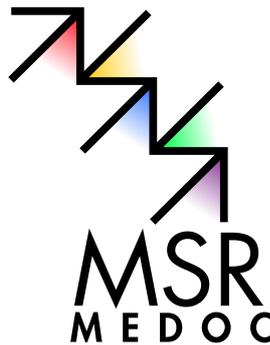




A Generic Algorithm for Merging SGML/XML Instances



MSR-MEDOC, Gerald Manger, BMW Group

Table of Contents

	Table of Contents	2
	List of Figures	3
	Introduction	4
1	Presentation	5
	Documentadministration	36

List of Figures

Figure 1	Folie 1	5
Figure 2	Folie 2	6
Figure 3	Folie 3	7
Figure 4	Folie 4	8
Figure 5	Folie 5	9
Figure 6	Folie 6	10
Figure 7	Folie 7	11
Figure 8	Folie 8	12
Figure 9	Folie 9	13
Figure 10	Folie 10	14
Figure 11	Folie 11	15
Figure 12	Folie 12	16
Figure 13	Folie 13	17
Figure 14	Folie 14	18
Figure 15	Folie 15	19
Figure 16	Folie 16	20
Figure 17	Folie 17	21
Figure 18	Folie 18	22
Figure 19	Folie 19	23
Figure 20	Folie 20	24
Figure 21	Folie 21	25
Figure 22	Folie 22	26
Figure 23	Folie 23	27
Figure 24	Folie 24	28
Figure 25	Folie 25	29
Figure 26	Folie 26	30
Figure 27	Folie 27	31
Figure 28	Folie 28	32
Figure 29	Folie 29	33
Figure 30	Folie 30	34
Figure 31	Folie 31	35



Introduction

Companies

MSR-MEDOC [MEDOC]

Name Roles	Departement	Address	Contact
Gerald Manger, BMW Group			

Version Information

Document Part	Editor			
	Company	Version	State	Remarks
1 RD 2002-11-03 For details refer to nr. 1, Page	Gerald Manger, BMW Group			
	MEDOC			

	A Generic Algorithm for Merging SGML/XML Instances XML Europe 2001 Chapter: Presentation	Page: 5/37 Date: 2002-11-03 State: RD
---	--	---

1 Presentation

A Generic Algorithm for Merging SGML/XML Instances.

Design and Implementation for the Use Case:
Function Documentation of Control Unit Software
in Automotive Industry.

Gerald Manger, Development Engineer.
25.05.2001, Berlin.

BMW Group



EU_Folie1.PNG

Figure 1: Folie 1



Overview



- **V-Model ECU-Software Development Process**
- **Data Model MSRSW.DTD for ECU-Software Documentation**
- **Function Documentation Process eDoc**
- **Intuitive demands for tree based instance merging**
- **Algorithm σ**
- **Examples**
- **Demonstration: Implementation SIGMA of Algorithm σ**

Figure 2: Folie 2

ECU Software Process

V-Model (simplified)

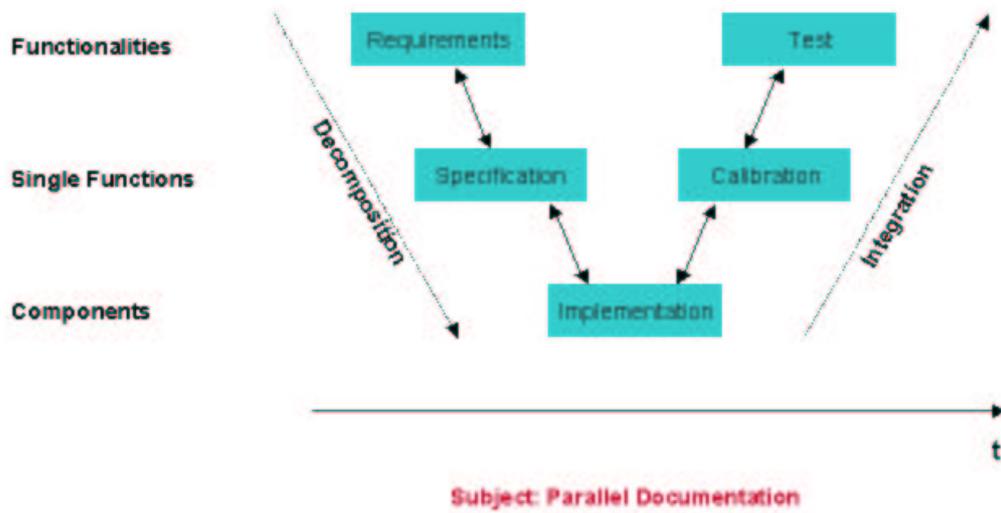


Figure 3: Folie 3

ECU-Software Documentation

Data Model MSRSW.DTD

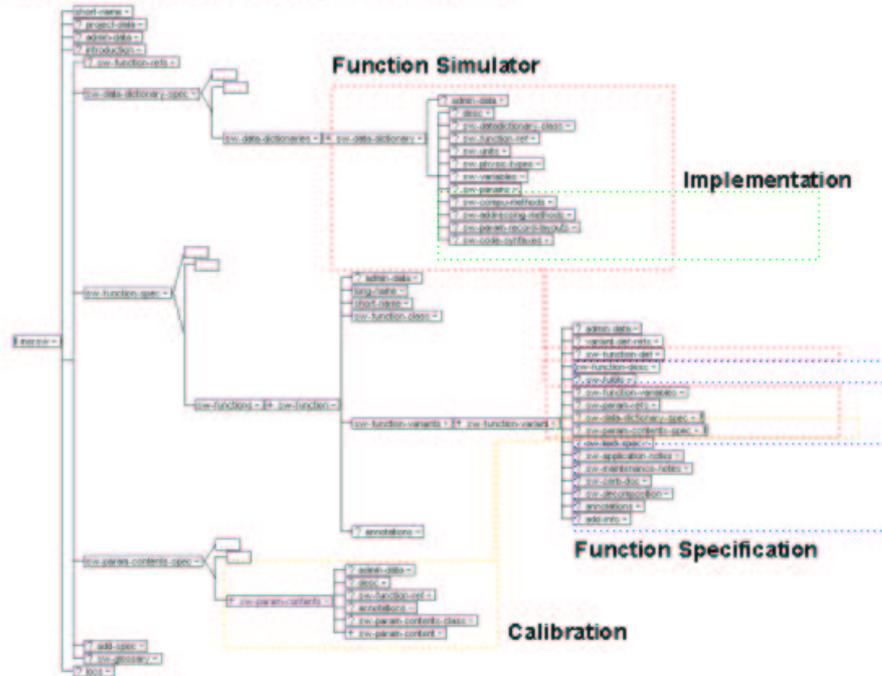
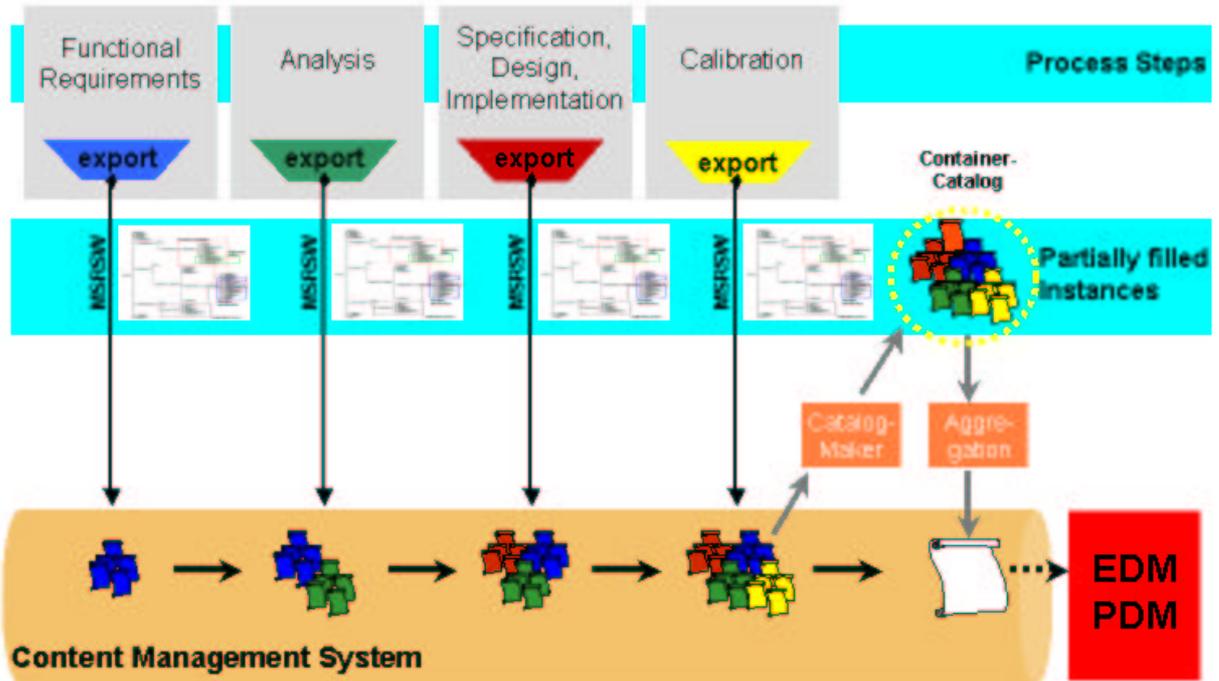


Figure 4: Folie 4

eDoc Process Phases

Control Unit Software Development



EU_Folie5.PNG

Figure 5: Folie 5

eDoc Modules

ECU Software Documentation Process

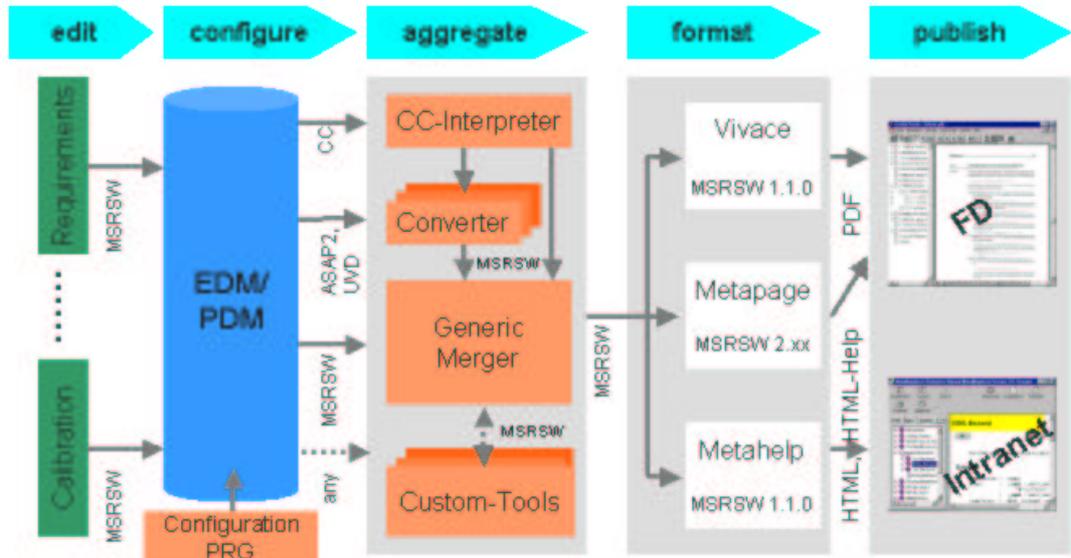


Figure 6: Folie 6

EU_Folie6.PNG

Algorithm σ

Tree-representation

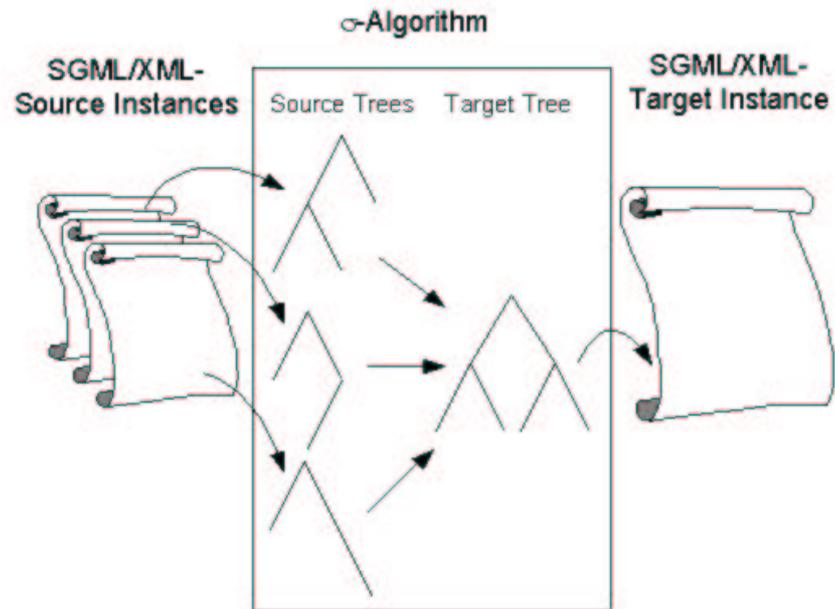


Figure 7: Folie 7

Algorithm σ

Example of a σ -tree representation

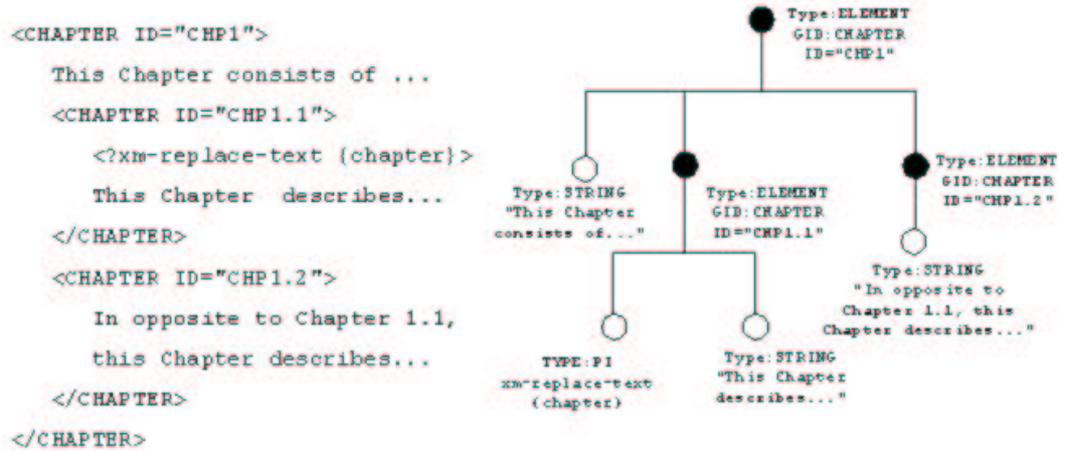


Figure 8: Folie 8

Algorithm σ

Merging σ -trees

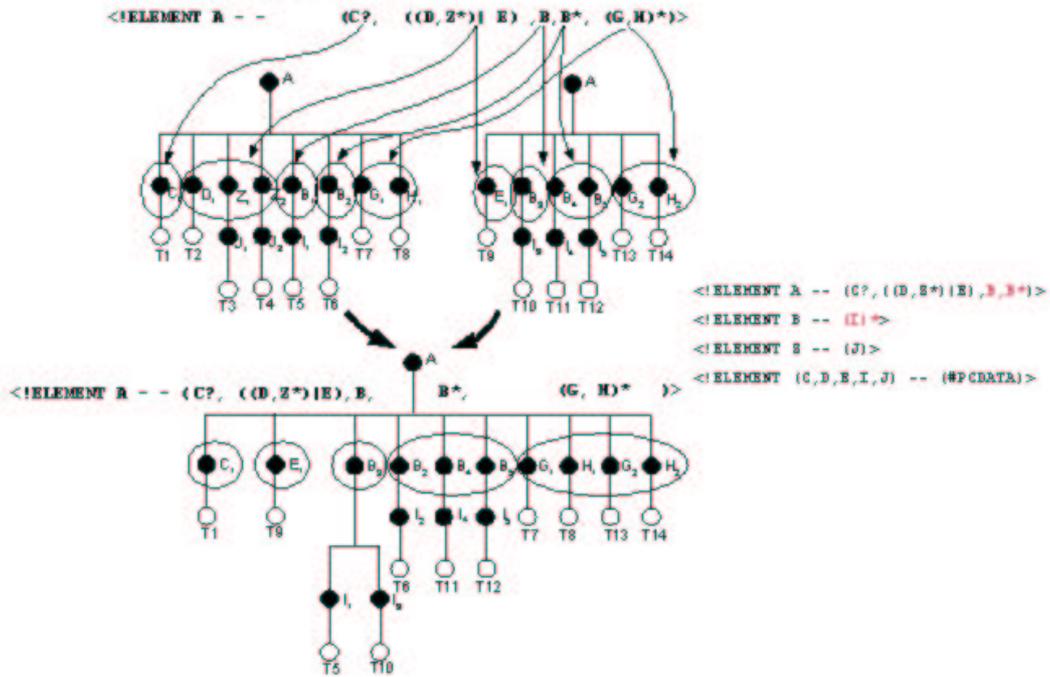
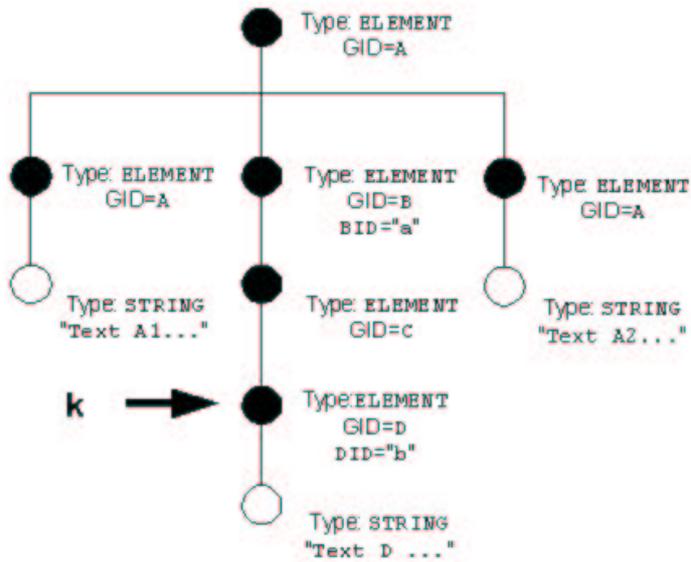


Figure 9: Folie 9

Algorithm σ

σ -path of a node k



$$\sigma p(k) = "A/B [BID=a] /C/D [DID=b] "$$

Figure 10: Folie 10

Algorithm σ

Definitions



δ : actual Document Type Definition (DTD) all sources are valid instances of

$\text{gid}(k)$: returns the generic identifier (gid) of a node k of type `ELEMENT`

$K_{\sigma p(k)}^{\delta}$: List of all σ -source nodes having the same σ -path as k

$\Gamma^{\delta}(\text{gid}(k))$: content model of the element corresponding to node k

$\Gamma_m^{\delta}(\text{gid}(k))$: σ -modelgroup no. m in content model $\Gamma^{\delta}(\text{gid}(k))$ of node k

Figure 11: Folie 11

Algorithm σ

σ -modelgroups



`<!ELEMENT A - - ((B,C)?, (D|E), (F,G)*)>`

$\Gamma^{\delta}_1(\text{gid}(A)) = (B,C)?, \Gamma^{\delta}_2(\text{gid}(A)) = (D|E), \Gamma^{\delta}_3(\text{gid}(A)) = (F,G)*$

`<!ELEMENT A - - ((B,C)+ | (D, (E | F))*)>`

$\Gamma^{\delta}_1(\text{gid}(A)) = \Gamma^{\delta}(\text{gid}(A))$

`<!ELEMENT A - - ((B,C)*,D)*>`

$\Gamma^{\delta}_1(\text{gid}(A)) = \Gamma^{\delta}(\text{gid}(A))$

Figure 12: Folie 12

Algorithm σ

Intuitive Demands for σ



- The target instance has to be valid concerning DTD δ .
- The source instances are represented by their corresponding σ -source trees.
- The σ -target tree is later converted to the target instance.
- Optional or mandatory σ - modelgroups imply the creation of **merge-priorities**.

Figure 13: Folie 13

Algorithm σ

Intuitive Demands for σ



- All nodes of the highest merge-priority σ -source tree shall be contained in the σ -target tree.
- Hence, the σ -target tree has to contain the root node of the highest priority σ -source tree.
- ELEMENT-type child nodes k_c of a σ -source tree node k_s may only be regarded as potential child nodes of an actual σ -target tree node k if k_s has the same σ -path as k .
- These Nodes k_c are appended below k grouped by instantiations of each $\Gamma_m^\delta(\text{gid}(k))$.

Figure 14: Folie 14

Algorithm σ

Intuitive Demands for σ



- A target instance with an ID-attribute value existing more than once is not valid.
- A target instance with IDREF(S)-attribute values pointing to not existing ID-attribute values is not valid.

Figure 15: Folie 15

Algorithm σ getInsertType(k)

**Input:**

actual σ -target tree node k with $\text{type}(k) = \text{ELEMENT}$

Output:

$\text{insert_type} \in \{\text{ONLY_FROM_ORIGIN}, \text{FROM_ALL_SOURCES}\}$

- The function $\text{getInsertType}(k)$ provides information whether the subtree of a σ -target tree node k can be copied directly from its origin node $\text{orig}(k)$.

Figure 16: Folie 16



Algorithm σ

σ without considering ID-attributes



Input:

σ -source trees $tr(S_1), \dots, tr(S_N)$, δ having no ID-attributes,
 S_1, \dots, S_N valid concerning δ .

Output:

σ -target tree Λ .

Figure 17: Folie 17

Algorithm σ

σ without considering ID-attributes



1. $\Lambda := \text{root}(\text{tr}(S_N))$
2. $k := \text{root}(\text{tr}(S_M))$
3. Traverse Λ top-down, left-right. Let k be the actual σ -target tree node.
 1. IF $\text{type}(k) \in \{\text{PI}, \text{STRING}, \text{COMMENT}\}$ THEN
 1. Copy all contents (i.e. the node text) of $\text{orig}(k)$ beneath k .
 2. Proceed traversing Λ top-down, left-right.
 2. ELSE
 1. IF $\text{getInsertType}(k) = \text{ONLY_FROM_ORIGIN}$ THEN
 1. Append subtree of $\text{orig}(k)$ beneath k .
 2. Stop traversing the subtree of k . Proceed traversing Λ .
 2. ELSE calculate $K_{\text{ap}(k)}^\delta$. For all $\Gamma_m^\delta(\text{gid}(k))$ in $\Gamma^\delta(\text{gid}(k))$ do:
 1. IF $\text{occ}(\Gamma_m^\delta(\text{gid}(k))) \in \{*, +\}$, THEN
Append $\text{contentsAt}(K_{\text{ap}(k)}^\delta, \Gamma_m^\delta(\text{gid}(k)))$ beneath k .
 2. ELSE Append $\text{first_ex_contentsAt}(K_{\text{ap}(k)}^\delta, \Gamma_m^\delta(\text{gid}(k)))$ beneath k .

Figure 18: Folie 18

Algorithm σ

Example: Step 1

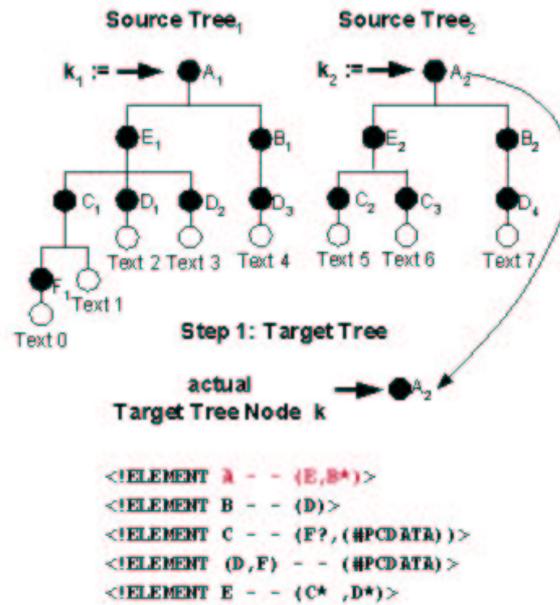
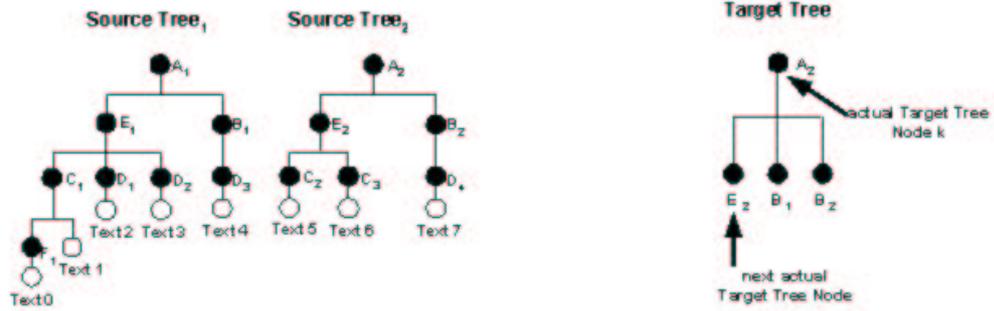


Figure 19: Folie 19

Algorithm σ

Example: Step 2



```

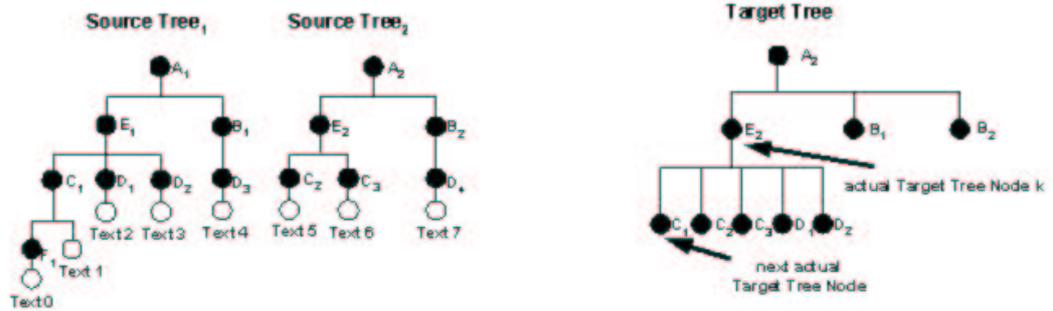
<ELEMENT A - - (E,B*)>
<ELEMENT B - - (D)>
<ELEMENT C - - (F?, (#PCDATA))>
<ELEMENT (D,F) - - (#PCDATA)>
<ELEMENT E - - (C*, D*)>

```

Figure 20: Folie 20

Algorithm σ

Example: Step 3



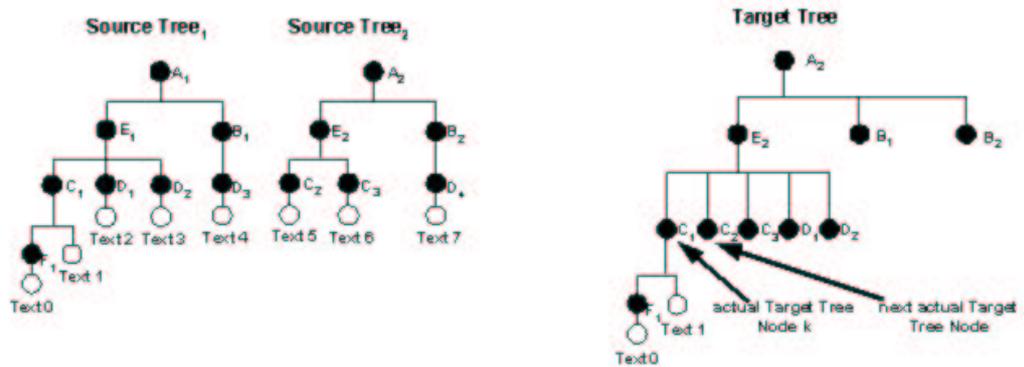
```

<ELEMENT A - - (E,B*)>
<ELEMENT B - - (D)>
<ELEMENT C - - (F?, (#PCDATA))>
<ELEMENT (D,F) - - (#PCDATA)>
<ELEMENT E - - (C*, D*)>
  
```

Figure 21: Folie 21

Algorithm σ

Example: Step 4



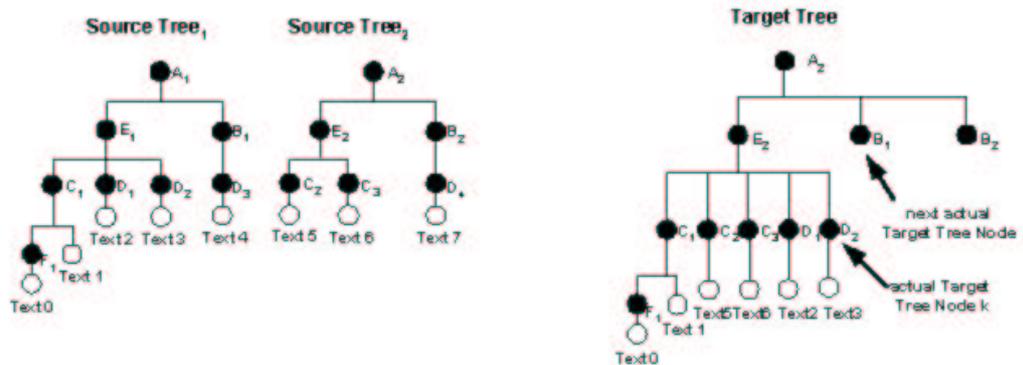
```

<ELEMENT A - - (E,B*)>
<ELEMENT B - - (D)>
<ELEMENT C - - (F?, (#PCDATA))>
<ELEMENT (D,F) - - (#PCDATA)>
<ELEMENT E - - (C*, D*)>
  
```

Figure 22: Folie 22

Algorithm σ

Example: Step 8



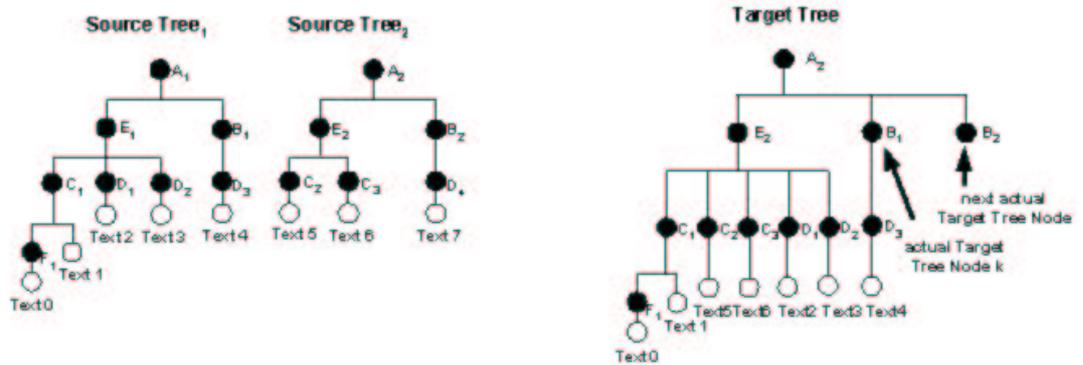
```

<ELEMENT A - - (E,B*)>
<ELEMENT B - - (D)>
<ELEMENT C - - (F?, (#PCDATA))>
<ELEMENT (D,F) - - (#PCDATA)>
<ELEMENT E - - (C*, D*)>
  
```

Figure 23: Folie 23

Algorithm σ

Example: Step 9



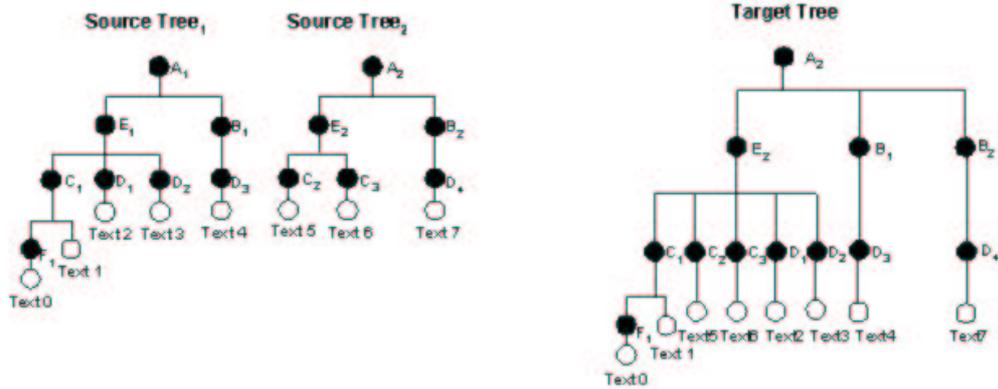
```

<ELEMENT A - - (E,B*)>
<ELEMENT B - - (D)>
<ELEMENT C - - (F?, (#PCDATA))>
<ELEMENT (D,F) - - (#PCDATA)>
<ELEMENT E - - (C*, D*)>
  
```

Figure 24: Folie 24

Algorithm σ

Example: Finished



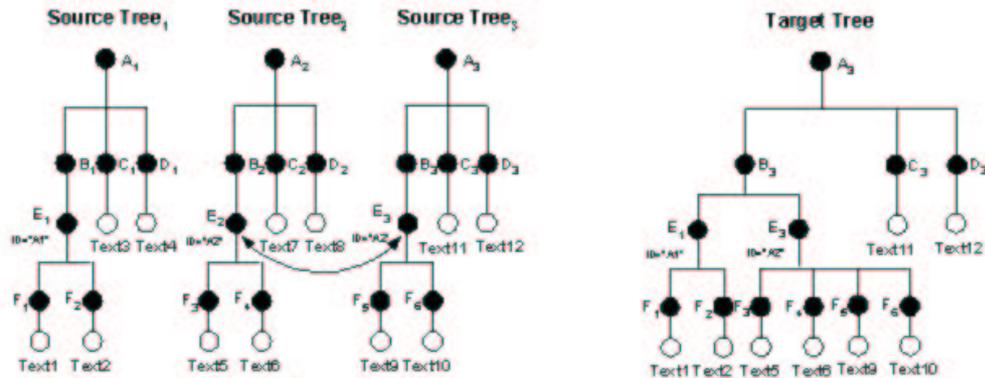
```

<ELEMENT A - - (E,B*)>
<ELEMENT B - - (D)>
<ELEMENT C - - (F?, (#PCDATA))>
<ELEMENT (D,F) - - (#PCDATA)>
<ELEMENT E - - (C*, D*)>
  
```

Figure 25: Folie 25

Algorithm σ

Example: ID-conflict of type 1



```

<ELEMENT A - - (B,C,D)>
<ELEMENT B - - (E)*>
<ELEMENT (C,D,F) - - (#PCDATA)>
<ELEMENT E - - (F)*>
<ATTLIST E ID ID #REQUIRED>
  
```

Figure 26: Folie 26

Algorithm σ

ID-conflict of type 2



```

<!ELEMENT A -- (Y?,B,Z?) >
<!ELEMENT B -- (C*,D*) >
<!ELEMENT (C,D,E,F) -- (#PCDATA) >
<!ATTLIST C ID ID #REQUIRED >
<!ATTLIST D ID ID #REQUIRED >
<!ELEMENT Y -- (E)* >
<!ELEMENT Z -- (F) >
<!ATTLIST E ID ID #REQUIRED >
<!ATTLIST F ID ID #REQUIRED >

```

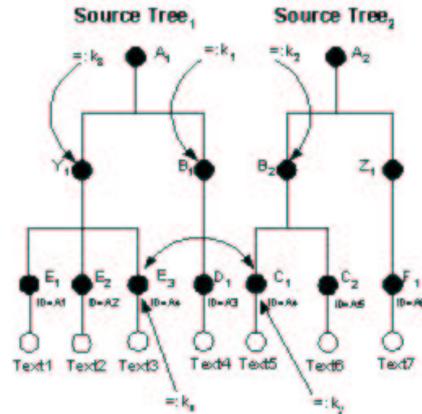


Figure 27: Folie 27

Algorithm σ

ID-conflict of type 2 with valid target tree

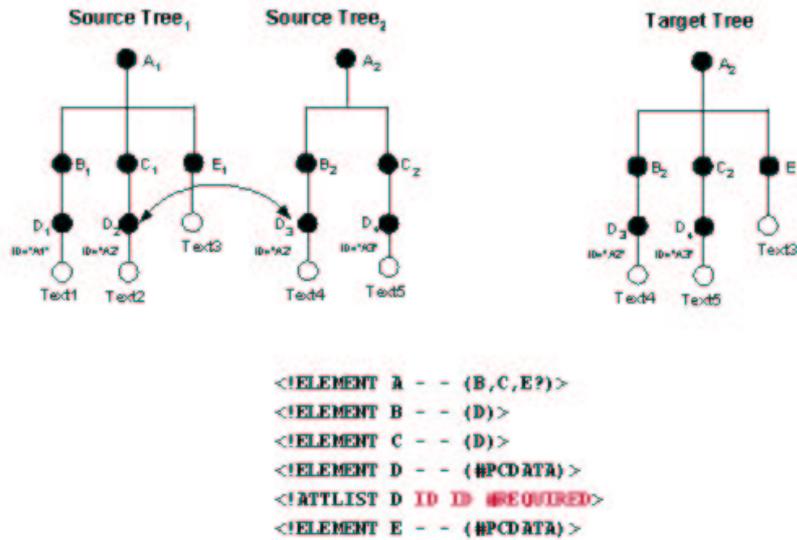


Figure 28: Folie 28

Algorithm σ IDREF-conflict

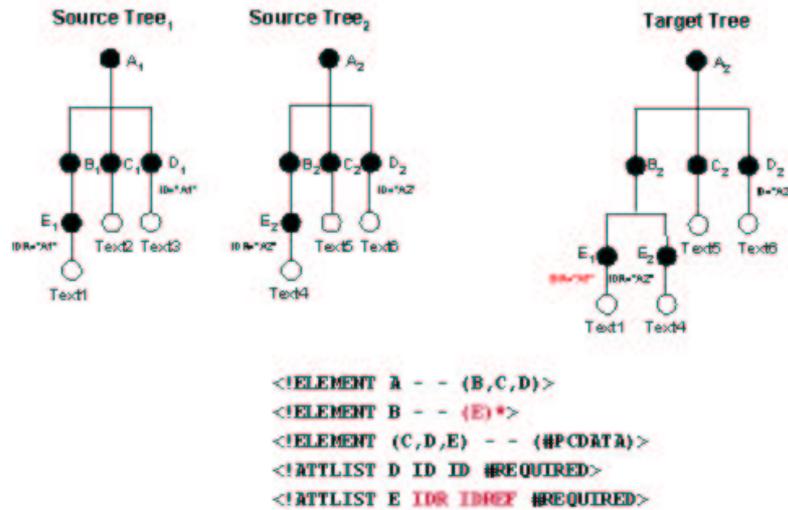


Figure 29: Folie 29

	A Generic Algorithm for Merging SGML/XML Instances XML Europe 2001 Chapter: Presentation	Page: 34/37 Date: 2002-11-03 State: RD
---	--	--

Implementation SIGMA

Demonstration



Figure 30: Folie 30

EU_Folie30.PNG

	A Generic Algorithm for Merging SGML/XML Instances XML Europe 2001 Chapter: Presentation	Page: 35/37 Date: 2002-11-03 State: RD
---	--	--

Contact



<http://www.msr-wg.de>

gerald.manger@bmw.de

Figure 31: Folie 31

EU_Folie31.PNG

Documentadministration

Versions Overview

Document Part	Date	Editor			
		Company	Version	State	Remarks
From page 4 	1	Gerald Manger, BMW Group			
	RD 2002-11-03 Changes 1	MEDOC			



Configuration Parameters

Company (**—company**)

MEDOC

Language (**—lang**)

English

Treatment of content for Xrefs (**—xrefcontent**)

Xref classes are shown

Specifying 'See' for XRefs

'See' is to be inserted for xrefs

Treatment of filenames in graphics (**—figname**)

Filenames for graphics are shown

Treatment of width and height attributes of graphics (**—figdimension**)

Width and height of graphics is not considered

Titlepage Graphic (**—graphic**)

No title graphic specified

Logo Graphic (**—head-logo**)

MSR_bw_sm.eps

Fixtext File (**—fixtext**)

C:\Programme\medoc\Metapage\mmapps\msrrep\lib\msrrep_ft.xml

Output of Local Administrative Data (**—admindata**)

Local administrative data is output

Filename

D:\Projekte\1052\new_seite\download\Literature/XML-Europe2001\xml\xml_europe.xml

MetaMorphosis-Version

3.2

Form Version

2.0 (MetaPage)

Date

21/05/2002 11:10:12