18

19

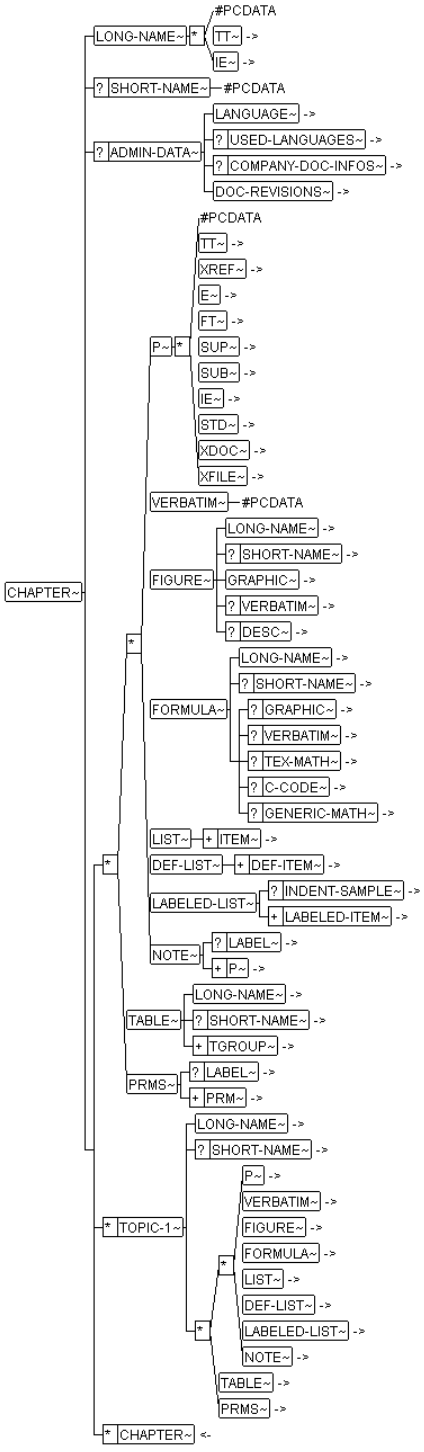

Figure 34: chapter content model

One advantage of using **<chapter>** for all levels[20] is the option to move a chapter using *cut & paste* to any place in the document at any level.

20

App. F.1.2 Topic

Use **<topic-1>** or **<topic-2>** to create bridge titles instead of one line paragraphs with entirely emphasized contents. Note that these elements can be referenced by **<xref>**. In difference to **<topic-1>**, **<topic-2>** has no **<prms>**.

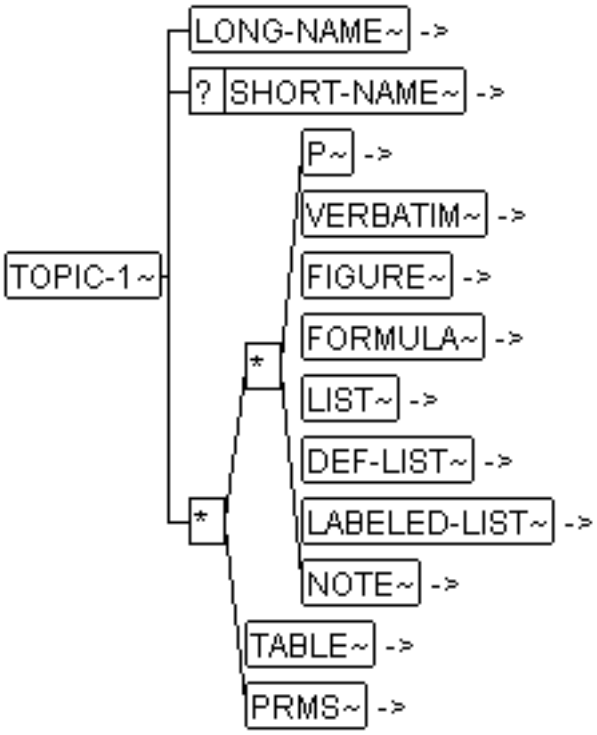

Figure 35: Structure of <topic-1>

App. F.1.3 Paragraph Level Elements

”Paragraph level elements” are elements which occur on the same level as **<p>**.

The user should first look for an appropriate one among the available elements before trying to simulate things by using inadequate elements. In that respect the following hints are given:

<p>	Paragraph
<verbatim>	Preformatted text which is usually set in monospaced font. Tabs, line spaces and carrige returns are considered. Use **<verbatim>** to print program listings etc. It can even be used to show simple diagrams.
<figure>	See chapter Topic App. F.1.3.2 Figure p. 109.
<formula>	See chapter Company App. F.1.3.3 Formula p. 110.
<list>	A ordered or unordered list of items. For an unordered set of items, use **<list type=”unnumbered”>**. For a ordered list of items use **<list type=”numbered”>** [21].

21

<def-list> Use **<def-list>** to create definition lists which might be collected into an overall definition list or a glossary. In this case **<labeled-list>** might lead to the same rendition but has no information about the fact that terms are defined[22].

<labeled-list> Use**<labeled-list>** to create explanations or even bridge titles for very short topics instead of bulleted lists with emphasized initial words. See also Topic App. F.1.3.1 Labeled List p. 107

Use **<labeled-list>** instead of two column tables if the first column cells almost contain one word.

<note> See chapter Topic App. F.1.3.4 Note p. 111

22

App. F.1.3.1 Labeled List

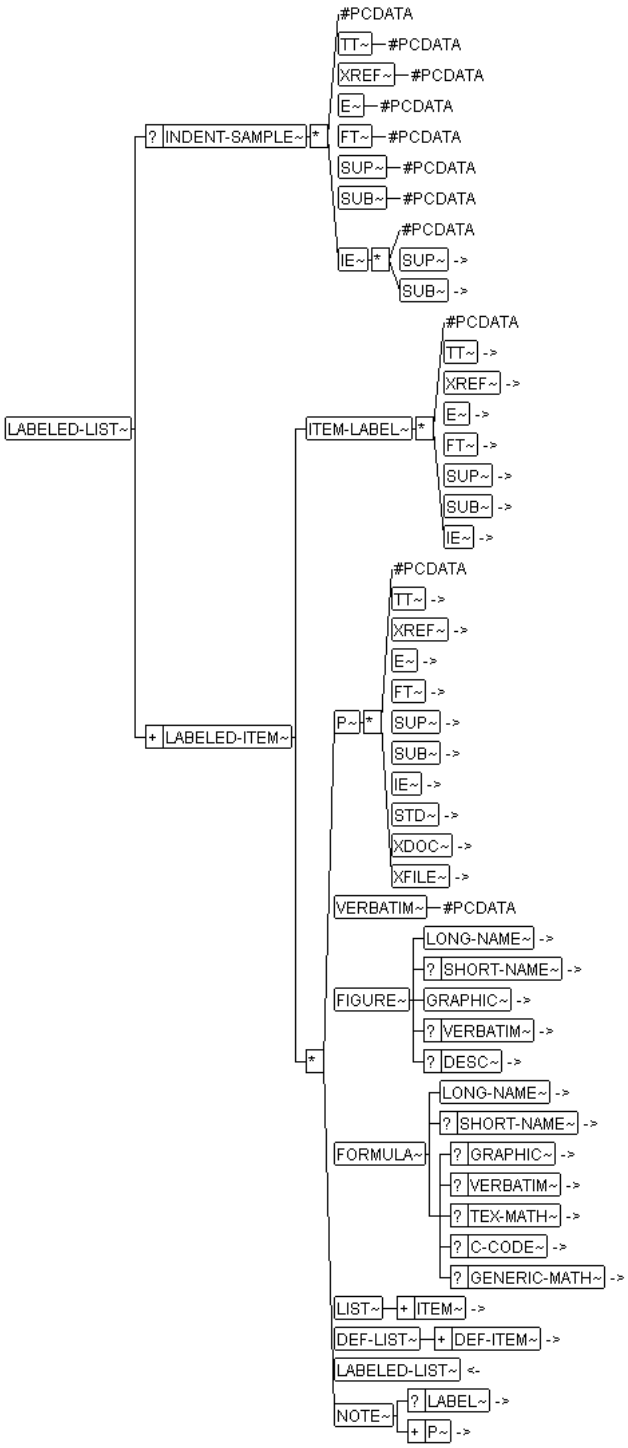

Figure 36: Structure of <labeled list>

<labeled-list> is one of the most powerful elements. If possible it is rendered as a label followed by the item body:

The indentation is determined by the *rendition system* which should take into account the biggest **<item-label>**.

Sometimes the author wants some influence to the indentation. For this respect **<indent-sample>** can receive any content which is used by the *rendition system* as a sample which must be rendered and measured to determine the indentation.

The attribute **[item-label-pos]** defines how the **<item-label>** should be handled. The default value of the attribute is **[item-label-pos]**=”no-newline”. If an **<item-label>** is wider than **<indent-sample>** the most general case is to start the item body in a new line if necessary(**[item-label-pos]** =”newline-if-necessary”):

If the attribute has the value **[item-label-pos]**=”newline” the item-body starts generally in a new line.

Note that **<indent-sample>** can be used to adjust the indentation if there are multiple **<labeled-list>**s which should have the same indentation.

App. F.1.3.2 Figure

<figure> is used to insert graphics into the document. A figure can be defined in three different ways.

1. as a real **<graphic>**
2. as an ASCII graphic (**<verbatim>**)
3. as a pure textual description (**<desc>**) of the graphic [23]

The treatment of the graphic is determined by the attributes of **<graphic>**:

Do not enter annotating text to **<long-name>**in**<figure>** or **<table>**(like *Figure 1: ...*). This embellishment is the task of the processing system, not of the author. If the author adds these things, they will be there twice since the *rendition system* will add it again.

[category] Denotes the category of the graphic. This information can be used to generate more specific list of figures

[filename] Denotes the system filename where the *rendition system* can find the graphic. This is not necessarily the final format. It is up to the *rendition system* to locate the graphic in the company specific environment, to change the file extension to get the appropriate graphic representation.

The type of this attribute can be turned from *SDATA* to*ENTITY* in the DTD file in order to allow *SGML tool*s access to the file using its *entity manager*. In this case, the entity name should be chosen in the style of a filename (e.g. *crpctmt.wmf*)[24].

[fit]

0 figure is placed in original size. If it does not fit on the page or the available space, it is scaled down.

1 the figure is scaled up or down to fit the page as possible. This value will be ignored if **[width]** or **[height]** is specified in addition.

2 the figure is rotated counterclockwise by 90° if it is landscape and is wider than the actual text area. It is scaled down to the page size if it does not fit otherwise. This value will be ignored if **[width]** or **[height]** is specified in addition.

23

24

3 the figure is always rotated counterclockwise by 90°. If it does not fit on the page it will be scaled down. If **[width]** or **[height]** is specified in addition, the figure will be rotated and then scaled to the specified values.

4 the figure is always rotated counterclockwise by 90° and scaled up or down for best fit on the page. This value will be ignored if **[width]** or **[height]** is specified in addition.

[height] If this attribute has a value, the figure will be scaled to the defined height which is a real value with dimensions (e.g. ”10cm”, ”150mm”, ”12.5in”). If also **[width]**is specified the figure will be distorted. This value always specifies the width of the ”figure box” on the page after possible scaling/rotating.

[notation] This attribute specifies the format of the graphic file if used by an *SGML Application* supporting notations.

[scale] If this attribute receives a value, the figure will be scaled by the given factor which must be a signed real number. Numbers greater 1 increase the size of the figure, values less than 1 make the figure smaller. For example with *scale=”0.5”* the a figure of the size 10x10 cm will appear as 5*5cm.

[width] If this attribute has a value, the figure will be scaled to the defined width which is a real value with dimensions (e.g. ”10cm”, ”150mm”, ”12.5in”). If also **[height]**is specified the figure will be distorted. This value always specifies the width of the ”figure box” on the page after possible scaling/rotating.

The scaling attribute precedence is:

- **[scale]** has precedence over all
- **[fit]**has precedence over **[width]**and/or **[height]**

App. F.1.3.3 Formula

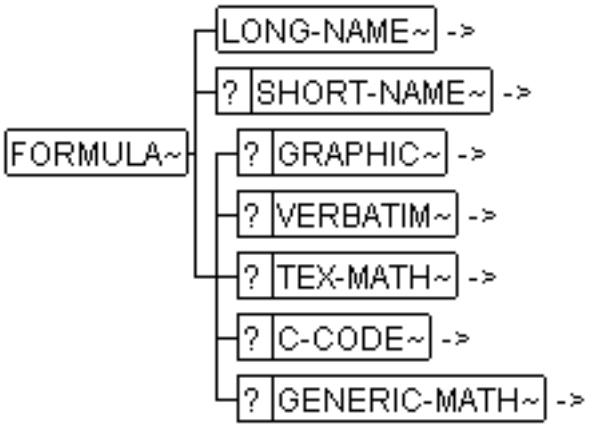

Figure 37: Structure of <figure>

A formula can be described in five different ways which can exist parallel. These are:

<graphic> A formula prerendered as a figure.

<verbatim> A simple ASCII formula.

<tex-math> A *TeX* math formula which can be processed by a *TeX* or *LaTeX* processor.

<c-code> A formula which is defined as c-code.

<generic-math> This element is intended for the definition of semantic math descriptions which can be processed by math processors. Actually there is no recommentation

for the language of the formula specification or usage of a special rendering system.

It is up to the rendering system which of the available representations is used.

App. F.1.3.4 Note

A note is an object to express a combination of an icon with descriptive text and an additional label. This is useful for things like cautions, hints etc..

The attribute **[notetype]** defines the note category. The following values are available:

- caution
- hint
- tip
- instruction
- exercise
- other

If the attribute **[notetype]** has a value of ”other” the user has to specify a own type within the attribute **[user-defined-type]**.

A formatter has to place the right icon before the descriptive text according to the value of **[notetype]** or **[user-defined-type]**. The optional **<label >** can be used to define a title of the note.

App. F.1.4 Character Level Elements

Character level elements can occur within element like **<p>**, **<item-label>**. There are rendition oriented elements like **<e>** (emphasis), **<sub>** as well as semantically oriented Elements as **<tt>** (technical term) or **<std>**(referring to an external standard). It is highly recommended to use rather semantically oriented elements than rendition oriented ones.

App. F.1.4.1 Rendition Oriented Character Level Elements

The rendition oriented character level elements are:

<e> Emphasizes the text. The attribute **[type]**determines the rendition style.

<sub> Subscript - places the contents with smaller font below the base line.

<sup> Superscript - places the contents with smaller font above the base line.

App. F.1.4.2

Semantically Oriented Character Level Elements

Table 7: semantically oriented character level elements

Element	use for	example
<tt>	Use for any technical term. The type of that term is determined by the attribute **[type]** [25]. This element could be treated as a back-door to markup information which is not totally semantic. The *SGML processing system* can generate list of technical terms which makes it easier to find misspellings and other errors.	This is an SGML tag **<tt type=sgmltag>** we can collect all **<tt>**s
<xref>	Used to create links in the document. The role of the target is determined by the attribute **[id-class]** receiving the value of the target's fixed attribute **[f-id-class]**. The attributes of **<xref>** should be maintained by the *authoring system*.	
<xdoc>	Used to refer to an external document which usually is not available electronically. **<xdoc>** receives a set of elements characterizing the external document	Details to architectural forms can be found in *[External Document: / URL: / Relevant Position:]*.
<ft>	Is used to create footnotes	Footnotes seem to be small and unimportant[26].
<ie>	creates index entries	It is not necessary to put SGML tags into the Index, since the processing for *MSRREP.DTD* recommends to create a list of SGML tags automatically.
<xfile>	Is used to create pointers to external files which are not to be processed by the native *SGML processing system*. The contents of **<xfile>** can be used to connect to appropriate systems in later steps of the processing chain.	The schematic is found in *[External FILE: MOTRONIC wiring diagram / URL: motronic.asc]*
<std>	Is used to refer to a standard.	SGML is defined in *[/ Standard: Information Processing - Text and Office Information Systems / Subtitle: Standard Generalized Markup Language / State: standard / Date: 1986 / URL: / Relevant Position: entire document]*

25

26

Table 8: usage of technical terms

type	use for	example
<tt type=sgmltag>	Used to describe SGML tags including attributes	To describe SGML tags use **<tt type=sgmltag>**.
<tt type=sgml-attribute>	Used to describe SGML attributes outside of tags	The sgmltag is denoted by the attribute **[type]**
<tt type=tool>	Used to mention tools used for example in a process. This can be software, as well as mechanical tools. The tool should be specified by its nature not by the specific product name.	SGML files are processed using an *SGML processing system*.
<tt type=product>	Used to mention specific products.	This document is processed using *MetaMorphosis*.
<tt type=variable>	Used to mention a variable informally. This is used to control the rendition as well as for generating variable lists. This is mainly for informal reports[27]. It is also possible to use this to mention a variable in the ECU software if no **<sw-data-dictionary>** is part of the document. In a later process step, this can be turned over to a formal **<xref>**	The initialization is controlled by the environment variable *MMRC*. The initial advanced angle is calculated based on *N*and*TL*.
<tt type=state>	Used to mention a state for example of a process.	The documents must at least be *revised* if they are submitted to the customer.
<tt type=prm>	Used to mention a state for example of a process. It is also possible to use this to mention a calibration parameter in the ECU software if no **<sw-data-dictionary>** is part of the document. In a later process step, this can be turned over to a formal **<xref>**	The initial advanced angle is calculated using a lookup table *KFZW*.
<tt type=material>	Used to mention material.	Furniture is usually made of *wood* and *plastic*
<tt type=control-element>	Used to mention control elements of tools like push-buttons, menu items, switches etc. as well as keyboard keys.	To finish the dialog push the *OK* button.
<tt type=code>	Used to markup program in line code sequences	*MetaMorphosis* is invoked with *mm crp.sgm*
<tt type=organisation>	Used to markup the name of an organization.	SGML is standardized by *ISO*

27

Table 8 (Cont.): usage of technical terms

type	use for	example
<tt type=other>	Used to mention a special term which does not fit to the other types. This is a back-door for the definition of user defined types. They have to be specified within the attribute **[user-defined-type]**. A formatter uses this user defined type only if **[type=other]**.	This is a *thing* not covered by **<tt>**.

Table 9: sub-elements for xdoc and xfile

Element	use for	example
<number>	Used to markup the document ISBN resp. the standard number	ISBN 0-7923-9432-1
<state>	Used to markup the state of the referred document resp. standard.	released
<date>	Used to markup the release date of the referred document resp. standard. This could be expressed as year only, if the exact date is not known.	1994
<publisher>	Markup the publisher of the document or the standard. This can be the author as well as the publishing organization.	Steven J. DeRose and David G. Durand / Kluwer Academic Publishers
<position>	Markup the relevant position in the referenced document resp. standard.	Chapter 5.2 - Architectural forms
<subtitle>	Used to markup the subtitle of the referenced document or standard if there is one.	HyTime
<short-name>	Used to markup the document identifier	SGML
<long-name>	Used to markup the main title of the referenced object.	Making Hypermedia work
<file>	Used to markup the file access information. This is intended to be processed by external systems.	*[External FILE: MOTRONIC wiring diagram / URL: motronic.asc]*

App. F.1.5 Table

<table> is implemented as *CALS table* (see *[External Document: CALS table spec / URL: / Relevant Position:]* at www.oasis.org). Capturing these kind of tables must be supported by the *SGML editor*, so only some hints are given here:

- *CALS table*s consist of mainly three parts within **<tgroup>**: **<thead>**, **<tbody>**, **<tfoot>**.
- Each part is made of **<row>**s of **<entry>**s. Each of these elements have attributes to control the layout of the table.
- **<tgroup>** also receives a set of **<colspec>**s having information about the table columns.
- One of the major problems if *CALS table*s do not work is, that the amount of **<colspec>** elements and **<entry>** does not match the value of the attribute **[columns]** in **<tgroup>**.
- Within **<entry>** most of the paragraph level elements are allowed.

Note It is highly recommended to insert **<thead>**. This creates a table heading which is repeated on each page, if a pagebreak falls into the table.

App. F.1.6 Parameter tables

User Definable Parameters

For structured documentation of individual numerical and/or alpha-numerical requirements, so-called parameters are available. They have the following structure:

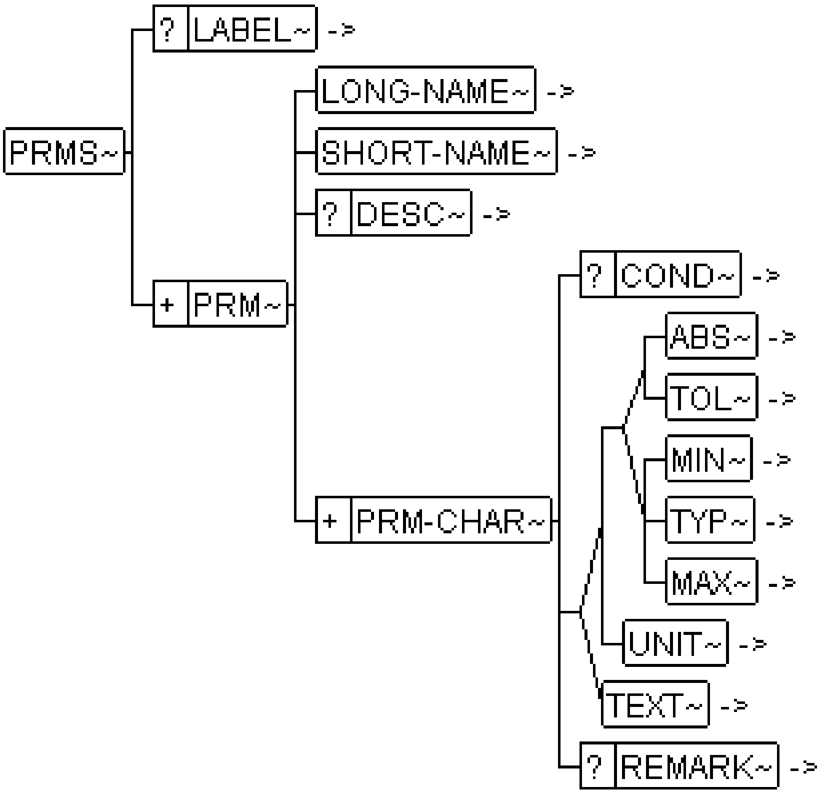

Figure 38: Structure of prms

* parameter * long-name

* short-name

* description

((* absolute value and tolerance[28] or

* minimum, typical, maximum value[29])

* unit) or

* text[30]

28

29

30

The following representation example can be drawn from this structure:

<short-name> UB

Table 10: Parameter structure

		<prm-char>						
Element: **<long-name>**	Element: **<short-name>**	Element: **<min>**	Element: **<typ>**	Element: **<max>**	Element: **<abs>**	Element: **<unit>**	Element: **<tol>**	
Operating voltage	U_B	10,8		14,2		V		
					13,5	V	5 %	
Colour of housing		red, green and blue						
Function state		active						

- Defined Parameters

  There are many pre-defined parameters in the MSR DOC DTD. The only difference between them and user defined parameters is that the designation (long-name element) of the parameter is pre-defined.

App. F.2 Predefined Document Structure

The automotive systems to be described with the help of this DTD possess very different specifications. Because of this, the specification of a particular topic, e.g. ”acoustic characteristics” might not make sense or might only become necessary later on, depending on the project.

This situation was also taken into account in the DTD through the elements ”**<na>**” (not applicable), ”**<tbd>**” (to be defined) and ”**<tbr>**” (to be resolved) as shown in Figure 39 Principles of information acquisition p. 116. This is a mechanism is located at each element on chapter level and works like a check list. A user has to make a statement for each topic.

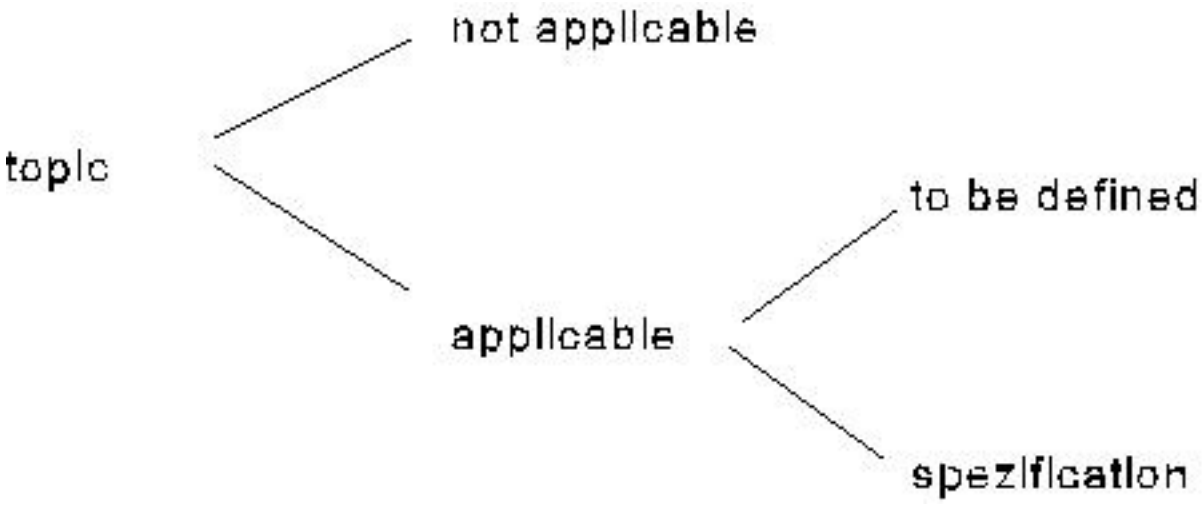

Figure 39: Principles of information acquisition

If a certain topic is not applicable it has to be marked with **<na>**. If it is applicable it can be marked with either with **<tbd>** which indicates that someone has to do a job, or it can be marked with **<tbr>** which indicates that a specification already exists but it hasn’t yet been included, or a detail specification can be defined.

The elements **<na>**and**<tbr>** can be described with a short description. Within the element **<tbd>** the persons responsible for the definitions that have to been made can be specified with **<team-member-ref>**s. The schedule for the definitions can be defined within **<schedule>**.

App. F.3 Project Data

Registering and documenting development of a MSR system is project-oriented, whereby there may be several versions of the product data of a project. The projects can be combined with the help of main projects. This can be defined within **<overall-project>** by a **<label>** an a short description in **<desc>**. Each project is assigned to a maximum of one main project.

The documentation and continuation of project phases occurs in versions. We differentiate between active versions, the data of which can still be modified, and fixed versions, the data of which can no longer be modified. New versions can be designed on the basis of a fixed version. New versions can reuse complete fixed versions of a document or even parts of such a document. This is illustrated by the following figure:

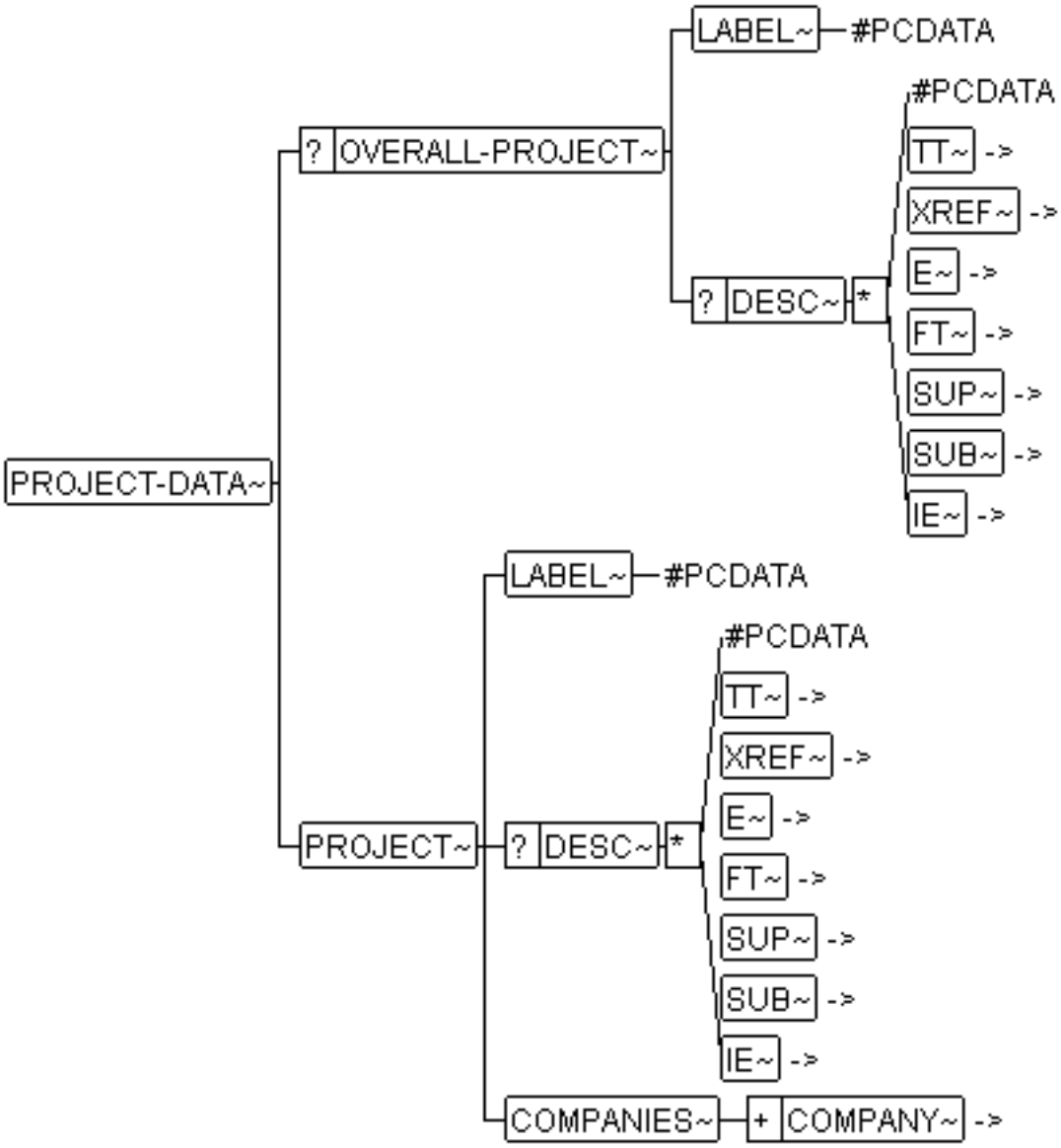

Figure 40: Structure of <project-data>

Project data can be described by a PDM system in an integrated SGML-Editor and PDM environment. This is information on the current project and possibly the main project. Company-specific details about the project can be specified in **<general-project-data>** on the following items:

System overview **<system-overview>**	This chapter can be used to define information about a global system, e.g. a certain car model.
Order justification **<reason-order>**	This may be used to specify information about the reasons for the order of the described component resp. for making the specification of such a component.
Objectives **<objectives>**	This chapter can be used to specify information about the project objectives. E.g. ”Development

	and system release of the engine-managment-system for the model NEW-BEETLE”
Models **<sample-spec>**	This structure is used to define development samples like A-,B-,C-,D-sample. These samples represent the results of the different development phases.
Variant specification **<variant-spec>**	This section is used to specify all variant definitions and their corresponding variant characteristics. See also Topic App. F.5 Variant Concept p. 121.
Limits to other projects **<demarcation-other-projects>**	This chapter is used to describe the demarcation to other projects.
Parallel developments **<parallel-design>**	This can be used to give an overview of the work in parallel projects.
Integration capability **<integration-capability>**	In this chapter requirements on the capabilities of integration in other systems can be described.
Acceptance conditions**<acceptance-cond>**	This chapter is used to define the general conditions for the acceptance of the described components.
Schedule and plans **<project-schedule>**	This chapter is used to define the project-schedule, e.g. project milestones, dates, time limits etc.
Purchasing conditions **<purchasing-cond>**	This is used to define purchasing conditions like amount of devices per year, delivery times, storage quantities, etc. .
Protocols, minutes of meeting **<protocols>**	This is the place where project minutes and other arrangements can be mentioned.
Handed over documents and data**<dir-hand-over-doc-data>**	This is the directory of the handed over documents and data.
Additional project specifications**<add-spec>**	Any kind of additional project description which can’t be described with the chapters mentioned above.

App. F.4 Administrative Data

Since the respective companies explode the interchange DTD into fragments and use it for the respective acquisition DTDs (perhaps in different departments), the administrative dataappears in many places in the DTD. Each of these places can be used as such a fragment(see below).

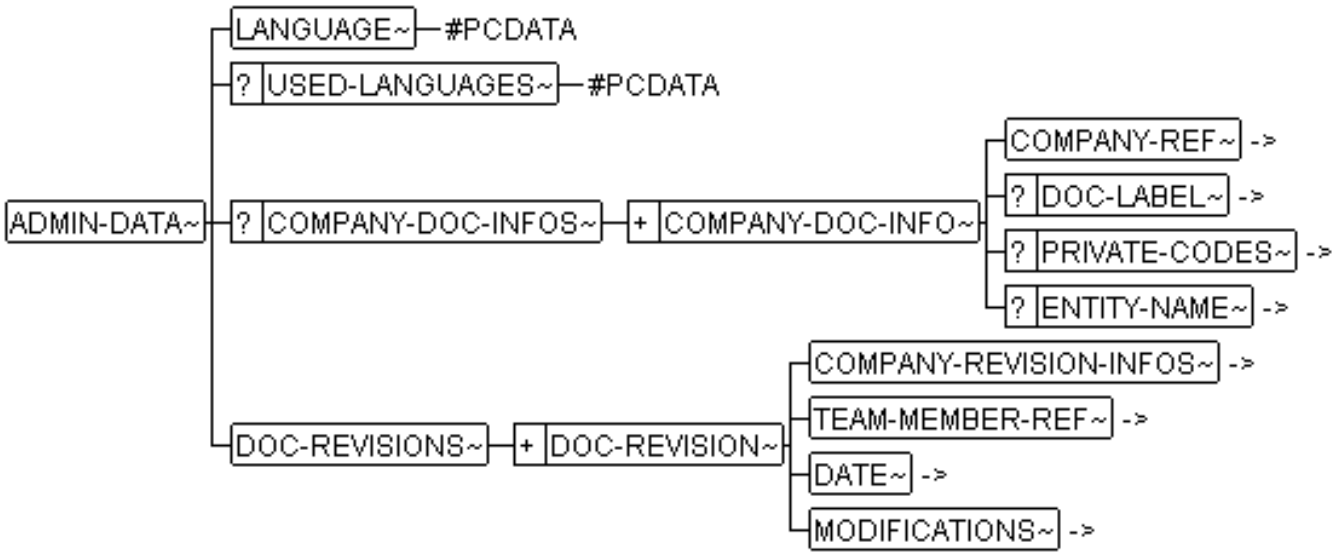

Figure 41: Support of DTD fragmentation through administrative data

The operating model is

- The document respectively the fragment is written in a certain language which can be defined in the element **<language>**. This element can be used to control a SGML system, e.g. to set the correct prefix strings for elements.
- The DTD can be configured for the multilingual operation. In this case **<language>** contains the language of the origin document. All languages used in a document have to be defined within **<used-languages>**, that is each language is defined with a **<l-10>**-element which contains the full language name and in the Attribute **[l]** the short language name (see Topic App. F.6 Multilinguality p. 121).
- The document (or the fragment) is handled in all companies participating in the project.
- The data management in the various companies is different. For that reason, each participant can enter information about their document management facilities in **<company-doc-info>**:

  <doc-label> this is the label under which the document is managed in the company denoted by **<company-ref>**

  <private-code> allows to transport company specific information in a private notation. This is the place, where for example *PDMS* (*Product Data Management System* s) can place pointers and document ids required to resynchronize after a document exchange.

  <entity-name> It might be the case that each participating company uses a different fragmentation strategy. In order to support this, **<entity-name>** can receive information useable by a *split utility* which creates the desired fragments out of the entire document.
- If a new release of the document or the fragment is given, each participating site may use a specific scheme for revision numbers. For that reason, each **<doc-revision>** can receive **<company-revision-info>** which holds the participant specific information for the actual document revision.

  It is up to a *semantical check utility* to keep sure that there is only one entry per company.
- nevertheless, the actual revision is initiated by one individual denoted by **<team-member-ref>** at one certain point of time denoted by **<date>**.

- Finally the modifications made in that revision are stored in **<modifcations>** where the actual **<change>** as well as the **<reason>** for that change is notified. If possible, the change can be located by **<xref>**.
- For each **<modifcation>** the attribute **[type]** determines, if the change is made to the document only (*doc-related*) or to the subject of the document (*content-related*).

App. F.5 Variant Concept

Especially in the automotive sector there is a multiplicity of different variants of a part type. Normally there is not only one variant documented in the system requirements respectively the product specification of such part types.

To understand the implementation of the variant concept in the MSR DTDs, first some definitions have to be made:

Variant Characteristic: Characteristics that lead to a new variant e.g. engine, product line, country, etc. Characteristics are defined in **<variant-char>**. The characteristics have to be subdivided in three classes. These are:

- characteristics which lead to a new subject number(<variant-char **[type="new-part-number"]**>). For this only the existence of such a characteristic is enough to establish a new subject number for this variant!
- characteristics which don't lead to a new subject number (<variant-char **[type="no-new-part-number"]**>).
- characteristics which lead to a new subject number according to shaping.

Variant Definition: Definition of several variants with their variant characteristics for a part type.

Variant: A variant of a part type is defined through the values of it's variant characteristics.

Variant Coding: Allocation of all variant definitions to their corresponding subject- and drawing- numbers and the respective development versions.

App. F.6 Multilinguality

The MSR DTDs can be configured for multilingual operation. To use the multilingual DTD configuration the DTD switch "multilinguality : YES or NO" have to be set.

The description of multilingual texts is made through multiple terminal elements that is multiple elements with content of #PCDATA. Multilingual elements get one of the additional language elements **<l1>**,**<l2>**, **<l3>**,**<l4>**,**<l10>** to build an aggregate of terminal elements. These language elements provide an attribute **[l]** where the language of this element can be specified. The content of the attribute **[l]** have to be defined as two-letter lower-case symbols according to the *[/ Standard: Code for the representation of names of languages / URL: / Relevant Position: Part1]*

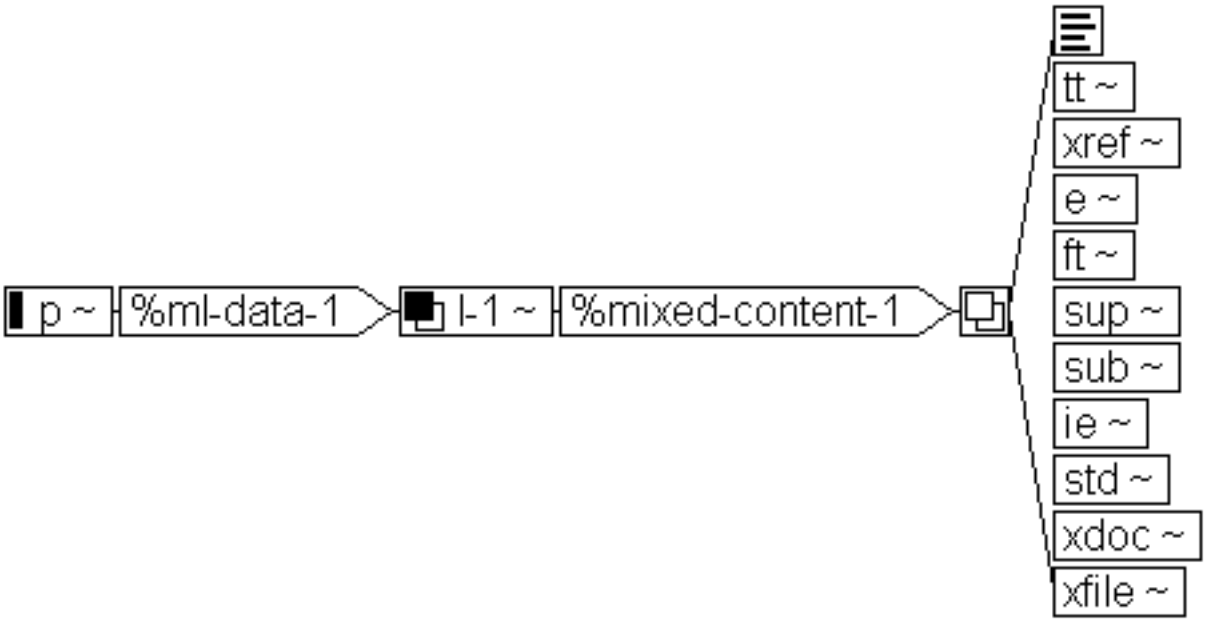

Figure 42: Multilingual Paragraph

App. G Zusätzliche Anmerkungen zum Dokumentstand 2 am 23.7.98

Status dieses Dokument: cd

Documentadministration

Table : team members

Name	Company
Dipl.-Ing.(FH) Uwe Bless	MSR Working Group MEDOC
Dipl.-Inform. Helmut Gengenbach	MSR Working Group MEDOC
Dipl. Ing. Eckard Jakobi	MSR Working Group MEDOC
Dipl.-Ing. Herbert Klein	MSR Working Group MEDOC
Dipl. Ing. Oliver Marcks	MSR Working Group MEDOC
Dipl.-Inform. Peter Rauleder	MSR Working Group MEDOC
Dipl.-Ing. Martin Trinschek	MSR Working Group MEDOC
Dipl.-Ing. Bernhard Weichel	MSR Working Group MEDOC

Table : version overview

Date	Publisher
23.7.98	Dipl.-Ing. Bernhard Weichel
10.7.98	Dipl.-Ing. Bernhard Weichel
27.4.98	Dipl.-Ing. Bernhard Weichel
5.3.98	Dipl.-Ing. Bernhard Weichel
16.12.97	Dipl.-Inform. Helmut Gengenbach
25.08.97	Dipl.-Inform. Helmut Gengenbach
12.08.97	Dipl.-Inform. Helmut Gengenbach
17.06.97	Dipl.-Inform. Helmut Gengenbach
10.04.1997	Dipl.-Inform. Helmut Gengenbach
17.02.1997	Dipl.-Inform. Helmut Gengenbach
13.01.1997	Dipl.-Inform. Helmut Gengenbach
04.08.96	Dipl.-Inform. Helmut Gengenbach
03.07.96	Dipl.-Inform. Helmut Gengenbach
24.06.96	Dipl.-Inform. Helmut Gengenbach
24.06.96	Dipl.-Inform. Helmut Gengenbach
3.6.96	Dipl.-Ing. Bernhard Weichel
15.5.96	Dipl.-Ing. Bernhard Weichel
23.4.1996	Dipl.-Inform. Helmut Gengenbach

Table : modifications

Change	Related to
updated description of application profile Reason:	Content
Migrated to the new msr-medoc docuemtation strategy (according to MSR-TR-DOV) Reason: adoption of new strategy	Content

Table (Cont.): modifications

Change	Related to
Content	Revised and corrections to structures Reason: Preparation of final document
Content	Start of editorial revisions Reason:
Content	Official release Reason: Release of first official version
Document	Initial release Reason: Release of first official version
Document	Changes to contents models of variables Reason: A distinction must be made for variables between the definition and the access to a variable.
Document	Document
Document	Content
Additional element <si-unit> introduced in the <sw-unit> diagram for one SI Unit. Reason: Required when using tools that can only work with SI units.	Document
Document	Document
Document	Introduction of physical data type for implementation of variables, basic data type/C-Typ introduced for parameters, units of measure in data dictionary, additional elements for various fragmentation models Reason: Extension, improvement, fragmentation
Document	Document
Examples included, editorial revisions Reason: Clarity	Document

Table (Cont.): modifications

Change	Related to
Document	Editorial revisions to Topic 1.2 Phase model for software generation p. 10 Reason: Clarity, comprehension
Document	Document
Content	Content
Content	Content
Content	Content
Handling variants, contents model software restructured Reason: Completeness	Document
Document	Content
Content	Document
Document	Current Near&Far graphics included Reason: Update
Document	Extended description of elements. Reason: Completeness.
Content	Standardization levels included Reason: Expansion
Document	Content
Content	Corrected typing errors Reason: Quality enhancement
Document	Content
Changes introduced Reason: For specific further developments	Content
Document	Changed format to S-GML. Reason: Study group resolution.
Document	Content
Content	Content
Content	Content
Content	Content

Table : modifications included

Date	Chapter	Change	Related to
Nr. 1, 23.7.98	Gesamt	updated description of application profile Reason:	Content
Nr. 2, 10.7.98	Gesamt	Migrated to the new msr-medoc docuemtation strategy (according to MSR-TR-DOV) Reason: adoption of new strategy	Content
		inserted chapter for application profile from (TR-CAP) Reason: to have all in one document	Content
Nr. 3, 27.4.98	Gesamt	Revised and corrections to structures Reason: Preparation of final document	Content
Nr. 4, 5.3.98	Gesamt	Start of editorial revisions Reason:	Content
Nr. 5, 16.12.97	Gesamt	Official release Reason: Release of first official version	Document
Nr. 6, 25.08.97	Gesamt	Initial release Reason: Release of first official version	Document
Nr. 7, 12.08.97	Gesamt	Changes to contents models of variables Reason: A distinction must be made for variables between the definition and the access to a variable.	Document
		Graphics update Reason: Completeness, update	Document
		Editorial revisions Reason: Completeness and update	Document
		Versions for variables Reason: In accordance with practice	Content
Nr. 8, 17.06.97	Gesamt	Additional element <si-unit> introduced in the <sw-unit> diagram for one SI Unit. Reason: Required when using tools that can only work with SI units.	Document
		Several data dictionaries Reason: For tools that treat data dictionaries on a function-specific basis.	Document
		Project data introduced Reason: Completeness	Document
		Parameter contents in the <sw-function-variant> changed analogous to data dictionaries. Reason: Correction	Document

Table (Cont.): modifications included

Date	Chapter	Change	Related to
Nr. 9, 10.04.1997	Gesamt	Introduction of physical data type for implementation of variables, basic data type/ C-Typ introduced for parameters, units of measure in data dictionary, additional elements for various fragmentation models Reason: Extension, improvement, fragmentation	Document
		Editorial revisions Reason: Clarity, corrections to typing errors	Document
Nr. 10, 17.02.1997	Gesamt	Examples included, editorial revisions Reason: Clarity	Document
		Editorial revisions Reason: Clarity, corrections to typing errors	Document

Table (Cont.): modifications included

Date	Chapter	Change	Related to
Nr. 11, 13.01.1997	Gesamt	Editorial revisions to Topic 1.2 Phase model for software generation p. 10 Reason: Clarity, comprehension	Document
		Action list included in Changes Reason: Standardization in procedures	Document
		Introduction of physical types in (Topic 2.2.3.2 Physical data types p. 22) Reason: Method of working for users (with Excel lists)	Content
		Introduction of units of measure for parameter contents (Change request 5.2 [paramunit] Parameter contents require units of measure p. 80). Reason: Parameter contents are used in the documentation of physical data and for exchangeability with other systems. The entire data dictionary is not necessarily available for an exchange. An option for the units of measure must therefore be provided.	Content
		Presentation of parameter contents not only as intergers. (Change request 5.3 [paramhex] Parameter contents at Integer/HEX/Address level p. 80). Reason: Meaningful for the documentation.	Content
		It shall be possible to assign parameter contents to functions (Change request 5.4 [paramfunc] Parameter contents must be assignable to functions p. 80). Reason: Function-oriented data management can be applied in autonomous mode.	Content
		Support of parameter structures (Change request 5.5 [kgrhierachie] Support of parameter hierarchies p. 81). Reason: Already used in practice.	Content
		Introduction of annotations (Change request 5.8 [annot] Annotation p. 82 <annotation> Reason: Refer to same	Content

Table (Cont.): modifications included

Date	Chapter	Change	Related to
Nr. 12, 04.08.96	Gesamt	Handling variants, contents model software restructured Reason: Completeness	Document
		Chapter 4.6 contents model software restructured with sub-paragraphs. Reason: Quality enhancement	Document
		Description of elements expanded. Reason: Completeness	Content
		Description of attributes included. Reason: Completeness	Content
		References within the software. Reason: Completeness	Document
		Description of general structures Reason: Quality enhancement	Document
Nr. 13, 03.07.96	Gesamt	Current Near&Far graphics included Reason: Update	Document
Nr. 14, 24.06.96	Gesamt	Extended description of elements. Reason: Completeness.	Content
Nr. 15, 24.06.96	Gesamt	Standardization levels included Reason: Expansion	Document
		Corrections to contents model software. Reason: Quality enhancement	Content
		Additional changes agreed in meeting 3.6.96 Reason: Supplement	Content
Nr. 16, 3.6.96	Gesamt	Corrected typing errors Reason: Quality enhancement	Document
		Included changes from meeting 3.6.96. Reason: Study group decision	Content
Nr. 17, 15.5.96	Gesamt	Changes introduced Reason: For specific further developments	Content
		Further changes from last meeting? Reason: To be added	Document

Table (Cont.): modifications included

Date	Chapter	Change	Related to
Nr. 18, 23.4.1996	Gesamt	Changed format to SGML. Reason: Study group resolution.	Document
		Inclusion of the structure in the form of NEAR&FAR. Reason: Study group resolution.	Content
		Extensions to the structure. Reason: Harmonization with ASAP and structures in use at Bosch.	Content
		Description of SGML elements. Reason: Extensions	Content
		<sw-params> included in data dictionary. Reason: Characteristics referenced as function-overriding characteristics.	Content
		<sw-base-type> <in sw-variable> Reason: The base type is variable-specific.	Content
		Deleted from <sw-desc> <prms>. Reason:	Content
		Direction implemented as attribute of <sw-variable-ref under <sw-function-variables>. Reason:	Content