MSR-Documentation

MSR-Report

Concepts of MSRREP.DTD

MSR-MEDOC Arbeitsgruppe DTD, Roman Reimer, STZ XI-Works

06/09/2002 14:46:15 msrrep-sp-en.xml

Table of Contents

Introduction

Companies

MSR-MEDOC Arbeitsgruppe DTD [MSR-AG-DTD]

Name **Roles**	**Departement**	**Address**	**Contact**
Dipl.-Inform. H. Gen-genbach			
Dipl.-Math. M. Krause			
Dipl.-Inform. P. Rauleder			
Dipl.-Ing. B. Weichel			
Dipl.-Inform. J. Wieland			
Roman Reimer	STZ XI-Works		

Version Information

Document Part	**Editor**			
	Company	**Version**	**State**	**Remarks**
2002-02-07	Roman Reimer			
For details refer to nr. 1, Page 33	MSR-AG-DTD	1.3	RD	

1 Introduction

MSRREP.DTD is an SGML doctype definition designed in the MSR environment considering the following objectives:

- useable to produce documentation and reports in the MSR working group
- gathering experiences with the **<ncoi>** type of element in the *MSRDOC.DTD*
- performing revision and change planning
- probably useful to carry reports not coverable by the very content oriented *MSRDOC.DTD*
- *%prose;* data should be interchangeable between *MSRREP.DTD* and *MSRDOC.DTD* without any changes.

This document is not intended to provide a 100% detailled description of all aspects of *MSR-REP.DTD*. It covers the main concepts of *MSRREP.DTD* and gives hints to take best advantages of these concepts.

In respect of *%prose;* this document can be helpful for users of *MSRDOC.DTD* also.

1 Concepts

MSRREP.DTD is derived from *MSRDOC.DTD*. Only a few elements are added. Therefore it is possible to derive most of the processing also.

The DTD mainly consists of three parts:

<report-head>(see Topic 2.1 report-head p. 8) carries all the document metadata as well as project related information

<report-body> (see Topic 2.3 report-body p. 9) carries the main part of the document as well as the change management support (see Topic 2.4.1 change objects p. 10)

<report-rear> (see Topic 2.5 report rear p. 12) taking all appendices

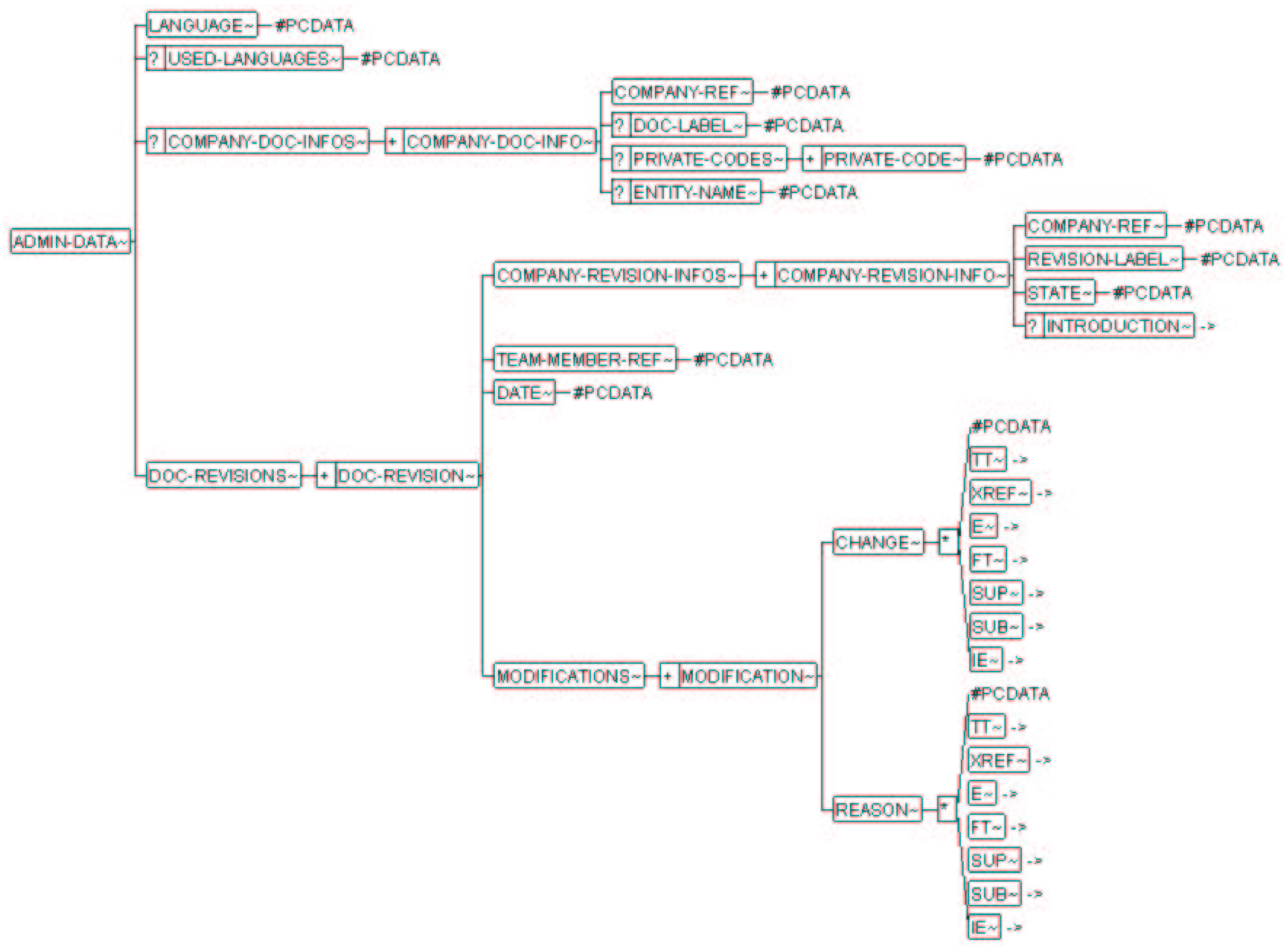

Figure 1: MSRREP.DTD and administrative data

In the top level element **<report-head>** there is also **<admin-data>** (see Topic 2.2 admin-data p. 9), providing means to describe the document versions, revisions etc. It is not treated as a document part since it is mainly metadata. This element is included also on a chapter level if the document is fragments into entities.

All three sections contain a generic structure which allows to define an unlimited chapter hierarchy (**<chapter>**). It is up to the user to take sure, that the number of chapter nesting does not exceed the capability of the *SGML processing systems*.

Cross-referencing is done in a non semantical way using **<xref>**. A subset of the *MSRDOC.DTD* id classes are supported in *MSRREP.DTD* also. These are mainly the organizational ones but not the technical ones of *MSRDOC.DTD*. *MSRREP.DTD* adds some id classes with the change management support. Some semantic references are also provided

MSRREP.DTD provides paragraph level markup as

paragraphs	an ordinary paragraph
tables	implemented as *CALS table*s
lists	provided as numbered, unnumbered and labelled lists. Each Item can mainly contain all paragraph level markup
figures	Allowing to include sizeable graphics
verbatim	for preformatted text
topic	which allows to insert bridge titles

There are some elements providing character level markup[1]. Most of them represent a certain semantics as it is the case in *MSRDOC.DTD*.

Elements taken from *MSRDOC.DTD* are not changed except **<xref>** which supports only the id classes in *MSRREP.DTD*. This leads to some structures which could be considered as overdesigned, mainly in the metadata where **<company>** still has all the project oriented elements of *MSRDOC.DTD*.

All elements have an attribute **[signature]** which is targeted to be filled by a *check routine*. It can be used to find out changes in the document.

1

2 User guide

When starting a new document, first the **<report-head>** (see Topic 2.1 report-head p. 8) has to be filled with all the meta information, interjectionally stuff etc. The actual document is entered in **<report-body>** which receives **<changes>** (see Topic 2.4.1 change objects p. 10) as the last chapter. This is the place to support the revision planning and change management. **<report-rear>** (Topic 2.5 report rear p. 12) receives all the appendices.

2.1 report-head

2.1.1 report-subject

<report-subject> allows to specify the subject of the document:

<overall-title>	specifies the overall environment the document belongs to. This could be the title of an overall project, an entire system, the overall title of a multipart book etc. If nothing of this matches, this element can be left out.
<main title>	specifies the major title of the document. This could be the title of a working group, the name of a subsystem, the title of a partial document etc.
<sub-title>	specifies the subtitle of the document. This could be the type of the document (e.g. **user manual**) etc.
<short-name>	specifies a short name under which the document is known in the project environment or the user community. This could be a mnemonic label.

2.1.2 companies

<companies> allows to enter all the information about the related companies, their team members etc. This element is taken from *MSRDOC.DTD* as is.

The persons involved in the document related project are listed as **<team-member>** within their **<company>**. Each **<team-member>** has a **<role>** in the process for example *project-leader expert secretary supervisor* etc.

If the document is mainly maintained by one person in a task force, the task force could be treated as an organization of its own behalf and entered as the only company. In this case, the native organization of the team members can be entered into **<department>** like the following example:

Usually, **<general-product-data>** is filled with **<na>**. If *MSRREP.DTD* is used for accompanying reports in a certain project context, **<general-product-data>** could be used to add project related information. For further details, see *[External Document: Concepts of the MSR-DOC.DTD / URL: / Relevant Position:]*

2.1.3 abstract

<abstract> can be used to give a short overview of the document. The contents of this element is bound to be transmitted to a document management system of a document catalog. So it should not have more than a few hundred words. In order to provide a unified content model, **<abstract>** can get all the paragraph level elements. But these means should be used with great care.

2.1.4 chapter in report-head

<chapter> in **<report-head>** can be used to enter excessive forewords, meta information etc. In order to provide a unified content model, there is no restriction here. Nevertheless, it is recommended not to enter the entire document at this place. For more details about **<chapter>** see Topic 3.1.1 Chapter p. 13.

2.2 admin-data

<admin-data> is used to markup document adminstrative data (see Figure 1 MSRREP.DTD and administrative data p. 6) such as revisions etc. The operating model is

- The document (or the fragment) is handled in all companies participating in the project.
- The data management in the various companies is different. For that reason, echa participant can enter information about their document management facilities in **<company-doc-info>**:

<doc-label>	this is the label under which the document is managed in the company denoted by **<company-ref>**
<private-code>	allows to transport company specific information in a private notation. This is the place, where for example *PDMS* (*Product Data Management System* s) can place pointers and document ids required to resynchronize after a document exchange.
<entity-name>	It might be the case that each participating company uses a different fragmentation strategy. In order to support this, **<entity-name>** can receive information useable by a *split utility* which creates the desired fragments out of the entire document.

- If a new release of the document or the fragment is given, each participating site may use a specific scheme for revision numbers. For that reason, each **<doc-revision>** can receive **<company-revision-info>** which holds the participiant specific information for the actual document revision.

  It is up to a *semantical check utility* to keep sure that there is only one entry per company.
- nevertheless, the actual revision is initiated by one individual denoted by **<team-member-ref>** at one certain point of time denoted by **<date>**.
- Finally the modifications made in that revision are stored in **<modifcations>** where the actual **<change>** as well as the **<reason>** for that change is notified. If possible, the change can be located by **<xref>**.
- For each **<modifcation>** the attribute **[type]** determines, if the change is made to the document only (*doc-related*) or to the subject of the document (*part-related*).

2.3 report-body

<report-body> receives an unlimited amount of **<chapter>**s or **<chg-chapter>**s, which receive the actual information.

2.4 Change management support

Change manangement support is provided using **<changes>** which is placed into **<chg-chapter>** . This allows to maintain multiple change mananagements within one document.

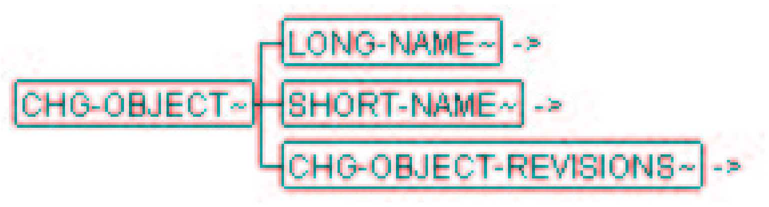

Figure 2: Change management support

This feature is based on the following model:

- Within one document, there are several **<chg-objects>** to be considered in the actual project domain. These objects can be any deliverable in the process, either documents, products or even libraries.
- These objects exist in multiple revisions (**<chg-object-revisions>**)within the process.
- Requested changes can be captured and attached to these **<chg-object-revision>**s.
- The treatment of the requested changes can be managed using element in **<chg-treatment>** .

2.4.1 change objects

Within one document, there are several **<chg-objects>** to be considered in the actual project domain. These objects can be any deliverable in the process, either documents, products or even libraries.

The development of **<chg-object>** takes place by creating new revisions. These **<chg-object-revision>**s are characteriezed by

<long-name>	receiving the work title of the revision (e.g. ”Summer 95 release”)
<short-name>	holding the revision number
<date>	holding the release date

The **<chg-object-revision>**s are referred to from within **<chg-request>**. This allows to generate **<chg-object-revision>** related tables and lists.

It is recommended not only to capture existing revisions but also future ones. This allows to refer them in order to receive complente release notes etc.

2.4.2 chg request

In the project there are any requested changes **<chg-requests>**. For each **<chg-request>** there is one or more related objects resp. object revisions (**<chg-object-revision-ref>**). The referred objects are the ones to be improved resp. causing the problem to be solved by the required change[2]. Thus no **<chg-request>** can be there without respect to a **<chg-object-revision>**. If necessary a fake **<chg-object-revision>** should be introduced.

For each **<chg-request>** the following can be entered:

<long-name>	A working title for the change request. This can be used in verbal communication as well as anntoatin in overview reports
<short-name>	A label for the change request. This can be used to refer to link the document to *Project Management Systems*

2

<chg-keywords>	it is possible to enter blank separated keywords here. These can be used for retrieval purposes as well as in overview reports. It could also be used to enter notes for the project manager like assignments to indiviuals etc.
<chg-proposed-by>	can receive the names or initials of the individual(s) who proposed the change request.
<chg-priority>	Allows to qualify the priority of the requested changes. The priorization model is up to the author. The *SGML processing system* can use this element to order the requested changes according to their priority. Therfore the highest priority should be of low lexical order. It can either be a simple number (high priorities get low values) determining the priority. Another model is a letter/digit combination. The letter characterizes the importance, the digit the urgency (e.g. *A 2* says *very important* and *medium urgent*)
<chg-class>	allows to classify the requested change. This element should receive something like *enhancement error*
<chg-related-objects>	This element receives links to the related object revidions. The semantic of this link is, that the referred object revision is causing the problem (e.g. with a bug report) or must be enhanced (in case of an enhancement request). All objects concerned by the change request should be mentioned.
<chg-subject>	This is the place to describe all the aspects of the change request. All paragraph level elements can be used.
<chg-reason>	This element receives the justification of the change request. In this element all problems should be mentioned which shall be solved by the requested change.

2.4.3 chg treatment

The treatment of the reuqested changes can be documented in **<chg-treatment>**.

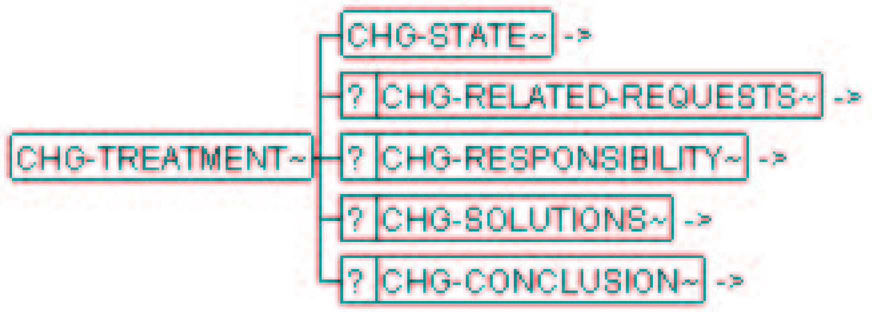

Figure 3: Change treatment

The treatment of one change request is expected in following steps:

- The request goes through a life cycle which is documented in **<chg-state>**. This element receives some notes about the next actions as well as a formal state using the attribute **[state]** with the selection of *open*, *in-progress*, *passed*, *rejected* or *done*:

open	The request is captured, but no further action has been taken on it.
in-progress	The request is accepted and is in implementation.

passed	The request is accepted, but no implementation started up to now.
rejected	The request is rejected. It will not be implemented.
done	The request is implemented. It is highly recommended, that the **<chg-release-notes>** is completely filled now for each changed object.

- If there are change requests related to the actual one, this can be documented using **<chg-related-requests>** and **<chg-request-ref>**. It is possible to express if the related change is a prerequisite to the actual one. Further informatione about the relationship can be entered as text in **<chg-request-ref>**.
- For the **<chg-request>** the responsibility is specified in (**<chg-responsibility>**). The assignment is done verbally using **<desc>** or as a set of **<team-member-ref>**s in **<chg-responsible>**. The latter style is treated formally and can be used to generate task lists for team members.
- The possible solutions to the problem can be documented in **<chg-solution>** where each option can be specifed in **<chg-solution-spec>** and discussed using **<chg-solution-pro>** **<chg-chg-solution-con>**.

  Also the estimated effort can be assinged using **<chg-effort>**. This is treated as a single value. It is up to the user to decide if the effort is measured in hours, days, weeks or months. Since a *SGML processing system* can sumarize the effort, it is recommended to use the same units.
- Finally there should be a conclusion, documented in **<chg-conclusion>** with the following aspects:

<chg-solution-spec>	This is the place to document internals of the chosen solution as well as technical consequences etc.
<chg-implementations>	is a set **<chg-implementation>**s, each consisting of a pointer (**<chg-object-revision-ref>**) to **<chg-object-revision>** to show, in which ones of the revisions the solution is or will be implemented. **<chg-implementation>** also receives the **<chg-release-notes>** which receives the information for the release notes regarding the change-object denoted by **<chg-object-revision-ref>**. This information should be extracted into an entity holding a **<chapter>** by an *SGML processing system*. If the author wants to specify the detailled location of the implementation (e.g. the changed files) he can do this by introducing **<topic>**s. When the release notes are collected, the *SGML processing system* could assort the topics based on their **<long-names>** [3]. **<chg-implementation>** can be ther more than once if the solution touches more than one **<chg-object>** [4].

2.5 report rear

<report-rear> receives a number of **<chapter>**. *SGML formatting systems* can add additional chapters like crossreference indexes or reports generated for **<tt>**.

3

4

3 Basic Structures of the MSR Application Profile

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

3.1 Not Content Orientated Information (ncoi)

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

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

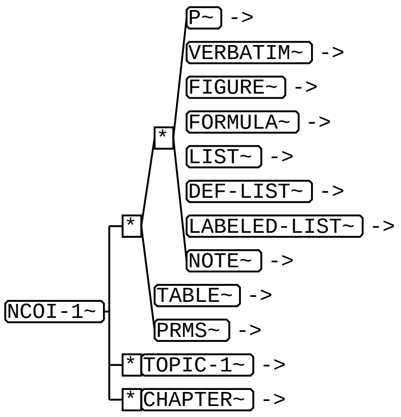

Figure 4: Structure of <ncoi-1>

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

The components of ncoi [5] are interchangeable between all MSR DTDs[6] without any changes.

3.1.1 Chapter

<chapter> is a sequence of paragraph level elements mixed with **<chapter>**. **<chapter>**s can be nested as deeply as required. It is up to the author to make sure, that the nesting of the chapters can be handled by the processing system[7].

5

6

7

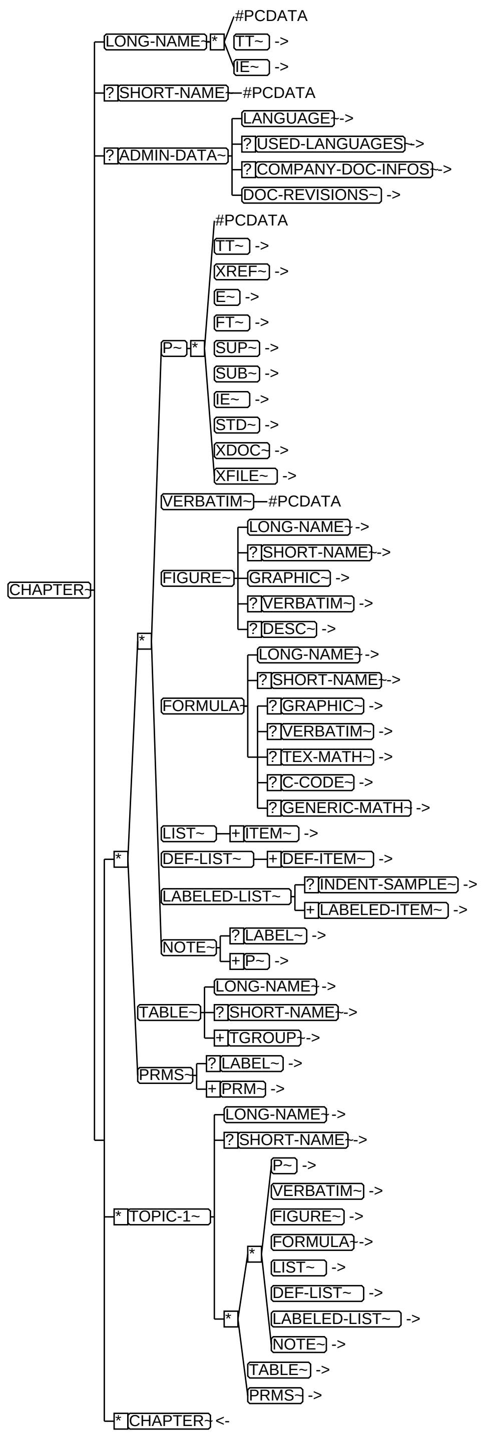

Figure 5: chapter content model

One advantage of using **<chapter>** for all levels[8] is the option to move a chapter using *cut & paste* to any place in the document at any level.

[8]

3.1.2 Topic

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

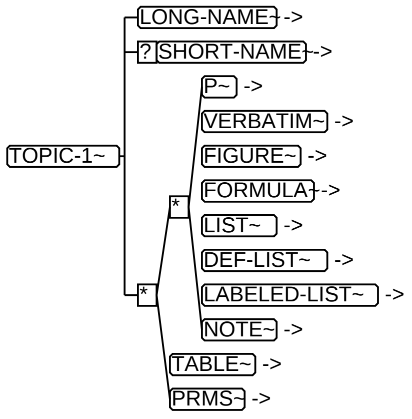

Figure 6: Structure of <topic-1>

3.1.3 Paragraph Level Elements

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

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

<p>	Paragraph
<verbatim>	Preformatted text which is usually set in monospaced font. Tabs, line spaces and carrige returns are considered. Use **<verbatim>** to print program listings etc. It can even be used to show simple diagrams.
<figure>	See chapter Topic 3.1.3.2 Figure p. 17.
<formula>	See chapter Company 3.1.3.3 Formula p. 18.
<list>	A ordered or unordered list of items. For an unordered set of items, use **<list type=”unnumbered”>**. For a ordered list of items use **<list type=”numbered”>** [9].
<def-list>	Use **<def-list>** to create definition lists which might be collected into an overall definition list or a glossary. In this case **<labeled-list>** might lead to the same rendition but has no information about the fact that terms are defined[10].
<labeled-list>	Use**<labeled-list>** to create explanations or even bridge titles for very short topics instead of bulleted lists with emphasized initial words. See also Topic 3.1.3.1 Labeled List p. 16 Use **<labeled-list>** instead of two column tables if the first column cells almost contain one word.

9

10

<note> See chapter Topic 3.1.3.4 Note p. 18

3.1.3.1 Labeled List

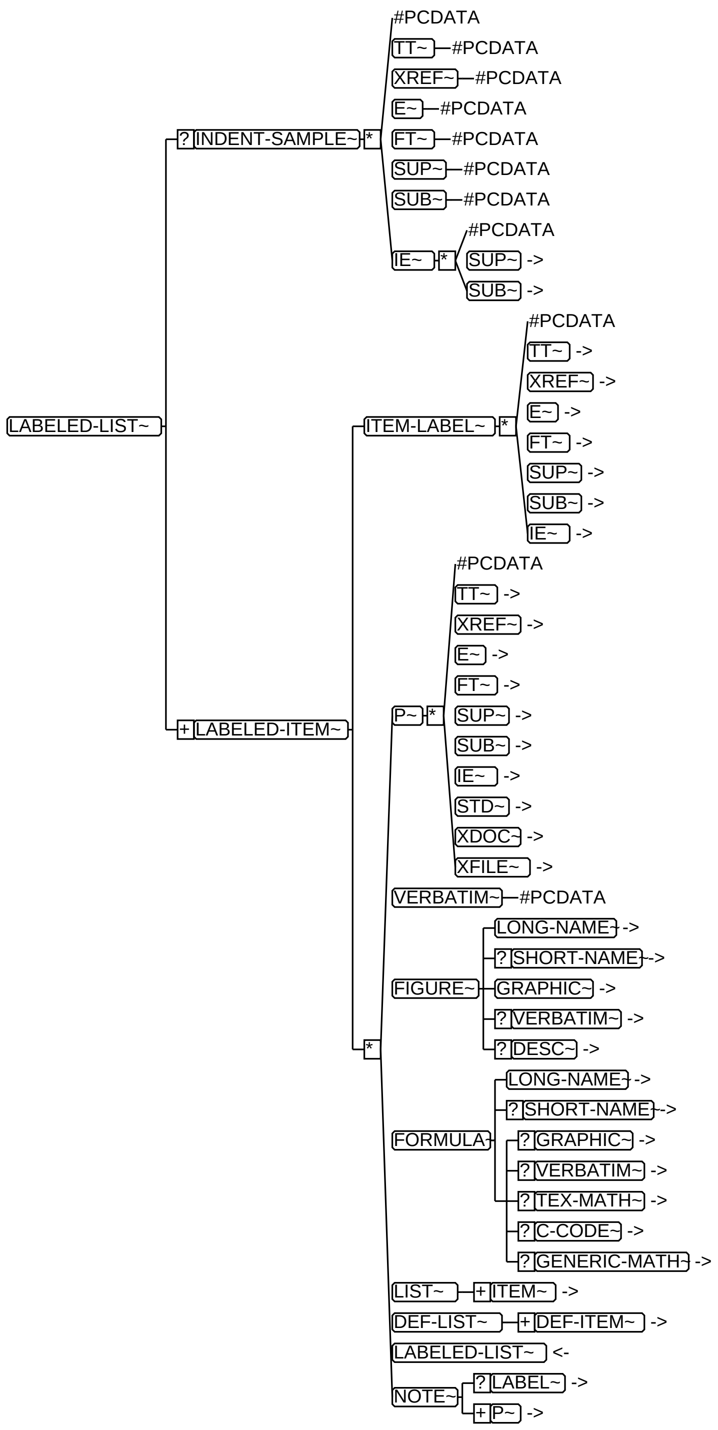

Figure 7: Structure of <labeled list>

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

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

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

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

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

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

3.1.3.2 Figure

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

1. as a real **<graphic>**
2. as an ASCII graphic (**<verbatim>**)
3. as a pure textual description (**<desc>**) of the graphic [11]

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

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

[category]	Denotes the category of the graphic. This information can be used to generate more specific list of figures
[filename]	Denotes the system filename where the *rendition system* can find the graphic. This is not necessarily the final format. It is up to the *rendition system* to locate the graphic in the company specific environment, to change the file extension to get the appropriate graphic representation. The type of this attribute can be turned from *SDATA* to*ENTITY* in the DTD file in order to allow *SGML tool*s access to the file using its *entity manager*. In this case, the entity name should be chosen in the style of a filename (e.g. *crpctmt.wmf*)[12].
[fit]	0 figure is placed in original size. If it does not fit on the page or the available space, it is scaled down. 1 the figure is scaled up or down to fit the page as possible. This value will be ignored if **[width]** or **[height]** is specified in addition. 2 the figure is rotated counterclockwise by 90° if it is landscape and is wider than the actual text area. It is scaled down to the page size if it does not fit otherwise. This value will be ignored if **[width]** or **[height]** is specified in addition.

11

12

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

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

[height]	If this attribute has a value, the figure will be scaled to the defined height which is a real value with dimensions (e.g. ”10cm”, ”150mm”, ”12.5in”). If also **[width]**is specified the figure will be distorted. This value always specifies the width of the ”figure box” on the page after possible scaling/rotating.
[notation]	This attribute specifies the format of the graphic file if used by an *SGML Application* supporting notations.
[scale]	If this attribute receives a value, the figure will be scaled by the given factor which must be a signed real number. Numbers greater 1 increase the size of the figure, values less than 1 make the figure smaller. For example with *scale=”0.5”* the a figure of the size 10x10 cm will appear as 5*5cm.
[width]	If this attribute has a value, the figure will be scaled to the defined width which is a real value with dimensions (e.g. ”10cm”, ”150mm”, ”12.5in”). If also **[height]**is specified the figure will be distorted. This value always specifies the width of the ”figure box” on the page after possible scaling/rotating.

The scaling attribute precedence is:

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

3.1.3.3 Formula

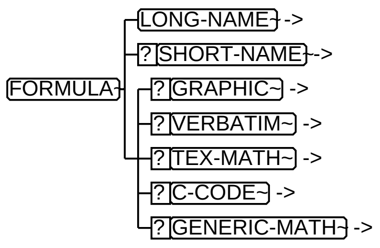

Figure 8: Structure of <figure>

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

<graphic>	A formula prerendered as a figure.
<verbatim>	A simple ASCII formula.
<tex-math>	A *TeX* math formula which can be processed by a *TeX* or *LaTeX* processor.
<c-code>	A formula which is defined as c-code.
<generic-math>	This element is intended for the definition of semantic math descriptions which can be processed by math processors. Actually there is no recommentation for the language of the formula specification or usage of a special rendering system.

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

3.1.3.4 Note

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

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

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

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

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

3.1.4 Character Level Elements

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

3.1.4.1 Rendition Oriented Character Level Elements

The rendition oriented character level elements are:

<e>	Emphasizes the text. The attribute **[type]**determines the rendition style.
<sub>	Subscript - places the contents with smaller font below the base line.
<sup>	Superscript - places the contents with smaller font above the base line.

3.1.4.2 Semantically Oriented Character Level Elements

Table 1: semantically oriented character level elements

Element	use for	example
<tt>	Use for any technical term. The type of that term is determined by the attribute **[type]** [13]. This element could be treated as a back-door to markup information which is not totally semantic. The *SGML processing system* can generate list of technical terms which makes it easier to find misspellings and other errors.	This is an SGML tag **<t-t type=sgmltag>** we can collect all **<tt>**s

13

06/09/2002 14:46:15 msrrep-sp-en.xml

Table 1 (Cont.): semantically oriented character level elements

Element	use for	example
<xref>	Used to create links in the document. The role of the target is determined by the attribute **[id-class]** receiving the value of the target's fixed attribute **[f-id-class]**. The attributes of **<xref>** should be maintained by the *authoring system*.	
<xdoc>	Used to refer to an external document which usually is not available electronically. **<xdoc>** receives a set of elements characterizing the external document	Details to architectural forms can be found in *[External Document: / URL: / Relevant Position:]* .
<ft>	Is used to create footnotes	Footnotes seem to be small and unimportant[14].
<ie>	creates index entries	It is not necessary to put SGML tags into the Index, since the processing for *MSRREP.DTD* recommends to create a list of SGML tags automatically.
<xfile>	Is used to create pointers to external files which are not to be processed by the native *SGML processing system*. The contents of **<xfile>** can be used to connect to appropriate systems in later steps of the processing chain.	The schematic is found in *[External FILE: MOTRONIC wiring diagram / URL: motronic.asc]*
<std>	Is used to refer to a standard.	SGML is defined in *[/ Standard: Information Processing - Text and Office Information Systems / Subtitle: Standard Generalized Markup Language / State: standard / Date: 1986 / URL: / Relevant Position: entire document]*

Table 2: usage of technical terms

type	use for	example
<tt type=sgmltag>	Used to describe SGML tags including attributes	To describe SGML tags use **<tt type=sgmltag>**.
<tt type=sgml-attribute>	Used to describe SGML attributes outside of tags	The sgmltag is denoted by the attribute **[type]**
<tt type=tool>	Used to mention tools used for example in a process. This can be software, as well as mechanical tools. The tool should be specified by its nature not by the specific product name.	SGML files are processed using an *SGML processing system*.

14

Table 2 (Cont.): usage of technical terms

type	use for	example
<tt type=product>	Used to mention specific products.	This document is processed using *MetaMorphosis*.
<tt type=variable>	Used to mention a variable informally. This is used to control the rendition as well as for generating variable lists. This is mainly for informal reports[15]. It is also possible to use this to mention a variable in the ECU software if no **<sw-data-dictionary>** is part of the document. In a later process step, this can be turned over to a formal **<xref>**	The initialization is controlled by the environment variable *MMRC*. The initial advanced angle is calculated based on *N*and*TL*.
<tt type=state>	Used to mention a state for example of a process.	The documents must at least be *revised* if they are submitted to the customer.
<tt type=prm>	Used to mention a state for example of a process. It is also possible to use this to mention a calibration parameter in the ECU software if no **<sw-data-dictionary>** is part of the document. In a later process step, this can be turned over to a formal **<xref>**	The initial advanced angle is calculated using a lookup table *KFZW*.
<tt type=material>	Used to mention material.	Furniture is usually made of *wood* and *plastic*
<tt type=control-element>	Used to mention control elements of tools like push-buttons, menu items, switches etc. as well as keyboard keys.	To finish the dialog push the *OK* button.
<tt type=code>	Used to markup program in line code sequences	*MetaMorphosis* is invoked with *mm crp.sgm*
<tt type=organisation>	Used to markup the name of an organization.	SGML is standardized by *ISO*
<tt type=other>	Used to mention a special term which does not fit to the other types. This is a back-door for the definition of user defined types. They have to be specified within the attribute **[user-defined-type]**. A formatter uses this user defined type only if **[type=other]**.	This is a *thing* not covered by **<tt>**.

15

Table 3: sub-elements for xdoc and xfile

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

3.1.5 Table

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

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

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

3.1.6 Parameter tables

User Definable Parameters

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

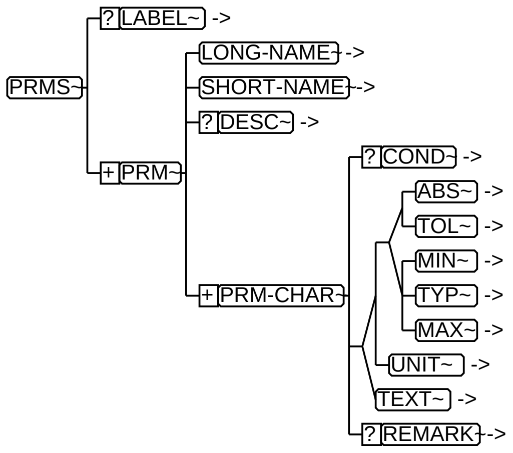

Figure 9: Structure of prms

* parameter * long-name

* short-name

* description

[17] [18] * condition

((* absolute value and tolerance[16] or

* minimum, typical, maximum value[17])

* unit) or

* text[18]

The following representation example can be drawn from this structure:

<short-name> UB

Table 4: Parameter structure

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

16

17

18

06/09/2002 14:46:15 msrrep-sp-en.xml

- Defined Parameters

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

3.2 Predefined Document Structure

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

This situation was also taken into account in the DTD through the elements ”**<na>**” (not applicable), ”**<tbd>**” (to be defined) and ”**<tbr>**” (to be resolved) as shown in Figure 10 Principles of information acquisition p. 24. This is a mechanism is located at each element on chapter level and works like a check list. A user has to make a statement for each topic.

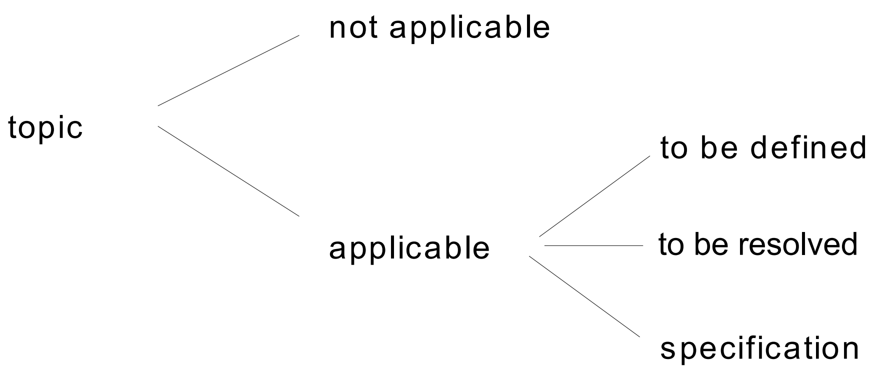

Figure 10: Principles of information acquisition

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

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

3.3 Project Data

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

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

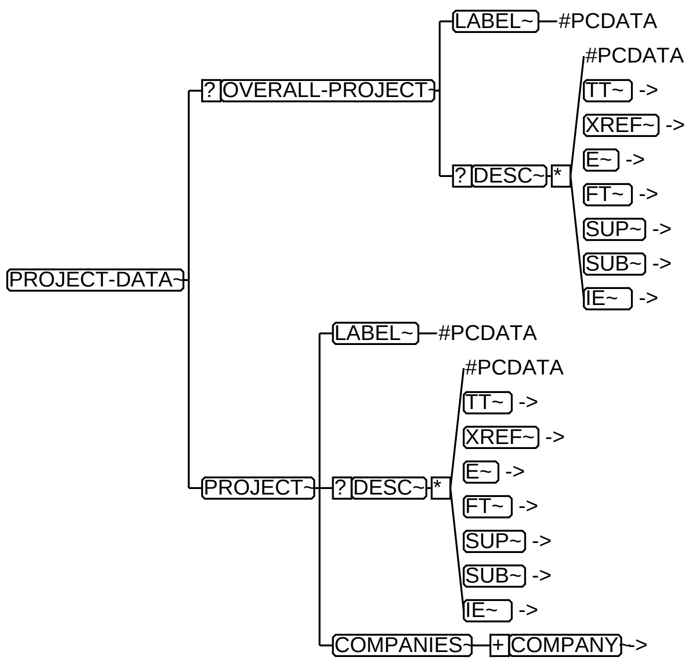

Figure 11: Structure of <project-data>

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

System overview **<system-overview>**	This chapter can be used to define information about a global system, e.g. a certain car model.
Order justification **<reason-order>**	This may be used to specify information about the reasons for the order of the described component resp. for making the specification of such a component.
Objectives **<objectives>**	This chapter can be used to specify information about the project objectives. E.g. ”Development and system release of the engine-managment-system for the model NEW-BEETLE”
Models **<sample-spec>**	This structure is used to define development samples like A-,B-,C-,D-sample. These samples represent the results of the different development phases.
Variant specification **<variant-spec>**	This section is used to specify all variant definitions and their corresponding variant characteristics. See also Topic 3.5 Variant Concept p. 27.

Limits to other projects **<demarcation-other-projects>**	This chapter is used to describe the demarcation to other projects.
Parallel developments **<parallel-design>**	This can be used to give an overview of the work in parallel projects.
Integration capability **<integration-capability>**	In this chapter requirements on the capabilities of integration in other systems can be described.
Acceptance conditions**<acceptance-cond>**	This chapter is used to define the general conditions for the acceptance of the described components.
Schedule and plans **<project-schedule>**	This chapter is used to define the project-schedule, e.g. project milestones, dates, time limits etc.
Purchasing conditions **<purchasing-cond>**	This is used to define purchasing conditions like amount of devices per year, delivery times, storage quantities, etc. .
Protocols, minutes of meeting **<protocols>**	This is the place where project minutes and other arrangements can be mentioned.
Handed over documents and data**<dir-hand-over-doc-data>**	This is the directory of the handed over documents and data.
Additional project specifications**<add-spec>**	Any kind of additional project description which can't be described with the chapters mentioned above.

3.4 Administrative Data

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

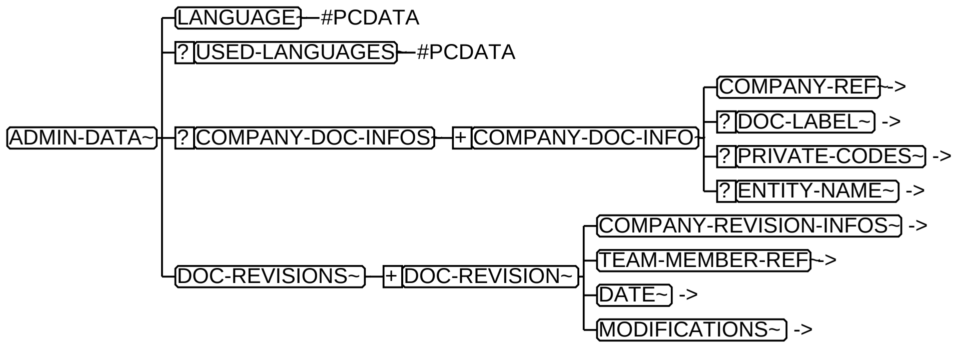

Figure 12: Support of DTD fragmentation through administrative data

The operating model is

- The document respectively the fragment is written in a certain language which can be defined in the element **<language>**. This element can be used to control a SGML system, e.g. to set the correct prefix strings for elements.
- The DTD can be configured for the multilingual operation. In this case **<language>** contains the language of the origin document. All languages used in a document have to be defined within **<used-languages>**, that is each language is defined with a **<l-10>**-element which contains the full language name and in the Attribute **[l]** the short language name (see Topic 3.6 Multilinguality p. 28).
- The document (or the fragment) is handled in all companies participating in the project.
- The data management in the various companies is different. For that reason, each participant can enter information about their document management facilities in **<company-doc-info>**:

<doc-label>	this is the label under which the document is managed in the company denoted by **<company-ref>**
<private-code>	allows to transport company specific information in a private notation. This is the place, where for example *PDMS* (*Product Data Management System* s) can place pointers and document ids required to resynchronize after a document exchange.
<entity-name>	It might be the case that each participating company uses a different fragmentation strategy. In order to support this, **<entity-name>** can receive information useable by a *split utility* which creates the desired fragments out of the entire document.

- If a new release of the document or the fragment is given, each participating site may use a specific scheme for revision numbers. For that reason, each **<doc-revision>** can receive **<company-revision-info>** which holds the participant specific information for the actual document revision.

  It is up to a *semantical check utility* to keep sure that there is only one entry per company.
- nevertheless, the actual revision is initiated by one individual denoted by **<team-member-ref>** at one certain point of time denoted by **<date>**.
- Finally the modifications made in that revision are stored in **<modifcations>** where the actual **<change>** as well as the **<reason>** for that change is notified. If possible, the change can be located by **<xref>**.
- For each **<modifcation>** the attribute **[type]** determines, if the change is made to the document only (*doc-related*) or to the subject of the document (*content-related*).

3.5 Variant Concept

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

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

Variant Characteristic	Characteristics that lead to a new variant e.g. engine, product line, country, etc. Characteristics are defined in **<variant-char>**. The characteristics have to be subdivided in three classes. These are: • characteristics which lead to a new subject number(<variant-char **[type="new-part-number"]**>). For this only the existence of such a characteristic is enough to establish a new subject number for this variant!

- characteristics which don’t lead to a new subject number (<variant-char **[type=”no-new-part-number”]**>).
- characteristics which lead to a new subject number according to shaping.

Variant Definition:	Definition of several variants with their variant characteristics for a part type.
Variant:	A variant of a part type is defined through the values of it’s variant characteristics.
Variant Coding:	Allocation of all variant definitions to their corresponding subject- and drawing- numbers and the respective development versions.

3.6 Multilinguality

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

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

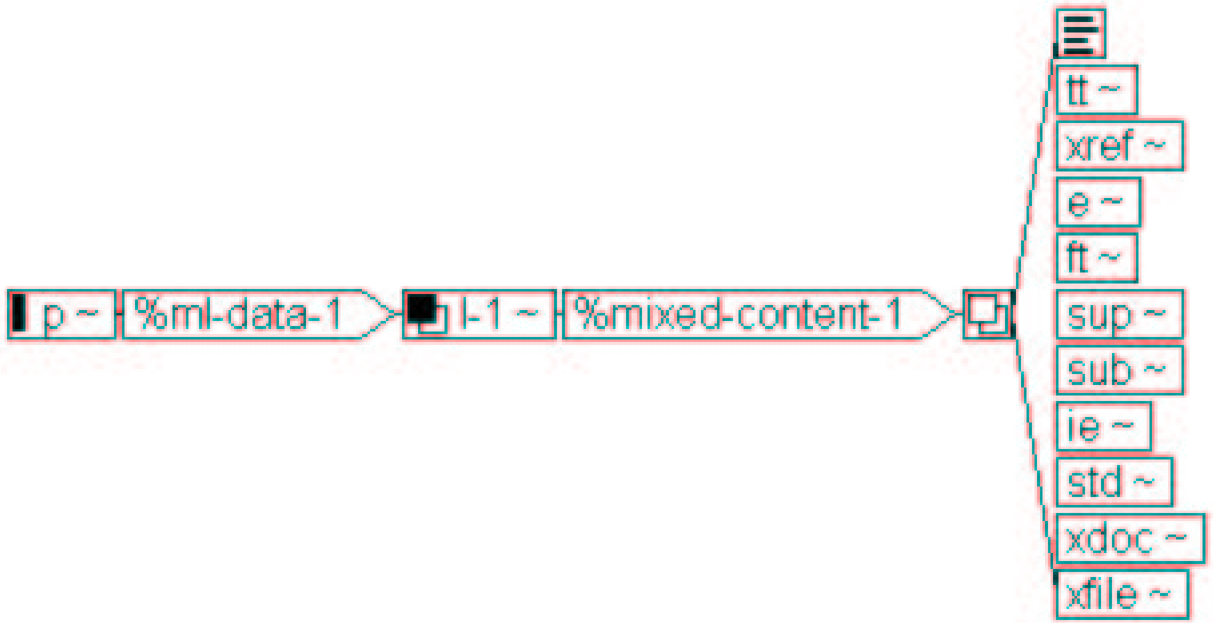

Figure 13: Multilingual Paragraph

App. A Processing hints

MSRREP.DTD is not very content oriented. Nevertheless some aspects of semantic processing shall be mentioned here.

App. A.1 technical terms

<tt> can be used to generate specific lists or indexes, even if all types are rendered in the same manner.

App. A.2 reference checking

all references should be checked

App. A.3 changes

The strongest semantics is there in **<changes>**. Therefore the following processing is recommended:

- Table of object revisions (**<chg-object-revisions>**) sorted by object revision with **<short-name>**, **<long-name>**, revision, release date (see Table 5 object revisions p. 30)
- Tables of requested changes sorted by **<chg-state>** (*open*, *in-progress*, *passed*, *rejected*) **<chg-priority>**, related objects (**<short-name>**). This will actally be one table per **<chg-state>**. Each entry should refer to the change itself. Thus these tables can serve as a directory of change requests (see examples in Table 6 open changes p. 30 resp. Table 7 in-progress changes p. 30 etc.)
- Table of requested changes sorted by **<chg-state>** (*open*, *in-progress*, *passed*, *rejected*) **<chg-priority>**, related objects (**<short-name>**). This will actally be one table per **<chg-object>**. Each entry should refer to the change itself. Thus these tables can serve as a directory of change requests.
- Table of planned revisions with summarized effort
- Release notes either one file per revision or only for one specific revision which is specified as a runtime argument. The release notes should be generated as fragment of a *MSRREP.DTD* instance, where a chapter is generated for each **<chg-request>**. It is recommended to include the release notes not directly into **<report-body>**, because **<chapter>** on level 1 usually starts a new page.
- If the author wants to specify the detailled location of the implementation (e.g. the changed files) he can do this by introducing **<topic>**s. When the release notes are collected, the *SGML processing system* could assort the topics[19].

The set of tables to produce should be controlled either by a *processing instruction* **<?chg 1246>** or by a *runtime argument*.

It is also possible to export the changes into other tools like *Project Management Systems* or *Spreadsheets*.

19

App. A.3.1 Sample Tables from changes

Table 5: object revisions

object	rev	title	effort	release date
msrrep.dtd	DTD for genereic reports			
	0.15	initial revision	5	April 96
	0.16	improved changes	7	july 96
msrdoc.dtd	DTD for MSR engineering doucumentation			
	0.15	performing major test	7	dec 95
	0.16	releas candidate	4	sep96

Table 6: open changes

change	priority	title	related object	revision	repsonsi-ble	page

Table 7: in-progress changes

change	priority	title	related object	revision	responsi-ble	page

Table 8: changes to msrrep.dtd

priority	change	title	state	revision	responsi-ble	page

Table 9: estimated efforts

priority	change	title	related object	effort	page
A 1	chg-imp	improve chg-treatment	msrrep.dtd	3	4
A 2	chg-tt	Introduce tt	msrrep.dtd msrdoc.dtd	1	9
Summary of A Changes				4	

App. B Zusätzliche Anmerkungen zum Dokumentstand 1.0 am 13.7.98

Status dieses Dokument: cd

Documentadministration

Table : team members

Name	Company	
Dipl.-Inform. H. Gengenbach	MSR-MEDOC Arbeitsgruppe DTD	
Dipl.-Math. M. Krause	MSR-MEDOC Arbeitsgruppe DTD	
Dipl.-Inform. P. Rauleder	MSR-MEDOC Arbeitsgruppe DTD	
Dipl.-Ing. B. Weichel	MSR-MEDOC Arbeitsgruppe DTD	
Dipl.-Inform. J. Wieland	MSR-MEDOC Arbeitsgruppe DTD	
Roman Reimer	MSR-MEDOC Arbeitsgruppe DTD	Department: STZ XI-Works

Table : version overview

Date	Publisher
2002-02-07	Roman Reimer
13.7.98	Dipl.-Ing. B. Weichel
10.7.96	Dipl.-Ing. B. Weichel
24.6.96	Dipl.-Ing. B. Weichel
21.6.96	Dipl.-Ing. B. Weichel

Table : modifications

Change	Related to
Create index, technical terms and reference. Convert to MSRREP V210 XML. Reason:	Content
Change to new document strategy Reason:	Content
chg-responsible can be specified formally as well as informally Reason: This maintains the flexibility of the informal assignment, and provides the option to link to project managment systems.	Content
Document	changes as alternate content model for chapter Topic App. A.3 changes p. 29 Reason: This allows the author to decide himself, wehre to put changes. It furhteron allows to run multiple change managements within one project.

Table (Cont.): modifications

Change	Related to
Content	Content
Content	first edition according to session from 19.6.96 Reason: there was no introduction to MSRREP.DTD esp. changes

Table : modifications included

Date	Chapter	Change	Related to
Nr. 1, 2002-02-07	Gesamt	Create index, technical terms and reference. Convert to MSRREP V210 XML. Reason:	Content
Nr. 2, 13.7.98	Gesamt	Change to new document strategy Reason:	Content
Nr. 3, 10.7.96	Gesamt	chg-responsible can be specified formally as well as informally Reason: This maintains the flexibility of the informal assignment, and provides the option to link to project managment systems.	Content
		added examples for chages-reports Topic App. A.3.1 Sample Tables from changes p. 29 Reason:	Document
Nr. 4, 24.6.96	Gesamt	changes as alternate content model for chapter Topic App. A.3 changes p. 29 Reason: This allows the author to decide himself, wehre to put changes. It furhteron allows to run multiple change managements within one project.	Content
		chg-responsible now is a set of teammember-ref Topic 2.4.3 chg treatment p. 11 Reason: make assignments formally correct	Content
		redesign implementation and placement of chg-release-notes Topic 2.4.3 chg treatment p. 11 Reason: to allow design object specific release note processing	Content

Table (Cont.): modifications included

Date	Chapter	Change	Related to
Nr. 5, 21.6.96	Gesamt	first edition according to session from 19.6.96 Reason: there was no introduction to M-SRREP.DTD esp. changes	Content