—— 4 ——
FMV Grund-DTD overview

In this section, a technical overview of FMV Grund-DTD is given. It is divided in three conceptual parts:

- *The principles for design and development that has been used in the project*
- *The general structure of the DTD, based on information types*
- *Highlights on specific areas of the DTD*

Contents

Design and development principles

FMV Grund-DTD is built on the following basic design principles:

- The information will be stored in information modules, identifying the maintenance object they belong to. An information module is limited to describe one task for one maintenance object, as far as possible. A higher level information module must not include information that can be placed somewhere in its substructure.
  The DTD shall describe the structure of such information modules.
- The DTD shall not cover a materiel system breakdown structure, or the relations between objects in such a structure. The DTD must however assume that such structures (both physical and functional) and relations exists.
- The development should be based on ISO 8879 SGML and ISO 10744 HyTime, and will exploit areas of interest in MIL-D-87269 IETMDB and other DTD's.
- The Grund-DTD should focus on the storage of various information modules at FMV, in order to achieve a DTD as neutral as possible. However, the DTD must incorporate identified functions (e.g. attributes) required for screen presentation, paper printout, and production.
- A declared element in the DTD should always have the same attributes and content (sub-structure), in order to avoid confusion and misunderstandings. A possible variant should be given a new name, and be treated as a separate element.
- Elements and attributes shall be given Swedish names that indicate the supposed use and relationship as far as possible, except for the table structure which is based on SGML Open Technical Resolution TR 9503:1995, Exchange Table model DTD module.
  [Due to software limitations, Swedish characters Å,Ä, and Ö in upper and lower case is however forbidden in names (i.e. entity, element and attribute names, as well as attribute declared values name token groups).]

Information types in the DTD

FMV Grund-DTD consists of 22 types of information substructures that has been identified as necessary for operation and maintenance of materiel systems.

The 22 substructures can be classified into the following categories:

- descriptive information
- procedural information
- data
- assembled information
- administrative information
- linking information
- information products

Descriptive information

The descriptive substructures are called:

- *System* (system description)
- *konstruktion* (construction)
- *funktion* (function)
- *konstr.funkt* (construction and function)

Furthermore, *foreskrift* (regulation) is a mix of descriptive and procedural information.

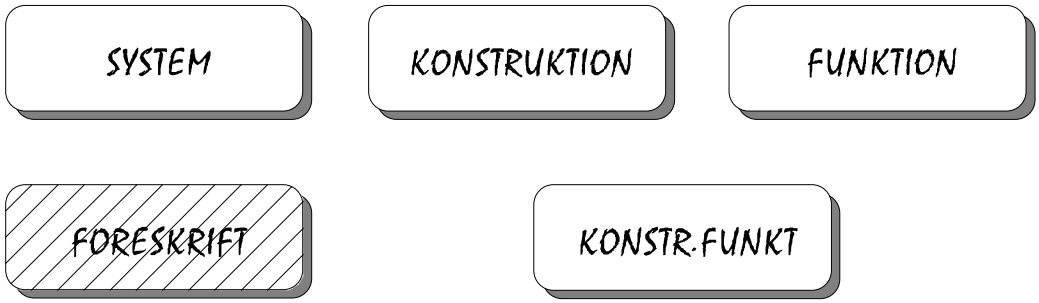

Figure 13. DTD modules for descriptive information types

The descriptive information is built up of sections (*avsnitt*) containing a title and ordinary text components. Sections can be placed inside each other, and are thereby regarded as subsections in a hierarchical fashion. There is no limit on how many subsections one can have, but if the breakdown of the materiel system is fairly thorough, there should rarely be need for more than two or three levels.

Procedural information

There are a number of procedural substructures, and some of them have attributes that further classifies them:

- *handhavande* (operation)
- *felsokning* (fault finding)
- *felavhjalpning* (corrective maintenance)
  - *-atgard.typ* (task type)
- *period.uh* (periodic maintenance)
  - *anvstatus* (operational status)
  - *tidsintervall* (time interval)
  - *driftintervall* (operational interval)
  - *driftlage* (usage status)
- *uh.atgard* (maintenance task)
  - *uhatgard.typ* (maintenance task type)
- *kontroll* (test)
  - *kontrolltyp* (test type)
- *oversyn* (overhaul)
  - *tidsintervall* (time interval)
  - *driftintervall* (operational interval)
- *renovering* (renovation)
- *modifiering* (modification)

- *forradsstallning* (storage instructions)
  - *forradslage* (storage status)
  - *utlamningslage* (hand-out status)

Furthermore, *foreskrift* (regulation) is a mix of descriptive and procedural information.

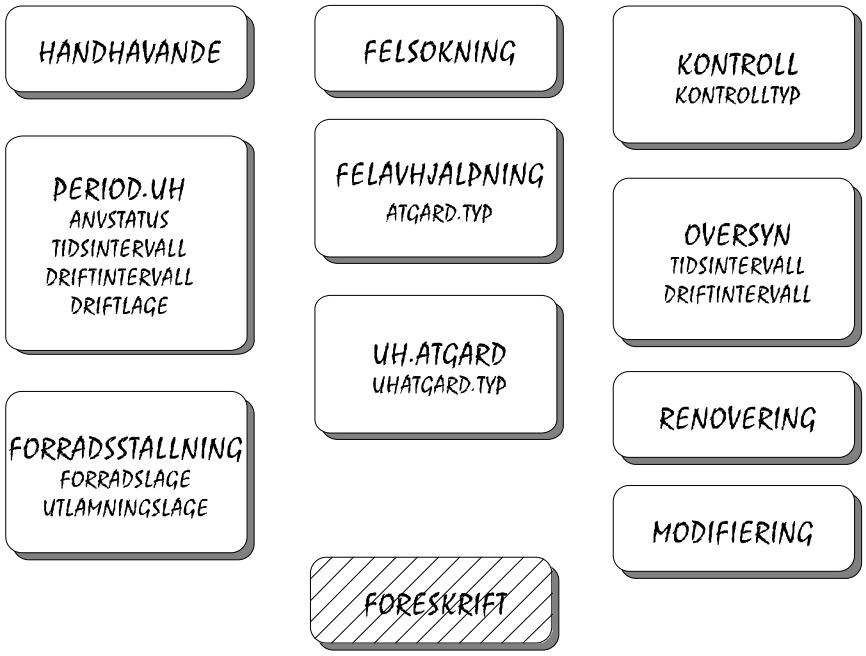

Figure 14. DTD modules for procedural information types

Procedural information is basically built up of the procedure itself, and its preconditions and postconditions. The basic idea with a procedure is that it gives information on **what** needs to be done. **How** it is to be done is described elsewhere (e.g. in a *uh.atgard*, maintenance task), or maybe as a procedure on a lower level (only if this is the only use of that procedure).

The quite large number of substructures for procedural information stems from the fact that the top structure differs somewhat. The only exception to this is *renovering* and *modifiering*, which have identical structures but are to their nature different types of information, so the design group therefore decided to use two substructures.

Data

The following substructures are of the type data:

- *tekngrunddata* (general technical data)
- *underhallsdata* (common maintenance data)

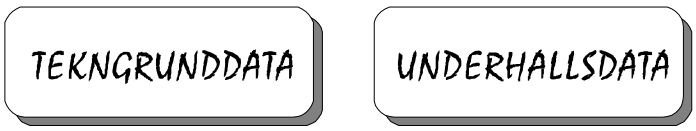

Figure 15. DTD modules for "data" information types

The data type substructures can be regarded as mirrors of databases. In the absence of one, general database solution for all information of this type, those substructures have been created to enable retrieval and re-use of information that might or might not be stored in a database.

If a real database, containing this type of information (no less), is available to the presentation system, all links to these modules may be redirected into the database and the true source information directly.

Note: This may pose legal problems in some cases, since databases do not use version control in the same sense as documentation.

Assembled information

Information modules that consists of one or more illustrations that are described by textual data, often found in databases, are called assembled information. The following substructures are of the type assembled information:

- *reservdelar* (spare parts)
- *tillbehor* (accessories)
- *schema* (diagram)
  - *schematyp* (diagram type)

Figure 16. DTD modules for assembled information

The common denominator for those modules is that the illustration and the text has a "cross reference" in the call-out system, which is specific to each illustration (or materiel system). Those information modules tend to be publication specific, but the combination of an illustration and text is very valuable to end users, and there is no other way of structuring this information at this detailed level.

Administrative information

The module *admindata* (administrative data) is used to supply administrative information regarding the other information modules, i.e. status, changes, responsible persons etc.

The idea is that for a delivery, one module of the type *admindata* is assembled, containing information on all other information modules in the delivery. In a storage environment, as well as during production and in a presentation application, this type of information is supposed to be handled by an administrative system, and there might not be any need of keeping an *admindata* module at that time.

The structure of the *admindata* module is heavily based on similar structures in FMV Base DTD, and earlier version of the Grund-DTD used e.g. in the FVSDUP project. The FVSDUP project was the only project available at FMV in which information of this type could be found. When other projects employs digital delivery, we suspect changes to this module to be required.

Linking information

The linking substructure HyTime addresses, *hytime.adresser*, is supposed to contain all name location elements. Links between the information module and other information modules are created by using a reference element, *hanvisning*, which points to a name location element located in *hytime.adresser*. All references are encoded with HyTime, ISO 10744, and are contextual links, i.e. the start anchor is bound to its context.

Information products

To be able to produce documents, publications, screen applications, etc. from a base of information modules, a means of putting them together in a specific hierarchy must be supplied. The module *infoproduktmodul*, information product module, defines the information used in an information product by pointing at the information modules.

Given the information product module, and the information modules it assembles, a presentation format can be applied and, possibly after some specific processing to re-organise the information slightly, a document can be obtained.

The *infoproduktmodul* is to its nature very different to the information modules described by the DTD. Its sole purpose is to organise other information modules in some hierarchical fashion, for a specific purpose, e.g. paper document, electronic presentation, etc.

Information product modules may also point at other information product modules. Thereby hierarchical structures of information product modules may be set up, e.g. to organise chapters of a book separately.

Deciding the level of granularity

Even if the DTD is created against what has previously been defined as an information module, the size or volume of an information module is dynamic. It depends on how far the materiel system has been broken down, i.e. system, sub-system, assembly, sub-assembly, component, line replaceable unit (LRU), etc.

If a logistic support analysis (LSA) has been conducted, the breakdown is probably quite detailed, but smaller systems may not have a defined breakdown structure at all.

Note: It is important that also the operational tasks are covered by the LSA, in order to supply a structure (or objects) for technical information purposes.

And even given a detailed breakdown structure, what's the decision for using information fragments for version and variant control? What's the best level of "smallest information chunk" for the defence? And what's the best level for each of the other parties in the deal?

The decision is probably specific for each project, but some guidance may be given.

Technical information that is going to be integrated with other information during its lifetime, in a product model or otherwise, must be broken down. Preferably this is done in a standardised way, e.g. through an LSA. The breakdown must be fairly detailed, if the integration of information shall have high functionality, probably down to LRU. The information modules for such a system will be very small, and this is the concept of information modules for which the DTD has been tailored.

Technical information that will be updated frequently in the future will gain of using information fragments for version and variant control. Most information need to be updated frequently, but the cost of updates makes it often impossible to update. Some materiel systems documentation has never been updated, and the documentation is totally wrong today. One must be conscious of that the production cost of using information fragments is much higher (at least today), since it requires advanced systems to handle the information, but that cost must be balanced against the cost of updates.

Tables

Most traditional tables in defence documentation will be handled in another way than with the "table" code. Fault finding, technical data, spare part lists etc. are all handled with true content coding, without mentioning the presentation form "table".

The table structure in FMV Grund-DTD is based on SGML Open Technical Resolution TR 9503:1995, Exchange Table model DTD module. A few changes has been made to the table structure:

- The element *spanspec* is added to the table structure.
- The attribute spanname is added to the *entry* element.
- The attribute tgroupstyle is added to the *tgroup* element.

A cell may contain text, notes *anm*, figures *bild* and verbatim text *tecken.for.tecken*. Verbatim text means that all line breaks and tabs are preserved, and it could therefore be used where line breaks and spaces are significant. The verbatim text is however only available in table cells.

Blocking of information

The DTD contains a number of elements that are used to block, or group information:

- *beskr.block*
- *foreskr.block*
- *grunddata.block*
- *uhdata.block*
- *moment.block*
- *sekv.block*
- *sekv.block.sub*

These elements are used to embrace elements that belong together in some way, e.g. text paragraphs and a figure, or a note and the text paragraph it belongs to. Blocks do not add another level of headings, it simply holds together information pieces. It is not possible to group non-contiguous elements with the DTD, i.e. all elements in between the first and last element of the block are a part of the block.

Blocks in procedural information, *moment.block*, *sekv.block*, and *sekv.block.sub*, must be used to enter warnings and cautions, *varn* and *obs*, inside a procedure. The warning must belong to some actions, steps, and the block indicates where the warning is valid (a warning at the beginning is valid throughout the entire procedure).

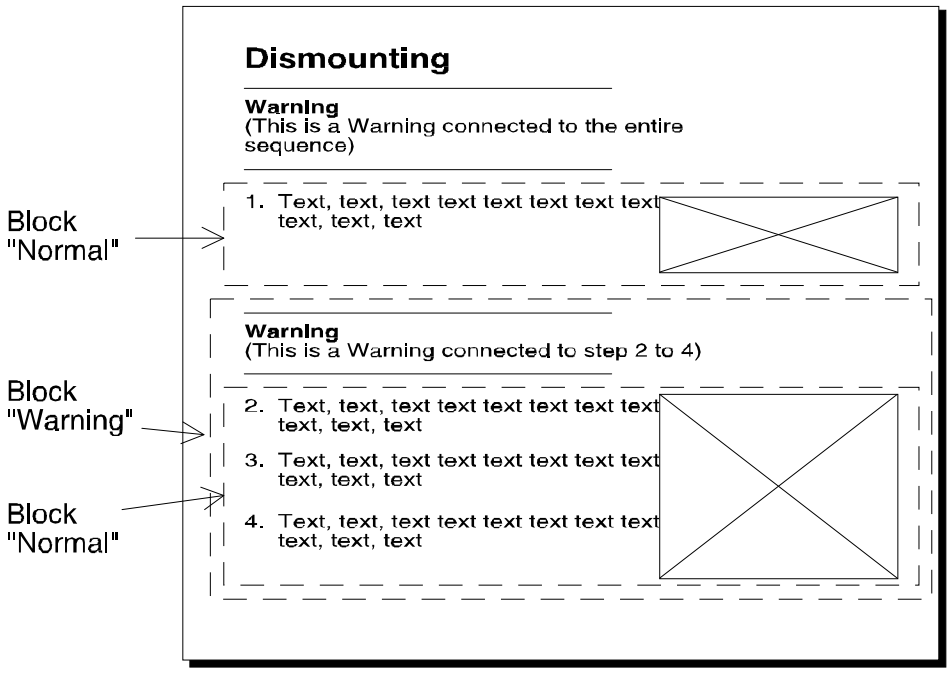

Figure 17. Example of a block in procedural information

HyTime and links

HyTime introduction

HyTime can be conceived as an extension to SGML. In FMV Grund-DTD, HyTime gives the possibility to create links between information modules in a standardised way.

HyTime consists of six modules that may be used in an application:

- **Base module**
- **Location address module**
- **Hyperlinks module**

- Measurement module
- Scheduling module
- Rendition module

In FMV Grund-DTD three of those are used: Base, location address and hyperlinks modules. Other modules of HyTime will probably be incorporated in future releases.

The use of HyTime and its modules are declared in HyTime declarations (see the end of file FMVGRUND.DCL). In every HyTime "document", i.e. all modules containing HyTime syntax, the HyTime declarations must be present to initiate HyTime processing.

HyTime defines a number of architectural forms, that can be used in SGML DTD's such as FMV Grund-DTD. HyTime does not define elements, but by declaring elements in your DTD as being conformant to a HyTime architectural form, it is possible to initiate HyTime processing on that element. An architectural form could be described as a meta-element.

Note: Neither SGML nor HyTime defines any aspects of presentation. The presentation is defined by the application that process and presents the SGML encoded information, and what might be suggested or implied by the codes can be overridden.

A simple link

A link in HyTime consists of several elements, and works similar to pointers in a programming language. HyTime defines two types of links, or two types of architectural forms for links, independent link and contextual. The contextual link is placed at one of its anchors (an anchor is the point where a HyTime link ends). The independent links position is independent of its anchors.

In FMV Grund-DTD we use only the contextual link, and further discussions regard only this type of link.

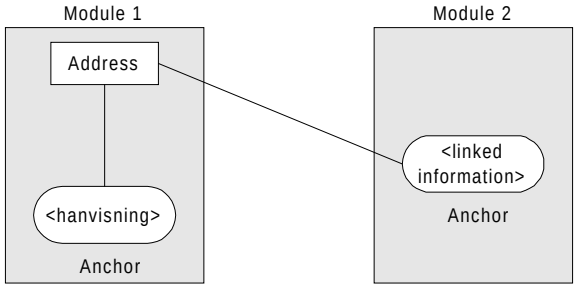

Figure 18. A simplified illustration of a HyTime contextual link

The link in the example above consists of the following elements (or architectural forms):

- The link element (HyTime architectural form *clink*) is the core of the link. It contains attributes that defines the link. One of those attributes points to the address element.
- Address consists of a name location element (architectural form *nameloc*), which in turn contains a name list element (*nmlist*).The name list element identifies module 2, and the anchor within it.

A link can be traversed in both directions, even if one of them is the "pre-defined" direction.

A *nameloc* may contain several *nmlists*, and a *nmlist* may point at another *nameloc*. In this way, very complicated and hierarchical links may be created.

Links in FMV Grund-DTD

All relations between modules in FMV Grund-DTD are encoded with HyTime. All the HyTime elements (that constitutes the link itself) are stored in the information modules.

Reference link
A reference link is a link from one place to one or many other places. Any type of information can be referenced, a section, figure or table in the same module, another module or part of another module, a document or publication that is not part of the same materiel system, in digital form or on paper. If the linked target is in digital form and accessible by the application, the link may be resolved by the application. If it is not, the location of the target information (or other information about the target) may be presented instead. In such a case, the application must be able to deal with possible HyTime errors due to the fact that the link is not fully resolvable.

The information that is referenced, the target information, does not belong to the module, but complements it.

In a paper presentation, the target information is not printed at the place of the reference. Instead, the reference is presented as e.g. "see section 4.2" or "see further 'Safety regulations for welding', TO1234-56".

In a computer based application, the user must interact with the application to see the target information, e.g. by clicking on a button or "hot spot".

In FMV Grund-DTD there is one link element of this type: *hanvisning*.

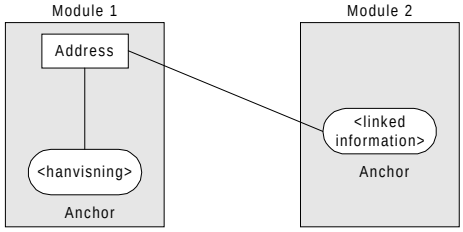

Figure 19. References into and to other modules

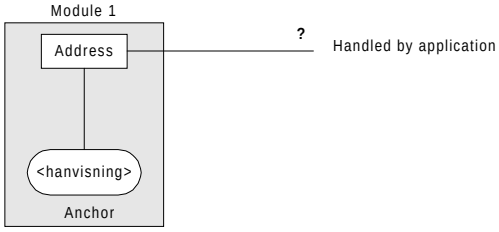

Figure 20. References to inaccessible documents

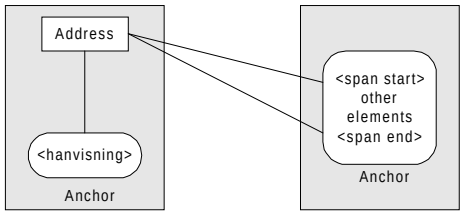

Figure 21. References to interval (span)

Administrative data link and object belongings

A link of the type administrative data (meta-data) describes the relation between information modules and their administrative data.

This link type requires that each information module has its own *reference* element, *adm.lank*, and that each administrative data entry has its own reference, *modul.lank*, back to the module.

An object is assumed to have a unique id, possibly stored in a database somewhere. The element *objekt* is the interface to the outer world, and contains all information necessary for an application to establish a query or connection with other information connected to the object, e.g. product data or logistic data in a database.

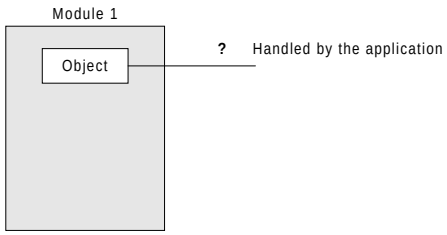

Figure 22. Link between module and object

The *adm.lank* is the connection between an information module and its administrative data.

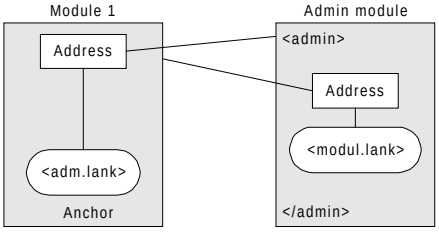

Figure 23. Link between Info module and Admin data

Versions and variants

When a new version or variant of an object (e.g. a changed component or exchangeable components) is created, it is assumed that the new version is given a new unique identifier. In FMV Grund-DTD this would be reflected as a new element of the type *objekt*. Thereby, information modules that are applicable for the new version must be updated with the new *objekt* element, and there is no further need to keep track of versions and variants of objects.

However, there is a need for version and variant management for information modules, regardless of objects.

From a linguistic perspective it is difficult to define what is a version and what is a variant. In FMV Grund-DTD we have decided to regard version as a superset of variants, i.e. a version of an information module can have different variants.

Information module versions

In FMV Grund-DTD, the version of an information module depends on its production status. The version is controlled through the attribute *modulstatus* (on the element *fmvgrund*) which has five recommended values:

- planned (*planerad*)
- work issue (*arbetsutg*)
- approved (*godkand*)
- confirmed and distributed (*faststalld*)
- revoked (*upphavd*)

Information module variants

Variants of information modules may exist if e.g. the information is tailored towards different target groups, i.e. operator versus technician. There are four attributes which can be used to identify and separate information module variants (all on the element *fmvgrund*):

- target group (*malgrupp*)
- maintenance level (*uh.niva*)
- special conditions, both internal and external (*specforutsattn*)
- language (*sprak*)

Each of these can be given project defined attribute values, and FMV Grund-DTD gives only examples of values to be used. It is the intention to establish recommended values for these in the future, if possible.