Strategies for implementing SGML/XML as a glue layer in engineering processes

MSR MEDOC, Bernhard Weichel, Robert Bosch GmbH

Table of Contents

Table of Contents 2

Introduction 3

1 **Strategies for implementing SGML/XML as a glue layer in engineering processes** **4**

1.1 Abstract: 4

2 **Engineering processes for engine management systems** **5**

3 **A flexible document architecture** **9**

4 **The document base as a glue layer** **11**

4.1 The role of the DTD in the document base 11
4.2 Building a database oriented DTD 11
4.2.1 Relational models 12
4.2.2 Object oriented models 12
4.3 Using a database oriented DTD 14
4.3.1 Supporting the entire life cycle 15
4.3.2 General approaches for readers 17
4.3.2.1 Source driven approach 17
4.3.2.2 Target driven approach 17
4.3.3 Query language 17
4.3.4 Authoring in SGML/XML directly 21
4.3.4.1 Link manager 21
4.3.4.2 Templates 22
4.3.5 Further experience with database oriented DTDs 23

5 **The document base in the MSR consortium** **25**

5.1 Natural addressing in MSR 26

6 **Desires for SGML/XML** **28**

7 **Conclusion** **29**

Documentadministration 30

Introduction

Companies

MSR MEDOC [MEDOC]

Name **Roles**	**Departement**	**Address**	**Contact**
Bernhard Weichel, Robert Bosch Gmb-H Author			

Version Information

Document Part	**Editor**			
	Company	**Version**	**State**	**Remarks**
1 RD 2002-11-03 For details refer to nr. 1, Page	Bernhard Weichel, Robert Bosch GmbH			
	MEDOC			

1 Strategies for implementing SGML/XML as a glue layer in engineering processes

Bernhard Weichel Robert Bosch GmbH,

1.1 Abstract:

The scope of engineering processes tends to strengthen the argument for requiring a single source approach throughout the entire process. SGML/XML can be used as an integration platform by providing a unique data exchange and archive format. This presentation shows a multiple layer architecture for document and data management and the essential role of SGML/XML. It will also explain possible strategies for implementing the data model in SGML/XML. It covers topics such as natural addressing of objects, subclassing, and conceptual versus logical versus physical data schemes. It also includes a discussion of desired enhancements of the standards for better support of data modeling.

2 Engineering processes for engine management systems

Engine management systems for modern cars are very complex (). They collect characteristical data like speed, load, various temperatures etc. from the engine and control the engine performance with respect to various optimization criteria. In addition to this other function like on board diagnosis, security etc. must be provided.

Motronic ME7

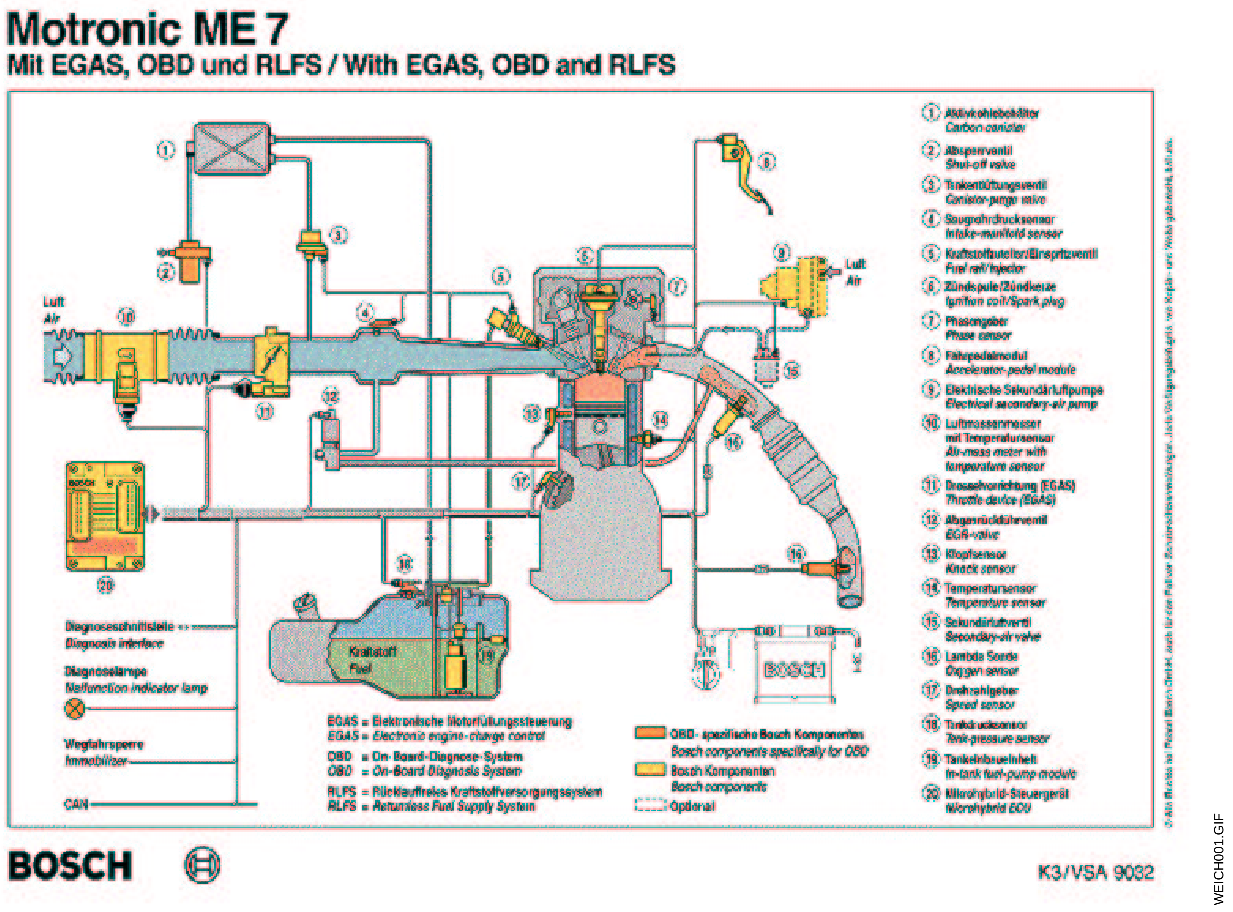


Figure :

The engineering process for such complex systems covers multiple phases () handled by many different people. These participants in the process are using various engineering tools to produce results appropriate to their process phase but also for the entire process. The problem is, that each sub process requires a set of documents and data which is not necessarily known by the other participants. This often leads to individual data files, documents and data formats where the data are captured many times.

Phases in the engineering process

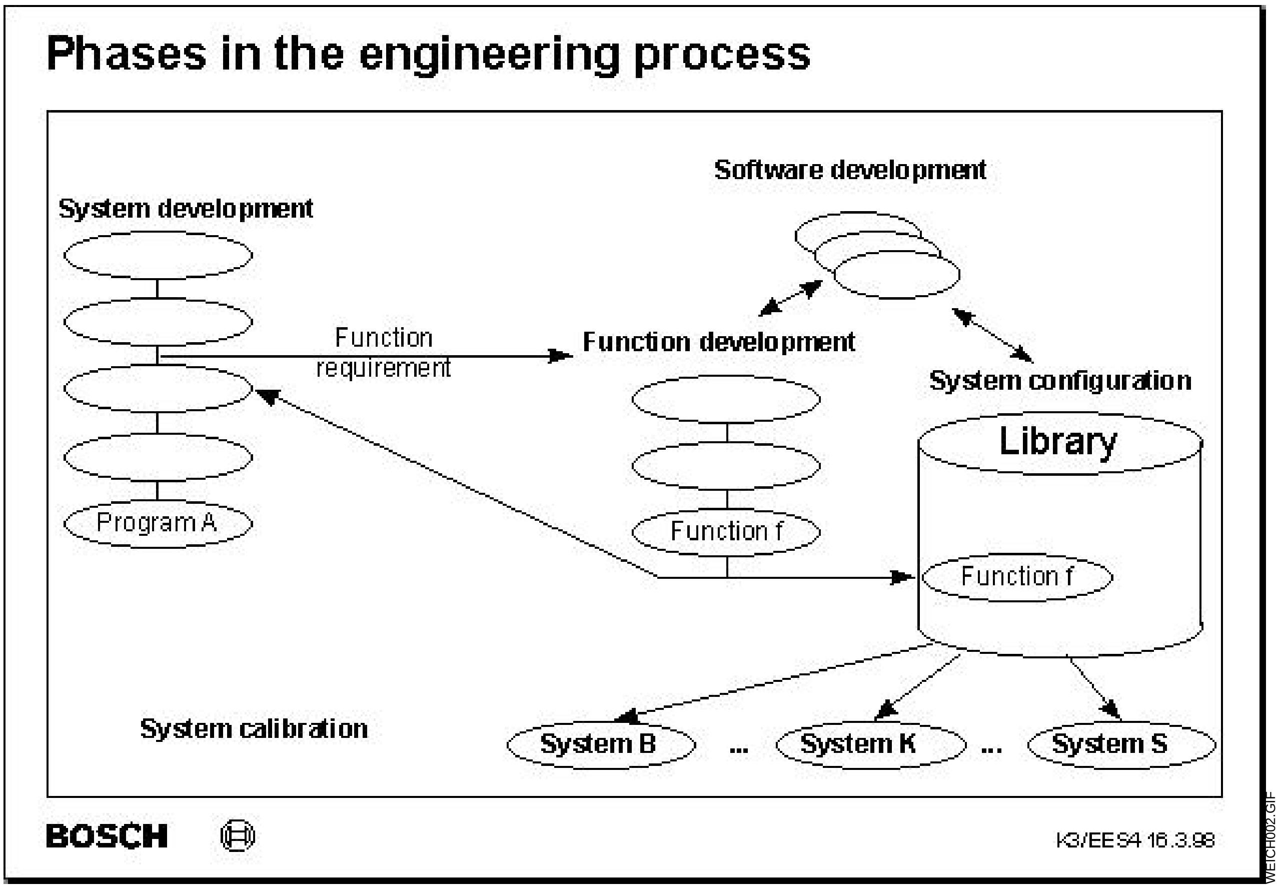


Figure :

of an engine management system. Every car manufacturer produces various engine types (different volumes, number of valves, turbo etc.). In addition to this, there are many variants of vehicles which impact the amount of variants for the control system. Worldwide we have different laws. All this leads to hundreds of different MOTRONIC programs.

These systems are based on platforms, configured by using generic functions, customer and project specific functions. The systems are applied to the individual engines by adjusting thousands of parameters. In conclusion, we see the following major areas in this process:

System development

Development of systems with complete programs built of functions from an existing library as well as new functions which are defined as specification for the function development. In addition to this, system development covers areas others than software (but which may influence software).

Function development

This discipline defines new functions performed in the ECU to control the engine. The function development is based rather on generic principles than on individual engines or implementations. Therefore this tasks is usually performed using control simulation tools where functions are defined using block diagrams, state machines etc. An example for this is given in

Software development

The functions are coded for the individual target platform. Although actually coding is a manual task, we will see more and more code produced by code generators built into the simulation tools which are fed by all the required information.

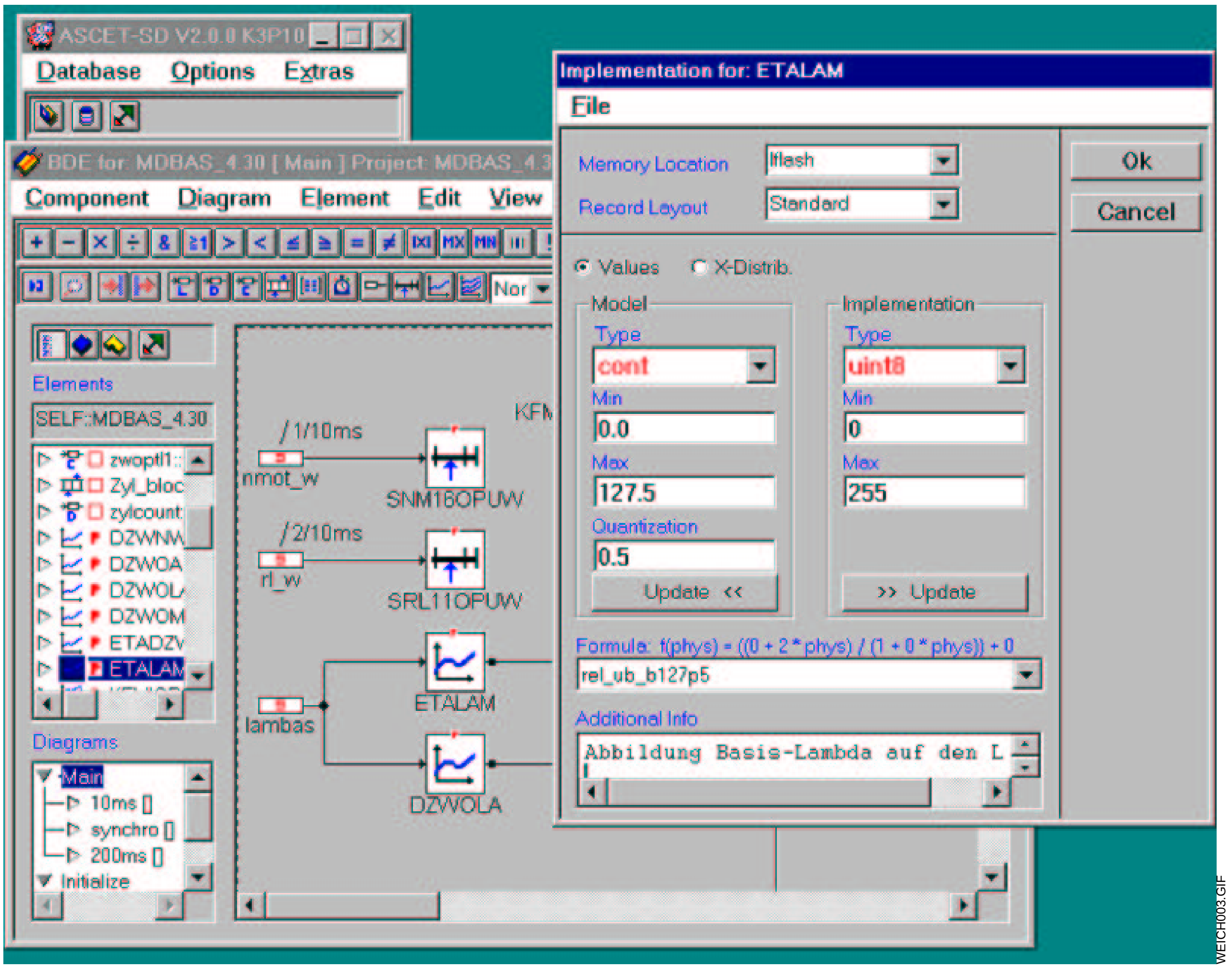
System configuration

The developed functions are integrated into complete systems. This is done by creating build descriptions, project configuration files etc. One of the software variants is built using about 250 out of 500 available functions.

Calibration

The entire system is applied to a certain engine type (variant). In this phase all of the about 3000 calibration parameters in the system are adjusted to match the performance objectives for the actual engine variant. In order to perform this calibration a detailed documentation as well as appropriate tool support is required. The basic information set for this task is created in the phases mentioned above.

Function development in ASCET simulator



Implementation for: ETALAM

Memory Location: Iflash
Record Layout: Standard
(● Values ○ X-Distrib.)

Model — Type: cont; Min: 0.0; Max: 127.5; Quantization: 0.5
Implementation — Type: uint8; Min: 0; Max: 255

Formula: f(phys) = ((0 + 2 * phys) / (1 + 0 * phys)) + 0
rel_ub_b127p5

Additional Info: Abbildung Basis-Lambda auf den L

Figure :

These areas in the process are performed by different people using different Tools. As already mentioned each sub process requires a set of documents which is not necessarily known by the other participants. Actually the data is handled using individual files with specific data formats. These files follow different strategies and are often incompatible, since they are not based on an overall data model. Therefore some documents and data are captured many times.

3 A flexible document architecture

While dealing with such complex engineering processes, it is often unclear what a document really is. The three layer architecture in can help in this respect:

Document layer architecture

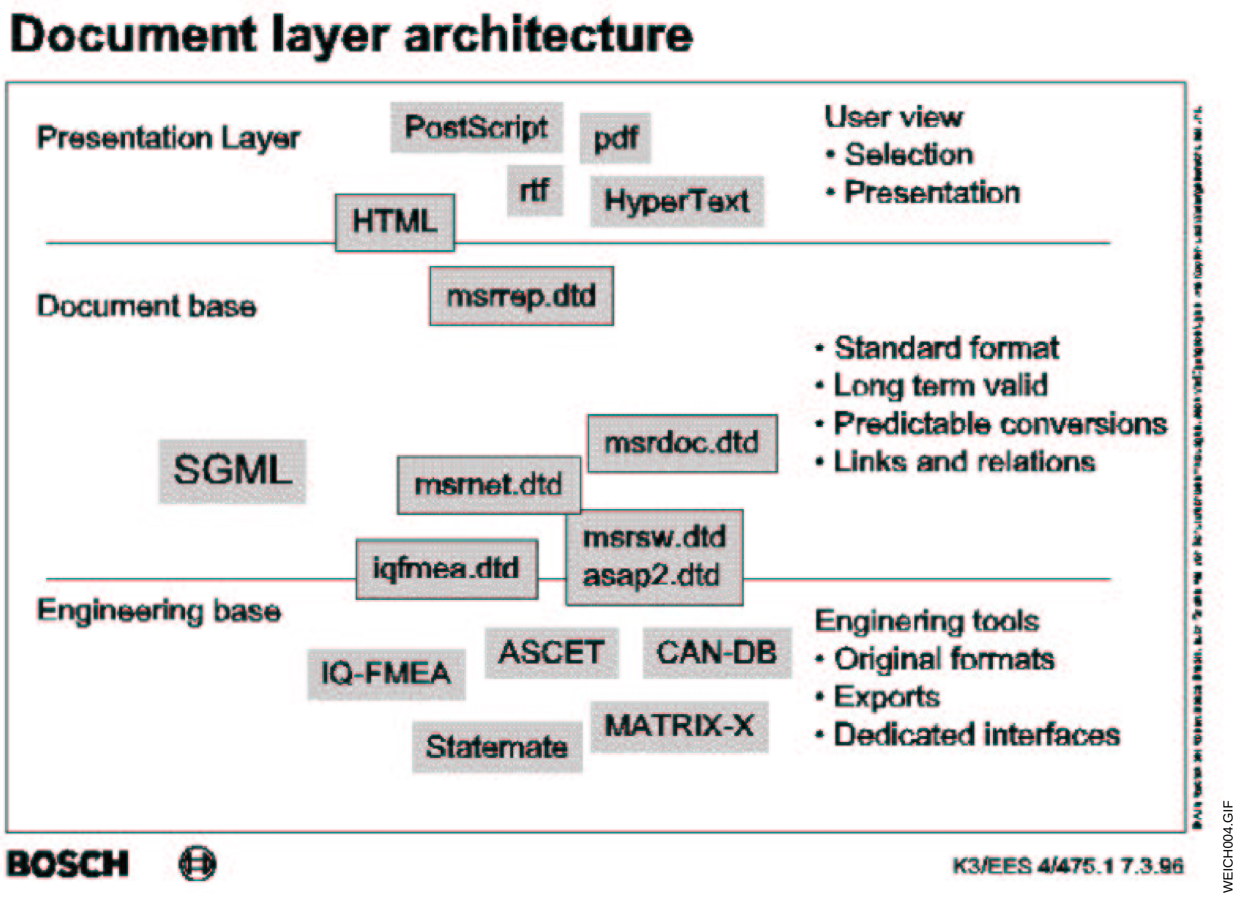


Figure :

Presentation view

This is the daily experience with a document. It is a piece of paper with a certain amount of information which is presented using various fonts, layouts and graphical elements. Progressive people use this document in electronic form.

The main focus of this layer is to present the right set of information in an appropriate layout. Therefore it is the primary domain of typewriters, word processors and electronic distribution media. I would even position HTML in this layer.

Engineering Base

which has no access to the content of the administrated data.

Nowadays engineering tools produce document views using report generators or interfaces to common word processors often proudly marketed as ”documentation interface”. In a broader sense, the information in the engineering base itself and not only the exported presentations must be treated as documents.

If views are required which reflect information from more than one engineering tool, then either some connection of the tools is required, or the documents must be assembled manually.

Obviously there is a gap between the presentation view and the engineering base.

Document Base

The existing gap between the two layers mentioned above can be closed by introducing a middle layer, the document base. This layer must be built upon a standard data format which can be used to generate all the required outputs (document views) in the presentation layer while still having the power to keep the semantics of the engineering base at least to some extent.

This layer is the domain of SGML/XML, which is easy to create, able to define powerful data structures and not too hard to read by humans as well as by machines.

4 The document base as a glue layer

As we have seen, the document base can be used to harmonize the documentation activities in the engineering process. But there is more! The document base can serve as a glue layer for documentation activities as well as for other important tasks:

- The standardized data format then enables us to generate the required views by using standard tools regardless what the original format was. The presentation view can even be generated if the engineering tool is not actually available. If the requirements of the presentation view are changing, additional views can be defined to meet them.
- The standard data format can even be used to overcome peer to peer data formats which tend to be very specific to tools and tasks. Thus the document base not only bridges the gap between engineering tools and document presentations but also existing gaps in the engineering base itself.
- The document base can be used for long term archiving. This protects the investment in information which remains accessible when engineering tools disappear. As we all know, the life of our data is several times longer than the availability of our tools.
- The document base can be used to integrate results of different engineering groups even if there are different tools in use. This can be utilized across companies. The MSR Chapter 5 The document base in the MSR consortium p. 25 consortium follows this approach (see).

4.1 The role of the DTD in the document base

The document base can be setup with all the flexibility and benefits provided by SGML. But it is obvious, that the design of the DTDs determines the usability of the document base. In principle, as I see it, there are three classes of DTDs

Layout oriented

The focus of this DTD style is rendition, therefore the capabilities are mostly oriented to publishing tools. This kind of DTD cannot be used for data exchange between tools. This DTD style is not even appropriate to build the document base.

Presentation oriented

Chapter 5 The document base in the MSR consortium p. 25 The focus of this DTD style is the structure of the presentation layer but not rendering. So such DTDs allow to support various layouts on many different media. An example for this class of DTDs is DOCBOOK.DTD or MSRREP.DTD (). This sort of DTD is well suited for the document base if the required presentations reflect layout and global rearrangement. It is not really possible to extract information based on the content of the document, since the semantic of the DTD is oriented on the structure of a presentation rather than on the a data model of the presented subjects.

Database oriented

This class of DTDs is essential if data exchange is performed via the document base. The focus of such DTDs is a data model reflecting the reality of the objects in the process. This means, that the semantic of the DTDs reflect the details of the presented subjects. For example we have part instead of chapter or interface-prms instead of table .

4.2 Building a database oriented DTD

When building a database oriented DTD it appears that all the methods of data analysis can be used. But if you do this, you might end up in a relational data model which does not necessarily reflect the real world of engineering objects. An SGML DTD is much more like an object oriented database scheme with all its power than a relational one.

4.2.1 Relational models

The relational data model is defining records of data which are related to each other. The tables are identified using an entity relationship analysis which identifies the relationships between the table rows. In order to store data without redundancy in relational tables, the data model must be normalized. It is obvious that this paradigm can very easy be implemented using SGML/XML. This makes it easy to use SGML/XML as an archive as well as an exchange format for relational database applications. I will illustrate this in the following figure ().

relational data model

company emps emp emp-no=”4321” name=”Bernhard Weichel” adress=”stuttgart”/ emp emp-no=”12345” name=”Helmut Gengenbach” adress=”schwieberdingen”/ /emps depts dept name=”K3/EES4” mgr=”4321”/ dept name=”FV/PLI” mgr=”0815”/ /dept /company

But for engineering applications the relational model is not really appropriate. The information structure is very complex, so that normalization is a very high effort and ends up in an awful lot of links. SGML/XML’s capabilities are far beyond the relational model.

4.2.2 Object oriented models

SGML/XML has the power to define very complex data models. The major reason for this is the fact that it represents a tree structure which is one of the most intuitive data models for the real world. I am using the term ”object oriented” rather in the sense that the data model is oriented towards the structure of the described objects than in the traditional sense of object oriented programming.

- The tree structure reflects the ”one to one” as well as the ”one to many” relationships of the relational database. But it goes far beyond this because this structure can be nested infinitely and some control of the cardinality is there.

There is no need for normalization, since SGML/XML can very well handle denormalized data in its tree structure.

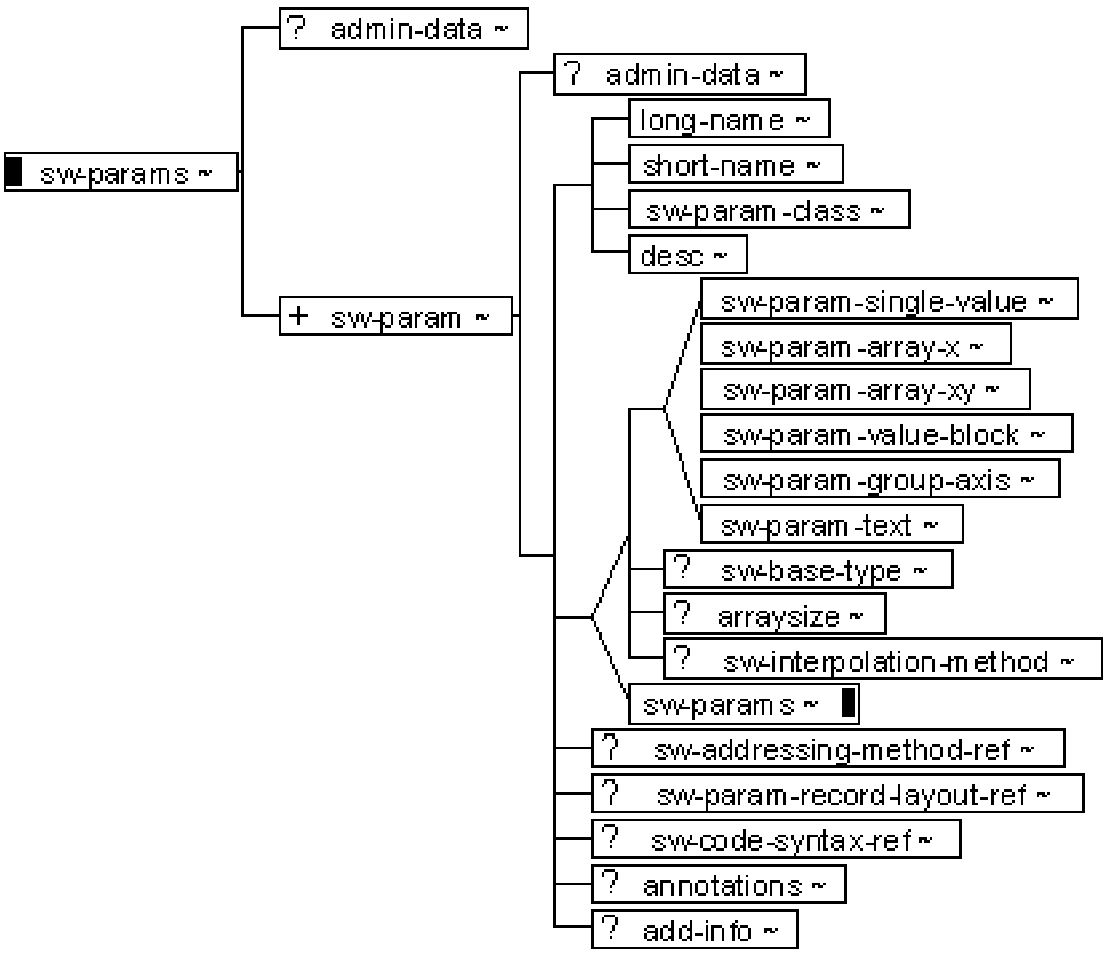
- Data objects are grouped in containers. This expresses most of the relationships directly and avoids the introduction of artificial access keys or links.
- Chapter 5.1 Natural addressing in MSR p. 26 Chapter 4.3.1 Supporting the entire life cycle p. 15 If there are ”many to many” relationships, those can be established using existing data (natural keys). Any data can be used to establish links within an instance (see also). These links have a very specific semantic (link types) which can be derived from the position of the anchor, the anchor element itself or be determined by architecture attributes. This approach for linking is based on the natural object hierarchy (also establishing a hierarchy of namespaces) and not on the element hierarchy in the instance. Therefore the resulting methods usually are not directly compatible with the

linking methods provided by SGML/XML such as ID/IDREF, HyTime or XLL. To use SGML/XML methods for linking, a transformation step is required (see also).

- The DTD provides a data model with implicit access paths to data. This allows to use very flexible navigation approaches and query languages.
- The occurrence operator allows various instances matching one DTD. This can be used for subclassing. Subclassing can be controlled by other elements (e.g. sw-param-class in) or by attributes (Architectural Forms).
- Object orientation (in the sense of object oriented programming) can be implemented using SGML/XML by defining the appropriate semantics for the elements. This covers the inheritance of data contents (not of data structures) as well as the option to keep certain methods as queries in elements (although there is no established standard for this).

As one example for an object oriented model is given in the following figure (). It comes from MSRSW.DTD which is used in the development of software for an engine control system. The DTD is the data scheme for calibration parameters in the ECU . Such a parameter can be a single value, a curve or even a map. This is reflected in the DTD by some alternative branches used to support alternative details. For example, a single value only has a value axis but no x or y axis. A curve has a value axis, an x axis but no y axis. The map however has three axis (value, x and y). sw-param-class denotes the class of the parameter and determines which alternatives must be used. This illustrates, how subclassing can be performed using object oriented DTDs .

Example of an object oriented DTD



WEICH006.GIF

Figure :

The following illustrates, how this structure is filled in the instance. It is one of the characteristics to calibrate an engine management system. The example (related to the one in) is a curve with individual set points.

Instance for calibration parameter

sw-param long-nameLambda Wirkungsgrad/long-name short-nameetalam/short-name sw-param-classcurve_individual/sw-param-class desc Abbildung Basis-Lambda auf den Lambda-Wirkungsgrad ohne Eingriff bezogen auf optimales Moment bei Lambda=1 /desc sw-param-array-x sw-param-axis-x sw-axis-individual sw-variable-reflambas/sw-variable-ref max-count10/max-count /sw-axis-individual /sw-param-axis-x sw-param-axis-values sw-compu-method-refrel_ub_b127b5/sw-compu-method-ref /sw-param-axis-values sw-addressing-method-refNearRomByte/sw-addressing-method-ref sw-param-record-layout-refKlAUbSstUbWUb/sw-param-record-layout-ref sw-code-syntax-refKlNearAUbSstUbUb/sw-code-syntax-ref /sw-param

4.3 Using a database oriented DTD

First I want to talk about how a database oriented DTD can support the entire life cycle. This is important for an understanding of the use cases for such a DTD presented later.

4.3.1 Supporting the entire life cycle

If the DTDs are bound to support all the phases of an engineering process (), a new set of problems must be solved:

- In most of the phases in the process not all information is available. This makes it nearly impossible to define the correct occurrence in the DTD. For the development of engine management software we found six different flavors of instances. Strictly speaking we need six different DTDs. The MSRSW.DTD has about 250 elements. Using SGML schema language it is impossible to manage this.
- Sometimes parts of the data are created in different, even in parallel process steps. An example for this situation is the fact that the data dictionary of an engine management software is generated using some CASE tool while the prose description is captured by the engineer using an SGML/XML editor. This requires sophisticated version control strategies and process management.
- In the intermediate steps in the process not all objects of the entire systems exist. This forces a method to handle broken links. For example when a function is defined, we know that someone else defines a variable which must be used. But this variable will be visible not before the integration phase. So the broken link is not a bug. It is a feature for the integration phase.

Chapter 5.1 Natural addressing in MSR p. 26 Chapter 4.2.2 Object oriented models p. 12 For the final results in the process it is very helpful to go back to standard methods to establish links. This makes it easier to circulate the final results etc. for treatment with standard viewers. For this reason, MSR DTDs have an MSR specific method () of natural addressing in parallel to standard ID/IDREF, HyTime and XLL based methods. All the links will be synchronized for exchange using SGML tools (see also).

- Different tools use different global data models. This is reflected in the capabilities of the SGML generators in those tools, especially in terms of the available iterators. For example there may be an entire data dictionary knowing the function for each variable. On the other hand there may be a specific function knowing all its variables. So the same information is available but structured using different approaches.

In the first place it appears to make sense to have a sub-DTD for each process phase or even one per tool. This would allow to define the data set per process phase by this DTD. On the other hand, this approach requires specific SGML applications for each process phase. The knowledge about the data structures would also be process phase specific.

One DTD for different process phases

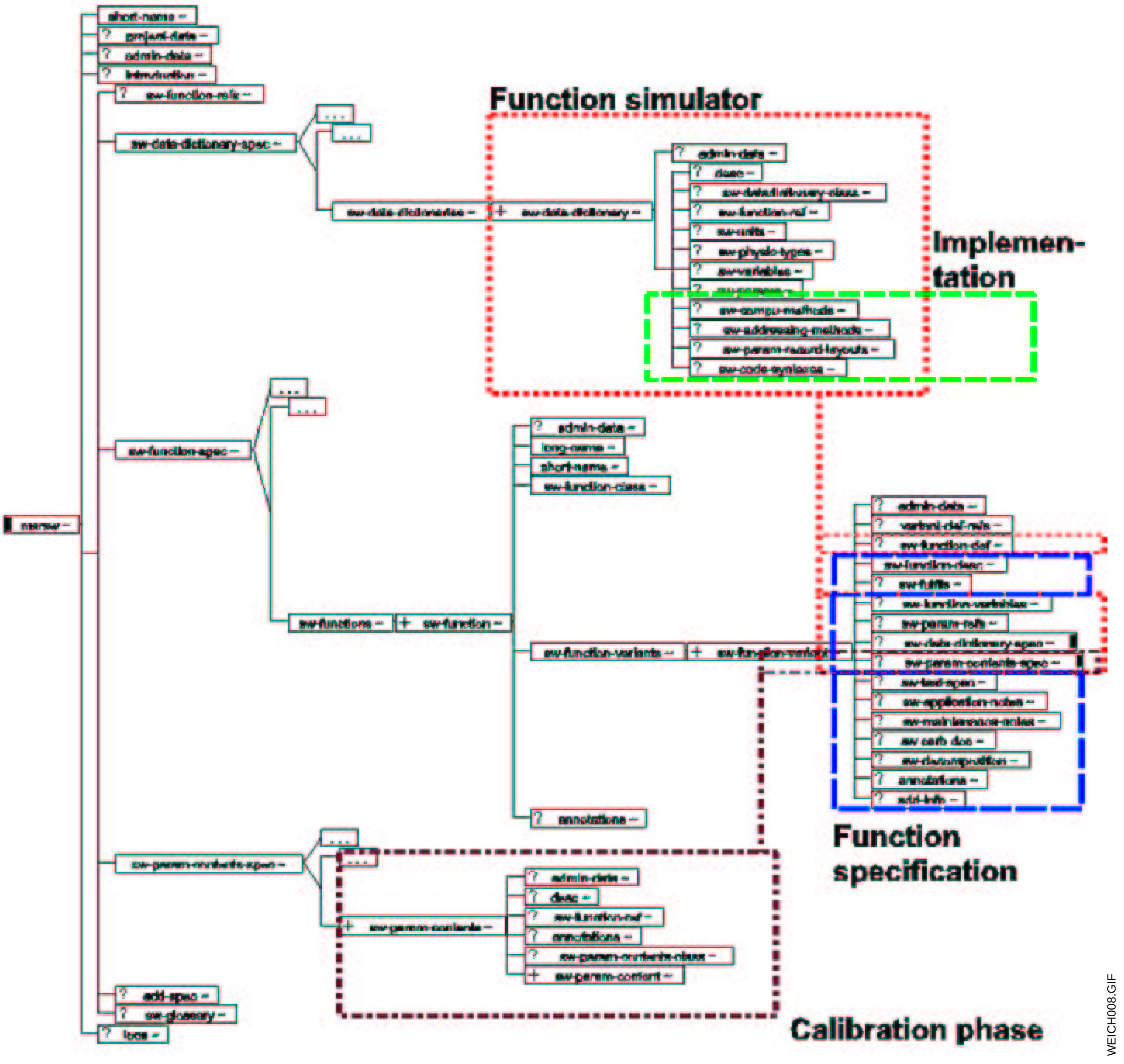


Figure :

For these reasons, we found it better to use same DTD for all the steps and information fragments instead of defining sub DTDs with different model roots. This makes sure that all applications can find all information using the same access path if the information is there. This makes application development and reuse of software much easier. This approach is the ”clou” which makes the glue layer really universal. Some examples:

- If there is a formatting engine producing paper documents for entire systems, it can also be used for instances in early process phases which are only filled partially. But the results of these intermediate steps can be presented without any further effort. Otherwise if a specific DTD is used, a specific application must be built.

- If there is an import filter for an engineering tool which usually handles files in early phases, this filter can also be used with enriched instances created in latter stages.
- If one of the writers is not capable to support the DTD in all details, the generated instances can be treated as Well Formed XML and upgraded to Valid XML using SGML/XML conversion tools.

Following this approach, there is only one DTD used in all process phases which supports a complete data model. This leads to the fact that in this DTD everything missed in one process phase should be optional. Practically speaking, any element must be optional. The DTD is still worthy because it defines a limited set of access paths to the data. The instances can still be handled as retrievable databases.

4.3.2 General approaches for readers

When dealing with database oriented SGML/XML instances all known methods of transformation respectively retrieval apply.

4.3.2.1 Source driven approach

Chapter 4.3 Using a database oriented DTD p. 14 In the source driven approach, the sequence and selection of the retrieved data is determined by the existence of the data in the instance. As explained in this is highly dependant on the process phase and the generator tool. For this reasons, a source driven approach is useful only if standardizes tasks (like adjusting the different link models etc.) are to be performed.

When using source driven readers it is essential that the sequence of information comes very close to the structure of the target system. Mostly it is easy for a tool to read its own output. But as soon as information appears which cannot be handled by a reader, special actions are necessary.

Therefore the source driven approach is not adequate in most of the cases.

4.3.2.2 Target driven approach

In the target driven approach, in any process step, the desired data structures are built on demand. This provides further independence of the capabilities of other tools in the process as long as they deliver the required information. So the use models for the SGML/XML instances are very flexible:

- The source instance can be queried to get all the information required by the tool. The data is then saved in the tool’s proprietary format and exported on request. This export in fact means to generate the SGML/XML instance from scratch. Information not supported by the tool is lost.
- can be applied.
- For migration phases, SGML/XML tools can be used to convert the data into existing proprietary import formats which then can handled according to existing processes. While doing this some advanced preprocessing can be performed (e.g. adjusting default values) which is not available in the existing environment.

4.3.3 Query language

When using a database oriented DTD, a powerful query language is essential to leverage all the power of this. Especially the following features are required.

- Multiple instances and trees must be queried. This is used for merge and lookup processes
- Information must be available from anywhere in the trees.
- Queries must be concatenated so that each object in the result is the base point for new queries.
- Any arbitrary condition must be applicable at any point
- There must be a way to keep results of subqueries for later reuse
- The result of a query can be anything from a single value up to an entire tree structure.
- It must be possible to establish auxiliary data structures for query optimization (e.g. indexes like in database systems)
- The results of subqueries are lists or sets which must be handled using set operations (union, intersection).

The following examples use the query language of MetaMorphosis (http://www.ovidius.com) to illustrate how these features are used. The first example () shows concatenated queries with nested conditions (the related instance is shown in . The result of the query is a list of short-name of all parameters with implausible values.

Finding invalid parameters

```
descendant  // consider all descendants

  [  // for which is true:

  @f-id-class==”PRM”   // Attribute f-id-class is ”PRM” AND

  child  // has a child

  [  // for which is true:

  ?prm-char   // GID is ”PRM-CHAR” and

  child[?min].data  // child with GID ”MIN”

 // is greater than one of

 // the children with GID ”TYP” or ”MAX”

  child[?typ | ?max].data  ]  ]  .  // for each of those

  child[?short-name]  // use children with GID ”SHORT-NAME”

  . data  // for each of those return PCDATA
```

The next example () shows how multiple instances are queried while data is retrieved from multiple places in each instance. The query is based on the MSRSW.DTD (see) The result of the query is all contents of all data dictionaries, in other words the united data dictionary.

Merging Data

```
source(”sim.sgm”, ”impl.sgm”)  // two sources ”sim.sgm” as well as ”impl.sgm”

  .(  // for each of those
  root,  // use root
```

```
 // as well as path to ...
  path("sw-function-spec/sw-function/sw-function-variants/sw-function-variant")).
  // for each of those

// path down to sw-data-dictionary
  path("sw-data-dictionary-spec/sw-datadictionaries/sw-data-dictionary").contents
  // for each of those return the contents
```

The last example (instance shown in , result displayed as , query explained in) shows how to retrieve all information about a parameter set and to return an entire tree. The result is in fact a tree representing a table structure similar to a CALS table. It also shows how intermediate results can be saved in variables.

MSR-Instance for parameter set

```
<PRMS>
 <PRM>
  <LONG-NAME>Long-name</LONG-NAME>
  <SHORT-NAME>sn</SHORT-NAME>
  <DESC>Das ist die Beschreibung des Parameters sn</DESC>
  <PRM-CHAR>
   <COND>
    <P>Bedingung1</P>
   </COND>
   <MIN>min-1</MIN>
   <TYP>typ-1</TYP>
   <MAX>max-1</MAX>
   <REMARK>
    <P>Das ist der Hinweis zur Bedingung 1</P>
   </REMARK>
  </PRM-CHAR>
  <PRM-CHAR>
   <COND>
    <P>Bedingung 2</P>
   </COND>
   <MIN>min 2</MIN>
<TYP>typ-2</TYP>
   <MAX>max-2</MAX>
   <UNIT>unit</UNIT>
  </PRM-CHAR>
 </PRM>
 <PRM>
<LONG-NAME>Abs-tol-Parameter</LONG-NAME>
  <SHORT-NAME>atsn</SHORT-NAME>
  <DESC>Das ist ein abs-tol-parameter</DESC>
  <PRM-CHAR>
   <ABS>abs</ABS>
   <TOL>tol</TOL>
   <UNIT>unit</UNIT>
   <REMARK>
    <P>Das ist die Anmerkung zu abstol</P>
   </REMARK>
  </PRM-CHAR>
 </PRM>
 <PRM>
<LONG-NAME>Textparameter</LONG-NAME>
  <SHORT-NAME>txt</SHORT-NAME>
  <PRM-CHAR>
   <TEXT>Das ist der text des Parameters</TEXT>
  </PRM-CHAR>
 </PRM>
</PRMS>
```

The desired result is as follows:

Table :

Result of query on parameter set

Table :

Beze-ichnung	Kürzel	Min	Typ	Max	Abs	Tol	Einh.	Hinw.
Long-name								

Table :

	- Be-din-gung 1	sn	min-1	typ-1	max-1			unit	1.
Long-name									
	- Be-din-gung 2	sn	min-2	typ-2	max-2			unit	
Abs-tol-Parameter	atsn				abs	tol			
Textpa-rame-ter	txt	Das ist der text des Pa-rame-ters							

The query to achieve this is given in . The interesting point is the fact that the presentation table is specified declaratively, its cells are populated by sub queries retrieving the desired data from the source document.

Query to retrieve a parameter set as a table

```
|st.table|(*hook:=this)  // create the table

 // keep hook in variable

  {  |st.title|node{*hook._GID | *hook.GID},  // make Title

  |st.cols|node  // define columns

  {  |st.col, @width:=”4cm”|node,
     |st.col, @width:=”2cm”|node,
     |st.col, @width:=”1.5cm”|node,
     |st.col, @width:=”1.5cm”|node,
     |st.col, @width:=”1.5cm”|node,
     |st.col, @width:=”1.5cm”|node,
     |st.col, @width:=”1.5cm”|node,
     |st.col, @width:=”2cm”|node,
     |st.col, @width:=”1.5cm”|node  },
     |st.thead|node  // Table head

  {  |st.row|node
    {|st.entry|node{”Bezeichnung”},
     |st.entry|node{”Kürzel”},
     |st.entry|node{”Min.”},
     |st.entry|node{”Typ.”},
     |st.entry|node{”Max.”},
     |st.entry|node{”Abs.”},
     |st.entry|node{”Tol.”},
     |st.entry|node{”Einheit”},
     |st.entry|node{”Hinw.”}
     }
   },
 |st.tbody|*hook  // Table body, hooked to parameter-table
  {  |st.row|child  .child[?prm-char]  // a table row for each prm-char
   {(*curent:=this).null  // save actual prm-char in variable
     |st.entry,  // an entry for title
     @hjust:=”l”|node{(*current.parent.child[?long-name].contents //use cousin long-name
    |  // or
     *current.parent.gid),  // or GID of table element
      |q.pb|that.child[?cond]  .contents  // add cond
      {” - ”,contents}},
      |st.entry|node{*current.left[?short-name].contents},  // an entry for short-name
      (|st.entry,  // spanned entry for text
       @w:=”9cm”,@hjust:=”l”,  @hspan:=6|child[?text]{contents}
       |    // or individual entries for
       (|st.entry|node{that.child[?min].contents},  // min
        |st.entry|node{that.child[?typ].contents},  // typ
        |st.entry|node{that.child[?max].contents},  // max
        |st.entry|node{that.child[?abs].contents},  // abs
        |st.entry|node{that.child[?tol].contents},  // tol
        |st.entry|node{that.child[?unit].contents}  // unit
        )
       ),
      |st.entry|node  // entry for remark
      {  that.child[?remark]  .(count(parent.parent  // number counter for remark
          .left.child[?prm-char]  // count grand cousins
          .child[?remark]  //
         ,parent  .left[?prm-char]  // and cousins
          .child[?remark]  )+1||”.”  )  |null  // leave cell blank if
          // there is no remark
       }
      }
     }
    }
```

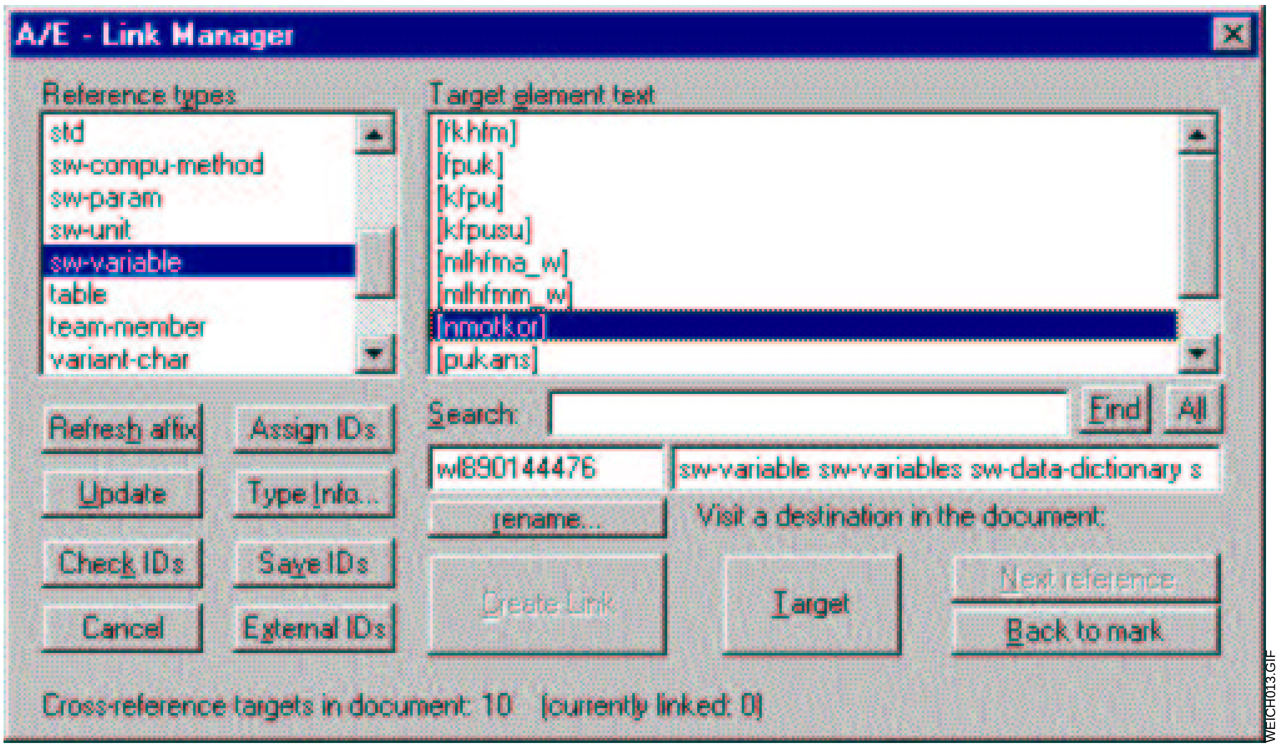
4.3.4 Authoring in SGML/XML directly

Obviously it is highly recommended that instances for database oriented DTDs are generated automatically by tools rather than edited manually. But in some circumstances it is required to do manual authoring (e.g. if no generator is available, for debug purposes etc.). If some basic funtionality is available in the SGML editor, direct authoring becomes practicable (but still not ideal).

4.3.4.1 Link manager

As discussed above () in a database oriented DTD there are links with well defined semantics. There are many different semantics so there are many different link types. For example if a function is importing a variable, the related link should not end at a company or at a measurement unit. For manual authoring, there must be means to follow these different link types. An example of a link manager for the MSRSW.DTD is shown in in the figure below (). In the left pane of this dialog, all link types are displayed. The available objects belonging to the selected link type (in this case variable) are shown in the right pane.

Link manager



wl890144476

sw-variable sw-variables sw-data-dictionary s

Cross-reference targets in document: 10 (currently linked: 0)

Figure :

Such a link manager is useful for establishing links as well as for navigation in the instance especially for quickly locating, previewing and editing of available objects. This link manager also supports cross instance linking.

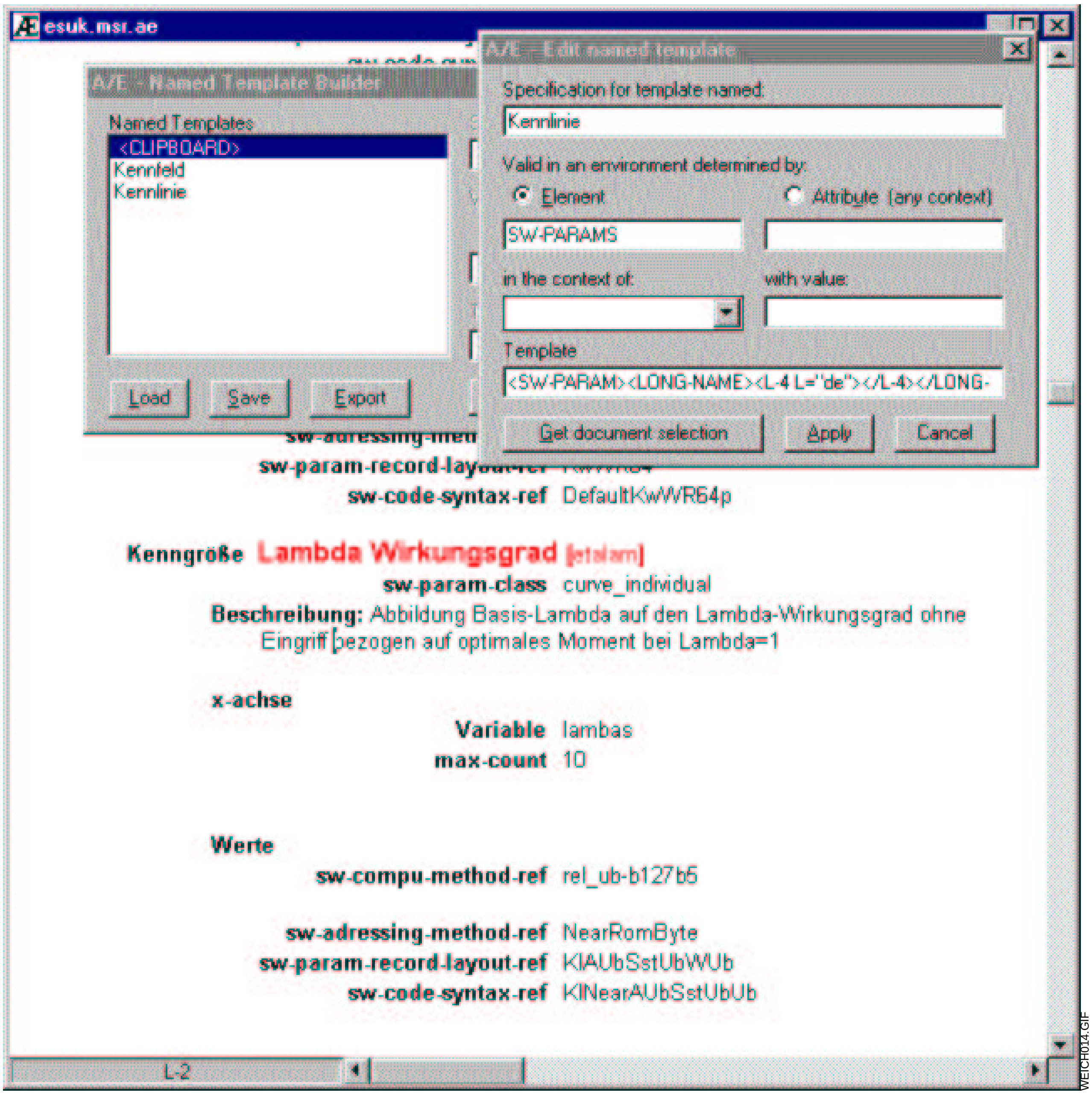
4.3.4.2 Templates

Database oriented DTDs tend to be very complex. There are a lot of elements with well defined semantics. On the other hand there are object classes as shown in . To support capturing such complex strictures, a template strategy can be used with the following features.

- Provide the supported classes as templates.
- Identify an appropriate insertion point by analyzing the template and the DTD. This is one of the most important topics, because it reduces the need for the author to know the DTD in all its structural details.
- Provide means for the author to define templates on the fly. This allows introduction of new classes by the engineer without any support organization.

An Example of such a link manager is shown in the following figure ().

Inserting and defining templates



esuk.msr.ae

A/E - Named Template Builder

Named Templates
<CLIPBOARD>
Kennfeld
Kennlinie

Load | Save | Export

A/E - Edit named template

Specification for template named: Kennlinie

Valid in an environment determined by:
(● Element) ○ Attribute (any context)

Element: SW-PARAMS

in the context of:

with value:

Template: <SW-PARAM><LONG-NAME><L-4 L="de"></L-4></LONG-

Get document selection | Apply | Cancel

sw-param-record-lay
sw-code-syntax-ref DefaultKwWR64p

Kenngröße **Lambda Wirkungsgrad** [etalam]
sw-param-class curve_individual
Beschreibung: Abbildung Basis-Lambda auf den Lambda-Wirkungsgrad ohne Eingriff bezogen auf optimales Moment bei Lambda=1

x-achse
Variable lambas
max-count 10

Werte
sw-compu-method-ref rel_ub-b127b5
sw-adressing-method-ref NearRomByte
sw-param-record-layout-ref KlAUbSstUbWUb
sw-code-syntax-ref KlNearAUbSstUbUb

L-2

WEICH014.GIF

Figure :

,

4.3.5 Further experience with database oriented DTDs

I will communicate some experience with database oriented DTDs as they appeared in my work using MSR DTDs as well as Bosch specific ones. These statements don't claim to be generally true in all circumstances.

- The semantics of the DTD must be defined as strictly as possible. This is the hardest part of the job especially since SGML/XML is pretty weak in terms of data types and formal methods for defining semantics are still missing. Mostly the final semantics is defined during the pilot implementations. It is essential that every definition is documented as exactly as possible.
- As long as instances are authored or viewed using SGML/XML tools it is better to use elements for terminal data rather than attributes. The general handling is more straightforward. The data typing provided by attributes is insufficient anyhow.
- *SGML/XML is pretty verbose (terseness is not a design goal for XML*). This causes instances to be very big. For example the data dictionary of an engine management system is about 10 MB where markup is 80% of this. The file size could be reduced substantially by using terse element names (but then the DTD is much harder to understand). Another option is to use empty end tags (which is not allowed in XML but SGML is still there :-). The file size can be a problem especially in pure tree oriented systems.
- If processing is performed using hardwired tools it is often easier to use existing data for linking purposes instaead of introducing artificial access keys (ID/IDREF). In some cases there is no need to really follow the link because the link recreates itself by the synergy of the natural names.
- It is highly recommended to define the data model as completely as possible. Any compromises must be fixed later with much more effort.
- It makes sense to keep the DTD flexible in terms of the possible containers. This makes it much easier for the engineering tools to generate valid instances. The information can very easily be rearranged by SGML/XML tools if one of the readers has different requirements. An example is given above ().

5 The document base in the MSR consortium

In the German automotive industry, there is a consortium of car manufacturers and automotive equipment suppliers made up of the companies BMW AG, Daimler Benz AG, Porsche AG, Volkswagen AG, Robert Bosch GmbH, Hella KG Hueck Co. and Siemens AG.

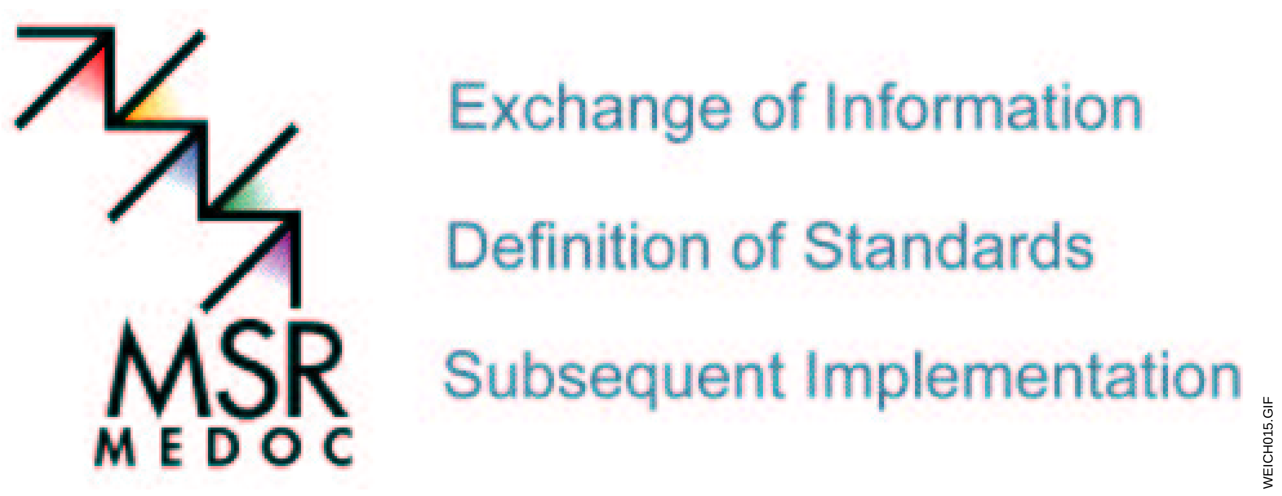


Figure :

MSR supports the joint development of car manufacturers and their electronic control system suppliers by enabling process synchronization and proper management of information exchange.

Within the work of MSR a whole set of related DTDs are developed and used in projects:

MSRSYS.DTD

This database oriented DTD is used to describe an to specify entire control systems with all its mechanical and electrical components. This DTD provides detailed structures for project data, architectures, signals, connections, electrical, properties, mechanical properties. This DTD was successfully used in multiple projects within a big automotive manufacturer and his suppliers.

MSRNET.DTD

This highly database oriented DTD is used to transport information about networks in the vehicle. It allows to specify the information transported on the network, how this information is packed into messages, the network topology etc. This DTD is implemented as import/export facility in the CAN tools from Vector Informatik (http://www.vector-informatik.de). This DTD is actually used by another automotive manufacturer to support databases for the next generation of automotive networks.

MSRSW.DTD

This highly database oriented DTD is used to specify the software for ECU s. The examples in this presentation were taken from applications with MSRSW.DTD. This DTD is successfully used in projects across companies as well as across different business units in one company while each partner uses different engineering tools and SGML/XML tools. This DTD is supported in the system design tools ASCET (http://www.etas.de)

In one case a single document of thousands of pages as well as on-line documentation is generated using MSRSW.DTD. Parts are contributed by the customer as well as Bosch engineers and merged on SGML/XML level. In the same project, data definitions are transferred in the same instance and introduced in the software build process.

MSRFMEA.DTD

. This data model was generalized to eliminate the tool specifics.

MSRREP.DTD

This presentation oriented DTD is used to write reports and specifications not yet covered by the other DTDs. All the MSR documentation is done using MSRREP.DTD.

These MSR DTDs while implementing specific data models for their domain have the same basic principles:

- Chapter 5.1 Natural addressing in MSR p. 26 Chapter 4.2.2 Object oriented models p. 12 Chapter 4.3.4.1 Link manager p. 21 Same link model for all DTDs (shown in). This allows instances of the MSR DTDs to be linked together. The link types (see and) are unique across the entire MSR DTD system.
- Same basic models (free style text sections, parameters, architectures)
- Same configuration capabilities
- Same subclassing methods using ...class elements.
- Elements to keep administrative data. These can be used for example to implement version control even for subtrees in one instance. This is useful if an instance is built of fragments delivered by different project partners.
- Same generic approaches for constructing the DTDs like naming conventions, architectures etc.

5.1 Natural addressing in MSR

Chapter 4.3.1 Supporting the entire life cycle p. 15 As already mentioned (e.g.), MSR has a specific method of adressing linked resources which exists in parallel to the methods provided by SGML/XML. This method is based on the following principles:

- All linked targets have a child (short-name) receiving a name for that object which can be used to construct natural address. In most cases this name is already there, e.g. a product name, a variable name etc. The engineers (e.g. a programmer) has these natural objects in mind when dealing with such objects.
- The natural hierarchy of objects is reflected in the MSR instances. For example, a team member is part of a company. Therefore the names for team members must be unique within a company. In a cross company project, there can be team members with the same name. So the name of the company is used as a prefix for the team member (example see).
- As there is an object hierarchy, there is also a hierarchy of name spaces.
- The name spaces for the objects is denoted in the DTD using fixed attributes.

- Objects are identified using the concatenation of the short-names of the object hierarchy. This concatenation can be shortcut by omitting the name of common ancestors. This represents a relative address which can be resolved by taking in to account the nearest ancestor of the linking element belonging to the same name space.
- Instances, of course, also have a short-name providing a unique name. So if the natural address starts with ”/”, it is taken absolute and can be interpreted as a cross instance link (e.g. port-ref/ecu/cpu/tmot/port-ref).
- All MSR DTDs have a branch of HyTime namelocs. This is used to reflect the ”natural linking” using SGML/XML methods.

Natural adressing in MSR

companies company short-namebosch/short-name

... team-members team-member short-namewl/short-name

long-nameBernhard Weichel/long-name ... company short-namefoo/short-name ... team-members team-member short-namewl/short-name

long-nameGerhard Walter/long-name /companies modifications modifcation team-member-refbosch/wl/team-member-ref

6 Desires for SGML/XML

While using SGML/XML in applications described above some desires came up which would make things easier:

- More means for DTD configuration in the SGML schema language. This means especially fully explicit occurrence operators. There should be an explicit operator for ”required” (e.g. ”!”). Furthermore there should be an operator for ”not allowed here” (e.g. ”ˆ”). This would enable the DTD designer to use parameter entities to configure the DTD structure by manipulating occurrences.
- The content model of an element should be definable in respect of the actual context. There must be a meta DTD, defining the superset of all possible models, so that processors can rely on this. But in the actual context there must be limitations. These limitations cover mostly the occurrences. It could be done using architectural forms but then we must use different element names.
- Integrated documentation facilities. Actually a DTD is written and commented. But the comments are pretty weak and DTD syntax is not understandable by users who want to use the DTD in an authoring system. This documentation must cover the semantics, possible values etc. The content models must be described individually since the semantics of an element is content dependant.
- More means to express links using existing data. The locator attribute for example, could be the data of the locator element. Multiple nested link types can be established which reflect the mutual object hierarchy.

7 Conclusion

As we see, SGML/XML offers a great potential to establish a glue layer in complex engineering processes. This layer can be used to harmonize different data structures of the engineering tools, to integrate information from many sources and to establish a long term available base of document and data assets. In order to achieve this, a careful DTD design and a well founded implementation strategy is required. There are systems on the market that can support the implementation of database oriented DTDs. But there is still a significant effort remaining to get it to work.

There are public DTDs developed in the MSR consortium which have been designed with this approach in mind and can be used to build a document base. These DTDs can be used in pilot and production projects.

This style of DTDs is possible but not yet in the mainstream focus of SGML/XML which is oriented towards traditional documents. But this situation is going to change.

It is my intention to contribute to this process and to encourage others to using the potential provided by SGML/XML.

Documentadministration

Versions Overview

Document Part	Date	Editor			
		Company	**Version**	**State**	**Remarks**
From page 3	1 RD 2002-11-03 Changes 1	Bernhard Weichel, Robert Bosch GmbH			
		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_cl_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\xi1052\new_seite\download\Literature\SGML_Europe_98\sgml_europe_98.xml

MetaMorphosis-Version
3.2

Form Version
2.0 (MetaPage)

Date
21/05/2002 08:05:12