XML-based Management Framework for CANopen Systems

Dr.-Ing. Martin Wollschlaeger

Web-based management solutions for fieldbus systems have become state of the art. A framework based on descriptions in XML (eXtensible Markup Language) allows an easy mapping of a device's management functionality to web pages used as management front-ends. In most cases, the XML representation can be used directly in the browser. Another way is an off-line generation of HTML-pages by means of scripts and style sheets.

The management framework supports several design scenarios. It is possible to generate XML descriptions from a functional decomposition of a device, or on top of application profiles and device descriptions. Using an extensively linked set of XML files, hierarchical descriptions describe the spectrum from single device up to complete systems.

Examples of a practical realization prove the feasibility of the concepts. Interfaces to the common software design process are pointed out. Problems and benefits are discussed, and further trends are highlighted.

1. Introduction

During the last years, web-based management solutions for fieldbus systems have been introduced. They allow to combine management tasks from different stages of the systems' life cycles within an unique environment, a framework [1], [2]. The framework uses web-browsers as well-known, widely accepted user-interfaces. The web-browsers act as containers hosting specialized software components, which implement the management functionality dedicated to a specific fieldbus component. The principle structure of such a web-based framework is shown in **Fig. 1** [1].

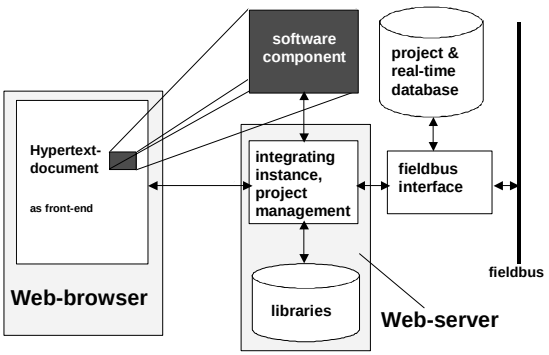

Fig. 1. Web-based management framework

In order to implement this kind of framework, different problems have to be solved. The first one is the design of a data exchange method between the fieldbus interface and the web-server as an integrating instance. Since PC-based technology has been widely adopted, concepts like OPC (OLE for Process Control) [3], based on the Component Object Model (COM) [4], can be used. Other concepts are based on Java technology.

Another implementation problem is the design of the front-end for the management functions. In web-based management systems, hypertext documents are used. They can host heterogeneous information, like graphics, forms, documentation, database front-ends, multimedia files, and so on. This allows web-based solutions to be used for presenting heterogeneous management information. In addition, there is an increasing number of management software for web-sites available, that also can be used in this framework.

Finally, the management software components themselves have to be developed and implemented. While the implementation of the components into websites can easily be done by systems' integrators or

users, the development of management components is usually performed by the vendor.

Since all these tasks require an excessive exchange of data between each other, software interfaces and file structures have been defined. These interfaces are mostly bus-system or vendor specific. Especially in heterogeneous system designs, the lack of an unique data exchange interface forces a problem for the seamless integration of tools from different vendors and different systems into a generic management framework.

2. XML as a general-purpose description language

A suitable solution for the problem described above would be the introduction of a general-purpose description. The description should have easy to access interfaces for electronic data exchange between different environments. Since this is not a new requirement in general, possible solutions already exist. They are frequently used in the description of heterogeneous documents, that can be found everywhere on the Internet. HTML has become a well-known and widely accepted method for description of heterogeneous, distributed and linked documents. However, HTML has some limitations and is not fully suitable for general-purpose descriptions.

The eXtensible Markup Language XML [5] expands the description language HTML with user-defined tags, data types and structures. Furthermore, it addresses one of the limitations in HTML – it introduces a clear separation between the data descriptions, the data themselves, and their representation in a browser (**Fig. 2**). In addition, declaring syntactical and semantical information in a separate file (Document Type Definition, DTD), allows reusing the description structure in different contexts. This provides a number of benefits when using the same XML description file for different tasks. Different views can be implemented on top of the same data. The description can be hierarchically organized. Depending on the functions to perform, the XML data can be filtered and associated to software components (controls, Java beans, etc.). The selection of the necessary information and the definition of their presentation details can be performed by means of style sheets [6]. The style sheets are part of the development of XML. In most cases, they are implemented using the extensible Style Language (XSL). By using different style sheets, context sensitive access to the XML data can be implemented. Another method for accessing XML files uses an object-oriented view of the XML tree. The Document Object Model (DOM) [7] provides platform-independent access to objects, that host dedicated parts of the XML tree. These objects implement methods and attributes for use in high-level programming languages, as well as in scripts (**Fig. 3**).

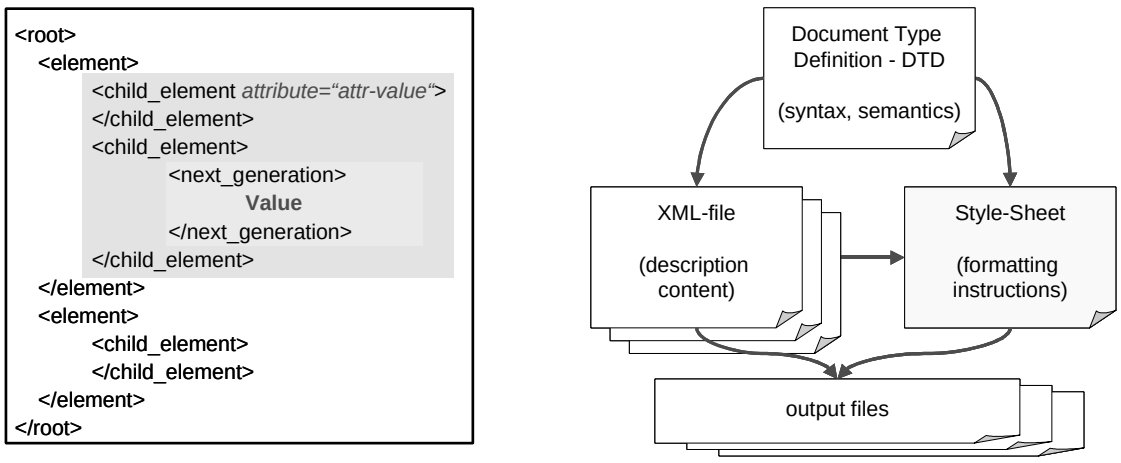

Fig. 2. Structure of an XML file (left) and XML environment with DTD and style sheet (right)

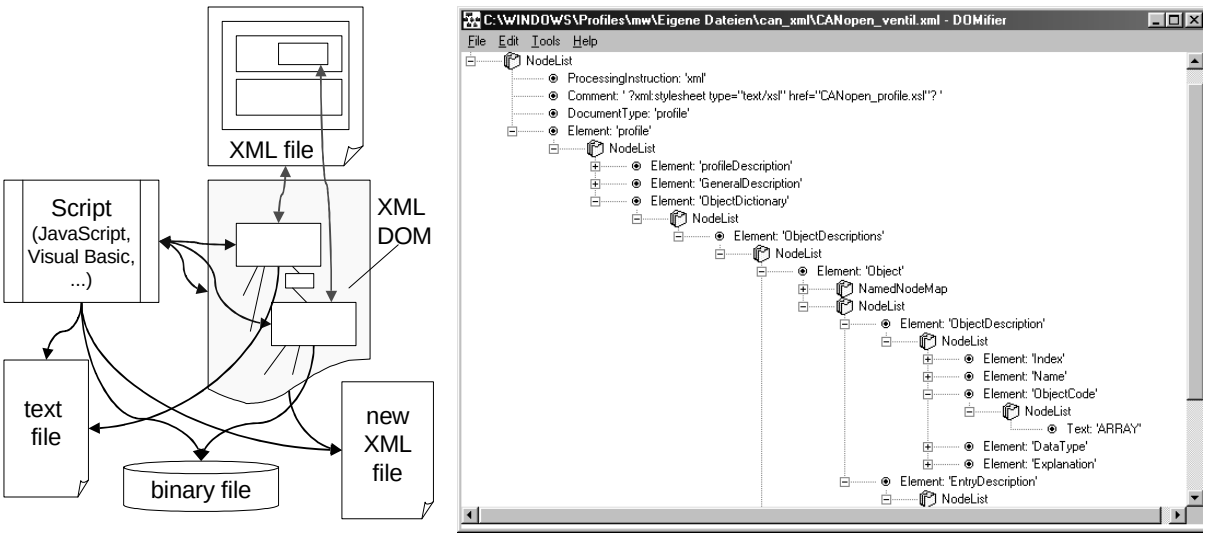

Fig. 3. An XML file and its tree representation in the Document Object Model (DOM)

3. XML-description of fieldbus components and systems

Different scenarios have to be considered in fieldbus management. They are characterized by a different focus and viewpoint, different environment, different stages of the systems' life cycles and so on. Typical scenarios are a topological view focusing on a system's and devices' topologies, or a functional view focusing on the automation and control application of the complete system. These different scenarios lead to different parameters that have to be managed, depending on the scenario. In other words, these parameters and the associated management functions are context-depending. It is very important, that the descriptions of fieldbus components and systems consider the context-depending parameters. Currently used electronic device descriptions normally do not recognize the context-depending character of the described information in a sufficient manner.

Using the concepts explained above, an XML description of a fieldbus component consists of different parts. First of all, appropriate fieldbus-related syntax and semantics have to be defined. This includes tags and attributes, their sequence, data types, and so on. By defining these, a dedicated content model for fieldbus-related descriptions is created. It contains a set of document type definitions, schemas, data type descriptions, templates, linking definitions, and transformation rules. By combining this information, a namespace for fieldbus related descriptions is set up (**Fig. 4**). In order to achieve a generic, fieldbus-independent content model, the user organizations are requested to provide XML descriptions of the fieldbus systems they are representing. Currently, there are first XML related activities defining generic concepts like [8], or creating XML representations and filters for bus-specific device description languages [9]. Furthermore, projects like NOAH [10] provide a good starting point for a harmonization of the descriptions.

An XML-based description consists of a distributed set of files, which contain specific parts of the description. They are hosted by those elements, which are described by the files. Typically, different XML files are created, each of them related to stages of the devices' and systems' life cycles. In order to prevent multiple definitions, the files are linked together. This creates a consistent set of descriptions. The users, as well as the applications using the description, usually don’t recognize the distributed character of the description.

XML provides outstanding linking capabilities, including on-line embedding of ressources and fragments of files. In addition, extended links support attributes, that consider a context. This way context depending links can be defined. The link target as well as the behaviour of the link differ depending on the context.

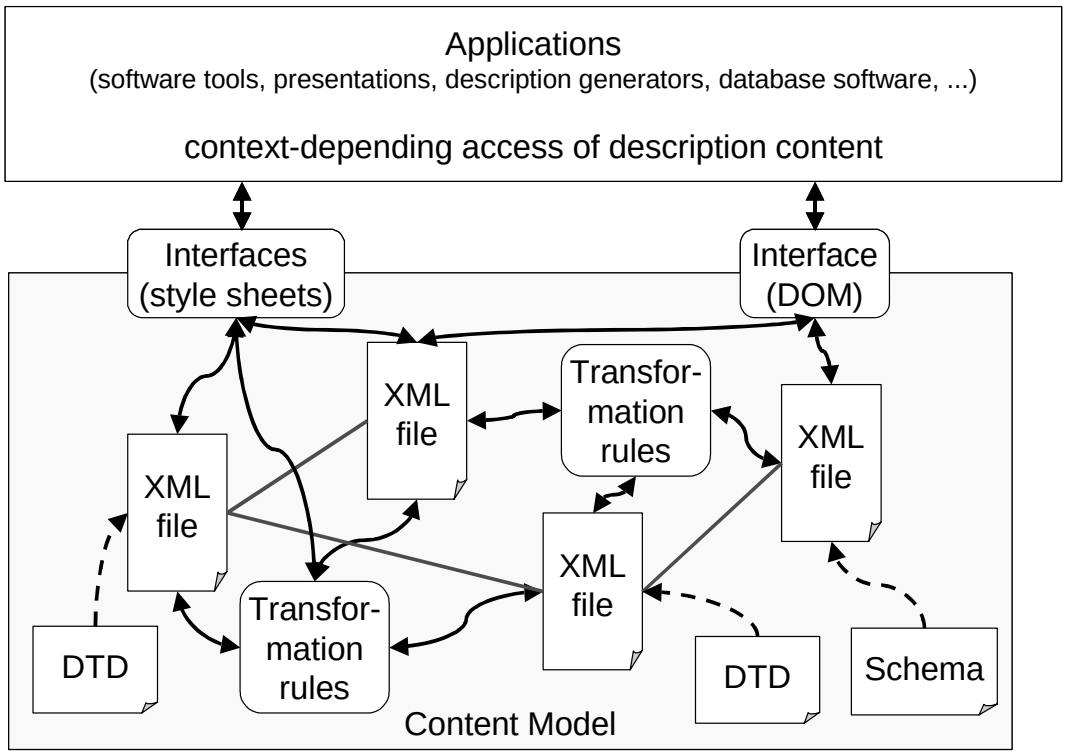

Fig. 4. A content model with structure and access methods

Although the linking principles and their definition syntax proposed by the World Wide Web Consortium (W3C) is quite stable and will shortly become a standard [11], there are currently only a few first implementations in parsers and tools. However, the focus on XML in the Internet's development lets expect really useful implementations very soon.

After having designed the content model, in a second step the XML files have to be created. Although this can be done manually, good tool support is desired to generate XML files from existing descriptions and specifications. Filters and translation software can be helpful here. In addition, there are some projects with an aim of defining interfaces between general-purpose modelling tools like the Unified Modeling Language (UML) [12] and XML-descriptions.

The third step is the design of filtering and formatting rules in order to generate the desired context-depending views to the unique XML description. This is done by scripting and declaring style sheets. While style sheets can be used for presentation tasks (generating output files in text, HTML, RTF or PDF formats), they are also suitable for transforming "input" XML files to resulting XML files, which may have different structures. So an information exchange between different contexts can be achieved.

The formatting instructions and translation specifications have to be developed once for every context and then can be re-used in combination with XML-files describing other fieldbus components and devices. Concerning web-based management concepts, HTML files can be created containing context-related data read from the XML files. This can be done off-line or "on the fly". Since built-in support for XML in browsers will be enhanced more and more, the generation "on the fly" in the client will be more promising. However, in order to support generic environments, it can be useful to create the HTML files at the server. This requires computational power at the server, which can be guaranteed in most application scenarios. So-called "thin clients", in an automation and control context, can be small HMI devices based on Windows CE or Java, that are designed to perform specific tasks of management or diagnosis. Such clients will benefit from a server-based scenario, since they are limited in computational power. As an opposite solution, a PC or

laptop with much more operating capacity could use a client-based implementation. Both scenarios are supported by the description concepts explained above.

4. Device descriptions in CANopen

In CANopen, devices and systems implement a unique communication profile. This profile defines global and system-wide information concerning application services, synchronization, data types and formats, the system's behavior, and so on. All CANopen devices and all software tools have to consider these definitions. Based on the communication profile, a number of parameters can be configured. These parameters are assigned to a management context focusing on topology and bus system's operation, but not on the application function.

In order to access CANopen devices in a convenient way, software tools require more information than the parameters from the communication profile. This additional information describes parameters, that refer to the application functionality and the behavior of the devices. Those parameters can be described in device profiles, or can be vendor-specific. Although using device profiles is the preferred way, vendor-specific parameters are often used in order to prevent opening of particular know-how.

The additional parameters and those of the communication profile, that are implemented in the device, are described by two different information stores. The first one defines a certain type of devices. This description is called the Electronic Data Sheet (EDS), and is unique for all devices of the same type. It can be used by software tools to retrieve the capabilities of a device type. EDS files can be used as templates containing specific information on the parameters, but usually not their application-related values. The second description is the Device Configuration File (DCF), which describes a single instance of a device. It is unique for every device within the system. DCF files rely on EDS files, but contain values for the parameters. These values depend on the specific configuration of a device.

The software tools for management of CANopen systems refer to both files in order to read out the capabilities and required configuration parameters of a device, and to store device-specific parameters. Other file are necessary to store information on the complete system's design. They are stored in project files with a specific data format.

The files mentioned above describe a CANopen system from a topological or device-centric viewpoint. No references to functional views are included. For example, function blocks according to IEC 61499 [13] are not considered. These function blocks are used in a functional oriented management scenario. They can be mapped to the appropriate devices.

Another information usually not considered in the descriptions, is that used for their graphical representation in SCADA systems or HMI devices. Of course, these parts of the description do not only depend on the devices, but also on the project. However, the characteristics of the devices have to be considered in order to create user-friendly interfaces.

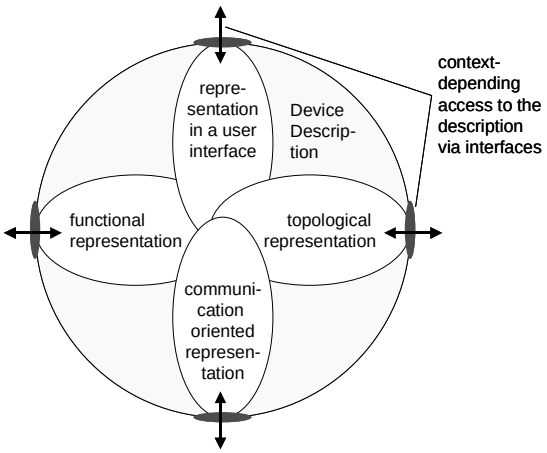

Fig. 5. Context-depending access to a description

Combining the information applicable in different contexts creates a description suitable for use in frameworks. The access to the context-specific representation is performed via an interface (**Fig. 5**). This interface encapsulates the internal structure of the information store, adds some specific information from other sources, and creates an appropriate view. Since common device descriptions are not ca-

pable of implementing the interfaces without having to write specific code, universal description methods like XML come into the focus of the description process [14]. The interfaces are implemented by means of style sheets, scripts and specific software tools as explained above.

5. XML-based management framework

Since XML allows hierarchically organized descriptions, the basic concepts suitable for describing a single device can also be used to describe a complete systems. This hierarchy shown in **Fig. 6** can also be found in the XML files. Within a general namespace, XML files are used to define several types of fieldbus devices. From an object-oriented perspective, these files describe classes of devices, with class-specific data models and management functions. They combine information from EDS files with information from other contexts.

The multiple instances of a device are described by assigning an XML file to each of them. This file contains the actual parameters of the device – similar to DCF files. Information concerning data types, units, value limits, dependability of other data etc. can be found in the class description, that is linked with the instance description. Since the file is in the same namespace, an identical semantic meaning of the XML tags is ensured. However, the information stored in the XML instance files is different from the class description. That's why the instance files have their own structure and DTD or schema. These files are automatically created from the class description, when a device is instanciated. Style sheet- and script-based transformation methods are invoked to create the instance-related XML files. The transformation rules are part of the content model, as described above.

The fieldbus installation project is internally structured into groups, subgroups, units and so on. Depending on the storage capabilities of the devices, descriptions may be implemented in a proxy object anywhere in the framework, The relations between the different files are expressed by linking. As an opposite to the Internet, in most cases a closed environment can be found in automation and control projects. This will help to reduce linking problems that are known from the Internet.

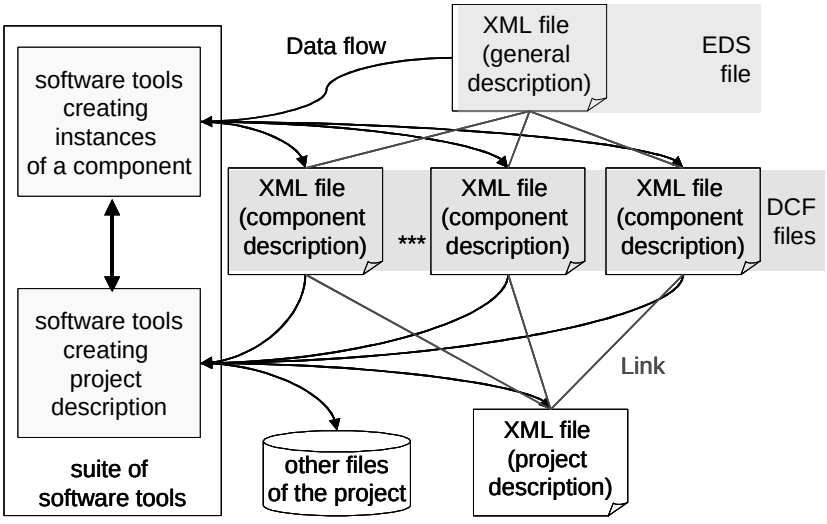

Fig. 6. Hierarchical description of a fieldbus project using XML

6. XML support for the generation of software components

The main idea of using XML descriptions was to re-use a single definition for different tasks with different scope and context. A very interesting problem is the development of software components, that map the management related functionality of a device to a user interface. Thus, the same XML files shall be used for that tasks. This requires filtering of the relevant data form the XML document and specific formatting of the results. From a methodological view, this process does not distinguish

from the one used to create HTML files - only data content and data format differ. So it is recommended to use the same methods as described above - style sheets and scripts. An XML document can be related to different style sheets, so that only the XSL references in the XML file has to be changed - according to the tasks to perform and to the changing context for using the XML description. This can easily be done by a script.

Besides style sheets, DOM-based tools can be used to generate resulting files. Data can be extracted and formatted to any output file format. This is especially important for creating input files for the software design process described above. Web-based management frameworks for fieldbusses use software components (ActiveX-Controls or Java components) for a mapping of the fieldbus related communication objects to web-pages. Such software components have to be developed for every specific device. Object descriptions are used for the development. For example, an ActiveX-control is based on a COM object definition. This definition describes objects with specific interfaces, properties and methods, defined using Object Definition Language (ODL). ODL files are text files with a specific structure and syntax, that can be derived from an appropriate XML file. The same approach can be used for the creation of Java source code, where a template can be completed with data from XML files. Both approaches use an unique (perhaps distributed, but consistent) XML description, reducing inconsistencies and errors in the software design process.

Since there are interfaces between XML and other web-related technologies, great new opportunities are possible for web-based management solutions. For example, databases can be easily accessed in order to provide the data covered by XML objects or data islands. Interactive documents can be created, containing plain text with embedded XML elements specifying input data for interactive components.

7. Conclusion

The use of a general-purpose modeling language like XML provides a number of advantages for design and application of fieldbus components. An XML-based description allows style sheets and scripts to be used for filtering and formatting tasks. This description method integrates existing solutions. It represents an upcoming standard in universal description methods, allowing platform- and vendor-independent access to data. Tools can be created to support the automatic generation of different types of output files. These files are used for creating templates, header files and object definitions for further use in software development. While the templates are generated for every class of devices, the template generator has to be implemented only once. This is an important advantage in software development. Hierarchical descriptions with extensive linking can be set up, so that object-oriented views can be implemented at description level. Automatic test and certification support can be derived from the same XML data.

Specific context-related Document Type Definitions (DTDs) can be introduced, that contain special description rules for fieldbus related data, depending on the tasks that have to be performed during the systems' life cycles. Combined with data type information, fieldbus related namespaces can be defined. These tasks should be initiated by the fieldbus systems' user organizations. It can be expected, that similar concepts will be used for a modeling of application functions. First steps can be recognized using XML as a description method and universal exchange format for Function Blocks according to IEC 61499 [13]. This will enable a functional view to distributed automation and control systems. In general, the errors introduced by re-defining same objects within different contexts, are reduced dramatically. For the vendor, all this will help to reduce design costs. The user participates from this cost reduction. In addition, software tools can be created, that extensively use XML descriptions. Although the acceptance of XML will grow in accordance with the growing acceptance of internet related technology, the software tools will hide XML from the user. He will be presented a user interface, that partly or totally covers XML, but without wasting its advantages.

Finally, the close relation between XML and web-technologies offers a new quality in web-based management solutions.

References

[1] Wollschlaeger, M.; Mapping of Fieldbus Components to WWW based Management Solutions. FeT'99 Fieldbus Technology, Magdeburg, 23.-24.09.1999. Published in: Dietrich, D.; Neumann, P.; Schweinzer, H.: Fieldbus Technology. Springer Wien New York, pp. 172-179

[2] Lainé, T.: Internet technologies and fieldbuses, ISA TECH/1999, Philadelphia, 05.-07.10.1999, proceedings "Productivity and Flexibility"

[3] n.n.: OPC Data Access Automation Specification, Version 2.0. OPC Foundation, October 14th 1998

[4] Brockschmidt, K.: Inside OLE. Second Edition. Microsoft Press, 1997

[5] Bray, T.; Paoli, J.; Sperberg-McQueen, C. M.: Extensible Markup Language (XML) 1.0. 1998, http://www.w3.org/TR/REC-xml

[6] Boumphrey, F.: Professional Style Sheets for HTML and XML. Wrox Press, 1998.

[7] n.n.: Document Object Model (DOM) Level 1 Specification, Version 1.0 W3C Recommendation 1 October, 1998, http://www.w3.org/TR/REC-DOM-Level-1

[8] Moss, W.H.: Report on ISO TC184/SC5/WG5 Open Systems Application Frameworks based on ISO 11898, 5th International CAN Conference (iCC'98), San Jose, 03-05.11.1998, proceedings "Industrial Automation" pp. 07-02 to 07-04

[9] Korsakas, J.V.; Moyne, J.R.: Moving DeviceNet Data Representations to XML, 6th Annual ODVA Meeting, Tampa, 08.03.2000, http://216.10.36.18/10_2/09_down/XML%20Overview.ppt

[10] Döbrich, U.; Noury, P.: ESPRIT Project NOAH - Introduction. FeT'99 Fieldbus Technology, Magdeburg, 23.-24.09.1999. Published in: Dietrich, D.; Neumann, P.; Schweinzer, H.: Fieldbus Technology. Springer Wien New York, pp. 414-422

[11] n.n.: XML Linking Language (XLink), W3C Candidate Recommendation 3 July 2000, http://www.w3.org/TR/2000/CR-xlink-20000703

[12] Kimber, W.E.: Using UML To Define XML Document Types, http://www.drmacro.com/hyprlink/uml-dtds.pdf

[13] n.n.: Committee Draft - Function Blocks for industrial-process measurement and control systems. Part 2 - Engineering Task Support, 2nd Committee Draft, ftp://ftp.cle.ab.com/stds/iec/tc65wg6/document/pt2cd2.zip

[14] Wollschlaeger, M.: CANopen Device Descriptions using general purpose modelling languages, 6th International CAN Conference (iCC'99) Turin (Italy), 02.-04.11.1999, proceedings pp. 03-06 - 03-13

Dr.-Ing. Martin Wollschlaeger
Otto-von-Guericke-Universität Magdeburg
Institute for Electronics, Signal Processing and Communications (IESK)
PO Box 4120, D-39016 Magdeburg, Germany
Phone:+49 (391) 67-1 46 53
Fax: +49 (391) 5 61 63 58
e-mail: mw@iesk.et.uni-magdeburg.de
http://www-nt.et.uni-magdeburg.de/