MSR-Teilprojekt MEDOC

Strukturelle Grundlagen

Bereich: Verhalten / Software

MSR-Arbeitskreis MEDOC, Dipl.-Ing. Bernhard Weichel

06/09/2002 12:14:25 msrsw-sp-de.xml

Zusammenfassung

Dieses Dokument beschreibt die strukturellen Grundlagen der MSR-Entwicklungsdokumentation MEDOC für den Bereich Software für Steuergeräte. Es werden Hinweise zum zugrundeliegenden Entwicklungsprozeß gegeben. Die Struktur der *MSRSW.DTD* wird detailliert beschrieben.

Inhaltsverzeichnis

Präliminarien

Projektbeteiligte Firmen

MSR-Arbeitskreis MEDOC [MSR-MEDOC]

Name Rollen	Abteilung	Adresse	Kontakt
Dipl.-Ing.(FH) Uwe Bless			
Dipl.-Inform. Helmut Gengenbach			
Dipl. Ing. Eckard Jakobi			
Dipl.-Ing. Herbert Klein			
Dipl. Ing. Oliver Marcks			
Dipl.-Inform. Peter Rauleder			
Dipl.-Ing. Martin Trinschek			
Dipl.-Ing. Bernhard Weichel			

Versionsinformation

Dokumententeil	Herausgeber			
	Firma	Version	Status	Anmerkungen
10.7.98 Details siehe Nr. 1, Seite 108	Dipl.-Ing. Bernhard Weichel			
	MSR-MEDOC	1.1.0a	CD	

1 Einleitung

Dieses Dokument beschreibt die strukturellen Grundlagen der MSR-Entwicklungsdokumentation MEDOC für den Bereich Software für Steuergeräte.

An dieser Stelle wird daraufhingewiesen, daß MSR keine Normung hinsichtlich der mit MEDOC beschriebenen Systeme oder deren Eigenschaften betreibt. MEDOC unterstützt die Verwendung von (inter-)nationalen Standards und Hausnormen sowie nicht genormten, für die Beschreibung von Systemen relevante Daten in einer Entwicklungsdokumentation.

2 Allgemeine Projektdaten

1 Hinweise zum Software-Entwicklungsprozeß

1.1 Zusammenarbeit zwischen Hersteller und Zulieferer

Die in Abbildung 1 Hersteller-Zulieferer-Szenarien S. 9 dargestellten Szenarien bzgl. Zusammenarbeit bei der Software-Entwicklung zwischen Hersteller und Zulieferer wurden bei der Definition der *MSRSW.DTD* berücksichtigt.

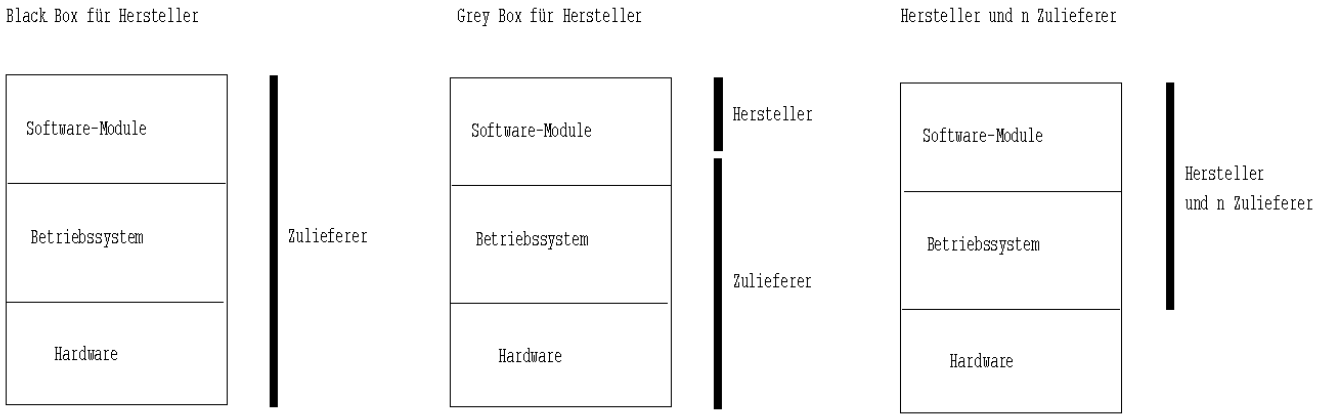

Abbildung 1: Hersteller-Zulieferer-Szenarien

Black Box Alle Entwicklungen laufen beim Zulieferer. Der Zulieferer erstellt eine komplette Dokumentation.

Grey Box Hersteller und Zulieferer haben jeweils einen definierten Satz von Steuergeräte-Funktionen für die sie verantwortlich sind. Die jeweiligen Entwicklungsergebnisse werden zusammengefaßt.

White Box Hersteller und Zulieferer arbeiten gemeinsam an Steuergeräte-Funktionen. Es findet ein laufender Austausch, auch von einzelnen Teilen (z.B. Variablendefinitionen usw.) statt.

1.2 Phasenmodell zur Erstellung von Software

In diesem Kapitel wird ein Phasenmodell für die Erstellung von Software skizziert. Es stellt eine gemeinsame Verständnisgrundlage aller MSR-Beteiligten dar. Die Phasen können in Schleifen wiederholt werden.

Auf Basis dieses Phasenmodells werden diejenigen Ergebnisse identifiziert, die zwischen Hersteller und Zulieferer ausgetauscht werden.

1.2.1 Systemanalyse

In der Systemanalyse wird das System auf logischer Ebene beschrieben. Das entstehende logische Funktionsmodell kann simulierbar sein (z.B. in *Matrix-X* bzw. *Ascet*). Für Systemanalyse und Systemdesign werden die gleichen Strukturen (und damit auch die gleiche DTD) verwendet.

Funktionsanalyse

Die Funktionsanalyse enthält die Beschreibung des Kontexts, der Funktionen (hierarchisch, inkl. Diagnosefunktionen, Sicherheitsfunktionen, etc.) und der Informationsflüsse (informell). Je Funktion werden eine formale Definition, eine informelle Beschreibung,

Applikationshinweise und Kundendiensthinweise dokumentiert (vgl Abbildung 18 Funktionsvariante S. 32).

Beschreibung der zeitlichen Abhängigkeiten

Die zeitlichen Abhängigkeiten können mittels RT-Analyse, Zustandsdiagrammen, Petri-Netzen, etc. modelliert werden.

1.2.2 Analyse - Review

Der Inhalt des Analyse-Reviews kann durch firmeninterne Richtlinien bestimmt werden.

Review-Protokoll

Das Protokoll kann die Feststellung der Vollständigkeit, der Widerspruchsfreiheit, der Testbarkeit, etc. der geforderten Funktionen enthalten.[1]

1.2.3 Systemdesign

Beim Systemdesign wird aus dem logischen Funktionsmodell das physikalische Funktionsmodell abgeleitet.

Physikalisches Funktionsmodell

Die Funktionsmodell enthält die Beschreibung des Kontexts, der implementierten Funktionen (hierarchisch, inkl. Diagnosefunktionen, Sicherheitsfunktionen, etc.) und der Informationsflüsse (informell).

Je Funktion werden eine formale Definition, eine informelle Beschreibung, Applikationshinweise und Kundendiensthinweise dokumentiert (vgl. Abbildung 18 Funktionsvariante S. 32).

Beschreibung der zeitlichen Abhängigkeiten

Beim Systemdesign werden z.B. Funktionen zu Tasks zusammengefaßt und Rechenebenen. zugeordnet (Scheduling-Tabellen)

MSRSW.DTD 1.1.0 besitzt außer bei Variablen keine expliziten Beschreibungsmittel für zeitliche Abhängigkeiten. Diese müssen daher informell bei der Funktionsbeschreibung (**<sw-function-desc>**) bzw. in **<add-info>** beschrieben werden..

Partitionierung in Hardware und Software

Die Partitionierung erfolgt unter Berücksichtigung von Resourcen und anderen Randbedingungen (z.B. Kosten, etc.).

Software-Architektur

Die Software-Architektur kann in der *MSRSW.DTD* beschrieben werden, in dem ein **<sw-function>** angelegt mit **<sw-function-class>** = *architecture*.

Hardware-Architektur

wird in *MSRSYS.DTD* beschrieben.

Testspezifikation

wird in *MSRSYS.DTD* beschrieben.

1.2.4 Design - Review

Der Inhalt des Design-Reviews kann durch firmeninterne Richtlinien bestimmt werden.

1

Review-Protokoll

Das Protokoll kann die Feststellung der Vollständigkeit, der Widerspruchsfreiheit, der Testbarkeit, etc. der geforderten Funktionen enthalten. Hierfür kann *MSRREP.DTD* verwendet werden.

1.2.5 Codierung

Funktionsmodule

Im Rahmen der Codierung werden die im Systemdesign definierten Funktionen implementiert. Diese Implementierung kann ggf. auch duch Codegeneratoren erfolgen.

In dieser Phase entstehen Details des Datenlexikons (**<sw-data-dictionaries>**) wie Kenngrößenstrukturen (**<sw-params>**), Variablen **<sw-variables>**, Umrechnungsformeln (**<sw-compu-methods>**). Darüber hinaus werden auch Quelltexte, Compilersteuerfiles u.s.w. erstellt.

1.2.6 Modultest

Formaler und funktionaler Test des codierten Definition Software-Modul S. 95Software-Moduls unter Laborbedingungen.

Testbericht

Der Testbericht beschreibt das Umfeld, die Beaufschlagung sowie die Ergebnisse des Modultests. Hierfür kann *MSRREP.DTD* verwendet werden.

1.2.7 Integration

In dieser Phase werden die einzelnen Module zusammengeführt. Die Integration wird auf der Test- und/oder der Zielumgebung durchgeführt.

Datenstand

Das konkrete Ergebnis der Integration ist der Definition Programmstand S. 95 Programmstand.

1.2.8 Applikation (Kalibrierung)

Bei der Applikation wird die Steuerungseinrichtung auf ein konkretes Fahrzeug bzw. einen konkreten Motor angepaßt. Dies geschieht durch Festlegung der jeweiligen Kenngrößeninhalte.

Datenstand

Das konkrete Ergebnis der Applikation ist der Definition Datenstand S. 94Datenstand.

1.2.9 Systemtest

für die Erstellung des Testberichtes kann *MSRREP.DTD* verwendet werden.

1.3 Sprachebenen

Bei Software-Funktionen in Steuergeräten kann man zwischen verschiedenen Sprachebenen unterscheiden (siehe Abbildung 2 Sprachebenen S. 12). Insbesonders gibt es eine Standardisierungsebene, in der Software-technische Standards bzgl. Dokumentation, Variablen- und Kenngrößen-Spezifikation, C-Header, C-Source und Betriebssystem-Anweisungen angesiedelt sind.

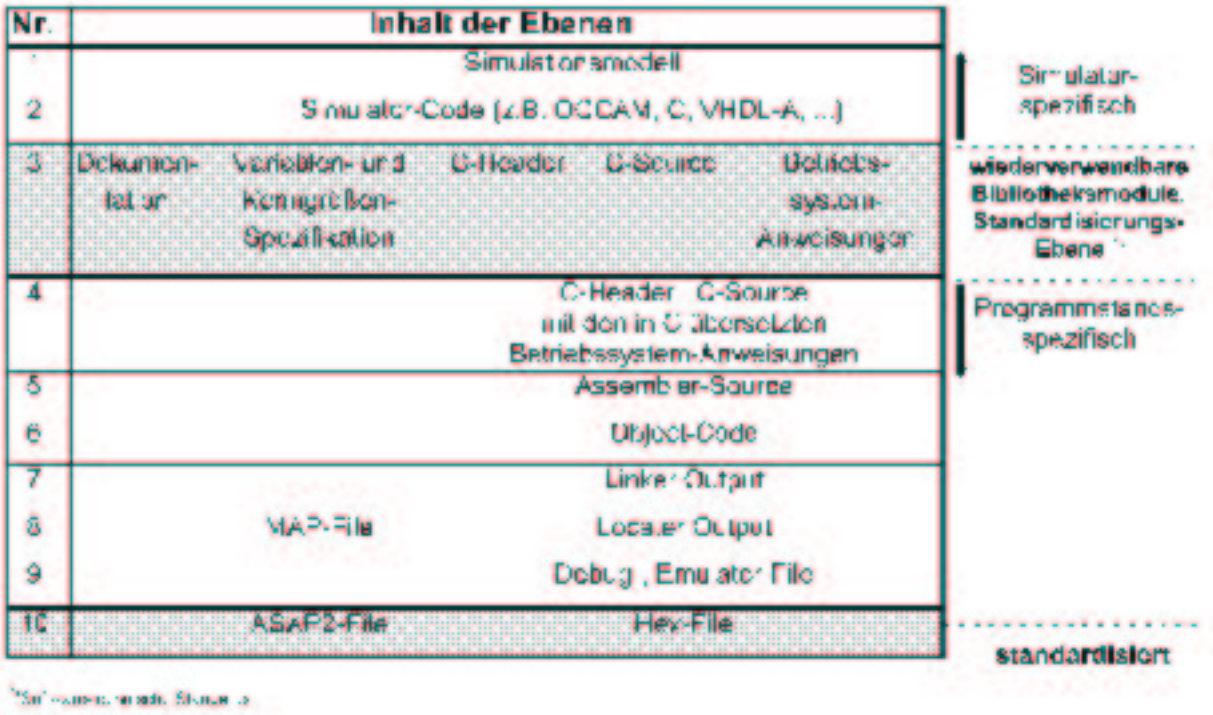

Abbildung 2: Sprachebenen

Die Standardisierung auf dieser Ebene wird in mehreren Standardisierungs-Projekten wie MSR-AG MEDOC, MSR-AG Codegeneratoren und OSEK betrieben (siehe Abbildung 3 Standardisierungs-Projekte S. 12).

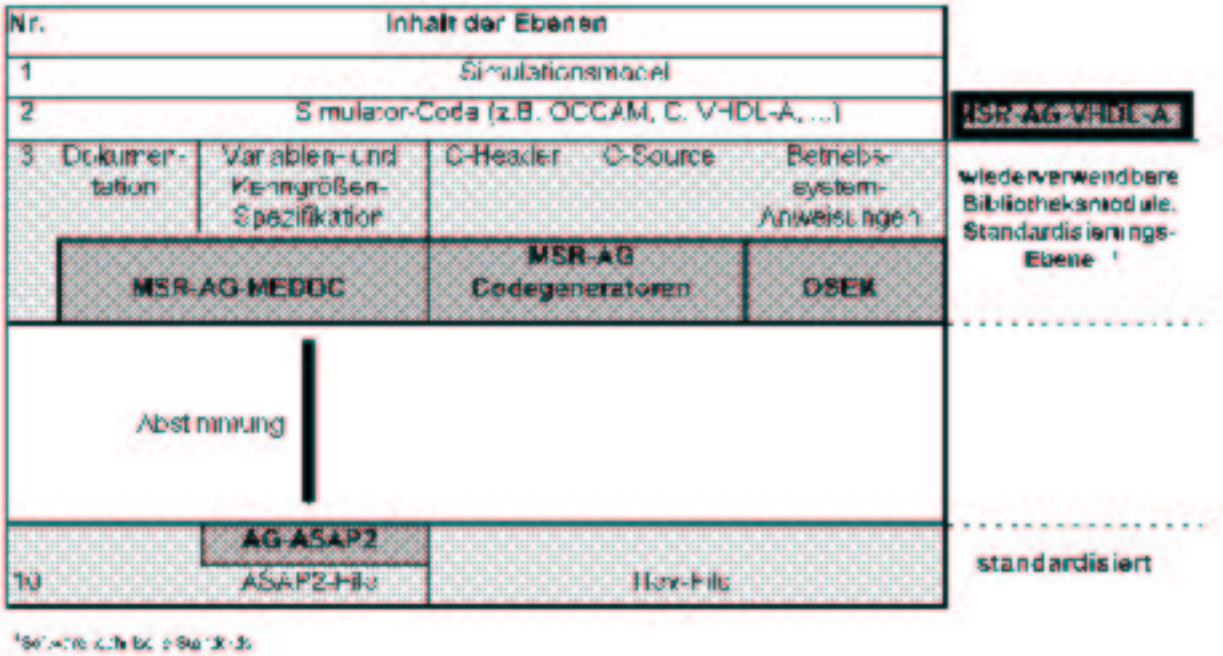

Abbildung 3: Standardisierungs-Projekte

Auf unterster Ebene hat eine Standardisierung bereits stattgefunden: ASAP2-File und Hex-File.

2 Strukturierung

2.1 Allgemeines

2.1.1 Verbindung zur MSRSYS

In der *MSRSYS.DTD* kann das Verhalten einer Komponente als Funktionshierarchie beschrieben werden (**<behaviour>**). Diese Beschreibung ist unabhängig von der Implementierung in Hardware bzw. Software.

Für die Realisierung in Software dient die *MSRSW.DTD*. Diese ist funktionsorientiert[2] aufgebaut. Der Zusammenhang zwischen Funktionen in *MSRSYS.DTD* und *MSRSW.DTD* kann über Referenzierung (**<sw-fulfils>**) hergestellt werden.

2.1.2 Variantenbehandlung

Bzgl. Varianten wurden folgende Szenarien betrachtet:

interne Varianten	Diese Varianten werden vom SG ohne massive Umprogrammierung beherrscht. Das Steuergeräte-Programm kann zwischen mehreren Varianten allein duch Umprogrammierung eines Codewortes (i.A. über Diagnoseleitung am Bandende) oder gar durch Anlegen von Steckerbrücken umschalten. Diese Variantenart ist letztlich eine Funktion der Software selbst.
externe Varianten	Diese Varianten betreffen Geräte, die sich sehr ähnlich sind, jedoch immer nur eine Variante beinhalten. Variantenaustausch geschieht durch weitgehende Umprogrammierung. In der Regel werden sowohl die unterschiedlichen programmierten Varianten als auch das unprogrammierte Steuergerät mit unterschiedlichen Sachnummern gehandhabt.

Variantenbschreibungen sind wichtig im Bereich der Funktionen (**<sw-function-spec>**) und der Kenngrößen-Inhalte (**<sw-param-contents>**).

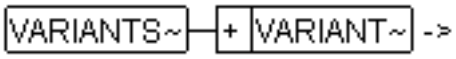

Abbildung 4: Funktionsvariante

2.1.2.1 Variantenabhängige Kenngrößen

Die Beschreibung der **<sw-param-contents>** ist möglich in **<sw-function-spec-variant>** und in **<sw-param-contents-spec>**. Die implizite Variantenbehandlung **<sw-param-content>** ist an beiden Stellen durch die Tatsache gegeben, daß in **<sw-param-content>** auf **<sw-param>** referenziert wird. **<sw-function-variant>** enthält einen expliziten Variantenverweis. Die Zuordnung von **<sw-param>** zu den Varianten ergibt sich aus den **<variant-def-ref>**s der **<sw-function-variant>**s in denen diese referenziert werden (implizite Zuordnung).

Die Zuordnung von Kenngrößen zu Varianten geschieht also ausschließlich über die Verwendung in Funktionsvarianten. Für die automatische Prüfung auf widerspruchsfreie Variantenbeschreibung von Funktionen und Kenngrößen wäre eine explizite Zuordnung von Kenngrößen zu Varianten (durch **<variant-def-refs>** in **<sw-param>**) erforderlich. Dies wird zunächst zurückgestellt, da nicht geklärt ist, ob die dadurch entstehende Komplexität erforderlich und gerechtfertigt ist[3].

2

3

2.1.2.2 Variantenabhängige Umrechnungsformeln

Umrechnungsformeln werden von der Hardware beeinflußt (Signalaufbereitung usw). Daher kann es vorkommen, daß unterschiedliche HW-Varianten auch unterschiedliche Umrechnungsformeln bedingen. Beispiel hierfür kann das Auswechseln eines Sensors sein, welcher gar nicht im SG eingebaut ist. Dies soll zunächst aber nicht explizt durch unser Daten/Dokumentationsmodell unterstützt werden. Die Behandlung dieses Falles geschieht vielmehr wie folgt:

- für jede Variante wird falls erforderlich eine eigene **<sw-compu-method>** angelegt, die sich dann mindestens durch ihren **<short-name>** und auch durch **<long-name>** unterscheiden müssen.
- Die variantenspezifischen **<sw-compu-method>**s werden referenziert in variantenspezifischen **<sw-param>**s bzw. **<sw-variable>**s. Diese müssen sich wiederum im **<short-name>** und **<long-name>** unterscheiden und können ihrerseits in **<sw-function-variant>** referenziert werden. Damit erfolgt die Variantenbehandlung nicht im Verborgenen (d.h. durch Aufbereitung in der DTD) sondern ausschließlich durch den Benutzer definiert.

2.1.3 Ablage des Datenlexikons

Das Datenlexikon kann sowohl global (unter **<msrsw>**) wie auch funktionslokal (unter **<sw-function-variant>**) im Dokument verteilt eingetragen werden. Es ist allerdings als ein logisches Lexikon zu behandeln. Diese Flexibilität erlaubt die Anwendung der *MSRSW.DTD* in verschiedenen Prozeßphasen, Prozeßmodellen. Darüberhinaus kann diese Flexibilität genutzt werden, um sich an die Möglichkeiten verschiedener Werkzeuge und deren Exportfilter anzupassen.

2.1.4 Ablage von Kenngrößeninhalten

Kenngrößeninhalte (**<sw-param-contents>**, vgl. Topic 2.2.5 Spezifikation von Kenngrößeninhalten S. 34) können sowohl lokal innerhalb von **<sw-function-variant>** (Topic 2.2.4.1.6 Funktionsbezogene (lokale) Kenngrößeninhalte S. 33) als auch global innerhalb von **<msrsw>** Topic 2.2.5 Spezifikation von Kenngrößeninhalten S. 34 abgelegt werden.

Hierfür gelten folgende Konventionen:

1. Die Zuordnung von Kenngrößen zu Funktionen geschieht ausschließlich über Referenzierung in den Funkionsbeschreibungen (**<sw-param-ref>** in **<sw-function-variant>**).
2. Es ist möglich, in **<sw-function-variant> <sw-param-content>**s anzugeben. Darin dürfen nur **<sw-param>**s referenziert werden, die auch in **<sw-function-variant>** referenziert sind. Die Angabe von Kenngrößeninhalten in Funktionen muß also konsistent sein zur Zuordnung von Kenngrößen zu Funktionen.
3. Wird für eine Kenngröße (**<sw-param>**) sowohl in **<sw-function-variant>** als auch in **<sw-param-content-spec>** ein Inhalt spezfiziert, hat letztere Angabe Vorrang.
4. Werden in zwei verschiedenen Funktionen für die selbe Kenngröße Inhalte angegeben, so muß auch in **<sw-param-content-spec>** ein Eintrag enthalten sein. Damit kann die o.g. einfache Vorrangregel verwendet werden.
5. Der Autor einer **<sw-function-variant>** hat durch Einfügung eines **<sw-param-contents>** die Möglichkeit zu steuern, welche Parameterinhalte er bei der Formatierung lokal mit angezeigt bekommt.

Da **<sw-param-contents>** sowohl bei den Funktionen als auch global für die gesamte Software (innerhalb von **<msrsw>** möglich ist, können verschiedene Prozesse mit der DTD betrieben werden. Es wird empfohlen:

- Kenngrößeninhalte als Programmiervorgabe werden innerhalb von **<sw-function-variant>** angegeben. Damit bleibt diese Beziehung auch bei Fragmentierung auf Funktionsebene erhalten.

- Kenngrößeninhalte als Ergebnis der Applikationsphase werden innerhalb von **<msrsw>** geschlossen funktionsübergreifend angegeben. **<sw-param-contents>** innerhalb **<msrsw>** (vgl. Topic 2.2.5 Spezifikation von Kenngrößeninhalten S. 34) nimmt also einen Applikationsdatenstand auf.

2.1.5 Namensräume

Allgemein gilt, daß der **<short-name>** identifizierenden Charakter besitzt, insbesondere für Bezüge nach außen (z.B. ASAP). Daraus ergibt sich, daß der **<short-name>** innerhalb gegebener Namens (Definitions-)räume eindeutig sein muß. Diese Namensräume ergeben sich aus definierten Hierarchien in der DTD [4]. Einzelheiten finden sich in *[Externes Dokument: Concepts of MSR-DTD / Status: in Erwartung / Datum: TBD / URL: / relevante Stelle:]*

2.1.6 Referenzen innerhalb der Software

Innerhalb der Software treten an vielen Stellen Referenzen auf. Durch die Verwendung von Referenzen kann Redundanz vermieden werden.

4

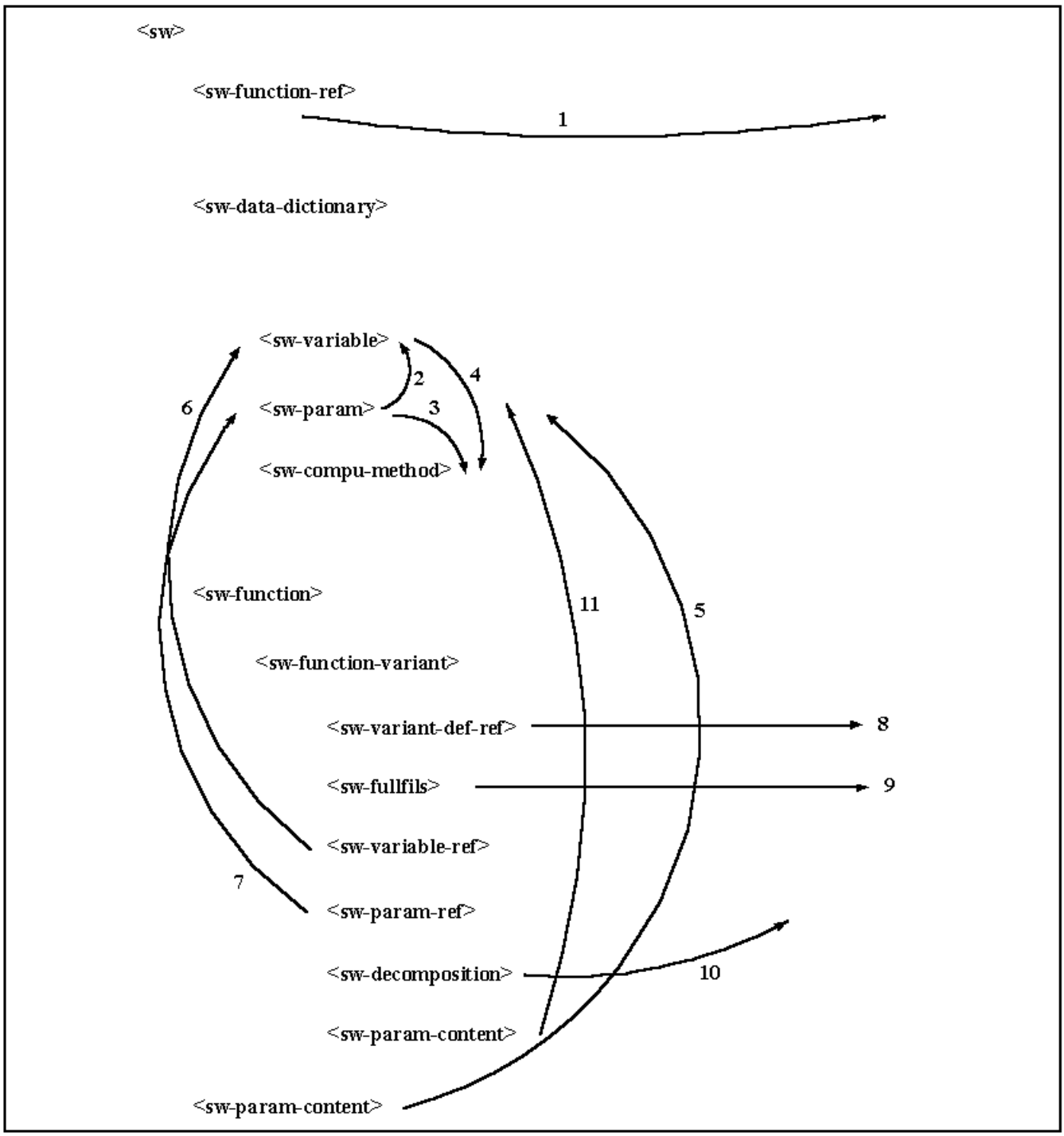

Abbildung 5: Referenzen innerhalb der Software

Siehe Kapitel Topic 2.3 Referenzen S. 36.

Tabelle 1: Referenzarten

Nr.	Bezeichnung	Quellelement	Zielelement	Bedeutung
1	**<sw-function-ref>**	**<msrsw>**	**<sw-function>**	Verweis auf verwendete Funktion außerhalb **<MSRSW>** wie z.B. Betriebssystem- oder Netzwerkfunktion.
2	**<sw-variable-ref>**	**<sw-axis-individual>**	**<sw-variable>** in **<data-dictionary>**	Verweis auf Eingangsvariable einer Kenngröße.
3	**<sw-compu-method-ref>**	**<sw-param-axis-values>**	**<sw-compu-method>**	Weist den Achswerten einer Kenngröße eine Umrechnungsformel zu.
4	**<sw-compu-method-ref>**	**<sw-variable>**	**<sw-compu-method>**	Weist einer Variablen eine Umrechnungsformel zu.
5	**<sw-param-ref>**	**<sw-param-content>**	**<sw-param>**	Ordnet einen Parameterinhalt einem Parameter zu.
6	**<sw-variable-ref>**	**<sw-passing-variables>** , **<sw-local-variables>**	**<sw-variable>** in **<data-dictionary>**	Verweis auf verwendete Variable einer Funktion.
7	**<sw-param-ref>**	**<sw-function-variant>**	**<sw-param>**	Verweis auf verwenete Parameter einer Funktionsvariante.
8	**<variant-def>**	**<variant-def>**	Variante in Projektdaten	Ordnet einer Funktionsvariante ein oder mehrere fiktive Varianten zu.
9	**<fulfills-ref>**	part, sw-function	function	Ordnet einem Part oder einer Software-Funktion diejenigen (Verhaltens-)Funktionen (keine, eine oder mehrere) zu, die es bzw. sie realisiert.
10	**<sw-function-ref>**	**<sw-data-dictionary>**	**<sw-function>**	Erlaubt es, ein Datenlexikon (fragment) einer bestimmten Funktion zuzuweisen. Damit kann man auch funktionsorientierte Datenlexika schreiben, wenn man **<sw-function>** nicht ausfüllt.
10	**<sw-function-ref>**	**<sw-param-contents>**	**<sw-function>**	Erlaubt es, Dateninhalte einer bestimmten Funktion zuzuweisen auch wenn man **<sw-function>** nicht ausfüllt ist.
10	**<sw-function-ref>**	**<sw-param-content>**	**<sw-function>**	Erlaubt es, den Inhalt einer Kenngröße einer bestimmten Funktion zuzuweisen auch wenn man **<sw-function>** nicht ausfüllt ist.
10	**<sw-function-ref>**	**<sw-decomposition>**	**<sw-function>**	Hier werden Verweise auf verwendete Funktionen eingetragen.
11	**<sw-param-ref>**	**<sw-param-content>**	**<sw-param>**	Ordnet einen Parameterinhalt einem Parameter zu.

2.2 Inhaltsmodell Software

Die Software umfaßt die folgenden Haupteile (siehe Abbildung 6 Softwarestruktur, oberste Ebene. S. 18) (Die Softwarestrukur wird nachfolgend in Form von *Near&Far* -Grafiken (vgl. Topic Anh. B Erklärung der Near&Far-Symbole S. 93) dargestellt).

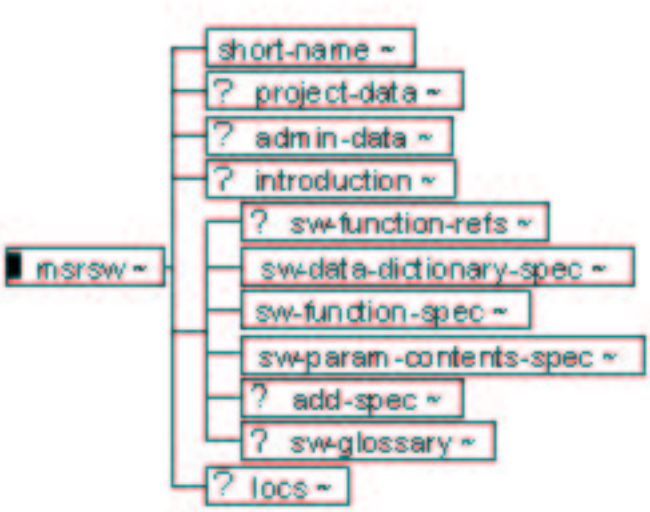

Abbildung 6: Softwarestruktur, oberste Ebene.

<short-name>	Kurzname für das gesamte Dokument.
<project-data>	Projektdaten (Topic 2.2.2 Projektdaten S. 19)
<admin-data>	Verwaltungsdaten (Topic Anh. E.1 Administrative Daten S. 97)
<introduction>	kurze Einführung (Topic Anh. E.7 Introduction S. 102)
<sw-function-refs>	Verweis auf externe Software (Topic 2.2.1 Verweise auf ”externe” Software S. 18)
<sw-data-dictionary-spec>	Globales Datenlexikon (Topic 2.2.3 Datenlexikon S. 19)
<sw-function-spec>	Detaillierte Spezifikation der Definition Software-Funktion S. 95 Software-Funktionen (Topic 2.2.4.1.1 Funktionsdefinition S. 32)
<sw-param-contents-spec>	Globale Parameterinhalte (Topic 2.2.5 Spezifikation von Kenngrößeninhalten S. 34)
<add-spec>	Zusatzliche, halbformale Spezifikationen (Topic 2.2.7 Zusätzliche Spezifikationen S. 36)
<sw-glossary>	Erlaubt die Erfassung eines Glossars (Topic 2.2.6 Glossar für das Dokument S. 36)
<locs>	Platzhalter für dokumentübergreifende Referenzierungen. Details siehe *[Externes Dokument: Concepts of MSR-DTD / Status: in Erwartung / Datum: TBD / URL: / relevante Stelle:]*

2.2.1 Verweise auf ”externe” Software

Mit Hilfe von **<sw-function-refs>** können Funktionen aus anderen Steuergeräten referenziert werden. Beispiele hierfür sind mehrfach verwendete Funktionen wie Betriebssystem- oder Netzwerkmanagementfunktionen.

2.2.2 Projektdaten

Hier können Informationen zum Projekt (**<project-data>**) eingetragen werden. Dies sind Angaben zur Firma, zu Mitarbeitern und allgemeine Projektdaten. In den allgemeinen Projektdaten können Angaben zu folgenden Punkten spezifiziert werden:

- Auftragsbegründung
- Ziele
- Muster
- Variantenspezifikation
- Abgrenzung gegeenüber anderen Projekten
- Parallelentwicklungen
- Integrationsfähigkeit
- Abnahmebedingungen
- Terminpläne
- Protokolle, Besprechungsnotizen

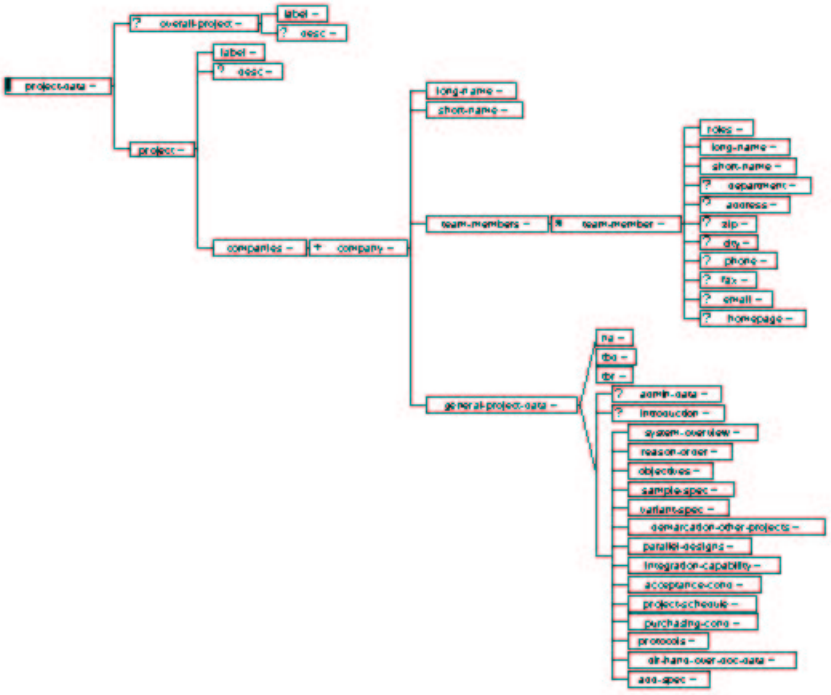

Abbildung 7: Projektdaten

2.2.3 Datenlexikon

Im Datenlexikon **<sw-datadictionary-spec>** werden die Datenstrukturen der beschriebenen Steuergerätesoftware.

Ein Datenlexikon (**<sw-datadictionary>**) umfaßt:

<desc>	Kurzbeschreibung
<sw-data-dictionary-class>	Dieses Element kann verwendet werden, um die Rolle des vorliegenden Datenlexikon im Prozeß zu kennzeichnen. So können z.B. alle geänderten Variablen in ein **<sw-datadictionary>** mit der Klasse *changed* abgelegt werden.
<sw-function-ref>	Erlaubt es, ein Datenlexikon einer bestimmten Funktion zuzuordnen (Topic 2.1.6 Referenzen innerhalb der Software S. 15)
<sw-units>	Maßeinheiten, die in der Dokumentation verwendet werden (Topic 2.2.3.1 Maßeinheiten S. 20)
<sw-physic-types>	Physikalische Datentypen (Topic 2.2.3.2 Physikalische Datentypen S. 21)
<sw-variables>	Variablen (Topic 2.2.3.3 Variablen S. 22)
<sw-params>	Strukturen von Kenngrößen (Topic 2.2.3.4 Kenngrößenstrukturen S. 23)
<sw-compu-methods>	Umrechnungsformeln (Topic 2.2.3.5 Umrechnungsformeln S. 27)
<sw-adressing-methods>	Adressierverfahren für Kenngrößen und Variablen (Topic 3.2 Beschreibung der Elemente S. 39)
<sw-param-record-layouts>	Vorschriften zur Ablage der Parameter im Steuergerätespeicher. Sie dienen zur Steuerung des Zugriffes von z.B. *Applikationssystem*en. (Topic 3.2 Beschreibung der Elemente S. 39)
<sw-code-syntaxes>	Vorschriften zur Generierung von Quellcode für den Compiler der Steuergerätesoftware (Topic 3.2 Beschreibung der Elemente S. 39)

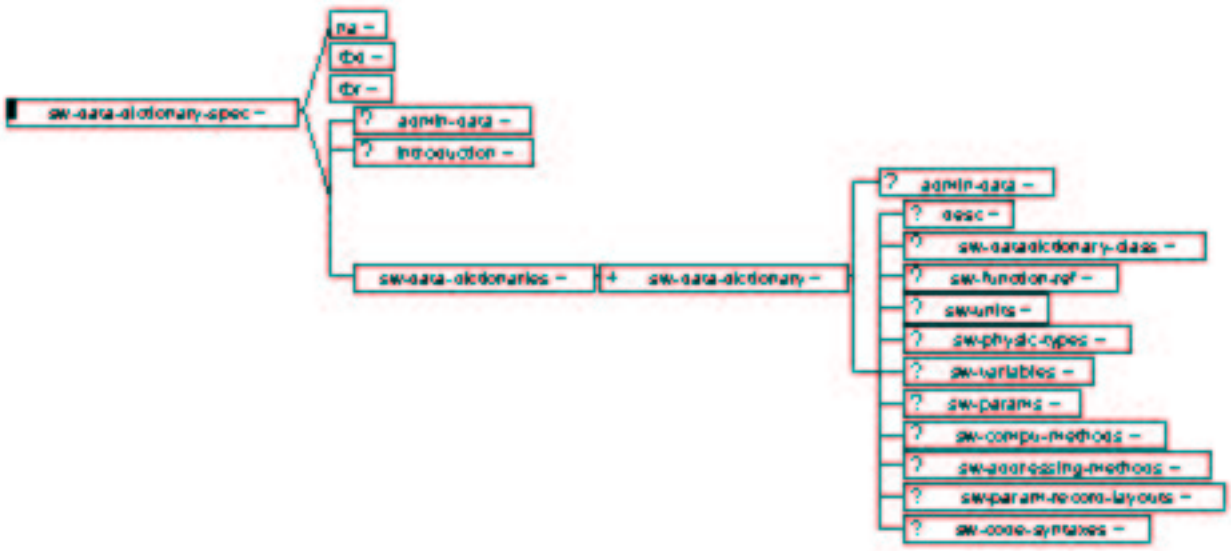

Abbildung 8: Data-Dictionary

2.2.3.1 Maßeinheiten

Unter **<sw-units>** können Maßeinheiten beschrieben werden. Eine Maßeinheit besitzt eine Lang- und eine Kurzbezeichnung (**<long-name>** bzw. **<short-name>**). Mit dem Element **<sw-unit-display>** kann die formatierte Ausgabe bestimmt werden.

Zur Abbildung der **<sw-unit>** auf eine SI-Einheit wird das Element **<si-unit>** benutzt. Dieses Element besitzt ein Attribut für jede SI-Basiseinheit mit dem die zugehörigen Exponenten angegeben werden[5]. Dies wird benötigt wenn man mit Werkzeugen arbeitet, die mit SI-Einheiten rechnen können.

Diese Abbildung auf SI-Einheiten macht die verwendeten Maßeinheiten vergleichbar.

Es ist möglich, einen Bezug zu anderen Maßeinheiten (vorzugsweise SI-Einheiten) über **<sw-unit-ref>** herzustellen. Die Vorschrift zur Umrechnung auf diese Einheit steht in **<sw-unit-to-ref-method>**. Die Vorschrift zur Umrechnung von dieser Einheit steht in **<sw-unit-from-method>**. Diese Methoden werden als 6 parameter entsprechend **<sw-asap-6-prm-method>** spezifiziert.

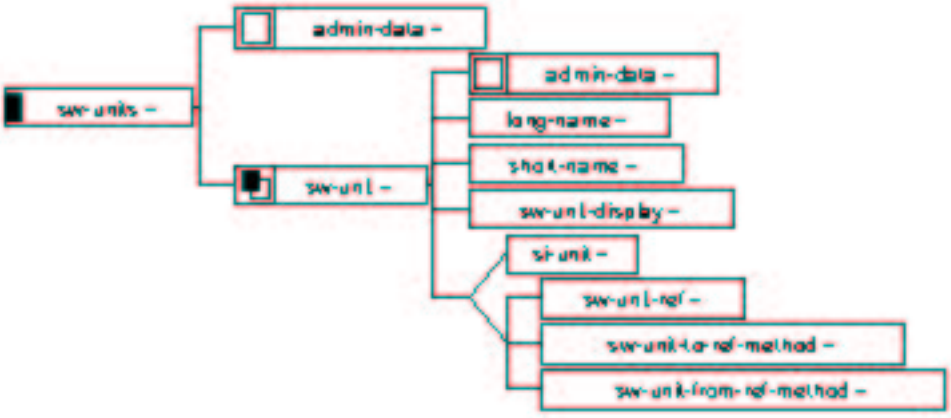

Abbildung 9: Maßeinheiten

Folgendes Beispiel verdeutlicht die Zusmmenhänge:

2.2.3.2 Physikalische Datentypen

In den frühen Phasen der Entwicklung von Steuergerätesoftware wird unabhängig von von der Implementierung in physikalischen Größen gearbeitet. Zur Unterstützung dieser Phase können im Datenlexikon global physikalische Datentypen (**<sw-physic-types>**) definiert werden.

Bei der Definition von Variablen (**<sw-variable>**) kann auf die bereits im vorhandenen physikalischen Datentypen durch Referenzierung (**<sw-physic-type-ref>**) Bezug genommen werden werden.

Es besteht jedoch auch die Möglichkeit direkt bei der Variablen den physikalischen Datentyp zu definieren (**<sw-physic-type-1>**). Weitere Einzelheiten siehe Abbildung 10 Physikalische Datentypen S. 22 .

5

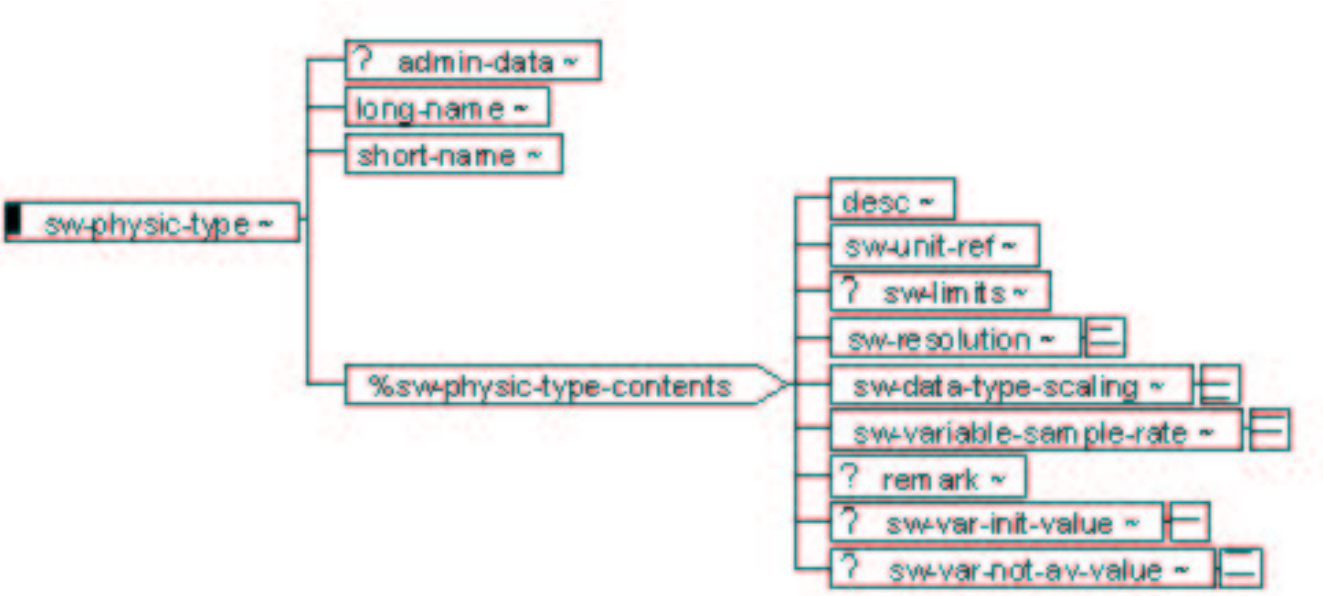

Abbildung 10: Physikalische Datentypen

Folgendes Beispiel zeigt die Definition eines physikalischen Datentyps:

2.2.3.3 Variablen

Variablen werden im Datadictionary definiert (**<sw-variables>**). Sie sind grundsätzlich global zu behandeln, d.h. ihr **<short-name>** muß eindeutig sein.

Funktionen verwenden diese Variablen zum Austausch von Informationen. Dazu werden diese Variablen entsprechend ihrer Verwendung referenziert (**<sw-function-variables>**).

Variablen werden aber auch als Eingangsgrößen von Kenngrößen und als Zwischenwerte verwendet.

Variablen können mit **<admin-data>** versehen werden, um z.B. Bibliotheksmanagement zu unterstützen.

Die Struktur zur Beschreibung der Software-Variablen ist rekursiv. Dies ermöglicht den Aufbau von hierarchischen Variablenstrukturen, die z.B. ein *STRUCT*-Konstrukt (in der Programmiersprache *C*) abbilden können.

Die Struktur von**<sw-variable>** ist in Abbildung 11 Variable S. 23 dargestellt. Beschreibung der Elemente siehe Topic 3.2 Beschreibung der Elemente S. 39.

Abbildung 11: Variable

Bei der Beschreibung von Datentypen (Variablen und Kenngrößen) ist zu beachten, daß Daten stets eine interne (coded) und eine externe (physikalische) Darstellung besitzen (z.B. für eine Motordrehzahl intern 0..200: extern 0..2000 U/min, intern 201..254: nicht definiert und intern 255: Motordrehzahl nicht verfügbar). Zwischen diesen Darstellungen wird über die in **<sw-compu-method-ref>** referenzierte Umrechnungsformel konvertiert.

Daher kann **<sw-limits>** sowohl physikalisch (**<phys>**)als auch codiert (**<phys>**) angegeben werden.

Der interne Wert für *nicht verfügbar* wird in **<sw-var-not-av-value>** eingetragen. Dies ist der Wert, den die Variable annimmt, wenn der zugehörige externe Wert nicht verfügbar ist (sei es daß er noch nicht erfaßt ist, oder aber ein Fehler vorliegt). Einen externen Wert für *nicht verfügbar* kann es nicht geben, da er ja nicht verfügbar ist.

Nicht alle Variablen werden in einem Applikatinssystem für die Kalibrierung benötigt. Diese Tatsache wird durch das Attribut **[calibration]** dokumentiert Es kann die Werte *calibration*, *no-calibration* und *not-in-memory* annehmen. Dieses Attribut wird genauso bei Kenngrößen verwendet.

2.2.3.4 Kenngrößenstrukturen

Ein Steuergeräte-Programm kann über Kenngrößen parametriert und quantitativ auf das jeweilige Zielfahrzeug angepaßt werden. Durch das Belegen dieser Kenngrößen mit geeigneten Werten wird das Steuergeräteprogramm an den zu bedienenden Motor in seinem Umfeld (z.B. Karosserie, Abgasgesetzgebung) angepaßt. Diesen Anpassungsvorgang nennt man Applikation bzw. Kalibrierung.

Es ist zu unterscheiden zwischen der Struktur einer Kenngröße und ihren Dateninhalten. Die Kenngrößenstrukturen werden im Datenlexikon unter **<sw-params>** (vgl. Abbildung 12 Parameter S. 24) beschrieben. Die Kenngrößeninhalte stehen in **<sw-param-contents>**.

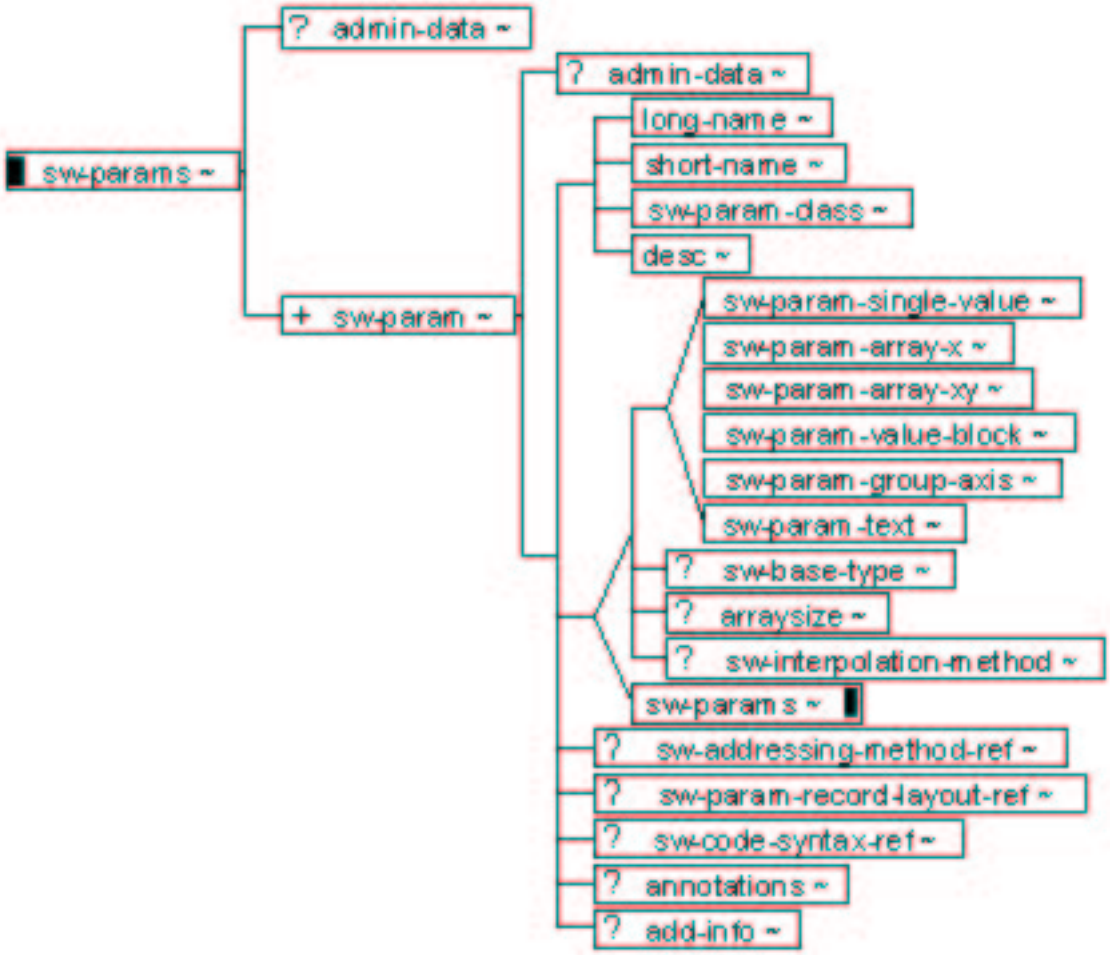

Abbildung 12: Parameter

Die Verwendung von Kenngrößen in Funktionen erfolgt durch Referenzierung (**<sw-param-ref>**). Dabei wird durch das Attribut [owns] mit dem möglichen Wert "noowns"[6] angegeben, ob die Kenngröße dieser Funktion zuzuordnen ist bzw. in dieser Funktion definiert wird.

Über ein Attribut **[calibration]** kann spezifiziert werden:

calibration Die Inhalte der Kenngröße werden in der Applikationsphase (vgl. Topic 1.2.8 Applikation (Kalibrierung) S. 11) auf das jeweilige Zielfahrzeug angepaßt (appliziert).

no-calibration Die Inhalte der Kenngröße können nicht durch Applikationssyteme usw. verändert, sondern nur gelesen werden.

not-in-memory Diese Kenngröße wird nicht im EPROM des Steuergerätes abgelegt. Ihre Inhalte werden während der Software-Entwicklung verarbeitet (z.B. zur Konfiguration über Headerfiles). Sie sind daher in der Applikationsphase weder lesbar noch veränderbar.

Es gibt verschiedene Klassen von Kenngrößen. Diese werden einerseits durch die Belegung von **<sw-param-class>** unterschieden, andererseits durch entsprechende Ausprägung der Strukturen gebildet.

6

Man unterscheidet folgende Kenngrößenklassen:

Kennwert Ein Kennwert (**<sw-param-single-value>**) ist eine Kenngröße, die aus einem einzelnen Wert besteht. Hiermit kann beispielsweise eine Drehzahlgrenze abgebildet werden.

Die Struktur des Kennwertes ist in Abbildung 13 Kennwert S. 26 dargestellt.

Systemkonstante Eine Systemkonstante ist ein spezieller Kennwert (**<sw-param-single-value>** gekennzeichnet durch **[calibration]**=*not-in-memory*), welcher z.B. zur Anpassung von Umrechungsformeln dient. Sie wird nicht im Steuergerät abgelegt.

Kenntext Ein Kenntext (**<sw-param-text>**) ist eine Kenngröße, die aus einem einzelnen Text besteht. Hiermit kann beispielsweise Meldungen für das Armaturenbrett dargestellt werden.

Kennlinie Bei einer Kennlinie (**<sw-param-array-x>**) wird die Ausgangsgröße in Abhängigkeit von einer Eingangsgröße durch das Steuergeräte-Programm berechnet. Mathematisch betrachtet ist die Kennlinie eine Funktion, bei der gilt: Ausgangswert = f(Eingangswert).

Diese Funktion ist dabei als Linienzug festgelegt, indem zu einer (durch **<max-count>** begrenzten) Anzahl von Stützstellen (**<sw-param-axis-x>**) der zugehörige Funktionswert (**<sw-param-axis-value>**) angegeben wird. Zwischen zwei Stützstellen wird im Steuergeräte-Programm üblicherweise linear interpoliert, außerhalb konstant extrapoliert.

Die Struktur der Kennline ist in Abbildung 14 Kennlinie S. 27 dargestellt.

Festkennlinie Bei Festkennlinien können die Stützstellen nicht frei festgelegt werden, sondern werden durch shift (**<shift>**) und offset (**<offset>**) im Steuergeräte-Programm berechnet (**<sw-axis-shift-offset>**). Die Festkennlinie wird ebenfalls als **<sw-param-array-x>** dargestellt, wobei **<sw-axis-shift-offset>** verwendet wird.

Kennfeld Bei einem Kennfeld (**<sw-param-array-xy>**) wird die Ausgangsgröße in Abhängigkeit von zwei Eingangsgrößen durch das Steuergeräte-Programm berechnet. Ansonsten ist der Aufbau wie bei einer Kennlinie.

Festkennfeld Wie bei einer Festkennlinie können auch hier die Stützstellen nicht frei gewählt werden. Das Festkennfeld wird ebenfalls als **<sw-param-array-xy>** dargestellt, wobei für beide Achsen **<sw-axis-shift-offset>** verwendet wird.

Gruppenkennlinie Gruppenkennlinien haben keine eigenen Stützstellen, sondern benutzen gemeinsam mit anderen Gruppenkennlinien oder -feldern die gleichen Stützstellen (**<sw-param-group-axis>** referenziert in **<sw-axis-grouped>**). Das Gruppenkennline wird ebenfalls als **<sw-param-array-x>** dargestellt, wobei für die Achse **<sw-axis-grouped>** verwendet wird.

Gruppenkennfeld Gruppenkennfelder haben keine eigenen Stützstellen, sondern benutzen gemeinsam mit anderen Gruppenkennlinien oder -feldern die gleichen Stützstellen (Gruppenstützstellen). Das Gruppenkennfeld wird ebenfalls als **<sw-param-array-xy>** dargestellt, wobei für beide Achsen **<sw-axis-grouped>** verwendet wird (vgl. Topic 2.2.3.4 Beispiel für ein Gruppenkennfeld S. 27).

Kennwerteblock Ein Kennwerteblock **<sw-param-value-block>** faßt mehrere (**<count>**) identisch aufgebaute Kennwerte zu einem Block zusammen. So könnte z.B. ein Kennwert, der für jeden Zylinder des Motors einzeln anzulegen ist, als Kennwerteblock mit 4 Elementen dargestellt werden.

Alle Kennwerte im Block besitzen die gleichen Eigenschaften. Jeder Kennwert erhält jedoch eine eigene Bezeichnung (**<label>**), in der die Verwendung des Wertes beschrieben werden kann. Der Kennwerteblock wird immer geschlossen behandelt. Daher können die Komponenten des Kennwerteblockes nicht einzeln referenziert werden.

Kennwertestruktur Eine Kennwertestruktur (**<sw-params>** in **<sw-param>**) faßt mehrere verschieden aufgebaute Kenngrößen zusammen. Damit können z.B. die Parameter einer Reglerstruktur dargestellt werden (bgl. Topic 2.2.3.4 Beispiel für eine Kennwertestruktur: S. 27).

Gruppenstützstellen Gruppenstützstellen (**<sw-param-group-axis>**) legen die Stützstellen fest, die von mehreren Gruppenkennlinien oder Gruppenkennfeldern gemeinsam benutzt werden (**<sw-param-ref>** in **<sw-axis-grouped>**).

Durch **<sw-param-class>**wird festgelegt, um welche Kenngrößenart es sich handelt (z.B. Kennlinie). Hierfür sollen die normierten Bezeichnungen von ASAP verwendet werden. Dies sind[7]:

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

Wert[8]	Bedeutung	Anmerkung
SYSTEM_CONSTANT	Systemkonstante	Entspricht strukturell einem Festwert mit **[calibration]**=*not-in-memory*
VALUE	Kennwert	
CURVE_INDIVIDUAL	Kennlinie	
MAP_INDIVIDUAL	Kennfeld	
CURVE_FIXED	Festkennlinie	
MAP_FIXED	Festkennfeld	
CURVE_GROUPED	Gruppenkennlinie	
MAP_GROUPED	Gruppenkennfeld	
ASCII	Kenntext	
STRUCTURE	Kenngrößenstruktur	
VALUE_BLOCK	Kennwerteblock	
AXIS_VALUES	Gruppenstützstellen	

Beispiel für einen Kennwert

Die Struktur eines Kennwertes ist in Abbildung 13 Kennwert S. 26 dargestellt.

Abbildung 13: Kennwert

Folgendes Beispiel illustriert die Anwendung:

Beispiel für eine Kennlinie

Die Struktur der Kennlinie ist in Abbildung 14 Kennlinie S. 27 dargestellt.

7

8

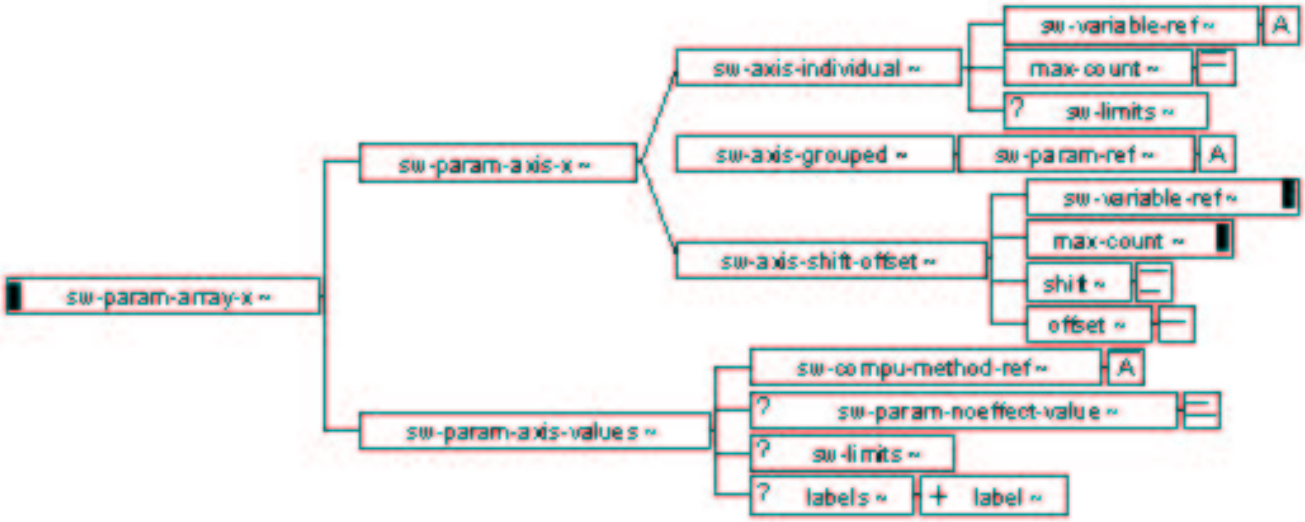

Abbildung 14: Kennlinie

Nachfolgendes Beispiel illustriert die Anwendung:

Beispiel für ein Gruppenkennfeld

Beispiel für eine Kennwertestruktur:

2.2.3.5 Umrechnungsformeln

In **<sw-compu-methods>** wird die Abildung von physikalischen Werten in Steuergeräte-interne Werte und umgekehrt definiert (vgl. Abbildung 15 Umrechnungsformeln S. 28). Man unterscheidet vier verschiedene Umrechnungsmethoden:

Polynomdarstellung Die Umsetzung wird durch sechs Parameter (**<sw-asap-6-prm-method>**) spezifiziert (vgl. Topic 2.2.3.5 Beispiel für Polynomdefinition S. 28). Die werden durch Leerzeichen getrennt und bilden folgende Formel ab:

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

Tabellarische Definition Die Umsetzung wird in **<sw-compu-method-table>** tabellarisch definiert (vgl. Topic 2.2.3.5 Beispiel für eine tabellarische Umrechnungsformel S. 28). Pro Wertepaar (**<sw-compu-method-value-pair>**) wird der Steuergeräteinterne Wert (**<cmt-int>**)[9] und der physikalische Wert (**<cmt-phys>**) angegeben.

Die Art der Interpolation wird in **[interpolation-style]** spezifiziert.

textuelle Darstellung Diese Umsetzung (**<sw-compu-method-text>**) bildet einen steuergeräteinternen (**<cmt-int>**) Wert in einen Text (**<cmt-text>**) (z.B. einen Zustandsbezeichner) ab. Pro Wertepaar wird ein **<sw-compu-method-text-pair>** angelegt (vgl. Topic 2.2.3.5 Beispiel für eine textuelle Umsetzung S. 28).

9

Programmcode — Unter **<prog-code>** kann die Umrechung in der Notation einer Programmiersprache spezifiziert werden. Programmiersprache (**[prog-lang]**), Programmiersprachendialekt (**[lang-subset]**) und verwendete Bibliotheken (**[used-libs]**) werden als Attribute angegeben.

Mathematische Darstellung — Die Umsetzung wird als allgemeiner math. Ausdruck (**<sw-compu-generic-math>**) spzeifizert. Der Inhalt ist noch offen, d.h. bilateral zu vereinbaren.[10]

Abbildung 15: Umrechnungsformeln

Beispiel für Polynomdefinition

Beispiel für eine tabellarische Umrechnungsformel

Beispiel für eine textuelle Umsetzung

2.2.3.6 Adressierverfahren

Die Kenngrößen und Variablen können im Steuergerät auf verschiedene Weise angesprochen werden. Die möglichen Adressierverfahren können unter **<sw-adressing-methods>** mit einem formalen Namen und einer informellen Beschreibung (**<sw-adressing-method-desc>**) aufgeführt werden. Damit können formale Verweise (**<sw-adressing-method-ref>**) aufgebaut werden. Weitere Einzelheiten siehe Topic 3 Beschreibung der Elemente und der Attribute S. 39

2.2.3.7 Speicherlayouts

Die Kenngrößen und Variablen können im Steuergerät auf verschiedene Weise abgelegt werden. Die möglichen Ablageschemata können unter **<sw-param-record-layout>** mit einem formalen Namen und einer informellen Beschreibung (**<sw-param-record-layout-desc>**) aufgeführt werden. Damit können formale Verweise (**<sw-param-record-layout-ref>**) aufgebaut werden. Weitere Einzelheiten siehe Topic 3 Beschreibung der Elemente und der Attribute S. 39.

2.2.3.8 Code-Syntaxen

Die Kenngrößen und Variablen können von Codegeneratoren in die Quelltexte für den Compiler eingebaut werden. Zur Steuerung dieser Codegeneratoren werden Code-Syntaxen verwendet. Die möglichen Adressierverfahren können unter **<sw-code-syntaxes>** mit einem formalen Namen und einer informellen Beschreibung (**<sw-code-syntax-desc>**) aufgeführt werden. Damit können formale Verweise (**<sw-adressing-method-ref>**) aufgebaut werden. Weitere Einzelheiten siehe Topic 3 Beschreibung der Elemente und der Attribute S. 39

10

2.2.3.9 Abgleich mit ASAP

Der Umfang des Datenlexikons wurde wesentlich durch einen Abgleich mit den Inhalten von *ASAP2* bestimmt. Dies betrifft vor allem:

- **<sw-variable>**
- **<sw-param>**
- und die Einführung von **<sw-compu-method>**.

Folgende Tabelle zeigt die bei *ASAP* verwendeten Begriffe und die MSR Entsprechungen.

Tabelle 3: Gegenüberstellung ASAP-/MSR-Begriffe

ASAP	***MSR***
characteristics	Parameter/Kenngrößen (**<sw-param>**)
measurement	Variable (**<sw-variable>**)
compu-method	Umrechnungsformel (**<sw-compu-method>**)

Die Elementnamen werden angelehnt an die in MSR verwendeten Begriffe gewählt, um Überschneidungen mit bereits exisiterenden MSR-Elementnamen (Characteristic, Measurement) zu vermeiden. Parameter/Kenngrößen (**<sw-param>**), Variable (**<sw-variable>**). Für die Umrechnungsformel wird **<sw-compu-method>** verwendet.

2.2.4 Funktionen

Eine Funktion ist eine bestimmte Anforderung oder Eigenschaft des Gesamtsystems, die durch Hardware und/oder Software realisiert werden kann. Eine Funktion ist aber auch die Lösung einer Steuerungs-, Regelungs- oder Anzeige-Aufgabe. Man kann solche Funktionen auch als Anwendungsfunktionen bezeichnen (vgl. Abbildung 16 Anwendungsfunktionen S. 30).

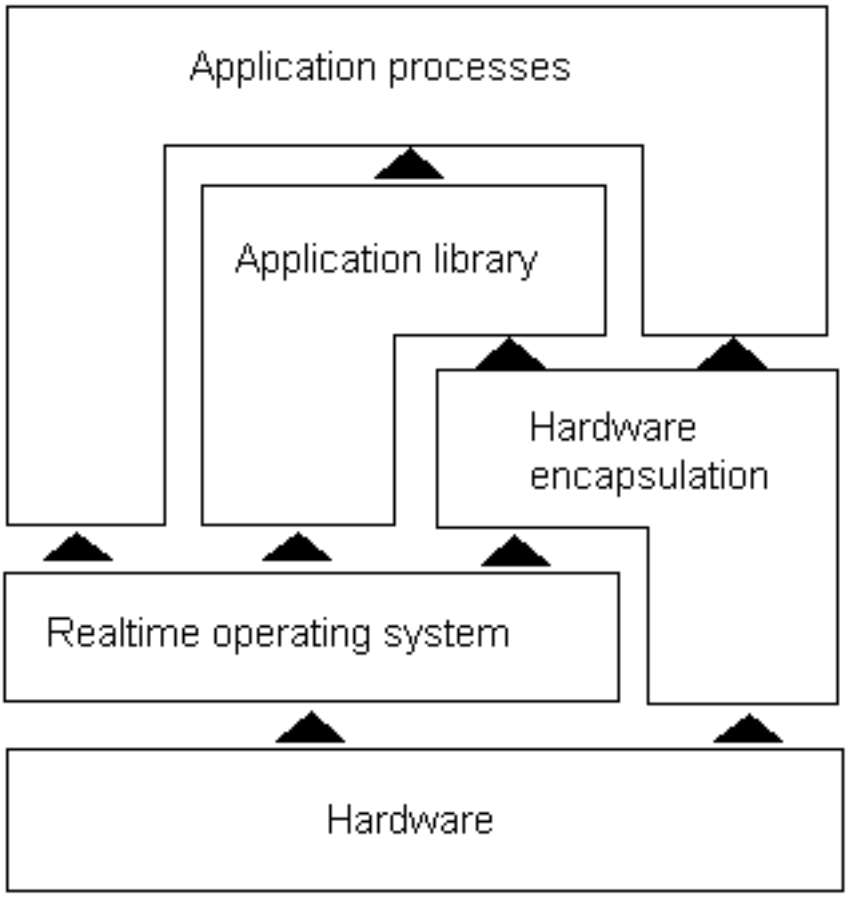

Abbildung 16: Anwendungsfunktionen

Die in Software realisierten Funktionen werden in **<sw-function-spec>** beschrieben. Diese Funktionen werden in Klassen eingeteilt (**<sw-function-class>**). Die Funktionen können variantenspezifisch beschrieben werden. Für jede Funktion muß mindestens eine Funktionsvariante existieren (**<sw-function-variant>**).

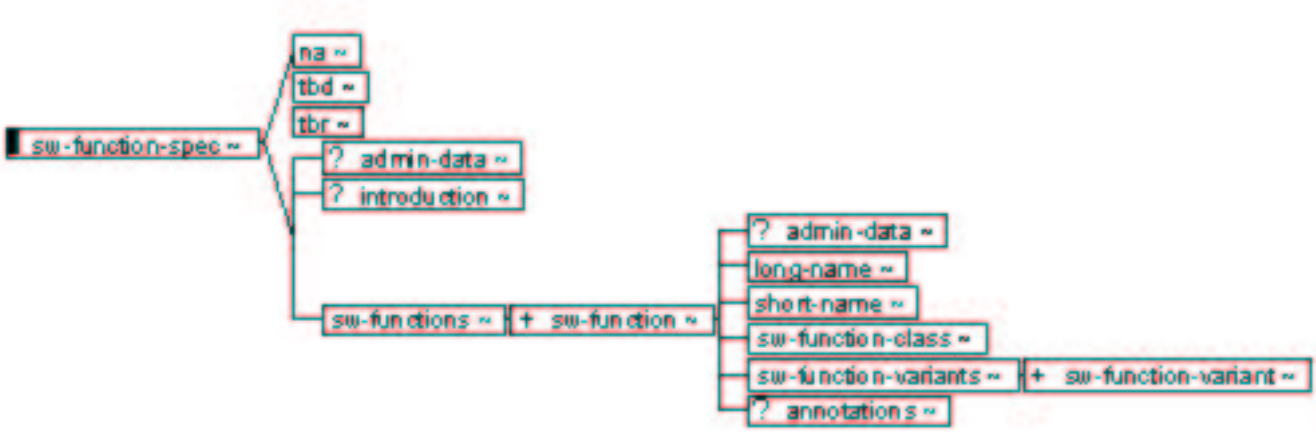

Abbildung 17: Funktionsspezifikation

2.2.4.1 Funktionsvarianten

Die Funktionsvariante (**<sw-function-variant>**) wird in folgenden Einzelheiten beschrieben (vgl. Abbildung 4 Funktionsvariante S. 13):

Variantenkennung Unter **<variant-def-refs>** werden Verweise auf die Varianten angegeben, für die die vorliegende Beschreibung gilt (vgl. Topic 2.1.2 Variantenbehandlung S. 13).

Funktionsdefinition Unter **<sw-function-def>** wird eine semiformale Definition der Funktion gegeben. Dieses Element wird i.d.R. aus Systementwurfswerkzeugen automatisch beschickt. Die obligatorischen Bilder sind zur Aufnahme von Diagrammen aus dem *Systementwurfswerkzeug* vorgesehen(vgl. Topic 2.2.4.1.1 Funktionsdefinition S. 32).

Funktionsbeschreibung Unter **<sw-function-desc>** wird eine informelle Beschreibung der Funktion abgelegt (vgl. Topic 2.2.4.1.2 Funktionsbeschreibung S. 32). Diese wird in aller Regel manuell erstellt.

Anforderungsverweis Unter **<sw-fulfils>** wird (dokumentübergreifend) auf die Systemfunktion in der *MSRSYS.DTD* verwiesen (vgl. Topic 2.2.4.1.3 Verweis auf Komponentenverhalten S. 32).

Funktionsvariablen Unter **<sw-function-variables>** werden die in der Funktion verwendeten Variablen aufgeführt. Dabei wird auch die Art der Verwendung spezifiziert (vgl. Topic 2.2.4.1.4 Funktionsvariablen S. 32).

Funktionskenngrößen Unter **<sw-param-refs>** werden die in der Funktion verwendeten Kenngrößen aufgeführt. Dabei wird über das Attribut **[owns]** auch die Art der Verwendung spezifiziert (vgl. Topic 2.2.4.1.5 Funktionskenngrößen S. 33).

Lokales Datenlexikon Unter **<sw-data-dictionary-spec>** wird ein funktionslokales Datenlexikon angelegt (vgl. Topic 2.2.3 Datenlexikon S. 19).

Lokale Kenngrößeninhalte Unter **<sw-param-contens-spec>** werden funktionslokale Kenngrößeninhalte abgelegt (vgl. Topic 2.2.4.1.6 Funktionsbezogene (lokale) Kenngrößeninhalte S. 33).

Testspezifikationen Unter **<sw-test-spec>** können Testspezifikationen abgelegt werden (vgl. Topic 2.2.4.1.7 Testspezifikation S. 33).

Applikationshinweise Unter **<sw-application-notes>** können Hinweise zur Kalibrierung (Applikation) der vorliegenden Funktion gegeben werden (vgl. Topic 2.2.4.1.8 Applikationshinweise S. 33).

Kundendiensthinweise Unter **<sw-maintenance-notes>** können Kundendiensthinweise mit Bezug auf die vorliegende Funktion gegeben werden (vgl. Topic 2.2.4.1.9 Kundendiensthinweise S. 33).

CARB-Dokumentation Unter **<sw-carb-doc>** kann die Dokumentation für *CARB* (*california air resource board*) beschrieben werden (vgl. Topic 2.2.4.1.10 CARB-Dokumentation S. 33).

Funktionszerlegung Unter **<sw-decomposition>** kann die Zerlegung der aktuellen Funktion in Teilfunktionen angegeben werden. Daraus kann eine Übersicht über die gesamte Funktionshierarchie generiert werden (vgl. Topic 2.2.4.1.11 Funktionszerlegung S. 34).

Notizobjekte Unter **<annotations>** können Informationen im Prozeß weitergereicht werden (vgl. Topic 3 Beschreibung der Elemente und der Attribute S. 39).

zusätzliche Informationen Unter **<add-info>** können Informationen und Beschreibungen abgelegt werden, für die es keine expliziten Strukturen in *MSRSW.DTD* gibt (vgl. Topic Anh. E.4 Zusätzliche Informationen S. 100).

Abbildung 18: Funktionsvariante

2.2.4.1.1 Funktionsdefinition

Für jede Software-Funktion kann in **<sw-function-def>** eine Definition mit formalen Anteilen (z. B. ein regelungstechnisches Blockschaltbild) dokumentiert werden. Dieses Element wird in der Regel durch die Ausgabe eines *CASE-Tools* bzw. *Systementwurfswerkzeug*s beschickt werden.

Dazu müssen die Report-Fähigkeiten der CASE-Tools genutzt werden. Die Ausgabe kann sowohl Grafiken als auch freie Texte umfassen.

Im Interesse reproduzierbarer Prozesse sollte dieses Element entweder voll manuell gepflegt werden, oder aber keine manuellen Eingaben enthalten.

2.2.4.1.2 Funktionsbeschreibung

In **<sw-function-desc>** kann eine detaillierte Beschreibung der Funktion erfolgen. Diese wird i. d.R. manuell erstellt. Formale, automatisch erstellte Anteile sollten in **<sw-function-def>** abgelegt werden. Unter **<prms>** können auch formale Parameter eingegeben werden, welche dann auch maschinell prüfbar sind. Dies betrifft Randbedingungen, Eigenschaften einer Funktion, welche nicht in **<sw-param>** spezifiziert werden, weil sie fest im Programm codiert sind, oder sich gar aus der Implementierungstechnik ergeben.

2.2.4.1.3 Verweis auf Komponentenverhalten

In dem Element **<sw-fulfils>** können Verhaltens-Funktionen des Steuergerätes referenziert werden, die durch die Software-Funktion realisiert werden. Dadurch ist es möglich, Funktionen zunächst implementierungsunabhängig (in einer *MSRSYS.DTD* - Instanz) zu beschreiben, und im Rahmen der Partitionierung die Korrespondenz zwichen Verhalten und Software-Funktion herzustellen.

2.2.4.1.4 Funktionsvariablen

In **<sw-function-variables>** werden die Variablen aufgeführt die von der Software-Funktion gelesen oder manipuliert werden. Die eigentliche Variablenbeschreibung befindet sich dabei im Datadictionary (**<sw-variables>**); dorthin wird referenziert. Bei den Variablen wird zwischen der Definition von Variablen und dem Zugriff auf Variable unterschieden.

Wenn eine Funktion eine Variable definiert, d.h. eine Variable bzw. den Wert zur Verfügung stellt, so so ist sie unter **<sw-function-export-variables>** aufzuführen.

In **<sw-function-import-variables>**) werden Variablen aufgeführt, die importiert bzw. von anderen Funktionen definiert werden. **<sw-function-local-variables>** legt fest, daß es sich nicht um eine Übergabevariable, sondern um eine lokale Variable handelt.

In **<sw-function-modelonly-variables>** werden Variablen aufgeführt, die nur im Modell (z.B. zur Vereinfachung der Dokumentation) existieren. Diese erscheinen nicht in der Implementierung im Steuergerät.

Hinsichtlich des Zugriffs auf Variable wird zwischen lesend (**<sw-variable-read>**), schreibend (**<sw-variables-write>**) und lesend und schreibend (**<sw-variables-readwrite>**) unterschieden.

Im nachfolgenden Beispiel wird die Verwendung von Variablen und Kenngrößen in Funktionen gezeigt.

2.2.4.1.5 Funktionskenngrößen

Die zugehörigen Kenngrößen der Software-Funktion werden in **<sw-param-refs>** aufgeführt. Die Kenngrößen werden durch Referenzierung (**<sw-param-ref>**) den Funktionen zugeordnet. Durch das Attribut **[noown]** kann bestimmt werden, ob eine Funktion eine Kenngröße definiert (die Kenngröße also zu der Funktion gehört oder ob die Funktion eine Kenngröße importiert (owns="noown")[11].

2.2.4.1.6 Funktionsbezogene (lokale) Kenngrößeninhalte

In **<sw-param-contents>** können innerhalb von **<sw-function-variant>** Kenngrößeninhalte spezifiziert werden. Sie werden dann als lokale Kenngrößeninhalte bezeichnet (vgl. Topic 2.1.4 Ablage von Kenngrößeninhalten S. 14).

Einzelheiten zur Kenngrößeninhalte sind unter Topic 2.2.5 Spezifikation von Kenngrößeninhalten S. 34 beschrieben.

2.2.4.1.7 Testspezifikation

Unter **<sw-test-spec>** können für jede Funktionsvariante die Testverfahren spezifiziert werden.

2.2.4.1.8 Applikationshinweise

Unter **<sw-application-notes>** können Hinweise auf Besonderheiten zur Applikation (Kalibrierung) der Funktion beschrieben werden.

2.2.4.1.9 Kundendiensthinweise

Unter **<sw-maintenance-notes>** können Informationen abgelegt werden, welche später z.B. in die Kundendienstunterlagen übernommen werden.

11

2.2.4.1.10 CARB-Dokumentation

Unter **<sw-carb-doc>** wird die für *CARB* benötigte Dokumentation eingetragen.

2.2.4.1.11 Funktionszerlegung

Die Software-Funktionen werden flach beschrieben. Die Zerlegung (**<sw-decomposition>**) der Software-Funktion in Details kann durch Referenzierung auf andere Software-Funktionen durchgeführt werden. Damit wird eine Mehrfachverwendung von Subfunktionen unterstützt[12].

2.2.5 Spezifikation von Kenngrößeninhalten

Kenngrößeninhalte können unter **<msrsw>** global (vgl. Topic 2.1.4 Ablage von Kenngrößeninhalten S. 14) beschrieben werden (**<sw-param-contents>**). Durch **<sw-function-ref>** kann ein Kenngrößeninhalt einer Funktion zugeordnet werden. Eine Klassifizierung des Kenngrößeninhalts ist durch **<sw-param-contents-class>** möglich, z.B.: "*Testdaten*", "*Rohapplikationsdaten*". Ein Kenngrößeninhalt kann mehrfach angegeben werden. Das ist sinnvoll, wenn für eine Kenngröße z.B mehrere Testdaten angegeben werden sollen.

Der Inhalt selbst wird durch **<sw-param-ref>** einer Kenngröße zugeordnet. Kenngrößenstruktur (in **<sw-param>**) und Kenngrößeninhalt (in **<sw-param-contents>**) müssen strukturell übereinstimmen (d.h. z.B. die gleiche **<sw-param-class>** tragen).

Die Kenngrößeninhalte gliedern sich in bis zu drei Achsen (**<sw-param-content-x>**, **<sw-param-content-y>**, **<sw-param-content-v>** bzw. **<sw-param-content-text>**) um die Strukturen in Topic 2.2.3.4 Kenngrößenstrukturen S. 23 mit Daten befüllen zu können. Bei Kennfeldern werden alle Werte zeilenweise in **<sw-param-content-v>** abgelegt, wobei der Index für die X-Achse schneller läuft.

Pro Achse können die Daten in verschiedenen Formaten spezifizert werden:

<sw-param-values-phys>	Physikalische Werte, angegeben ggf. als Fließkommzahl
<sw-param-values-coded>	Steuergeräte-interne Werte, angegeben als Integer bzw. als Fließkommazahl
<sw-param-values-coded-hex>	Steuergeräte-interne Werte, angegeben als Hexadezimalzahl im "*C*"-Format (z.B. "*0x7f2a*")
<sw-param-values-adr>	Adresse des Wertes im Steuergerät angegeben als Hexadezimalzahl im "*C*"-Format (z.B. "*0x7f2a*"). Diese Adresse dient vorwiegend zur Dokumentation (z.B. Umprogrammierungsanleitung).
<sw-param-values-generic>	Hier können weitere Formate abgelegt werden. Die Formate selbst werden durch das Attribut **[type]** festgelegt.

Die Kenngrößeninhalte sind so konzipiert, daß sie auch ohne Datenlexikon verwendet werden können, um z.B. eine physikalische Archivierung vorzunehmen. Daher sind die Elemente **<sw-param-ref>**, **<sw-param-class>**, **<sw-unit>** optional nocheinmal vorhanden. Die dabei mögliche Redundanz wird in Kauf genommen, da diese Elemente ohnehin maschinell generiert werden. Der Verweis **<sw-param-ref>** kann nicht mehr über ID/IDREF geführt werden, da in der Instanz nur Kenngrößeninhalte und nicht die Parameterdefinition enthalten sein können. Es bleibt also nur die natürliche Adressierung[13].

12

13

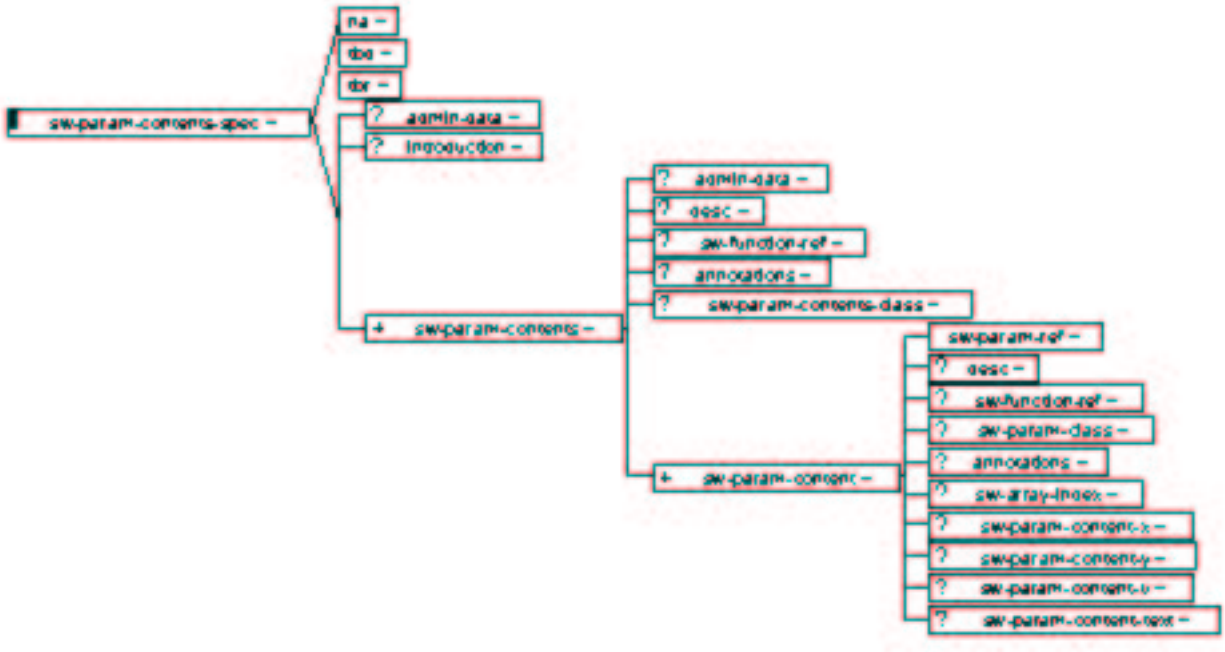

Abbildung 19: Parameterinhalte (Übersicht)

Abbildung 20: Parameterinhalte (Details)

Folgende Beispiele zeigen den Kenngrößeninhalt eines Kennwertes und einer Kennlinie:

2.2.6 Glossar für das Dokument

Es kann ein Glossar für das entsprechende Dokument angelegt werden (**<sw-glossary>**).

2.2.7 Zusätzliche Spezifikationen

<add-spec> erlaubt die Erfassung zusätzlicher Spezifikationen, die in der DTD nicht explizit vorgesehen sind (beipielsweise ein Abkürzungsverzeichnis).

2.3 Referenzen

Zwischen den oben aufgeführten Elementen werden zusätzliche Beziehungen eingeführt, die als Referenzen implementiert werden. Die folgende Tabelle enthält eine Auflistung aller in diesem Kontext zu betrachtenden Referenzarten, inklusive der bereits existierenden.

Tabelle 4: Referenzarten

Nr.	Bezeichnung	Quellelement	Zielelement	Bedeutung
1	**<system-ref>**	**<requirements>**, **<product-spec>**	**<part-type>**	Legt für jede der beiden Sichten den obersten Part-Type (genau einen) als System fest.
2	**<part-type-ref>**	**<part>**	**<part-type>**	Ordnet einem Part den eigenen Part-Type (genau einen) zu.
3	**<sw-function-ref>**	**<software>**	**<sw-function>**	Ordnet der Software eines Part-Type Software-Funktionen (keine, eine oder mehrere) zu, die innerhalb der Software eines anderen Part-Types beschrieben sind, z.B. mehrfach verwendete Funktionen wie Betriebssystem- oder Netzwerkmanagement-funktionen.
4	**<sw-function-ref>**	**<sw-function>**	**<sw-function>**	Ordnet einer Software-Funktion Software-Funktionen (keine, eine oder mehrere) innerhalb derselben Softwarebeschreibung zu, welche die Dekomposition der erstgenannten darstellen.
5	**<fulfills-ref>**	part, sw-function	function	Ordnet einem Part oder einer Software-Funktion diejenigen (Verhaltens-)Funktionen (keine, eine oder mehrere) zu, die es bzw. sie realisiert.
6	**<function-view-ref>**	function	function	Ordnet einer spezifizierten (Verhaltens-) Funktion in der Sicht Zusage diejenige geforderte (Verhaltens-)Funktion in der Sicht Vorgabe zu, welche durch erstgenannte realisiert wird.[14]

Die folgende Abbildung stellt die Referenzarten exemplarisch dar (mit Ausnahme der Nr. 6).

14

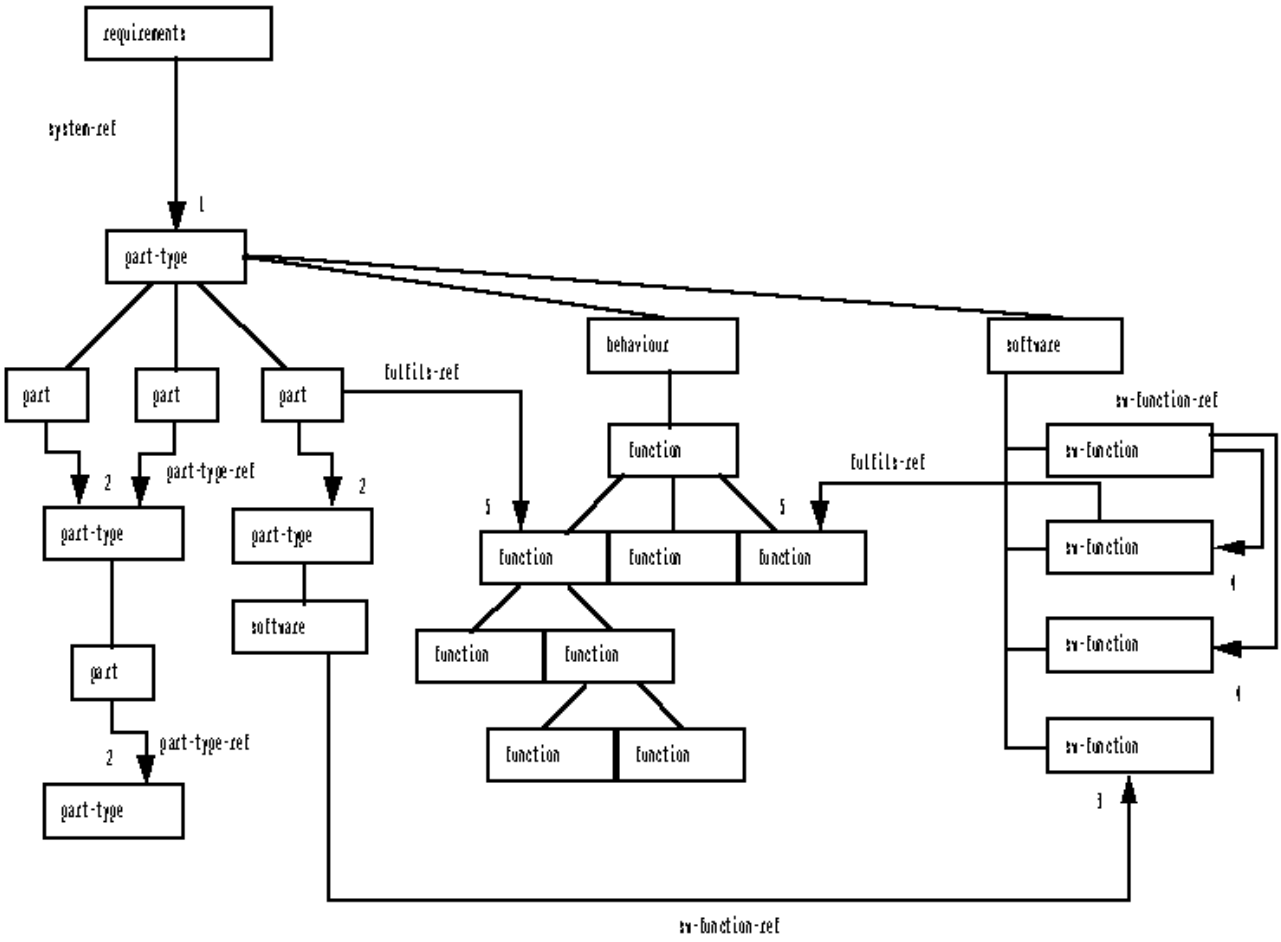

Abbildung 21: Erweiterte Struktur mit Referenzen

3 Beschreibung der Elemente und der Attribute

3.1 Klassen-Elemente

In der Software-DTD werden folgende Klassen-Elemente verwendet:

- 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

Alle Klassen-Elemente haben als Endung des Elementnamens ”-class”. Die Klassen sind nicht normiert bzw. nicht in der DTD definiert. Es wird jedoch empfohlen, bestimmte Klassenbezeichnungen z.B. entsprechend ASAP zu verwenden. Hierfür wurden Vorschläge gemacht. Siehe **<sw-param-class>**, z.B. ”value”. Die Klassenbezeichnungen sind zwischen den Projektpartnern zu vereinbaren.

3.2 Beschreibung der Elemente

Nachfolgend werden die Elemente der Software-DTD beschrieben. Die Beschreibung eines Elementes besteht aus: Beschreibung des Elements, Angabe der Attribute, Angabe des Kontextes und einem Beispiel.

Es gibt Elemente, die allgemeingültig sind und einer Anwendungsarchitektur zuzuordnen sind, sie haben im Elementnamen nicht den Vorspann ”sw-”. Diese Elemente werden in allen DTD’s, die einer bestimmten Anwendungsarchitektur entsprechen, verwendet. Deshalb werden sie hier nicht alle nochmals aufgeführt. Sie werden in dem Dokument ”Concepts of the DTD” beschrieben.

Es gibt ein Attribut s (Signatur), das in allen Elementen vorkommt. Deshalb wird dieses Attribut nicht mehr aufgeführt.

Bei den Beispielen werden zur Übersichtlichkeit teilweise auch fixe Attribute aufgeführt (z.B. **[f-namespace]**). Bei einer Instanz aufgrund der Software-DTD tauchen die fixen Attribute nicht mehr auf, d.h. die Instanz ist nur zusammen mit der DTD vollständig.

<abs>

Beschreibung	Absolutwert bei Parametercharakteristik. Siehe Parametermodell (**<sw-prm>**).	
Attribute	**[s]**	Signatur
Enthalten in	**<prm-char>**	
Beispiel		

<add-info>

Beschreibung	Zusätzliche Hinweise. In einer Funktion könnte beispielsweisemit Hilfe dieses Elementes die verwendete Programmiersprache dokumentiert.
Attribute	-
Enthalten in	**<sw-function-variant>**,**< sw-param>**, **<sw-variable-implementation>**

Beispiel

<add-spec>

Beschreibung

Attribute -

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

Beispiel

<admin-data>

Beschreibung Innerhalb der Struktur dieses Elementes können Angaben zur separaten Verwaltung von Teilinstanzen gemacht werden. Deshalb existiert dieses Element an all den Stellen, die poteniell für Export und Import zur separaten Verwaltung vorgesehen sind auf.

Attribute

[f-child-type]	Für Anwendungen, um unterschiedliche ”child-types” zu ermöglichen(Typprüfung). Z.B. ”language:selection”, d.h. das Kindelement language ist vom Typ selection, kann also aus einer Auswahlliste ausgewählt werden.
[s]	Signatur

Enthalten in Allen Elementen, die für eine Fragmentierung vorgesehen sind.

Beispiel

<annotation>

Beschreibung Im Laufe eines Prozesses können verschiedene Informationen entstehen, (z.B. Bearbeitungsinformationen, Notizen usw.) die an die Variablen, Kenngrößen bzw. Funktion zu koppeln und im Prozeß weiterzureichen sind. Derartige Informationen werden in **<annotation>** abgelegt.

Attribute -

Enthalten in **<annotations>**

Beispiel

<arraysize>

Beschreibung Bei Variablen kann es vorkommen, daß sie nicht als Einzelelement, sonder als array angelegt werden. Dann gibt dieser Wert die Größe des arrays an. Bei mehrdimensionalen arrays Trennung der Dimensionen durch Leerzeichen.

Es können ebenso Kenngrößen-Felder/-arrays angelegt werden.

Attribute

[S]	Signatur

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

Beispiel

<bit-base-name>

Beschreibung In**<sw-bit-representation>** in **<sw-variable>**. Name der Variablen, in der das Bit liegt.

Attribute -

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

Beispiel

<bit-base-type>

Beschreibung In**<sw-bit-representation>** in **<sw-variable>**. Typ (byte, word, long) der Variablen, in der das Bit liegt.

Attribute -

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

Beispiel

<bit-count>

Beschreibung In**<sw-bit-representation>** in **<sw-variable>**. Anzahl der zusammengehörigen Bits.

Attribute -

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

Beispiel

<bit-pos>

Beschreibung In**<sw-bit-representation>** in **<sw-variable>**. Postion des Bits in der Basis-Größe.

Attribute -

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

Beispiel

<change>

Beschreibung Element, in dem die Art einer Änderung beschrieben werden kann.

Attribute -

Enthalten in **<Modification>**

Beispiel

<chapter>

Beschreibung In einem Kapitel können informelle, nicht anwendungsstrukturierte Informationen beschrieben werden. Kapitel können hierarchisch aufgebaut sein. Ein Kapitel kann Paragraphen, Tabellen, Grafiken, Listen, usw. enthalten.

Attribute -

Enthalten in

Beispiel

<cmt-int>

Beschreibung Interner Wert bei einer Umrechnungsformel

Attribute -

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

Beispiel

<cmt-phys>

Beschreibung Physikalischer Wert bei einer Umrechnungsformel.

Attribute **[s]** Signatur

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

Beispiel

<cmt-text>

Beschreibung Textueller Wert bei einer Umrechnungsformel.

Attribute -

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

Beispiel

<code>

Beschreibung

Attribute -

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

Beispiel

<coded>

Beschreibung Codierte/interne Werte bei limits.

Attribute -

Enthalten in **<sw-limits>**

Beispiel

<coded-max>

Beschreibung Maximaler codierter Grenzwert

Attribute -

Enthalten in **<coded>**

Beispiel

<coded-min>

Beschreibung Minimaler codierter Grenzwert.

Attribute -

Enthalten in **<coded>**

Beispiel

<companies>

Beschreibung Firmenspezifische Angaben, bestehend aus 1 .. n Firmen.

Attribute -

Enthalten in **<project>**

Beispiel

<company>

Beschreibung Firmenspezifische Angaben für eine am Projekt beteiligte Firma.

Attribute

[f-child-type] Für Anwendungen, um unterschiedliche ”child-types” zu ermöglichen.

[f-id-class]

[f-name-space]

[id]

[role] Rolle, der am Projekt beteiligten Firma. ”Manufacturer”, ”supplier”.

Enthalten in **<companies>**

Beispiel

<company-doc-info>

Beschreibung

Attribute -

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

Beispiel

<company-doc-infos>

Beschreibung Firmenspezifische Angaben bei den administrativen Daten

Attribute -

Enthalten in **<admin-data>**

Beispiel

<comany-ref>

Beschreibung Verweis auf eine Firma.

Attribute -

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

Beispiel

<company-revision-info>

Beschreibung Firmenspezifische Angaben zu einer bestimmten Revision/Variante.

Attribute **[f-child-type]**

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

Beispiel

<company-revision-infos>

Beschreibung Revsions- bzw. Variantenangaben zu einer Instanz oder Fragment.

Attribute **[s]** Signatur

Enthalten in **<doc-revision>**

Beispiel

<cond>

Beschreibung Bedingung bei Parametern.

Attribute -

Enthalten in **<prm-char>**

Beispiel

<count>

Beschreibung Anzahl der Elemente bei einem Kenngrößenblock.

Attribute -

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

Beispiel

<date>

Beschreibung Datumsangabe, mehrsprachig möglich.

Attribute **[s]** Signatur

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

Beispiel

<date-1>

Beschreibung Datumsangabe, nicht mehrsprachig möglich.

Attribute **[s]** Signatur

Enthalten in **<std>**, **<xdoc>**

Beispiel

<demarcation-other-project>

Beschreibung Abgrenzung zu anderen Projekten

Attribute -

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

Beispiel

<department>

Beschreibung Abteilung

Attribute -

Enthalten in **<team-member>**

Beispiel

<desc>

Beschreibung Beschreibungselement

Attribute -

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

Beispiel

<doc-label>

Beschreibung

Attribute -

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

Beispiel

<doc-revision>

Beschreibung

Attribute **[f-child-type]**

Enthalten in **<doc-revisions>**

Beispiel

<doc-revisions>

Beschreibung

Attribute -

Enthalten in **<admin-data>**

Beispiel

<entity-name>

Beschreibung

Attribute -

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

Beispiel

<figure>

Beschreibung Durch dieses Element können Grafiken eingebunden werden.

Attribute -

Enthalten in

Beispiel

<file>

Beschreibung Angabe eines Filenamens, um auf ein externes File, einen Standard zu verweisen.

Attribute

[filename]	Name des Files.
[notation]	Typ des Files, z.B. ”wmf”, ”cgm”.
[tool]	Werkzeug, mit dem das File angezeigt oder bearbeitet werden kann.
[tool-version]	Version des Werkzeugs, mit dem das File erzeugt wurde.

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

Beispiel

<function-ref>

Beschreibung Verweis auf Funktion bzw. Requirements-Funktion.

Attribute -

Enthalten in **<sw-fulfils>**

Beispiel

<general-project-data>

Beschreibung Allgemeine Projektdaten.

Attribute -

Enthalten in **<company>**

Beispiel

<generic-math>

Beschreibung Mathematische Formel.

Attribute -

Enthalten in **<formula>**

Beispiel

<graphic>

Beschreibung Angabe eines Grafikfiles.

Attribute **[category]**

[filename]

[fit]

[height]

[notation]

[scale]

[width]

Enthalten in **<ml-graphic>**

Beispiel

<ie>

Beschreibung

Attribute **[type]**

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

Beispiel

<introduction>

Beschreibung Einführungsteil bzw. Einführungskapitel.

Attribute -

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

Beispiel

<label>

Beschreibung Ein label ist eine Bezeichnung für ein Objekt, das nicht referenziert werden muß und kann, also keinen**< short-name>** und keine **[id]** besitzt.

Ein **<label>** bei **<sw-param-value-block>** ist eine Langbezeichnung für die Kennwerte eines Kennwerteblocks.

Attribute -

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

Beispiel

<labels>

Beschreibung Menge der labels bei Achswerten.

Attribute -

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

Beispiel

<locs>

Beschreibung Dieses Element wird für dokumenten-/instanzübergreifende Referenzierung verwendet (HighTime-Referenzierungskonzept über namelocs)

Attribute -

Enthalten in **<msrsw>**

Beispiel

<language>

Beschreibung Bezeichnet die Masterlanguage bei **<admin-data>**

Attribute -

Enthalten in **<admin-data>**

Beispiel

<long-name>

Beschreibung Langbezeichnung , z.B. ”Motortemperatur”.

Attribute -

Enthalten in

Beispiel

<max>

Beschreibung Maximalwert bei einer Parametercharakteristik.

Attribute -

Enthalten in **<prm-char>**

Beispiel

<max-count>

Beschreibung Maximale Anzahl Stützstellen.

Attribute -

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

Beispiel

<min>

Beschreibung Minimalwert bei prm-char

Attribute -

Enthalten in **<prm-char>**

Beispiel

<modification>

Beschreibung

Attribute **[type]** content-related, doc-related

Enthalten in **<modifications>**

Beispiel

<modifications>

Beschreibung

Attribute -

Enthalten in **<doc-revision>**

Beispiel

<msrsw>

Beschreibung Wurzelelement

Attribute **[f-namespace]**

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

[HyTime]

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

Enthalten in -

Beispiel

<na>

Beschreibung Falls gewisse Angaben nicht relevant sind, wird dieses Element anstelle von Teilstrukturen verwendet (”not applicable”).

Attribute -

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

Beispiel

<ncoi-1>

Beschreibung Dies ist ein allgemeines Element, das informelle und nicht softwarespezifischen Strukturen enthält (”none coded information”).

Attribute -

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

Beispiel

<offset>

Beschreibung Bei Festkennlinien, -feldern werden die Stützstellen üer einen Algorithmus festgelegt. Beispiel für einen Algorithmus: Stützstelle[i] = (... shift) * x + offset

Attribute -

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

Beispiel

<p>

Beschreibung Paragraph. Ein Paragraph kann aus Text und aus den Elementen tt (technischer Benennung), xref (Querverweis), e (Textattribut, wie Fettdruck), ft (Fußnote), sup (hochgestellt), sub (tiefgestellt), ie (Indexeintrag), std (Standard), xdoc (externes Dokument), xfile (externes File) in beliebiger, jedoch nicht hierarchisierter Reihenfolge bestehen.

Attribute **[help-entry]** Eintrag für ein Hilfesystem

Enthalten 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

Beispiel

<private-code>

Beschreibung

Attribute **<type>** -

Enthalten in **<private-codes>**

Beispiel

<private-codes>

Beschreibung

Attribute -

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

Beispiel

<prm>

Beschreibung In MSR wurde ein Parametermodell implementiert. Hierbei gibt es 2 Möglichkeiten. Ein Parameter kann entweder durch Angabe von **<abs>** ,**<tol>** oder durch Angabe von **<min>**, **<typ>** und**<max>** spezifiziert werden.

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

[id]

Enthalten in prms

Beispiel

<prms>

Beschreibung Liste von Parametern.

Attribute -

Enthalten 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>**

Beispiel

<prm-char>

Beschreibung Beschreibung der Parametercharakteristik eines Parameters.

Attribute -

Enthalten in **<prm>**

Beispiel

<prog-code>

Beschreibung Beschreibung von Umrechnungsformeln in Programmiersprachennotation.

Attribute **[lang-subset]**

[prog-lang] Programmiersprache

[used-libs] Benutzte Bibliotheken

Enthalten in sw-compu-method

Beispiel

<project>

Beschreibung Angaben zu einem Projekt.

Attribute -

Enthalten in **<project-data>**

Beispiel

<project-data>

Beschreibung Projektdaten

Attribute -

Enthalten in **<msrsw>**

Beispiel

<shift>

Beschreibung Bei Festkennlinien, -feldern werden die Stützstellen üer einen Algorithmus festgelegt. Beispiel für einen Algorithmus: Stützstelle[i] = (... shift) * x + offset

Attribute -

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

Beispiel

<short-name>

Beschreibung Kurzbezeichnung, z.B. ”TMOT”. Der **<short-name>** besitzt identifizierenden Charakter insbesondere für Bezüge nach außen (z.B. *ASAP*, *ASCET*).

Attribute -

Enthalten 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

Beispiel

<si-unit>

Beschreibung Bei Maßeinheiten gibt es SI-Maßeinheiten. STEP (ISO/DIS 10303-41. S96ff) stützt sich bzgl. SI-Einheiten auf auf sieben Basis-Einheiten ab (length, mass, time, electric_current, thermodynamic_temperature, amount_of_substance, luminous_intensity). Diese Basiseinheiten wurden über Attribute realisiert.

Attribute **[amount-ofsubstance-expo]**

[electric-current-expo]

[length-expo]

[luminous-intensity-expo]

[mass-expo]

[thermodynamic-temperature-expo]

[time-expo]

Enthalten in **<sw-unit>**

Beispiel

<reason>

Beschreibung

Attribute -

Enthalten in **<modification>**

Beispiel

<revision-label>

Beschreibung Revision

Attribute -

[]

Enthalten in **<company-revison-info>**

Beispiel

<state>

Beschreibung

Attribute -

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

Beispiel

<std>

Beschreibung

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

[f-id-class] std

[id]

Enthalten in

Beispiel

<sw-addressing-method>

Beschreibung Beschreibt das Addressierschema. Es wird beschrieben, wie eine Kenngröße oder eine RAM-Größe vom Steuergerät adressiert wird. Beispiele für unterschiedliche Adressierungsarten sind die direkte Adressierung und die indirekte Adressierung über Vektor. Über das Adressierschema kann aber auch festgelegt werden, in welchem RAM-Bereich eine RAM-Größe liegt (z.B. im internen oder externen RAM). Adressierschemata können von Kenngrößen und RAM-Größen benutzt werden.

Dieses Element ist eine Namensreferenz auf eine Zugriffsmethode, welche in nachfolgenden Systemen (z.B. *Verstellystemen*) nachgebildet werden muß. Die Verbindung zwischen MSR-Instanz und nachfolgenden Systemen erfolgt über diesen Namen.

Adressierschemas sind nicht standardisiert bzw. in der DTD vordefiniert. Sie sind auch nicht ausmodelliert und werden in **<sw-addressing-method-desc>** lediglich textuell beschrieben.

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

[id]

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

Beispiel

<sw-addressing-method-class>

Beschreibung Gibt die Klasse des Adressierschemas an. Kenngrößen bzw. Variable können nur Adressierschemas einer bestimmten Klasse verwenden. Dies kann jedoch nicht mit SGML-Mitteln sichergestellt werden, es muß die Anwendung tun.

Adressierschema-Klassen sind nicht standardisiert bzw. in der DTD vordefiniert.

Attribute -

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

Beispiel

<sw-addressing-method-desc>

Beschreibung Textuelle Beschreibung des Adressierschemas.

Attribute -

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

Beispiel

<sw-addressing-method-ref>

Beschreibung Verweis auf Adressierschema, z.B. durch Angabe des short-names.

Attribute **[HyNames]**

[HYTIME]

[sw-addressing-method] idref

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

Beispiel

<sw-addressing-methods>

Beschreibung In **<sw-data-dictionary>** werden alle Adressierschemas definiert.

Attribute -

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

Beispiel

<sw-application-notes>

Beschreibung Applikationshinweise

Attribute -

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

Beispiel

<sw-array-index>

Beschreibung Verweis auf die entsprechende Teilstruktur bei Parameterstrukturen.

Attribute -

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

Beispiel

<sw-asap-6-prm-method>

Beschreibung 6-Parameterformel entsprechend ASAP (Polynomdarstellung. Angabe der Elemente der Formel durch white spaces getrennt. Formel: int = (a $\times x^2$ + b $\times x^1$ + c) / (d $\times x^2$ + e $\times x^1$ + f)

Attribute -

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

Beispiel

<sw-axis-grouped>

Beschreibung Mehrere Kennlinien haben aus Laufzeitgründen diesselben Stützstellen auf die über Name referenziert wird.

Attribute -

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

Beispiel

<sw-axis-individual>

Beschreibung Beschreibung einer Achse, deren Stützstellen individuell festgelegt werden können. Jede Kennlinie oder jedes Kennfeld hat die eigenen, privaten Stützstellenwerte (im Gegensatz zu Gruppenkennlinie).

Attribute -

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

Beispiel

<sw-axis-shift-offset>

Beschreibung Spezielle aus Laufzeitoptimierungsgründen verwendete Form der Stützstellen. Die Stützstellen berechnen sich nach folgendem Algorithmus: Stützstelle[i] = ... + Offset.

Attribute **[]**

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

Beispiel

<sw-base-type>

Beschreibung Basistyp der sw-variable (z.B. char, integer).

Attribute -

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

Beispiel

<sw-bit-representation>

Beschreibung Dieses optionale Element enthält Informationen zur Bit-Darstellung in Form von Elementen (Bitanzahl, Bitposition, Bitbasisname, Bitbasistyp) bei Variablen mit dem Datentyp ”bit”.

Attribute -

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

Beispiel

<sw-carb-doc>

Beschreibung CARB-Dokumentation.

Attribute -

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

Beispiel

<sw-code-syntax>

Beschreibung Namensreferenz auf eine Codesyntax. Durch die Codesyntax wird die Darstellung der RAM-Größen und Kengrößen in der Datenquelldatei festgelegt. Die Codesyntaxen werden im **<sw-data-dictionary>** beschrieben. Diese dort definierten Codesyntaxen können von Variablen und Kenngrößen durch Referenzierung verwendet werden.

Der Inhalt einer Codesyntax ist nicht ausmodelliert, d.h. die Codesyntax wird nur textuell in **<sw-code-syntax-desc>** beschrieben. Für die Codesyntax gibt es keine standardisierten bzw. in der DTD vordefinierten Werte.

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

[id]

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

Beispiel

<sw-code-syntax-class>

Beschreibung Gibt die Klasse der Codesyntax an. Kenngrößen bzw. Variable können nur die Codesyntax einer bestimmten Klasse verwenden. Dies kann jedoch nicht mit SGML-Mitteln sichergestellt werden, es muß die Anwendung tun. Eine Codesyntax ist zum Beispiel für eine Kenngröße oder eine Variable geeignet, oder auch nur für eine Kennline. Durch die Klasse könnte auch die Kombierbarkeit einer bestimmten Codesyntax mit einem bestimmten Ablageschema ausgedrückt werden.

Für die Codesyntax-Klassen gibt es keine standardisierten bzw. in der DTD vordefinierten Werte.

Attribute -

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

Beispiel

<sw-code-syntax-desc>

Beschreibung Beschreibung der Code-Syntax

Attribute -

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

Beispiel

<sw-code-syntax-ref>

Beschreibung Innerhalb einer RAM-Größe oder einr Kenngröße kann optional auf eine Code-Syntax verwiesen werden.

Attribute **[HyNames]**

[HYTIME]

[sw-code-syntax]

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

Beispiel

<sw-code-syntaxes>

Beschreibung Liste der Code-Syntaxen.

Attribute -

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

Beispiel

<sw-compu-generic-math>

Beschreibung Die Umrechnungsformel wird durch einen allgemeinen math. Ausdruck beschrieben.

Attribute -

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

Beispiel

<sw-compu-method>

Beschreibung Umrechnungsformel. Rechenvorschrift, nach der eine Steuergeräte-intern verwendete Größe in ihren physikalischen Wert umgerechnet werden kann. Die Umrechnungsformel muß umkehrbar sein.

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

[fid]

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

Beispiel

<sw-compu-method-ref>

Beschreibung Verweis auf eine Umrechnungsformel

Attribute **[HyNames]**

[HYTIME]

[sw-compu-method]

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

Beispiel

<sw-compu-method-table>

Beschreibung Bei dieser Form der Umrechnungsformel wird der Zusammenhang zwischen interner und physikalischer Darstellung durch eine Tabelle von int-phys-Wertepaaren angegeben. Es gibt ein Attribut *interpolation-style*, das die Werte *interpolation* (default) und *discrete* annehmen kann.

Attribute **[interpolation-style]** Mögliche Werte: interpolation, no-interpolation, discrete. Default: interpolation.

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

Beispiel

<sw-compu-method-text>

Beschreibung Die Umrechnungsformel wird in textueller Form beschrieben. Im Gegensatz zu den übrigen Umrechnungstypen stellt die physikalische Seite nicht eine Zahl dar, sondern einen Text. Anwendungen sind denkbar bei Schaltern (”ein”/”aus”) oder Länderkennungen (”D”, ”F”, ”US”).

Attribute -

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

Beispiel

<sw-compu-method-text-pair>

Beschreibung Enthält bei textartigen Umrechnunsformeln einen internen und einen textuellen Wert.

Attribute -

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

Beispiel

<sw-compu-method-value-pair>

Beschreibung Enthält bei tabellenartigen Umrechnungsformeln einen internen und einen physikalischen Wert.

Attribute -

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

Beispiel

<sw-compu-methods>

Beschreibung Liste der Umrechnungsformeln.

Attribute -

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

Beispiel

<sw-data-dictionaries>

Beschreibung Menge der Datadictionaries.

Attribute -

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

Beispiel

<sw-data-dictionary>

Beschreibung Datenlexikon. Hier können Datentypen, Variablen, Parameter, Maßeinheiten, Adressierschemas, Ablageschemas, Codesyntaxen und Umrechnungsformeln beschrieben werden.

Attribute -

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

Beispiel

<sw-data-dictionary-spec>

Beschreibung Datadictionary-Spezifikation.

Attribute -

Enthalten in **<msrsw>**

Beispiel

<sw-data-dictionary-class>

Beschreibung In diesem Element können Angaben zur Entstehung, zur Änderung von Datadictionaries gemacht werden, z.B. ”Datadictionary von Firma xy”.

Attribute -

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

Beispiel

<sw-data-type-scaling>

Beschreibung Quantisierung einer Variablen. Beispiel: ”16 Bit”.

Attribute -

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

Beispiel

<sw-decomposition>

Beschreibung Bildung von Funktionshierarchien durch Referenzierung von eingebauten bzw. verwendeten Funktionen.

Attribute -

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

Beispiel

<sw-fullfils>

Beschreibung Ordnet einer Software-Funktion diejenige bzw. diejenigen (Verhaltens-)Funktionen zu, die sie realisiert.

Attribute -

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

Beispiel

<sw-function>

Beschreibung Software Funktion

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

[id]

Enthalten in **<sw-functions>**

Beispiel

<sw-function-class>

Beschreibung Funktionsklasse. Die Funktionen können in verschiedene Klassen aufgeteilt werden. Es könnten beispielsweise Funktion, die nur in der Dokumentation/Analysephase auftreten, zu einer Klasse zusammengefaßt werden.

Attribute -

Enthalten in **<sw-function>**

Beispiel

<sw-function-def>

Beschreibung Jede Funktion wird in geeigneter Darstellungsart kurz beschrieben. In der Regel wird eine grafische und eine kurze textuelle Darstellung verwendet. Beispiel: ASCET-Funktionsmodelle, Zustandsdiagramme, SA/SD-Diagramme.

Attribute -

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

Beispiel

<sw-function-desc>

Beschreibung Ausführliche textuelle Beschreibung der Funktion.

Attribute -

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

Beispiel

<sw-function-export-variables>

Beschreibung List der exportierten Variablen. Wenn eine Funktion eine Variable definiert, d.h. eine Variable bzw. den Wert zur Verfügung stellt, so ist sie hier aufzuführen.

Attribute -

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

Beispiel

<sw-function-import-variables>

Beschreibung Liste der importierten Variablen. Hier werden Variablen aufgeführt, die von anderen Funktionen zur Verfügung gestellt werden.

Attribute -

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

Beispiel

<sw-function-local-variables>

Beschreibung Liste der lokalen Variablen, d.h. keine Übergabevariable.

Attribute -

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

Beispiel

<sw-function-modelonly-variables>

Beschreibung Liste der Variablen, die nur im Modell existieren und in der Implementierung nicht mehr vorkommen.

Attribute -

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

Beispiel

<sw-function-ref>

Beschreibung In **<msrsw>**Verweis auf verwendete Funktionen, die außerhalb von SW beschrieben sind. In **<sw-decomposition>** Verweis auf Funktionen, die innerhalb **<msrsw>**beschrieben sind.

Attribute **[HyNames]**

[HYTIME]

[sw-function]

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

Beispiel

<sw-function-refs>

Beschreibung Verweise auf verwendete Funktionen, die außerhalb des Dokumentes beschrieben werden. Z.B. Betriebssystemfunktionen, Netzwerkservices.

Attribute -

Enthalten in **<msrsw>**

Beispiel

<sw-function-spec>

Beschreibung Funktionsspezifikation. Eine Funktionsspezifikation kann mehrere Funktionen enthalten. Die Funktionen können hierarchisiert sein, indem über **<sw-decomposition>** auf eine Funktion referenziert wird.

Attribute -

Enthalten in **<msrsw>**

Beispiel

<sw-function-variables>

Beschreibung Liste der von der Funktion verwendeten Variablen

Attribute -

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

Beispiel

<sw-function-variant>

Beschreibung Funktionsvariante. Falls keine Funktionsvarianten verwendet werden, ist trotzdem mindestens eine anzugeben. In der Implementierung V1.1.0 muß jede Funktionsvariante komplett beschrieben werden, d.h. es gibt keine Wiederverwendung.

Mittels**< variant-def-ref>** kann auf eine Merkmalsleiste in **<project-data>** ... **<var-def>** (Funktionsschalter) referenziert werde.

Attribute -

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

Beispiel

<sw-function-variants>

Beschreibung Menge der Funktionsvarianten.

Attribute -

Enthalten in **<sw-function>**

Beispiel

<sw-functions>

Beschreibung Menge der Funktionen in der Funktionsspezifikation/in dem Dokument. Hüllelement.

Attribute -

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

Beispiel

<sw-glossary>

Beschreibung Glossar

Attribute -

Enthalten in **<msrsw>**

Beispiel

<sw-interpolation-method>

Beschreibung Interpolationsmethode bei Kenngrößen. Die Methoden sind nicht normiert bzw. in der DTD definiert. Die Methoden werden textuell beschrieben.

Attribute -

Enthalten in **<sw-param>**

Beispiel

<sw-limits>

Beschreibung Hier werden minimale und maximale Werte für die physikalischen und die Steuergeräte-interne Werten angegeben.

Attribute -

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

Beispiel

<sw-maintenance-notes>

Beschreibung Kundendiensthinweise.

Attribute -

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

Beispiel

<sw-param>

Beschreibung Software-Parameter. Software-parameter sind Kenngrößen (in ASAP characteristics): Kennlinien, Kennfelder u. Festwerte.

In der Regel sind Kenngrößen im EPROM des Steuergerätes abgelegt. Sie sind applizierbar, d.h. sie können mittels Applikationswerkzeugen im Rahmen der Plausibilität und im Rahmen gesetzter Zugriffsschutzattribute gelesen oder verändert werden. Ausnahmen: z.B. Zylinderzahl, nicht im EPROM, nicht applizierbar. Es gibt ein Attribut *calibration* mit

den möglichen Atributwerten*calibration* (default), *no-calibration* u. *not-in-memory*.

Attribute **[calibration]** Mögliche Werte: calibration(default), no-calibration, not-in-memory

[f-child-type]

Enthalten in **<sw-params>**

Beispiel

<sw-param-array-x>

Beschreibung Kennlinie

Attribute -

Enthalten in **<sw-param>**

Beispiel

<sw-param-array-xy>

Beschreibung Kennfeld

Attribute -

Enthalten in **<sw-param>**

Beispiel

<sw-param-axis-values>

Beschreibung Werteachse. Beschreibung eines Kennwertes oder der Werte einer Kennlinie oder Kennfeldes.

Attribute -

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

Beispiel

<sw-param-axis-x>

Beschreibung Beschreibt die x-Achse einer Kenngröße (**<sw-param>**). Diese Beschreibung besteht aus der Eingangsgröße (als Referenz auf Variable), min, max, max-count (max. Anzahl Stützstellen).

Attribute -

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

Beispiel

<sw-param-axis-y>

Beschreibung Beschreibt die y-Achse einer Kenngröße (**<sw-param>**). Diese Beschreibung besteht aus der Eingangsgröße (als Referenz auf Variable), min, max, max-count (max. Anzahl Stützstellen).

Attribute -

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

Beispiel

<sw-param-class>

Beschreibung Gibt die Art einer Kenngröße an (Kennlinie, Kennfeld, Kennwert, Festkennlinie, Gruppenkennlinie, Festkennfeld).

Attribute -

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

Beispiel

<sw-param-content>

Beschreibung Kenngrößenwerte

Attribute -

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

Beispiel

<sw-param-contents-class>

Beschreibung Gibt die Art der Dateninhalte/Datensätze an, z.B.: ”Testdaten”, ”Rohapplikationsdaten”.

Die Kenngrößenklassen sind nicht normiert bzw. in der DTD definiert.

Attribute -

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

Beispiel

<sw-param-content-text>

Beschreibung Hier stehen die Werte eines Textparameters (sw-param-text).

Attribute -

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

Beispiel

<sw-param-content-v>

Beschreibung Hier steht bei einem Kennwert ein einzelner Wert. Bei einer Kennlinie stehen hier die Werte der v-Achse. Bei einem Kennfeld stehen hier die Werte der v-Achse. Dabei gilt die Konvention: Index der x-Achse läuft schneller. Werte sind durch Leerzeichen oder Zeilenvorschübe getrennt.

Attribute -

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

Beispiel

<sw-param-content-x>

Beschreibung Hier stehen die physikalischen, internen, ... Werte der x-Achse.

Attribute -

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

Beispiel

<sw-param-content-y>

Beschreibung Hier stehen die physikalischen, internen, ... Werte der y-Achse.

Attribute -

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

Beispiel

<sw-param-contents-spec>

Beschreibung Spezifikation der Kenngrößeninhalte.

Attribute -

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

Beispiel

<sw-param-group-axis>

Beschreibung Gruppenkennlinie

Attribute -

Enthalten in **<sw-param>**

Beispiel

<sw-param-noeffect-value>

Beschreibung Wert, der die Wirkung einer Kenngröße aufhebt (typischerweise 0 oder 1).

Attribute -

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

Beispiel

<sw-param-record-layout>

Beschreibung Ablageschema (in *ASAP record layout*). Ein Ablageschema beschreibt die Form der Ablage einer Kenngröße im Eprom. Über das Ablageschema wird die Reihenfolge der Bestandteile der Kenngröße und deren Datentyp bestimmt. So könnte ein Ablageschema für eine Kennlinie festlegen, daß zuerst die Adresse der Eingangsgröße als byte, dann die Stützstellenanzahl als unsigned byte, dann die Stützstellen als signed word und als letztes die Werte der Kennlinie als unsigned word abzulegen sind.

Dieses Element ist eine Namensreferenz auf ein Ablageschema. Dieses Ablageschema benennt den Aufbau des Speicherobjekts im Speicher des Steuergerätes. Nachfolgende Systemen (z.B. *Verstellsysteme*) müssen diese Ablageschemata auswerten. Die Verbindung zu derartigen Systemen geschieht über diesen Namen.

Codesyntax-Objekte beschreiben die Darstellung eines Objektes in einer Programmiersprache, die Ablageschema-Objekte ihre Ablage im Eprom. Codesyntax-Objekte und Ablageschema-Objekte müssen konsistent sein. D.h. eine Kenngröße kann bei dem Ablageschema nicht Wort-Ablage und bei der Code-Syntax Byte-Ablage verwenden.

Das Ablageschema ist nicht ausmodelliert. Das eigentliche Ablageschema kann textuell in **<sw-param-record-layout-desc>** beschrieben werden.

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

[id]

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

Beispiel

<sw-param-record-layout-class>

Beschreibung Gibt die Klasse des Ablageschemas an. Kenngrößen bzw. Variable können nur Ablageschemas einer bestimmten Klasse verwenden. Dies kann jedoch nicht mit SGML-Mitteln sichergestellt werden, es muß die Anwendung tun. Die Ablageklassen sind nicht standardisiert und nicht in der DTD definiert.

Attribute -

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

Beispiel

<sw-param-record-layout-desc>

Beschreibung Hier kann das Ablageschema textuell bechrieben werden.

Attribute -

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

Beispiel

<sw-param-ref>

Beschreibung Verweis auf **<sw-param>** Kenngröße. Ist bei **<sw-param-ref>** in **<sw-function-variant>** das Attribut owns nicht definiert, so bedeutet das, daß die Kenngröße in der Funktion definiert wird und ihr somit gehört.

Attribute **[HyNames]**

[HYTIME]

[owns]

[sw-param]

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

Beispiel

<sw-param-single-value>

Beschreibung Applizierbare Einzelgröße. Eine Einzelgröße wird wie eine Kennlinie mit einem Element behandelt. Die Kenngröße wird in**< sw-param>** definiert, der Wert steht in **<sw-param-content>**. Die Zuordnung geschieht durch Referenzierung in**<sw-param-ref>** von **<sw-param-content>** auf **<sw-param>**.

Attribute -

Enthalten in **<sw-param>**

Beispiel

<sw-param-target>

Beschreibung Referenz auf Name einer Variablen/RAM-Zelle in welcher das Ergebnis der Interpolation stehen soll.

Attribute -

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

Beispiel

<sw-param-text>

Beschreibung Hier können Kenngrößen beschrieben werden, die als Werte Texte haben. Beispiel: Meldungstext für Fahrerrückmeldungen. Die Werte stehen in **<sw-param-content-text>**.

Attribute -

Enthalten in **<sw-param>**

Beispiel

<sw-param-value-block>

Beschreibung Ein Kenngrößenblock besteht aus einem array von gleichartigen Kennwerten. Die Anzahl wird in **<count>** angegeben.

Attribute -

Enthalten in **<sw-param>**

Beispiel

<sw-param-values-adr>

Beschreibung Dieses Element enhält die Adressen von Kenngrößen. Die Werte stehen in dem Subelement **<v>**, das bei einem einzelnen Kennwert einmal und bei Kenngrößen mehrfach vorkommt.

Attribute -

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

Beispiel

<sw-param-values-coded>

Beschreibung Dieses Element enhält die Werte von Kenngrößen in codierter Form. Die Werte stehen in dem Subelement **<v>**, das bei einem einzelnen Kennwert einmal und bei Kenngrößen mehrfach vorkommt.

Attribute -

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

Beispiel

<sw-param-values-coded-hex>

Beschreibung Dieses Element enhält die Werte von Kenngrößen in hexadezimaler codierter Form. Die Werte stehen in dem Subelement **<v>**, das bei einem einzelnen Kennwert einmal und bei Kenngrößen mehrfach vorkommt.

Attribute -

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

Beispiel

<sw-param-values-generic>

Beschreibung Hier können Werte in einem nicht direkt unterstützten Format abgelegt werden. Dieses Format wird im Attribut **[type]** angegeben. Die Verwendung dieses Elementes ist prozeßspezifisch.

Attribute **[type]**

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

Beispiel

<sw-param-values-phys>

Beschreibung Dieses Element enhält die Werte von Kenngrößen in physikalischer Form. Die Werte stehen in dem Subelement **<v>**, das bei einem einzelnen Kennwert einmal und bei Kenngrößen mehrfach vorkommt.

Attribute -

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

Beispiel

<sw-params>

Beschreibung Liste der Kenngrößen (Hüllelement).

Attribute -

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

Beispiel

<sw-physic-type>

Beschreibung Definition eines physikalischen, implementierungsunabhängigen und wiederverwendbaren Datentyps. Physikalische Datentypen werden in den frühen Phasen der Steuergerätesoftwareentwicklung definiert.

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

[id]

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

Beispiel

<sw-physic-type-1>

Beschreibung Physikalische Datentypen können bei **<sw-physic-types>** im Datadictionary und auch direkt bei der Variblendefinition definiert werden. Die direkt bei der Variablen definierten physikalischen Eigenschaften haben keinen **<long-name>** und keinen **<short-name>** und sind nicht referenzierbar.

Attribute -

Enthalten in **<sw-variable>**

Beispiel

<sw-physic-type-contents>

Beschreibung Beschreibung eines physikalischen Datentyps.

Attribute -

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

Beispiel

<sw-physic-type-ref>

Beschreibung Verweis/Verwendung eines physikalischen Datentyps durch eine Variable.

Attribute **[HyNames]**

[HYTIME]

[sw-physic-type]

Enthalten in **<sw-variable>**

Beispiel

<sw-physic-types>

Beschreibung Menge der physikalischen Datentypen.

Attribute -

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

Beispiel

<sw-resolution>

Beschreibung Auflösung, z.B: ”100 U/m/s”. Wobei nur ”100” anzugeben ist, ”U/m/s” ist die Einheit, die in **<sw-unit>** steht.

Attribute -

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

Beispiel

<sw-unit>

Beschreibung Definition einer Maßeinheit.

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

[id]

Enthalten in **<sw-units>**

Beispiel

<sw-unit-display>

Beschreibung Angabe, wie die Maßeinheit auf einem Ausgabemedium dargestellt werden soll.

Attribute -

Enthalten in **<sw-unit>**

Beispiel

<sw-unit-ref>

Beschreibung Verweis auf eine bereits definierte Maßeinheit zur Verwendung.

Attribute **[HyNames]**

[HYTIME]

[sw-unit]

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

Beispiel

<sw-var-init-value>

Beschreibung Initialisierungswert einer Variablen.

Attribute -

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

Beispiel

<sw-var-not-av-value>

Beschreibung Interner Wert für nicht verfügbar, externen Wert kann es nicht geben.

Attribute -

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

Beispiel

<sw-variable>

Beschreibung Variable (in ASAP "*measurement*"). Variable Größe, die für den Algorithmus einer Steuergerätefunktion benötigt wird. Variablen werden in der Regel im RAM abgelegt. Man unterscheidet zwischen Eingangs-, Zwischen- (lokale) und Ausgangsgrößen.

Attribute **[calibration]**

[f-id-class]

[f-namespace]

[id]

Enthalten in **<sw-variables>**

Beispiel

<sw-variable-implementation>

Beschreibung Element bei sw-variable, bei dem Implementierungshinweise eingetragen werden können (nicht manuell zu füllen).

Attribute -

Enthalten in **<sw-variable>**

Beispiel

<sw-variable-kind>

Beschreibung z.B. Datenfluß, Kontrollfluß.

Attribute -

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

Beispiel

<sw-variable-ref>

Beschreibung Verweis auf **<sw-variable>** / RAM-Größe.

Attribute **[HyNames]**

[HYTIME]

[sw-variable]

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

Beispiel

<sw-variable-sample-rate>

Beschreibung Abtastrate bzw. zeitliche Auflösung (z.B. Kurbelwellensynchron, alle 10 ms).

Attribute -

Enthalten in sw-physic-type-contents

Beispiel

<sw-variables>

Beschreibung Menge der Variablen eines Datadictionaries. Die Struktur zur Beschreibung der Variablen ist rekursiv, somit können hierarische Variblenstrukturen aufgebaut werden (z.B. Struct-Konstrukt in C).

Attribute -

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

Beispiel

<sw-variables-read>

Beschreibung Variablen, auf die von einer Funktion lesend zugegriffen werden.

Attribute -

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

Beispiel

<sw-variables-readwrite>

Beschreibung Variablen, auf die von einer Funktion lesend und schreibend zugegriffen werden.

Attribute -

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

Beispiel

<sw-variables-write>

Beschreibung Variablen, auf die von einer Funktion schreibend zugegriffen werden.

Attribute -

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

Beispiel

<team-member-ref>

Beschreibung Verweis auf ein team-member

Attribute **[HyNames]**

[HYTIME]

[team-member]

Enthalten in **<team-members>**

Beispiel

<tbd>

Beschreibung ”to be defined”

Attribute -

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

Beispiel

<tbr>

Beschreibung "to be resolved"

Attribute -

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

Beispiel

<topic-1>

Beschreibung Kapitelstruktur. Im Gegensatz zu chapter erscheint es jedoch nicht als eigenständiges Kapitel im Inhaltsverzeichnis.

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

[help-entry]

[id]

Enthalten 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>**

Beispiel

<topic-2>

Beschreibung Einführungskapitel, Kapitelstruktur. Gleich mit topic-1, nur keine prms. Im Gegensatz zu chapter erscheint es jedoch nicht als eigenständiges Kapitel im Inhaltsverzeichnis.

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

[help-entry]

[id]

Enthalten in **<introduction>**

Beispiel

<used-languages>

Beschreibung Liste der verwendeten Sprachen bei **<admin-data>**. Die Mastersprache wird in **<language>** angegeben.

Attribute -

Enthalten in **<admin-data>**

Beispiel

<variant-char>

Beschreibung Hier können die Variantenmerkmale beschrieben werden.

Attribute **[id]**

[f-id-class]

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

Enthalten in **<variant-chars>**

Beispiel

<variant-def>

Beschreibung Beschreibung der Variantendefinitionen, auf die in **<sw-funtion-variant>** referenziert werden kann. Bei einer Variantendefiniton werden hier einer Varianten die entsprechenden Variantenmerkmale (**<variant-def>**) zugeordnet und es findet die Zuordnung von Variantenmerkmal zu Merkmalswert (**<variant-char-value>**, **<value>**) statt.

Attribute -

Enthalten in **<variant-defs>**

Beispiel

<variant-def-ref>

Beschreibung

Attribute **[HyNames]**

[HYTIME]

[variant-def]

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

Beispiel

<variant-def-refs>

Beschreibung

Attribute -

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

Beispiel

3.3 Beschreibung der Attribute

[amount-of-substance-exp]

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

Mögl. Werte

Default IMPLIED

Enthalten in si-unit

Beispiel

[calibration]

Beschreibung Beschreibt die Kalibrierbarkeit einer Kenngröße (**<sw-param>**) oder einer Variablen.

Mögl. Werte calibration, no-calibration, not-in-memory

Default calibration

Enthalten in sw-param

Beispiel

[category]

Beschreibung Typ einer Grafik

Mögl. Werte barcode, conceptual, engineering, flowchart, graph, logo, schematic, waveform

Default IMPLIED

Enthalten in graphic

Beispiel

[electric-current-expo]

Beschreibung Electric current (ampere).

Mögl. Werte

Default IMPLIED

Enthalten in si-unit

Beispiel

[ext-id-class]

Beschreibung Externe Id-Klasse

Mögl. Werte

Default IMPLIED

Enthalten in xref

Beispiel

[f-child-type]

Beschreibung

Mögl. Werte

Default

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

Beispiel

[f-id-class]

Beschreibung Alle Objekte, die einen Namen tragen/einen **<short-name>** haben und damit eine **[id]** haben, werden durch das Attribut **[f-id-class]** markiert. Der Wert von **[f-id-class]** wird gleich dem Elementnamen des Objekts gewählt.

Mögl. Werte 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

Enthalten in xref

Beispiel

[idref]

Beschreibung

Mögl. Werte

Default REQUIRED

Enthalten in

Beispiel

[f-namespace]

Beschreibung Um z.B. unterschiedliche Teilbäume ohne Namenskonflikte zusammenfügen zu können, werden Namensraumgrenzen eingeführt. Diese Namensraumgrenzen werden durch das Attribut **[f-namespace]** markiert, dessen Wert mit dem Attribut **[f-id-class]** des jeweiligen Objekttyps übereinstimmt. Alle Objekte eines Typs müssen innerhalb dieses Teilbaumes verschieden benannt werden.

Mögl. Werte

Default

Enthalten in

Beispiel

[f-pubid]

Beschreibung

Mögl. Werte

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

Enthalten in msrsw

Beispiel

[id]

Beschreibung Identifier

Mögl. Werte

Default REQUIRED

Enthalten 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

Beispiel

[interpolation-style]

Beschreibung Art der Interpolation bei einer Umrechnungstabelle.

Mögl. Werte interpolation, no-interpolation, discrete

Default interpolation

Enthalten in sw-compu-method-table

Beispiel

[length-expo]

Beschreibung Length in metre.

Mögl. Werte

Default IMPLIED

Enthalten in si-unit

Beispiel

[luminous-intensity-expo]

Beschreibung Luminous intensity in candela.

Mögl. Werte

Default IMPLIED

Enthalten in si-unit

Beispiel

[mass-expo]

Beschreibung Mass in gram.

Mögl. Werte

Default IMPLIED

Enthalten in si-unit

Beispiel

[owns]

Beschreibung In **<sw-param-ref>** mit möglichen Attributwerten ”noown”.

Mögl. Werte noown

Default IMPLIED

Enthalten in sw-param-ref

Beispiel

[s] *Beschreibung* Signature. Anhand der Signatur kann festgestellt werden, was sich beim Austausch in einer Instanz geändert hat. Die Signatur wird durch ein Werkzeug erzeugt (z.B. timestamp).

Mögl. Werte

Default IMPLIED

Enthalten in Allen Elementen.

Beispiel

[sw-compu-method]

Beschreibung In **<sw-compu-method-ref>** Verweis auf Umrechnungsformel.

Mögl. Werte

Default REQUIRED

Enthalten in sw-compu-method-ref

Beispiel

[sw-function]

Beschreibung	In **<sw-function-ref>** Verweis auf Funktion
Mögl. Werte	
Default	REQUIRED
Enthalten in	sw-function-ref
Beispiel	

[sw-param]

Beschreibung	In **<sw-param-ref>** Verweis auf Parameter.
Mögl. Werte	
Default	REQUIRED
Enthalten in	sw-param-ref
Beispiel	

[termodynamic-temperature-expo]

Beschreibung	Temperature in kelvin.
Mögl. Werte	
Default	IMPLIED
Enthalten in	sw-unit
Beispiel	

[time-expo]

Beschreibung	Time in seconds.
Mögl. Werte	
Default	IMPLIED
Enthalten in	si-unit
Beispiel	

4 Übersicht Changes

[SSW] MSR DOC Strukturelle Grundlagen Bereich Software

4.1 Übersicht zu Zustand nach Abgleichung mit ASAP

geplant für 26.04.96

4.2 Übersicht zu Zweite Überarbeitung

geplant für 29.04.96

4.3 Übersicht zu Bereinigung

geplant für 03.06.96

Tabelle 5: Requests zu Bereinigung

Name	Art	Thema	Status	Prio	MT	geplant für	S.
owns	enh	sw-param-ref owns="owns"	**Offen**	A 2			S. 80
paramunit	enh	Parameterinhalte brauchen eine Maßeinheit	**in Arbeit** abgeschlossen			Bereinigung	S. 80
paramhex	enh	Parameterinhalte auf Integer/ HEX/Adressniveau	**in Arbeit** abgeschlossen			Bereinigung	S. 80
paramfunc	enh	Parameterinhalte müssen Funktionen zuordenbar sein	**in Arbeit** abgeschlossen			Bereinigung	S. 81
kgrhierachie		Unterstützung von Kenngrößenhierarchien	**in Arbeit** abgeschlossen			Bereinigung	S. 81
kgarray	bug	Kenngrößeninhalte für arrays	**in Arbeit** abgeschlossen			Bereinigung	S. 81
sprmref	enh	SW-param-ref in Kenngrößeninhalten semantisch	**angenommen** vgl. semantische Referenz in			Bereinigung	S. 82
annot	enh	Notizobjekte	**in Arbeit** abgeschlossen			Bereinigung	S. 82

Tabelle 5 (Forts.): Requests zu Bereinigung

Name	Art	Thema	Status	Prio	MT	geplant für	S.
ndim-array	enh	Unterstützung mehrdimensionaler arrays	**angenommen** abgeschlossen	SW 09			S. 83
unknown-SW	enh	Behandlung bisher nicht betrachteter Software Parameterformen	**abgelehnt**				S. 83
cleanup-sw-datatypes	enh	SW-Datatypes sollte bereinigt werden	**angenommen**	10			S. 84
prog-code	enh	prog-code in sw-compu-method	**in Arbeit** abgeschlossen	B1			S. 84
sw-gen-math	enh	sw-compu-generic-math	**Offen** H. Rauleder verfolgt diesen Punkt weiter.				S. 85
Osek	enh	OSEK-Aspekte	**Offen** H. Bless kümmert sich um eine Element-Liste				S. 85
diagnose	enh	Behandlung von Diagnose/ Diagnosestand	**Offen**				S. 85
literatur	doc	Literaturverzeichnis vervollständigen	**Offen**				S. 86
security	enh	Sicherheitsrelevanz von SW-Funktionen behandeln	**Offen**				S. 86
glosssar	doc	Glossar abstimmen	**Offen**	C1			S. 86
schedule	enh	Beschreibung von zeitlichen Abhängigkeiten	**Offen**			Bereinigung	S. 87
va-sample		Implementierung bei Variablen erweitern	**in Arbeit** abgeschlossen				S. 87

Tabelle 5 (Forts.): Requests zu Bereinigung

Name	Art	Thema	Status	Prio	MT	geplant für	S.
param-basic		Basistyp bei Kenngrößen einführen	**in Arbeit** abgeschlossen				S. 87
msrsw	enh	Root-Element sw in msrsw umbenennen	**in Arbeit** abgeschlossen	C1			S. 88
fktvars	enh	Funktionsvariable optional	**in Arbeit** abgeschlossen	C1			S. 88
label	enh	Platzhalter für Long-name bei param-content einführen	**in Arbeit**	A1			S. 88
admin-in-pcont		Administrative Daten bei Kenngrößeninhalten	**angenommen**				S. 89
fref-in-pcont		Funktionsreferenz bei Kenngrößeninhalte	**Offen**				S. 89
list-adr		Listen von Adressierschemas, Ablageschemas und Codesyntaxen.	**in Arbeit**				S. 89
glosopt		Glossar optional	**Offen**				S. 90
units		Maßeinheiten	**Offen** open				S. 90
variables		Variable	**Offen** open				S. 90

Tabelle 6: Lösungen in Bereinigung

Name	Art	Thema	Status	Prio	MT	Aufgetreten in	S.
paramunit	enh	Parameterinhalte brauchen eine Maßeinheit	**in Arbeit** abgeschlossen			Bereinigung	S. 80
paramhex	enh	Parameterinhalte auf Integer/ HEX/Adressniveau	**in Arbeit** abgeschlossen			Bereinigung	S. 80
paramfunc	enh	Parameterinhalte müssen Funktionen zuordenbar sein	**in Arbeit** abgeschlossen			Bereinigung	S. 81

Tabelle 6 (Forts.): Lösungen in Bereinigung

Name	Art	Thema	Status	Prio	MT	Aufgetreten in	S.
kgrhiera-chie		Unterstützung von Kenngrößenhierar-chien	**in Arbeit** abge-schlossen			Bereinigung	S. 81
kgarray	bug	Kenngrößeninhalte für arrays	**in Arbeit** abge-schlossen			Bereinigung	S. 81
sprmref	enh	SW-param-ref in Kenngrößeninhal-ten semantisch	**angenom-men** vgl. seman-tische Re-ferenz in			Bereinigung	S. 82
annot	enh	Notizobjekte	**in Arbeit** abge-schlossen			Bereinigung	S. 82
schedule	enh	Beschreibung von zeitlichen Abhän-gigkeiten	**Offen**			Bereinigung	S. 87

5 Changes

5.1 [owns] sw-param-ref owns=”owns”

Art	Vorgeschlagen von	Status	Prio	Tage	Aufgetreten in	Geplant für
enh	Wl	**Offen**	A 2		[10] S. 76	

Sache Das Attribut **[owns]** in **<sw-param-ref>** muß auch den Wert ”owns” zulassen.

Begründung Sonst bleibt das Attribut implied, was die Verarbeitung erschwert.

5.2 [paramunit] Parameterinhalte brauchen eine Maßeinheit

Art	Vorgeschlagen von	Status	Prio	Tage	Aufgetreten in	Geplant für
enh	Hünerfeld	**in Arbeit** abgeschlossen			[10] S. 76	[10] S. 76

Sache Parameterinhalte **<sw-param-contents>** brauchen Maßeinheiten.

Begründung Die Parameterinhalte werden verwendet, um physikalische Daten zu dokumentieren wie auch mit anderen Systemen auszutauschen. Bei einem derartigen Austausch ist ggf. nicht das ganze Datenlexikon verfügbar. Daher müssen die Maßeinheiten zumindest optional angebbar sein.

Beschlossene Lösung

Element **<sw-unit>** einführen.

Release notes

5.3 [paramhex] Parameterinhalte auf Integer/HEX/Adressniveau

Art	Vorgeschlagen von	Status	Prio	Tage	Aufgetreten in	Geplant für
enh	weichel	**in Arbeit** abgeschlossen			[10] S. 76	[10] S. 76

Sache Die Parameterinhalte müssen nicht nur als Integergroessen darstellbar sein.

Begründung Für die Dokumentation ist es sinnvoll, auch integer-Darstellungen wie auch Adressinformationen abzulegen.

Beschlossene Lösung

Die Kinder von **<sw-param-content>** (**<sw-param-content-x>** usw.) bekommen für jede Repräsentation (Binäre, Hex, Adresse usw). ein eigenes Element.

Release notes

5.4 [paramfunc] Parameterinhalte müssen Funktionen zuordenbar sein

Art	Vorgeschlagen von	Status	Prio	Tage	Aufgetreten in	Geplant für
enh	hünerfeld	**in Arbeit** abgeschlossen			[10] S. 76	[10] S. 76

Sache Die Parameterinhalte sollten Funktionen zuordenbar sein - zumindest optional.

Begründung Im Autonom-Betrieb kann man eine Funktionsorientierte Datenverwaltung aufsetzen

Beschlossene Lösung

<sw-param-content> bekommt die Möglichkeit, den Inhalt einer Funktion zuzuordnen, d.h. auf eine Funktion zu referenzieren (**<sw-function-ref>**)

Release notes

5.5 [kgrhierachie] Unterstützung von Kenngrößenhierarchien

Art	Vorgeschlagen von	Status	Prio	Tage	Aufgetreten in	Geplant für
	hünerfeld	**in Arbeit** abgeschlossen			[10] S. 76	[10] S. 76

Sache Es müssen auch Kenngrößenstrukturen unterstützt werden können.

Begründung Wenn man Variablenstrukturen bildet, muß das auch für Kenngrößen gehen. Solche Strukturen werden bereits eingesetzt.

Beschlossene Lösung

Rekursive Wiederholung von **<sw-params>** in **<sw-param>** können auch Kenngrößenhierarchien dargestellt werden.

Release notes

5.6 [kgarray] Kenngrößeninhalte für arrays

Art	Vorgeschlagen von	Status	Prio	Tage	Aufgetreten in	Geplant für
bug	Hünerfeld	**in Arbeit** abgeschlossen			[10] S. 76	[10] S. 76

Sache Kenngrößeninhalte müssen auch für Arrays spezifizierbar sein.

Begründung Sonst ist die Sache unvollständig

Beschlossene Lösung

<arraysize> wird in **<sw-param>** eingeführt, ebenso **<sw-array-index>** in **<sw-param-content>** .

Release notes

5.7 [sprmref] SW-param-ref in Kenngrößeninhalten semantisch

Art	Vorgeschlagen von	Status	Prio	Tage	Aufgetreten in	Geplant für
enh	Hünerfeld Weichel	**angenommen** vgl. semantische Referenz in			[10] S. 76	[10] S. 76

Sache **<sw-param-ref>** in **<sw-param-contents>** muß semantisch sein.

Begründung
- Unterstützung von reinen Kenngrößeninhaltsdateien
- ein Leser hat ggf. das Datenlexikon nicht zur Verfügung

Beschlossene Lösung

Wird im Rahmen der semantischen Referenzierung allgemein behandelt. Diese wird auf einer Hierarchie von **<short-names>** beruhen.

Release notes

5.8 [annot] Notizobjekte

Art	Vorgeschlagen von	Status	Prio	Tage	Aufgetreten in	Geplant für
enh	Hünerfeld	**in Arbeit** abgeschlossen			[10] S. 76	[10] S. 76

Sache Aus dem Datenlexikon bzw. aus den Funktionsdefinitionen sollten Notizobjekte extrahierbar sein, welche in nachgeschalteten Systemen wieder angezeigt werden.

Begründung Damit kann man aus der Spezifikation heraus Anzeigen in nachgeschalteten Systemen steueren. Solche Systeme können auch Informationen über in die Dokumentation zurückggeben. Diese Notizobjekte müssen vorhanden sein in:
- Variablen
- Kenngrößenstrukturen
- Funktionen
- Kenngrößeninhalten (Funktionszusammenfassung)

- Kenngrößeninhalten (einzeln)

Beschlossene Lösung

Die Notizobjekte können in **<annotations>** angelegt werden.

Release notes

5.9 [ndim-array] Unterstützung mehrdimensionaler arrays

Art	Vorgeschlagen von	Status	Prio	Tage	Aufgetreten in	Geplant für
enh	??	**angenommen** abgeschlossen	SW 09		[10] S. 76 [10] S. 76	

Sache Software-Variable sollte auch als mehrdimensionales Array auftreten können.

Begründung Weil es vorkommen kann.

Beschlossene Lösung

Das Inhaltsmodell von *arraysize* ist *#PCDATA*. Bei mehrdimensionalen Arrays werden die Dimensionen als Integerzahlen getrennt durch Leerzeichen eingegeben.

5.10 [unknown-SW] Behandlung bisher nicht betrachteter Software Parameterformen

Art	Vorgeschlagen von	Status	Prio	Tage	Aufgetreten in	Geplant für
enh	??	**abgelehnt**			[10] S. 76 [10] S. 76	

Sache Einführung einer ”Hintertür” für bislang nicht betrachtete Software-Parameter-Strukturen.

Begründung Es könnte sein, daß eine Software-Parameterform erfunden wird, welche nicht zu den bisherigen paßt.

Lösungsansatz 1

Einführung einer **<ncoi>** als weitere Option

Pro Nutzt vorhandene Elemente.

Bietet bei der Erfassung die notwendige Freiheit.

Con Software-Parameter-Strukturen werden sinnvollerweise nicht nur auf Papier ausgegeben, sondern auch maschinell im Entwicklungsprozeß bearbeitet. Diese wäre mit **<NCOI>** nicht mehr möglich.

Beschlossene Lösung

Dieser Fall wird zunächst nicht abgedeckt. Die Einführung weiterer Parameterformen hat Auswirkungen auf die gesamte Prozeßkette. Daher wäre es nicht ausreichend, hier eine Hintertür zu eröffenen.

5.11 [cleanup-sw-datatypes] SW-Datatypes sollte bereinigt werden

Art	Vorgeschlagen von	Status	Prio	Tage	Aufgetreten in	Geplant für
enh	??	**angenommen**	10		[10] S. 76	

Sache **<sw-datatype-scaling> <sw-resolution>** sollten bereinigt werden.

Begründung Diese Elemente scheinen redundant insbesondere bezogen auf **<sw-compu-method>**

Lösungsansatz 1

Man sollte versuchen, **<sw-data-type>** und **<sw-compu-mehtod>** generell zu verschmelzen.

Pro Damit können weitere Überdeckungen behandelt werden.

Con
- Es könnte sein, daß zwei gleichartige (d.h. vom gleichen Typ seiende) Variablen mit unterschiedlichen Umrechnungsformeln implementiert werden. In diesem Fall hätte man eine hohe Redundanz bei den Umrechnungsformeln.
- Vielleicht braucht man mal einen allgemeineren Datentyp als nur einen, der auf Berechungsformeln basiert. In diesem Fall müßte man eine Pseudo-Umrechnungsformeln definieren.
- Es kann Zwischengrößen geben, welche gar nicht mit Umrechungsformeln versehen werden. In diesem Fall müßte man eine Pseudo-Umrechnungsformel definieren[15].

Beschlossene Lösung

Die Bereinigung ist ausgeführt. Die Elemente wurden konsequent getrennt nach frühen Phasen (**<sw-physic-types>** und **<sw-varable-implementation>**)

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

Art	Vorgeschlagen von	Status	Prio	Tage	Aufgetreten in	Geplant für
enh	AG-SW	**in Arbeit** abgeschlossen	B1		[10] S. 76	

Sache Beschreibung von Umrechnungsformeln in Programiersprachennotation.

15

Begründung Immer mehr Verarbeitungswerkzeuge unterstützen die Formulierung von Umrechnungformeln als Programmiersprachenfragment.

Beschlossene Lösung

<prog-code > wird parallel zu **<sw-compu-method-text>** eingeführt.

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

Art	Vorgeschlagen von	Status	Prio	Tage	Aufgetreten in	Geplant für
enh	??	**Offen** H. Rauleder verfolgt diesen Punkt weiter.			[10] S. 76	

Sache Auf SGML-Europe-96 wird ein semantisches Mathematik-Modell vorgestellt. Dieses sollte in MSR-DOC verwendet werden.

Begründung Damit können proprietäre Ansätze überwunden werden.

Lösungsansatz 1

5.14 [Osek] OSEK-Aspekte

Art	Vorgeschlagen von	Status	Prio	Tage	Aufgetreten in	Geplant für
enh	??	**Offen** H. Bless kümmert sich um eine Element-Liste			[10] S. 76 [10] S. 76	

Sache OSEK-Belange sind ebenfalls in die MSR-DTD bzw. die Strukturellen Grundlagen Software aufzunehmen.

Begründung Wenn *OSEK*-konform entwickelt wird, ist es notwendig, eine strukturierte Dokumentation zu haben. Wichtig sind hierbei insbesondere Generierungsparameter usw.

Lösungsansatz 1

Formulierung der Beschreibungssprache in SGML im Rahmen von Medoc.

Pro Ist sicher konform zu MSR-DOC.

Man könnte mit Hilfe von SGML-Werkzeugen eine Eingabeumgebung realisieren.

Con • SGML-Editoren sind nicht notwendigerweise in die Codireungsumgebungen der Software-Entwickler eingebunden. Die OSEK-Anweisungen und Konfigurationsdateien stehen üblicherweise im Quelltext der Steuergeräte-Software (z.B. *C-Editor*).

Beschlossene Lösung

5.15 [diagnose] Behandlung von Diagnose/Diagnosestand

Art	Vorgeschlagen von	Status	Prio	Tage	Aufgetreten in	Geplant für
enh	AG	**Offen**			[10] S. 76	

Sache Es ist zu klären, inwieweit die Themen Diagnose und Diagnosestand durch die bisherige Modellierung abgedeckt.

Begründung AG hat sich bislang nicht intensiv mit Diagnose befaßt.

Beschlossene Lösung

5.16 [literatur] Literaturverzeichnis vervollständigen

Art	Vorgeschlagen von	Status	Prio	Tage	Aufgetreten in	Geplant für
doc	Rauleder	**Offen**			[10] S. 76	

Sache Das Literaturverzeichnis muß vervollständigt werden.

Begründung

Beschlossene Lösung

5.17 [security] Sicherheitsrelevanz von SW-Funktionen behandeln

Art	Vorgeschlagen von	Status	Prio	Tage	Aufgetreten in	Geplant für
enh	AG	**Offen**			[10] S. 76	

Sache Es ist prinzipiell zu klären, wie Aspekte der Sicherheit bzw. Sicherheitsrelevanz bezogen auf Softwarefnktionen und allgemein abgebildet werden soll.

Begründung Anwendung der *msrsw.dtd* auch für sicherheitsrelevante Systeme.

Beschlossene Lösung

5.18 [glosssar] Glossar abstimmen

Art	Vorgeschlagen von	Status	Prio	Tage	Aufgetreten in	Geplant für
doc	AG	**Offen**	C1		[10] S. 76	

Sache Das Glossar muß abgestimmt werden.

Begründung Damit es stimmt.

Beschlossene Lösung

5.19 [schedule] Beschreibung von zeitlichen Abhängigkeiten

Art	Vorgeschlagen von	Status	Prio	Tage	Aufgetreten in	Geplant für
enh	AG	**Offen**			[10] S. 76	[10] S. 76

Sache Die Beschreibungsmittel für zeitliche Abhängigkeiten und Scheduling-Informationen müssen analysiert und eingebaut werden.

Begründung System und Designtools berücksichtigen zunehmend diese Informationen.

Siehe auch **sonst:** Kapitel 5.14 [Osek] OSEK-Aspekte S. 85

Beschlossene Lösung

Release notes

5.20 [va-sample] Implementierung bei Variablen erweitern

Art	Vorgeschlagen von	Status	Prio	Tage	Aufgetreten in	Geplant für
	RB	**in Arbeit** abgeschlossen			[10] S. 76	

Sache **<sw-variable-sample-rate>** auch bei **<sw-variable-implementation>** aufnehmen.

Begründung Bei den physikalischen Daten wird ein Minimalforderung angegeben, bei der Implementierung die tatsächliche Realisierung.

5.21 [param-basic] Basistyp bei Kenngrößen einführen

Art	Vorgeschlagen von	Status	Prio	Tage	Aufgetreten in	Geplant für
	RB	**in Arbeit** abgeschlossen			[10] S. 76	

Sache Bei **<sw-param>** sollte**<sw-base-type>** (C-Datentyp) bzgl. des Wertes der Kenngröße optional eingeführt werden.

5.22 [msrsw] Root-Element sw in msrsw umbenennen

Art	Vorgeschlagen von	Status	Prio	Tage	Aufgetreten in	Geplant für
enh	RB	**in Arbeit** abgeschlossen	C1		[10] S. 76	

Sache Das Root-Element **<sw>** in **<msrsw>** umbenennen.

Begründung Entsprechend allgemeiner Namensvergabe in DTDs.

5.23 [fktvars] Funktionsvariable optional

Art	Vorgeschlagen von	Status	Prio	Tage	Aufgetreten in	Geplant für
enh	RB	**in Arbeit** abgeschlossen	C1		[10] S. 76	

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

Begründung Gleichbehandlung wie Kenngrößen.

Beschlossene Lösung

<sw-function-variables> wird optional.

5.24 [label] Platzhalter für Long-name bei param-content einführen

Art	Vorgeschlagen von	Status	Prio	Tage	Aufgetreten in	Geplant für
enh	RB	**in Arbeit**	A1		[10] S. 76	

Sache Bei **<sw-param-contents>** und bei **<sw-param-content>** einen**< label>**l optional einführen. Der Label entspricht der Langbezeichnung der Funktion.

Begründung Für die Kalibrierungsphase werden Kenngrößeninhalte ohne Datenlexikon ausgetauscht. Eine Langbezeichnung für die Funkiton ist wünschenswert.

Beschlossene Lösung

Das Element **<label>** könnte bei Softwareentwicklern zur Verwechslung mit Assembler-Labeln führen. Elemente, die eine Abbildung eines anderen Namen, z.B. eines anderen **<long-name>** darstellen oder die nur für Layoutzwecke z.B. zur Definition von Navigatoren in Panorama Pro eingeführt werden, sollen durch ein neues Element ausgedrückt werden. Dieses Element heißt **<auto-text>** .

5.25 [admin-in-pcont] Administrative Daten bei Kenngrößeninhalten

Art	Vorgeschlagen von	Status	Prio	Tage	Aufgetreten in	Geplant für
	RB	**angenommen**			[10] S. 76	

Sache **<admin-data>** bei Kenngrößeninhalten einführen.

Begründung In der Kalibrierungsphase werden verschiedene Versionenen von Kenngrößeninhalten erzeugt.

5.26 [fref-in-pcont] Funktionsreferenz bei Kenngrößeninhalte

Art	Vorgeschlagen von	Status	Prio	Tage	Aufgetreten in	Geplant für
	RB	**Offen**			[10] S. 76	

Sache Einführung von **<sw-function-ref>** in**<sw-param-content>**.

Begründung Während der Kalibrierungsphase werden Instanzen mit funktionsbezogenen Kenngrößeninhalten ausgetauscht. Dieser Austausch erfolgt projektübergreifend und muß deshalb ohne Datenlexikon[16] erfolgen.

5.27 [list-adr] Listen von Adressierschemas, Ablageschemas und Codesyntaxen.

Art	Vorgeschlagen von	Status	Prio	Tage	Aufgetreten in	Geplant für
	RB	**in Arbeit**			[10] S. 76	

Sache Einführung von **<sw-addressing-methods>**, **<sw-code-syntaxes>** und **<sw-record-layouts>** in **<sw-data-dictionary>**.

16

Begründung Normierung und Beschreibung der Objekte, Vorbereitung für Codegenerierung.

5.28 [glosopt] Glossar optional

Art	Vorgeschlagen von	Status	Prio	Tage	Aufgetreten in	Geplant für
	RB	**Offen**			[10] S. 76	

Sache .

Begründung

5.29 [units] Maßeinheiten

Art	Vorgeschlagen von	Status	Prio	Tage	Aufgetreten in	Geplant für
	RB	**Offen** open			[10] S. 76	

Sache Einführung von Maßeinheiten im**<sw-data-dictionary>** Formale Referenzierung.

Begründung Die Maßeinheiten gehen direkt in die Verwendtbarkeit von physikalischen Größen ein und müssen deshalb formal behandelt werden. Normierung der Maßeinheiten.

Lösungsansatz 1

Im Datadictionary werden die verwendeten Maßeinheiten mit Langbezeichnung, Kurzbezeichnung aufgeführt. Falls es sich nicht um eine SI-Einheit handelt, wird auf die entsprechende SI-Einheit verwiesen und die Umrechnung zur bzw. von der SI-Einheit angegeben. Zusätzlich wird eine layoutspezifische Darstellung angegeben. Bei einer SI-Einheit entfällt der Verweis auf eine SI-Einheit und die Umrechnungen. Das Element **<sw-unit>** wird durch **<sw-unit-ref>** ersetzt. Siehe folgendes Beispiel.

5.30 [variables] Variable

Art	Vorgeschlagen von	Status	Prio	Tage	Aufgetreten in	Geplant für
	RB	**Offen** open			[10] S. 76	

Sache Ändern des Inhaltsmodells von Variablen **<sw-function-variables>**.

Begründung Bei Variablen muß zwischen der Definition einer Variablen und dem Zugriff auf eine Variable unterschieden werden.

Lösungsansatz 1

Variable, die von einer Funktion definiert werden, werden unter **<sw-function-export-variables>** aufgeführt. Wenn die Variable nur von der Funktion selbst benützt wird unter **<sw-function-**

local-variables>. Variable, die von der Funktion benutzt bzw. gelesen oder geschrieben werden, aber nicht von der Funktion definiert werden, werden unter **<sw-function-import-variables>** aufgeführt.

Bezüglich des Zugriffs auf Variablen wird zwischen lesend (**<sw-variables-read>**), schreibend (**<sw-variables-write>**) und lesend + schreibend (**<sw-variables-readwrite>**) unterschieden.

Anh. A Hinweise zum Processing

Zu jeder Software-Funktion sollen alle Software-Funktionen aufgelistet werden, die jene aufrufen. Dieser Zusammenhang ist in der obigen Struktur über die Software-Variablen und deren Richtung abgebildet.

In **<sw-limits>** werden Minimal- und Maximalwerte angegeben für die codierten (Steuergeräte-internen) und die physikalischen Werte. Ein Semanitk-Prüfer sollte die Konsistenz dieser Werte prüfen. Ggf. kann auch in einem nachgeschalteten Prozeß der eine aus dem andereren Wert errechnet werden, wenn dieser in der Lage ist, die Umrechnungsformeln auzuwerten.

Anh. B Erklärung der Near&Far-Symbole

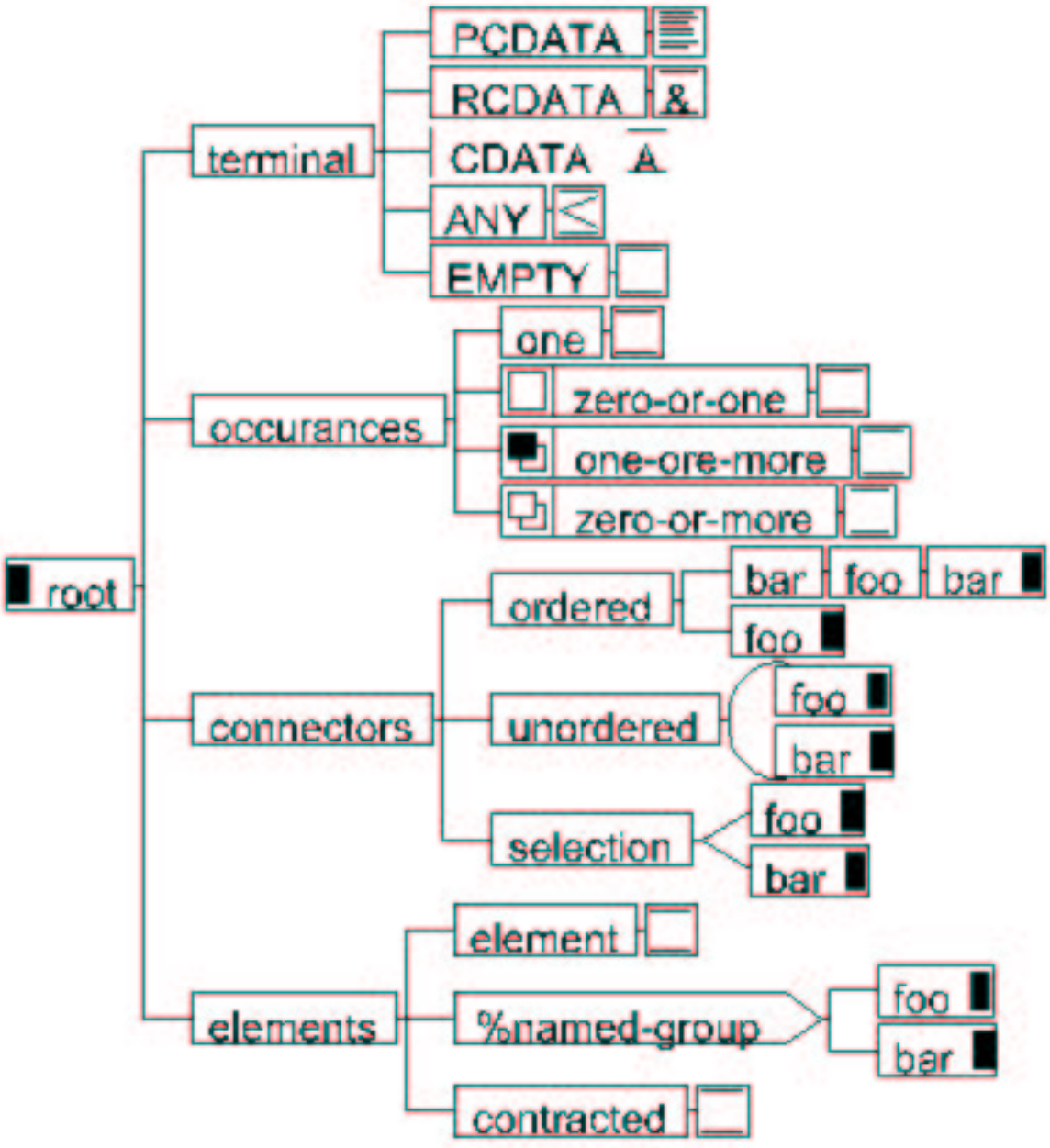

Abbildung 22: NearFar-Symbole

Anh. C Glossar

Das Glossar enthält einige zentrale Begriffe. Die Erläuterungen nehmen z.T. Bezug auf Begrifflichkeiten innerhalb der *MSRSYS.DTD*.

add-info, add-spec Elemente innerhalb der *MSRSYS.DTD*, die es ermöglichen, neben vorgegebenen Themen/Überschriften zusätzliche Spezifikationen (add-spec) und für Objekte neben der vorgegebenen Beschreibungsstruktur zusätzliche Informationen (add-info) freistrukturiert zu dokumentieren.

ASAP Die ASAP-Schnittstellen wurden vom ”Arbeitskreis zur Standardisierung von Applikationssystemen (*ASAP*)” vereinbart. Mitglieder dieses Arbeitskreises sind deutsche Automobilhersteller und Firmen der Zuliefererindustrie.

Daten Inhalte der Software-Parameter (Kenngrößeninhalte)

Datenstand Festgeschriebene Inhalte der Software-Parameter (Kenngrößeninhalte) zu einem bestimmten Zeitpunkt. Der Datenstand wird durch eine Versionsnummer gekennzeichnet.

Dekomposition Zerlegung einer Software-Funktion in Unterfunktionen.

Funktion Unter Funktion versteht man eine bestimmte Anforderung oder Eigenschaft des Gesamtsystems, die entweder durch Hardware, Software oder eine Kombination aus beidem realisiert werden kann.

Funktionsanalyse Die Funktionsanalyse enthält die Beschreibung des Kontexts, der Funktionen (hierarchisch, inkl. Diagnosefunktionen, Sicherheitsfunktionen etc.) und der Informationsflüsse (informell). Je Funktion werden eine formale Definition, eine informelle Beschreibung, Applikationshinweise und Kundendiensthinweise dokumentiert.

Hardware-Funktion Funktion innerhalb des Verhaltens, die in Hardware realisiert ist.

Hardware-Modul Bestandteil einer Hardware-Komponente (Part-Type); ggf. auch die Komponente selbst.

Integration Phase, in der die einzelnen Module zusammengelinkt werden. Die Integration kann auf der Test- und/oder der Zielumgebung durchgeführt werden.

Logische Funktion Eine Funktion innerhalb der Verhaltensbeschreibung.

MSRDOCDTD Innerhalb des Projekts MSR/MEDOC im Standard SGML (ISO 8879) definierte Austauschstruktur.

OSEK OSEK steht für: ”Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug” oder auf englisch: ”Open Systems and the Corresponding Interfaces for Automotive Electronics”. OSEK ist im Mai 1993 von Firmen der deutschen Automobilindustrie gegründet worden. Gründungspartner waren: BMW, Bosch, Daimler-Benz, Opel, Siemens und VW. Projektkoordinator ist das IIT an der Universität Karlsruhe (Institue of Industrial Information Sytems). 1994 kamen Peugot und Renault mit dem französischen Projekt VDX-approach (Vehicle Distribute eXecutive) hinzu. Es wurden die deutschen und die französichen Projekte in OSEK/VDX zusammengeführt. In OSEK/VDX werden die Themen Betriebssysteme, Kommunikation und Netzwerkmanagement behandelt. Siehe *[Externes Dokument: Proceedings of the 1st International Workshop on Open Systems in Automotive Networks / Dokumentnummer: ISBN 3-00-000259-6 / Datum: 9.10.1995 / Herausgeber: IIIT (Institut für Industrielle Informationstechnik), Universität Karlsruhe / URL: / relevante Stelle:]*

part-type, part Zentrale Elemente innerhalb der MSRDOCDTD für die Beschreibung der Hardware-Komponentenstruktur.

Processing Das Processing beschreibt (in dem Dokument ”Processing Guide für die MSR-DOCDTD”) die Prozessierungs- und Layoutdefinitionen für die MSRDOCDTD. D.h. es wird beschrieben, wie eine SGML-Instanz dieser DTD durch einen Formatierer layouttechnisch in ein druckfähiges Dokument umgesetzt wird.

Programm Ablauffähiger Teil der Definition Software S. 95Software im Steuergerät. Daten werden separat, also nicht als Bestandteil des Programmes behandelt, da sie in einem eigenen Prozeß entwickelt werden (vgl. Topic 1.2.8 Applikation (Kalibrierung) S. 11).

Programmierbarkeit Eigenschaft einer Komponente (z.B. eines Steuergeräts), die beschreibt, in welcher Form und in welchem Umfang die Komponente programmierbar ist (Bandende-, Offline- und Feldprogrammierbarkeit).

Programmstand Ausführbarer Code im Steuergerat ohne Daten (Kenngrößen usw.)

Sicht Die *MSRSYS.DTD* kann Daten in zwei Sichten (views) aufnehmen: ”requirements” und ”product-spec”.

Software Lauffähige Software, d.h. Programm und Daten in einem Steuergerät.

Software-Funktion Eine in Software realisierte Definition Funktion S. 94Funktion in einem Steuergerät.

Software-Modul Eine durch einen Compiler übersetzbare Einheit, die auch mehrere Software-Funktionen umfassen kann.

Softwarestand Kombination aus Definition Programmstand S. 95Programmstand und Definition Datenstand S. 94Datenstand

Systemanalyse In der Systemanalyse wird das System auf logischer Ebene beschrieben. Das entstehende logische Funktionsmodell kann simulierbar sein.

Systemdesign Beim Systemdesign wird aus dem logischen Funktionsmodell das physikalische Funktionsmodell abgeleitet.

Anh. D Literaturverzeichnis

Tabelle :

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. *[Externes Dokument: Proceedings of the 1st International Workshop on Open Systems in Automotive Networks / Dokumentnummer: ISBN 3-00-000259-6 / Datum: October 9,1995 / URL: / relevante Stelle:]*

Anh. E Allgemeine Strukturen

In dieser Anlage werden allgemeine Strukturen grafisch dargestellt, die in der MSRSYS.DTD definiert wurden und auch in der Softwaredokumentation verwendet werden oder in die von der Software referenziert wird.

Anh. E.1 Administrative Daten

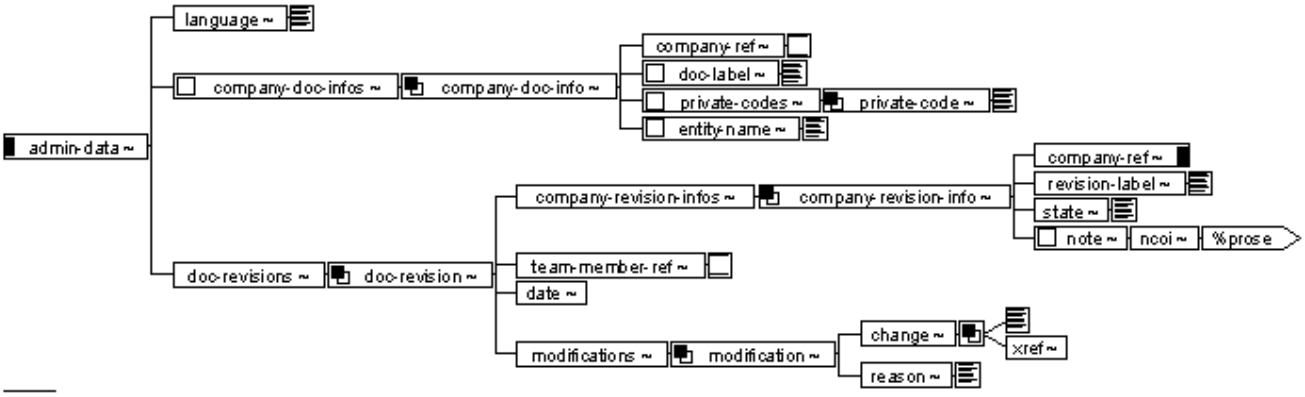

Abbildung 23: Administrative Daten

Anh. E.2 Parametermodell

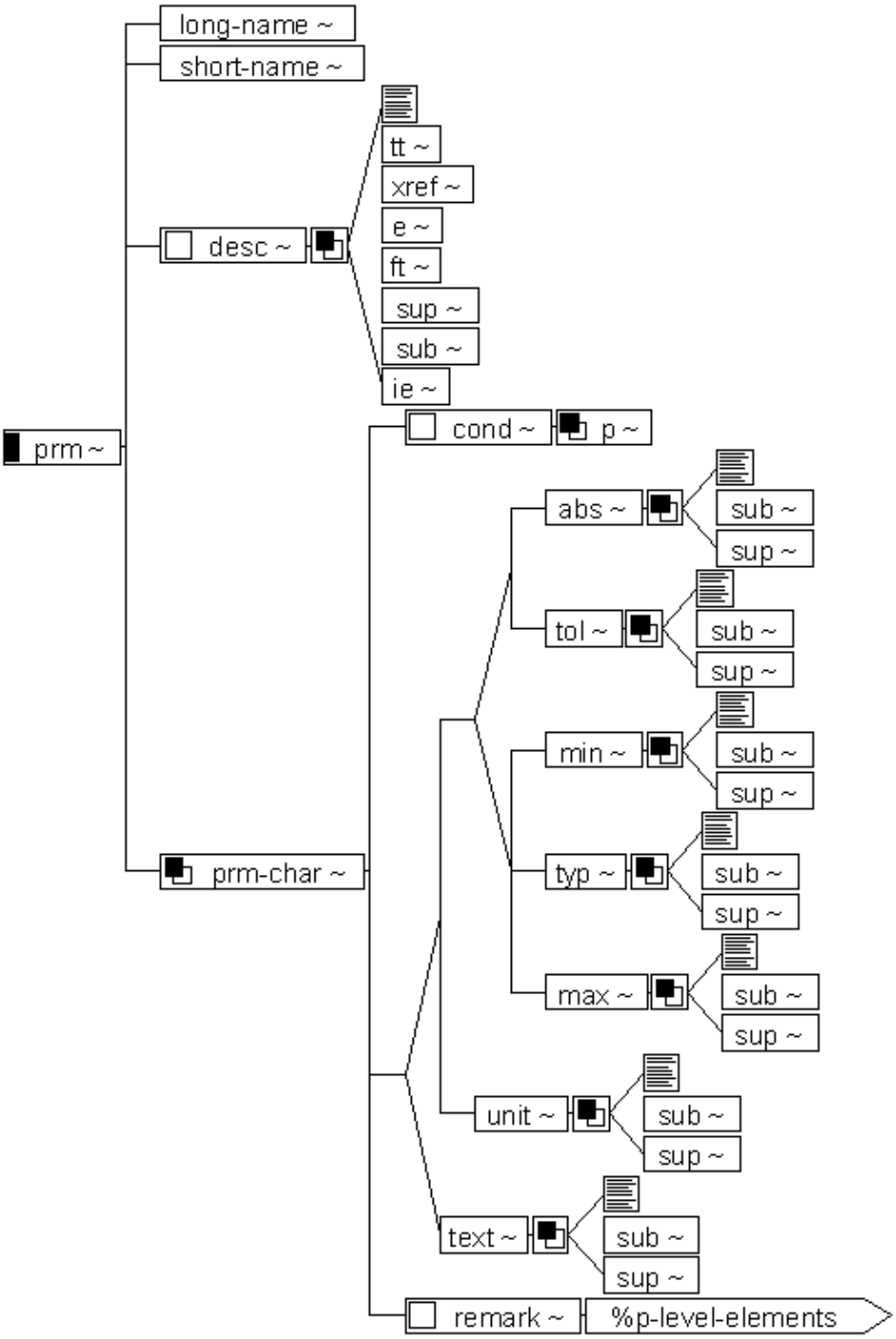

Abbildung 24: Parametermodell

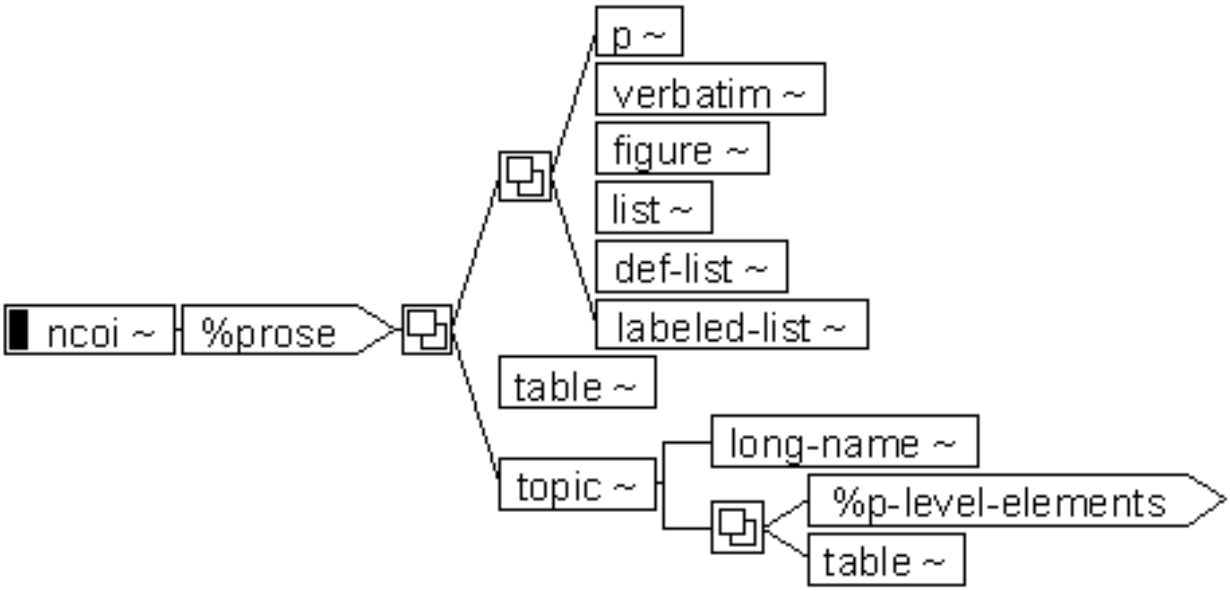

Abbildung 25: ncoi

Anh. E.3 Annotations

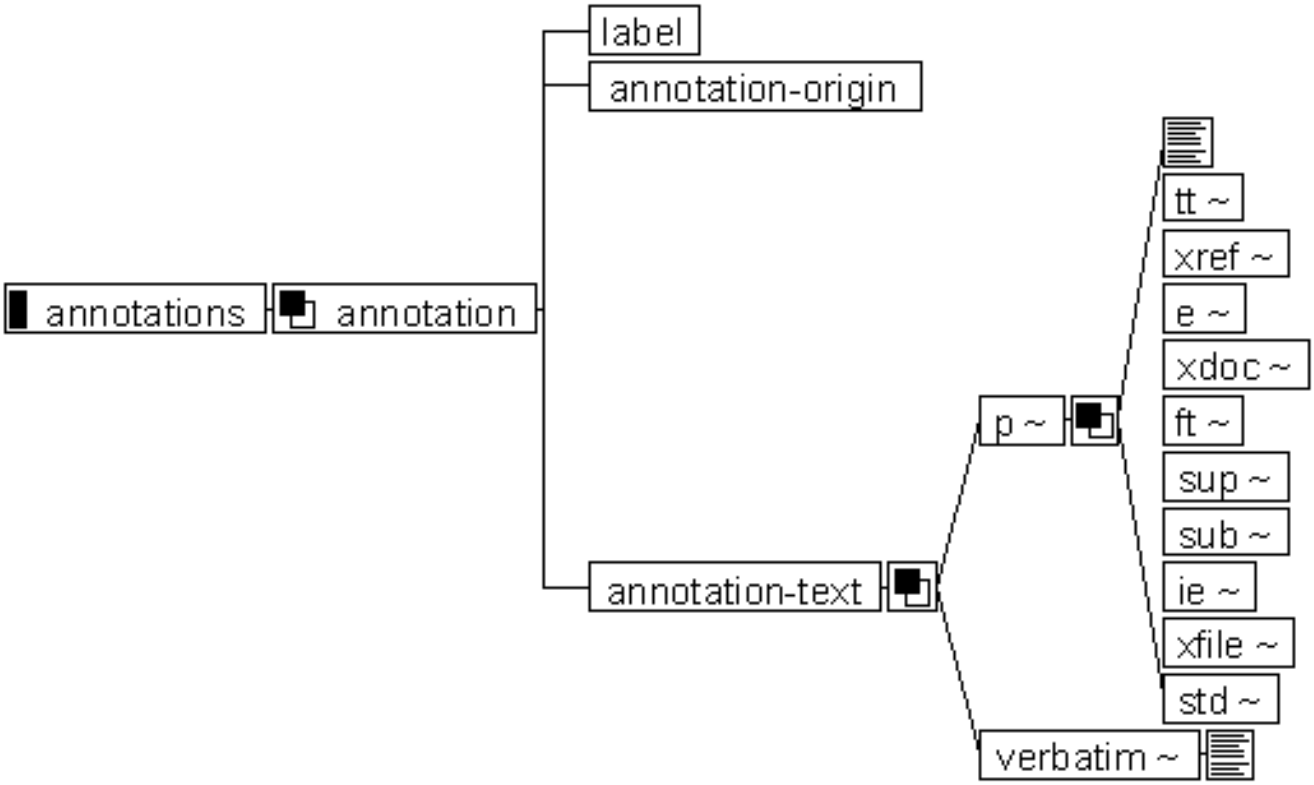

Abbildung 26: Annotationen

Anh. E.4 Zusätzliche Informationen

Erlaubt die Erfassung weiterer Informationen, die über die in der DTD vorgesehene Detaillierung hinausgehen.

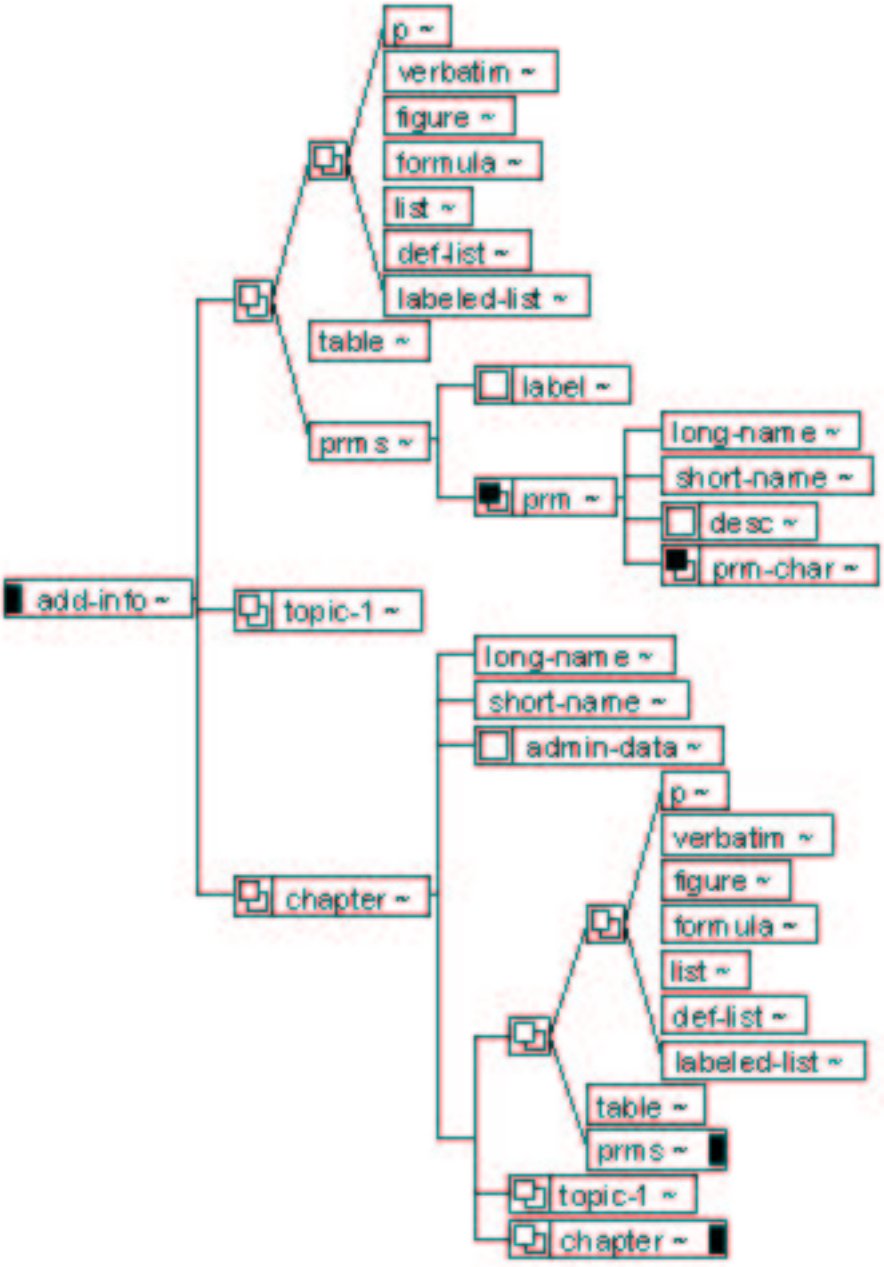

Abbildung 27: add-info

Anh. E.5 Topics

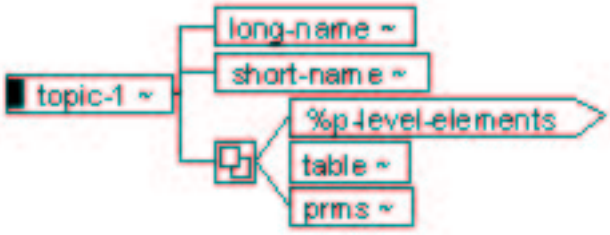

Abbildung 28: topic-1

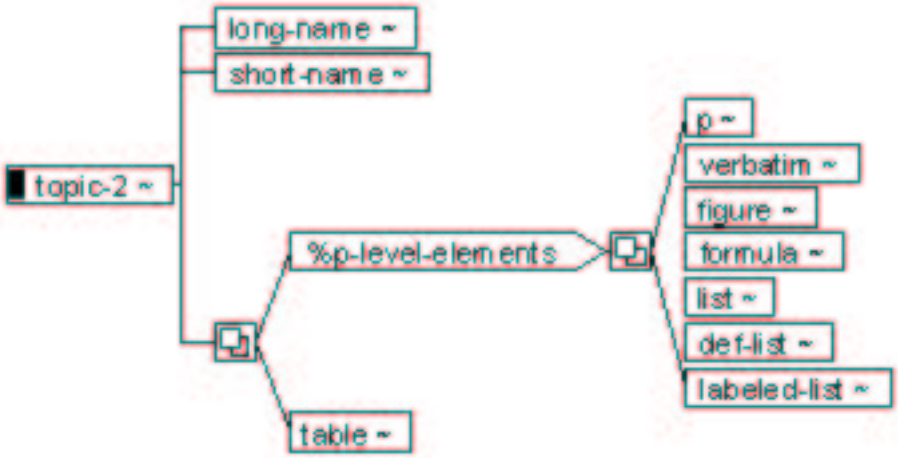

Abbildung 29: topic-2

Anh. E.6 Namelist

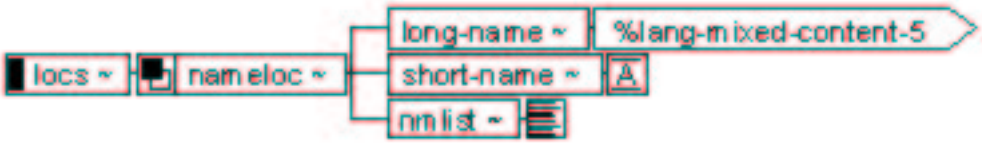

Abbildung 30: locs

Anh. E.7 Introduction

In der **<introduction>** kann eine kurze Einführung zum jeweiligen Objekt gegeben werden. Detaillierte Beschreibungen sollten in **<add-spec>** untergebracht werden.

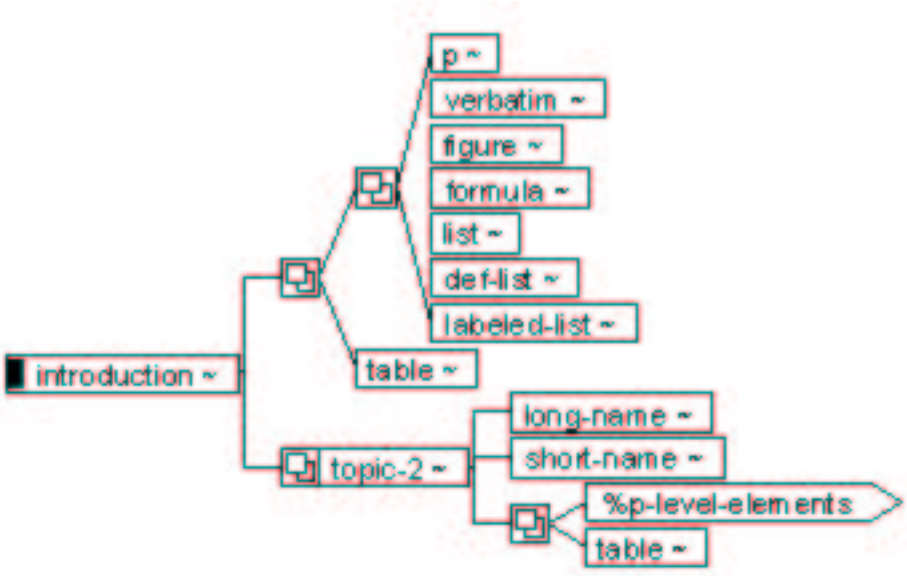

Abbildung 31: intro

Anh. E.8 SI-Einheiten

STEP (ISO/DIS 10303-41, S96 ff) stützt sich bzgl. SI-Einheiten auf die folgenden sieben Basis-Einheiten ab:

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

Anh. E.9 Mehrsprachige Dokumente

Über ein Konfigurationsfile kann die Software-DTD einsprachig oder mehrsprachig benutzt werden. Es wurde nicht für alle Elemente mit PCDATA-Content die Mehrsprachigkeit eingeführt. Elemente, die für die Referenzierung verwendet werden, müssen über Sprachgrenzen hinweg eindeutig sein. Beispiel **<short-name>**. Ein Beispiel für die Mehrsprachigkeit ist der **<long-name>**, siehe folgendes Bild und folgendes Beispiel.

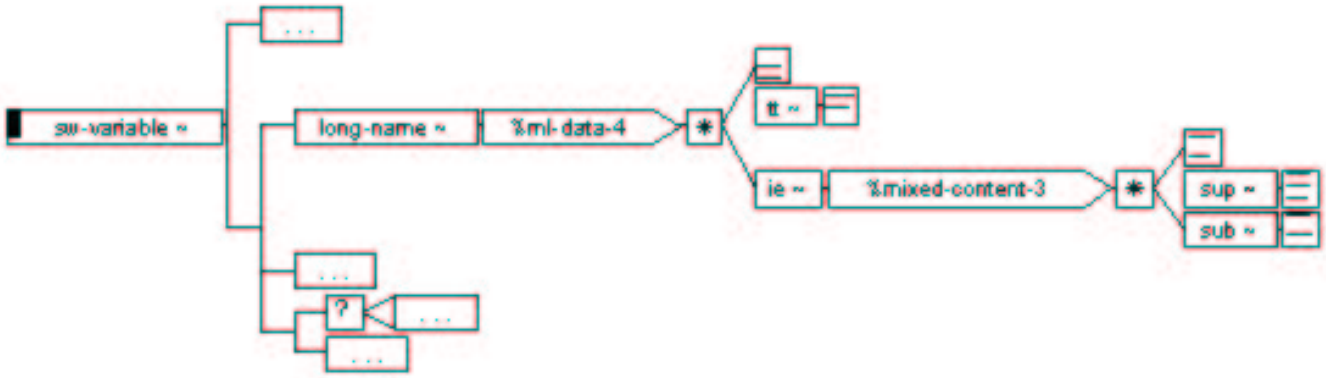

Abbildung 32: Beispiel long-name

Anh. F Zusätzliche Anmerkungen zum Dokumentstand 1.1.0a am 10.7.98

Status dieses Dokument: CD

Dokumentverwaltung

Tabelle : Beteiligte Personen

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

Tabelle : Versionsübersicht

Datum	Herausgeber
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

Tabelle : Änderungen

Änderung	Bezug
Dokument in neue Dokumentationsstruktur (entprechend TR-DOV übernommen) Grund: Neue Struktur wurde festgelegt	Inhalt
Redaktionelle Überarbeitung, Bereinigung der Strukturen Grund: Erstellung des Abschlußdokumentes	Inhalt
Beginn der redaktionellen Überarbeitung Grund:	Inhalt

Tabelle (Forts.): Änderungen

Änderung	Bezug
Ersterstellung Grund: Veröffentlichung der ersten offiziellen Version	Dokument
Ersterstellung Grund: Veröffentlichung der ersten offiziellen Version	Dokument
Ändern des Inhaltsmodells von Variablen Grund: Bei Variablen muß zwischen der Definition und dem Zugriff auf eine Variable unterschieden werden.	Dokument
Dokument	Dokument
Inhalt	In der Abbildug der <sw-unit> auf eine SI-Einheit wird ein zusätzliches Element <si-unit> eingeführt. Grund: Dies wird benötigt, wenn man mit Werkzeugen arbeitet, die nur mit SI-Einheiten rechnen können.
Dokument	Dokument
Dokument	Dokument
Physikalischer Datentyp bei Implementierung von Variablen eingeführt, Basisdatentyp/C-Typ bei Parametern eingeführt, Maßeinheiten im Datadictionary, weitere Elemente für verschiedene Fragmentierungsmodelle Grund: Erweiterung, Verbesserung, Fragmentierung	Dokument
Dokument	Beispiele hinzugefügt, Redaktionelle Überarbeitung Grund: Verständlichkeit
Dokument	Dokument
Redaktionelle überarbeitung von Topic 1.2 Phasenmodell zur Erstellung von Software S. 9 Grund: Übersichtlichkeit, Verständlichkeit	Dokument
Dokument	Inhalt
Inhalt	Inhalt
Inhalt	Inhalt
Inhalt	Variantenbehandlung, Inhaltsmodell Software neu strukuriert Grund: Vervollständigung
Dokument	Dokument
Inhalt	Inhalt

Tabelle (Forts.): Änderungen

Änderung	Bezug
Dokument	Dokument
aktuelle Near&Far Grafiken eingebaut Grund: Aktualisierung	Dokument
Erweiterung Beschreibung der Elemente. Grund: Vervollständigung.	Inhalt
Ebenen der Standardisierung eingebaut Grund: Erweiterung	Dokument
Inhalt	Inhalt
Schreibfehler korrigiert Grund: Qualitätssteigerung	Dokument
Inhalt	Changes eingeführt Grund: Zur gezielten Weiterentwicklung
Inhalt	Dokument
Umstellung auf SGML. Grund: Arbeitskreisbeschluß.	Dokument
Inhalt	Inhalt
Inhalt	Inhalt
Inhalt	Inhalt

Tabelle : Enthaltene Änderungen

Datum	Kapitel	Änderung	Bezug
Nr. 1, 10.7.98	Gesamt	Dokument in neue Dokumentationsstruktur (entprechend TR-DOV übernommen) Grund: Neue Struktur wurde festgelegt	Inhalt
Nr. 2, 27.4.98	Gesamt	Redaktionelle Überarbeitung, Bereinigung der Strukturen Grund: Erstellung des Abschlußdokumentes	Inhalt
Nr. 3, 5.3.98	Gesamt	Beginn der redaktionellen Überarbeitung Grund:	Inhalt
Nr. 4, 16.12.97	Gesamt	Ersterstellung Grund: Veröffentlichung der ersten offiziellen Version	Dokument
Nr. 5, 25.08.97	Gesamt	Ersterstellung Grund: Veröffentlichung der ersten offiziellen Version	Dokument

Tabelle (Forts.): Enthaltene Änderungen

Datum	Kapitel	Änderung	Bezug
Nr. 6, 12.08.97	Gesamt	Ändern des Inhaltsmodells von Variablen Grund: Bei Variablen muß zwischen der Definition und dem Zugriff auf eine Variable unterschieden werden.	Dokument
		Aktualisierung der Grafiken Grund: Vollständigkeit, Aktualisierung	Dokument
		Redaktionelle Überarbeitung Grund: Vollständigkeit, Aktualisierung	Dokument
		Versionierung von Variablen Grund: Entspricht der Praxis	Inhalt
Nr. 7, 17.06.97	Gesamt	In der Abbildug der <sw-unit> auf eine SI-Einheit wird ein zusätzliches Element <si-unit> eingeführt. Grund: Dies wird benötigt, wenn man mit Werkzeugen arbeitet, die nur mit SI-Einheiten rechnen können.	Dokument
		Mehrere Datadictionaries Grund: Es gibt Werkzeuge, die Datadictionaries funktionsspezifisch behandeln.	Dokument
		Projektdaten eingeführt Grund: Vollständigkeit	Dokument
		In <sw-function-variant> die Parameterinhalte analog zu den Datadictionaries geändert. Grund: Bereinigung	Dokument
Nr. 8, 10.04.1997	Gesamt	Physikalischer Datentyp bei Implementierung von Variablen eingeführtt, Basisdatentyp/C-Typ bei Parametern eingeführt, Maßeinheiten im Datadictionary, weitere Elemente für verschiedene Fragmentierungsmodelle Grund: Erweiterung, Verbesserung, Fragmentierung	Dokument
		Redaktionelle Überarbeitung Grund: Übersichtlichkeit, Schreibfehler beseitigt	Dokument
Nr. 9, 17.02.1997	Gesamt	Beispiele hinzugefügt, Redaktionelle Überarbeitung Grund: Verständlichkeit	Dokument
		Redaktionelle Überarbeitung Grund: Übersichtlichkeit, Schreibfehler beseitigt	Dokument

Tabelle (Forts.): Enthaltene Änderungen

Datum	Kapitel	Änderung	Bezug
Nr. 10, 13.01.1997	Gesamt	Redaktionelle überarbeitung von Topic 1.2 Phasenmodell zur Erstellung von Software S. 9 Grund: Übersichtlichkeit, Verständlichkeit	Dokument
		Aktionsliste in Changes eingebaut Grund: Vereinheitlichung der Vorgehensweise	Dokument
		Einführung von physical types in (Topic 2.2.3.2 Physikalische Datentypen S. 21) Grund: Arbeitsweise von Anwendern (mit Excel-Listen)	Inhalt
		Einführung von Maßeinheiten bei Parameterinhalten (Änderungsanforderung 5.2 [paramunit] Parameterinhalte brauchen eine Maßeinheit S. 80). Grund: Die Parameterinhalte werden verwendet, um physikalische Daten zu dokumentieren und mit anderen Systemen auszutauschen. Bei einem Austausch ist ggf. nicht das ganze Datenlexikon verfügbar. Daher müssen die Maßeinheiten zumindest optional angebbar sein.	Inhalt
		Die Parameterinhalte müssen nicht nur als Integergrößen darstellbar sein (Änderungsanforderung 5.3 [paramhex] Parameterinhalte auf Integer/HEX/Adressniveau S. 80). Grund: Für die Dokumentation sinnvoll.	Inhalt
		Parameterinhalte sollen Funktionen zuzuordnen sein (Änderungsanforderung 5.4 [paramfunc] Parameterinhalte müssen Funktionen zuordenbar sein S. 81). Grund: Im Autonom-Betrieb kann man eine funktionsorientierte Datenverwaltung aufsetzen.	Inhalt
		Unterstützung von Kenngrößenstrukturen (Änderungsanforderung 5.5 [kgrhierachie] Unterstützung von Kenngrößenhierarchien S. 81). Grund: Werden in der Praxis bereits eingesetzt.	Inhalt
		Einführung von Notiz-Objekten (Änderungsanforderung 5.8 [annot] Notizobjekte S. 82<annotation> Grund: siehe daselbst	Inhalt

Tabelle (Forts.): Enthaltene Änderungen

Datum	Kapitel	Änderung	Bezug
Nr. 11, 04.08.96	Gesamt	Variantenbehandlung, Inhaltsmodell Software neu strukuriert Grund: Vervollständigung	Dokument
		Kapitel 4.6 Inhaltsmodell Software mit Unterkapiteln neu strukturiert. Grund: Qualitätssteigerung	Dokument
		Beschreibung der Elemente erweitert. Grund: Vervollständigung	Inhalt
		Beschreibung der Attribute neu aufgenommen. Grund: Vervollständigung	Inhalt
		Referenzen innerhalb der Software. Grund: Vervollständigung	Dokument
		Beschreibung von allgemeinen Strukturen Grund: Qualitätssteigerung	Dokument
Nr. 12, 03.07.96	Gesamt	aktuelle Near&Far Grafiken eingebaut Grund: Aktualisierung	Dokument
Nr. 13, 24.06.96	Gesamt	Erweiterung Beschreibung der Elemente. Grund: Vervollständigung.	Inhalt
Nr. 14, 24.06.96	Gesamt	Ebenen der Standardisierung eingebaut Grund: Erweiterung	Dokument
		Inhaltsmodell Software bereinigt. Grund: Qualitätssteigerung	Inhalt
		Weitere Änderungen aus Sitzung vom 3.6.96 Grund: Nachtragen	Inhalt
Nr. 15, 3.6.96	Gesamt	Schreibfehler korrigiert Grund: Qualitätssteigerung	Dokument
		Änderungen aus Sitzung vom 3.6.96 eingebaut. Grund: AG-Beschluß	Inhalt
Nr. 16, 15.5.96	Gesamt	Changes eingeführt Grund: Zur gezielten Weiterentwicklung	Inhalt
		Weitere Änderungen aus letzter Sitzung? Grund: noch nachtragen	Dokument

Tabelle (Forts.): Enthaltene Änderungen

Datum	Kapitel	Änderung	Bezug
Nr. 17, 23.4.1996	Gesamt	Umstellung auf SGML. Grund: Arbeitskreisbeschluß.	Dokument
		Einbau der Struktur in From von NE-AR&FAR. Grund: Arbeitskreisbeschluß.	Inhalt
		Erweiterung der Struktur. Grund: Abgleich mit ASAP u. Bei Bosch verwendeten Strukturen.	Inhalt
		Beschreibung der SGML-Elemente. Grund: Erweiterung	Inhalt
		<sw-params> wurden ins Datendictionary übernommen. Grund: Kenngrößen werden funktions-übergreifend referenziert.	Inhalt
		<sw-base-type> <in sw-variable> Grund: Der Basistyp ist variablen-spezifisch.	Inhalt
		In <sw-desc> <prms> entfernt. Grund:	Inhalt
		Richtung (direction) als Attribut von <sw-variable-ref unter <sw-function-variables> implementiert. Grund:	Inhalt