Intranet-Based Management of CAN Devices

Dr.-Ing. Martin Wollschlaeger, Dipl.-Ing. Stefan Wehrmann

With the steadily growing use of CAN devices within complex automation systems, the need for effective management solutions becomes more and more obvious. The spreading integration of CAN systems into Local Area Networks (LAN) provides the opportunity to build such solutions on intranet-based technologies like HTTP or COM/DCOM. Particularly the chances of a link between the intranet-stored information and the fieldbus layer let the development of a new generation of management tools be more than likely.

Based on the illustration of principles for connecting fieldbus and LAN new concepts for the representation of CAN Higher Layer Protocol structures and data by intranet-based objects are introduced. Examples of practical realizations of the concepts for managing CAN modules prove the feasibility and show the prospects.

1. Introduction

One of the most outstanding trends in information technology's recent developments is the trend towards network-centric systems. Special attention has to be paid to the overwhelming development of the Internet and its enabling technologies in hard- and software, like TCP/IP, componentware, browser technology, Java etc. Using these technologies in an Intranet on top of a LAN enables the integration of different hard- and software platforms, as well as the integration of data with different sources, contents and representation. The user interface of an Intranet is implemented using a browser, a de-facto standard software package not only offering an unique look and feel, but a well known, user-accepted interface to the underlying heterogeneous information system. In addition to the representation of data, more and more management functions of the networked resources are accessible by a browser. This introduces a new quality for the management of industrial communication systems.

The steadily growing use of networked industrial PCs within fieldbus based systems offers an interesting platform for the integration of fieldbusses and LANs. These enterprise networks represent the solution for data exchange in process information systems. The main feature of a process information system is the seamless integration of fieldbus-based components into systems of the technical and economical management of the enterprise, providing easy access to manufacturing related data from standard office environment. This offers a starting point for the implementation of management functions for the underlying fieldbus and its components, that are accessible from any point within the process information system.

2. Integration concepts

There are different scenarios for the integration of fieldbus systems into LANs (**Figure 1**). PLC based implementations may be integrated using an industrial PC (IPC) as a gateway. This IPC is connected to both, the fieldbus and the network, by means of appropriate interface cards. When using an IPC with PC control applications instead of the PLC, the integration becomes more easy. In this case the IPC can offer the gateway functions in addition to the control applications. Both scenarios allow data exchange between the fieldbus components and any other networked equipment. This enables the access from applications of the business management as well as from configuration and monitoring tools and diagnostic systems to fieldbus data. Besides the process related data, installation-specific data and descriptions from databases will be opened for data access as well. The necessary conversion and representation of both classes of data - process data and descriptive data - are more and more performed using browser-based software components.

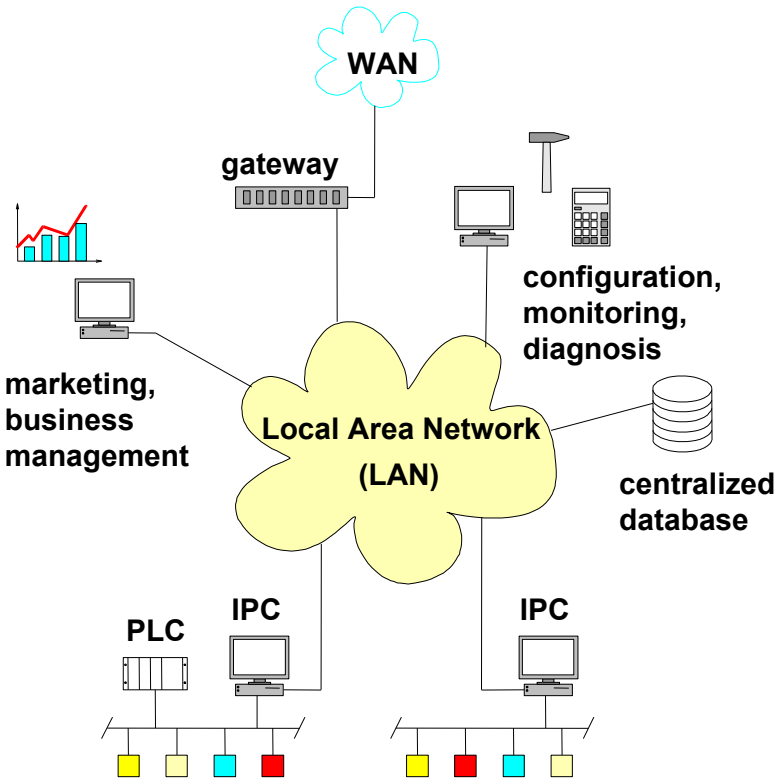

Figure 1: Scenarios for the integration of fieldbusses into a local area network

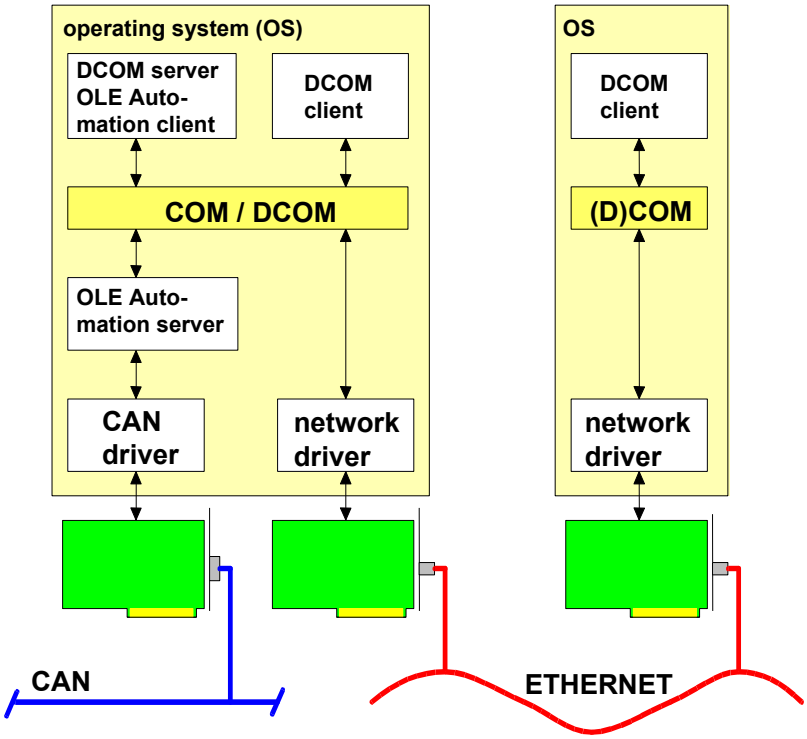

Figure 2: Software structure for the integration of a CAN system into a LAN

A mapping of the fieldbus protocol's functionality and data structures to appropriate objects is the major prerequisite for the integration of a fieldbus into an Intranet. These objects have to be organized in a way suitable for easy access from the applications relying on fieldbus related data.

More and more PC applications use the Component Object Model (COM) [1]. Therefor a fieldbus has to be mapped to COM-objects. The access of applications

to fieldbus data can be performed using OLE Automation techniques. Such a mapping enables access to fieldbus data from any OLE Automation capable local client application, without having to implement application-specific drivers for the fieldbus interface card. In addition, the mapping to COM objects is a prerequisite for the implementation of Distributed COM (DCOM). DCOM allows access from remote OLE client applications over the network in the same way as local clients do. **Figure 2** shows the principle structure of the integration of a CAN system into an Ethernet LAN.

The access to fieldbus data throughout the LAN environment is necessary for an Intranet based management of fieldbus components. Intranet based solutions are characterized by:

- the use of the Hypertext Transport Protocol (HTTP) as a network-wide unique logical transport layer,
- the use of web servers (HTTP servers) as data source, and
- the use of web browsers (HTTP clients) as a user interface.

The access of the browser to fieldbus objects is performed using server based scripts (**Figure 3**), or as a distributed application using DCOM-Clients (for example ActiveX-Controls) (**Figure 4**).

When using the script based implementation, a request from a browser initiates the execution of a script at the web server. This script starts an instance of an OLE Automation client, that accesses the COM objects of the fieldbus mapped to the OLE Automation server. The other implementation concept instantiates an ActiveX-Control [3] as a DCOM client. The web server provides the control for download on demand. The control uses DCOM for access to a DCOM server across the network. The DCOM server then uses the OLE Automation server to access the fieldbus data.

Both concepts are suitable for using a web browser as an unique interface widely accepted by the users. It provides access to fieldbus data as well as to any other information provided by the networked environment. Especially the direct access to data in fieldbus related design and configuration tools, and to database driven applications will support future complex, but scalable software tools for fieldbus handling.

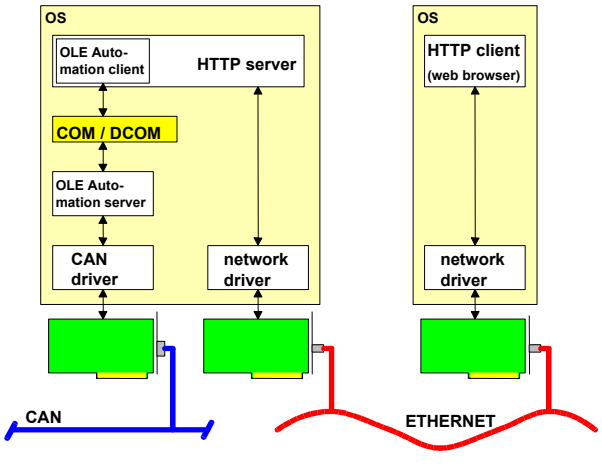

Figure 3: Server based integration

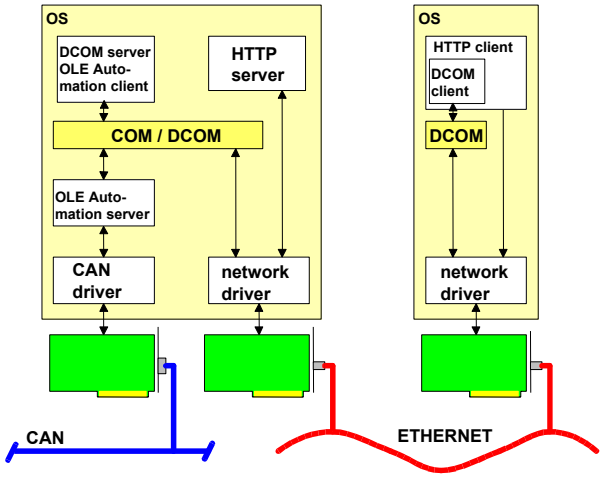

Figure 4: Integration as a distributed application

3. OLE Automation Server for CAN

An important part of the implemented prototype solution is the OLE Automation server for CAN. Its tasks are the encapsulation of the underlying driver for the CAN interface card, and the providing of a COM based interface for accessing the fieldbus. The COM model defines an unique, universal expandable binary interface enabling data exchange between different address spaces, access to properties and methods of objects, and the analysis of the interface of unknown objects. It is the prerequisite for other complex technologies, e.g. OLE Automation, Drag and Drop, OLE Documents and OLE Controls [2]. **Figure 5** shows the COM object "CAN" with its interfaces in a schematic view.

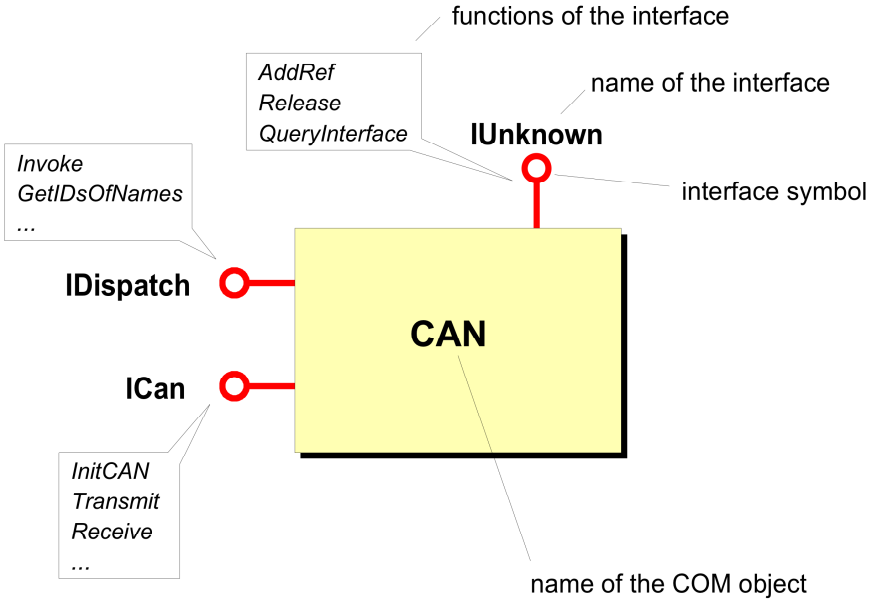

Figure 5: Structure of the COM Object "CAN" and its interfaces

The interface *IUnknown* represents the basic interface for all other interfaces. It is the only one, that every COM object has to implement. *IUnknown* provides three methods for two fundamental properties of the COM object. These properties are used for life-time control of the object by reference counting, and for navigating between the object's different interfaces.

Reference counting is performed using the method *AddRef* called by each client. Parallel to incrementing the value of the internal reference counter, this initial conversation loads the object into the memory. By calling *Release* a client closes the connection with the object. The method decrements the reference counter. If the counter reaches zero, the object terminates and is removed from the memory. The method *QueryInterface* allows requesting for special properties and provides access to the appropriate interfaces. A client, that wants to access certain properties of an object has to use *QueryInterface* in order to get a pointer the interfaces required to handle the properties. This method allows further development of an object's functionality by implementing additional interfaces without changing existing ones. The clients can use existing functions without any modification.

The interface *IDispatch* extends the object CAN by implementing the functions necessary to act as an OLE Automation server. The interface's methods allow the object to be easily accessed from any script based language like Visual Basic as well as from any other programming language like C/C++. The main method of *IDispatch* is *Invoke*. It calls functions of the COM object and passes the given parameters to these functions. The selection of the appropriate function is performed using the parameter DISPID. For example, an OLE Automation client that wants to call the object's function InitCAN passes the function's name to the function *GetIDsOfNames* of the IDispatch interface.

GetIDsOfNames returns the DISPID corresponding to InitCAN. Calling *Invoke* with this DISPID at last results in the execution of InitCAN.

A language like Visual Basic encapsulates this sequence. The user only has to code the statement `object.InitCAN`. It should be mentioned, that this technique works across the boundaries of processes, and with DCOM across the boundaries of computers.

As an opposite to *IUnknown* and *IDispatch*, the structure of *ICan* is not predefined by COM or by technologies based on COM. The interface *ICan* represents an application specific interface. It has to provide functions suitable for access to the CAN fieldbus. These functions include for example initialization of the CAN interface card and transmitting and receiving of CAN objects. The direct access of the interface's functions allows, compared with an access using *IDispatch*, a significantly lower access time. However, this advantage can only be used within a given process and not between different processes nor different computers. This is a disadvantage according to an access using *IDispatch*.

The CAN driver implemented in the prototype system is a 16-bit driver. This required the OLE Automation server to be a 16-bit program as well. The direct access from 32-bit applications to the interface *ICan* is impossible. So the gap between 16-bit and 32-bit applications had to be closed by using *IDispatch*. Practical experiences have shown, that the timing conditions are okay for purposes of the management of CAN modules.

4. Exemplary solutions for the management of CAN modules

Based on a CAN installation using Smart Distributed System (SDS) [4] as an application layer, the feasibility of the integration concepts are shown. The modules are connected to a PC running Windows 95 by means of an interface card. This PC implements the gateway function between CAN and a local area network. The web server is implemented as Microsoft Personal Web Server on that PC.

4.1. Installation using scripts

This implementation uses the OLE Automation server for CAN described above. Scripts coded in Visual Basic Script language (VBScript) are running on top of the web server. They use the interface *IDispatch* provided by the OLE Automation server in order to access data and services of the SDS objects. Furthermore, these VBScripts are used to construct the user interface, which is provided by means of HTML pages. This interface contains different functional layers, that allows access to the functionality of SDS objects, as well as to its parameters. Compound documents represent an excellent possibility for an integration of SDS objects, installation-specific data and descriptions. **Figure 6** shows a code fragment of a HTML page containing the PDF coded online documentation of an SDS Binary Output Object and an interface to scripts for manipulating the status of the Output Object. A screenshot of that page is shown in **Figure 7**.

```
<SCRIPT LANGUAGE=VBScript
 RUNAT=Server></SCRIPT>
<HTML>
<HEAD><TITLE>  SDS Binary
 Output</TITLE></HEAD>
<BODY>
<H2>  SDS Binary Output</H2>
<HR WIDTH=100% ALIGN=center SIZE=5>
<table border="0">
<tr>
<td><FORM METHOD="POST" ACTION="on.asp">
<P><INPUT NAME="BtnOn" TYPE="SUBMIT"
 VALUE="Switch Binary Output to On">
</FORM></td>
<td><FORM METHOD="POST"
 ACTION="off.asp">
<INPUT NAME="BtnOff" TYPE="SUBMIT"
 VALUE="Switch Binary Output to Off">
</FORM></td>
</tr></table>
<FORM METHOD="POST" ACTION="read.asp">
<INPUT NAME="Read Binary Output State"
 TYPE="SUBMIT" VALUE="Read Binary
 Output State  ">
</FORM>
...
Code omitted here for brievity
...
<H3>SDS Component Modeling Specification
 ( Online ):</H3>
<body leftmargin=0 topmargin=0
 scroll=no>
<embed width=100% height=50%
 fullscreen=no src=http://141.44.61.248/
  NT/CAN/sds/gs052107.pdf>
</BODY>
</HTML>
```

Figure 6: Code fragment of an SDS related compound document

This installation requires only the OLE Automation server as fieldbus-related application. However, the management-related functions have to be provided by the scripts. It has to be mentioned, that security relevant data has to be transmitted as plain text, so that the implementation of further security methods have to be considered.

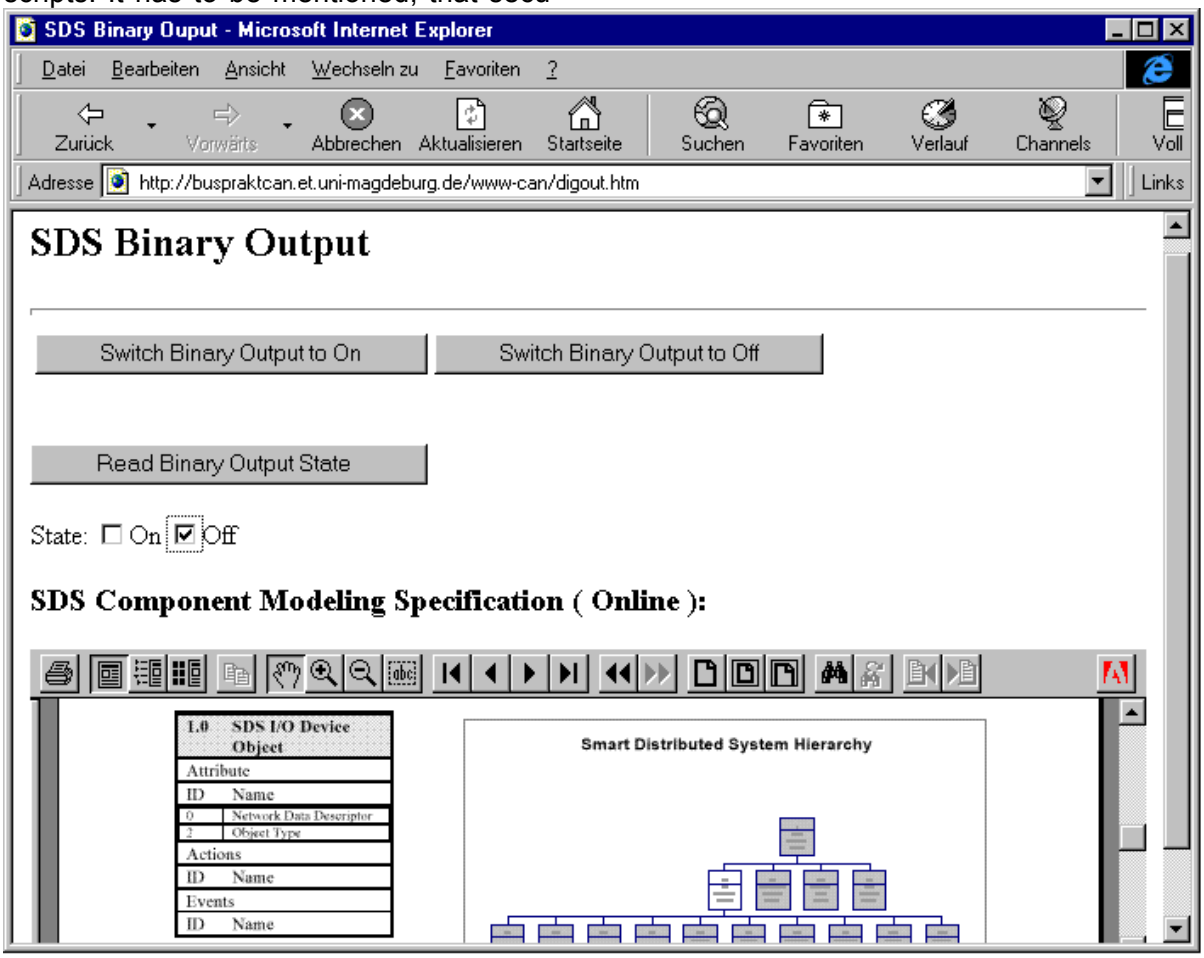

Figure 7: Screenshot of a compound document

4.2. Installation using ActiveX-Controls

This solution requires a DCOM server, that presents the functions of the OLE Automation server to the networked environment. The interaction with the user is performed by ActiveX-Controls. After a download from the web server, these controls are automatically instantiated on the client system, where they reside for further use. ActiveX-Controls can be integrated into any ActiveX-Container application. More and more container applications are used as a framework for specialized software components. This allows the reuse of the controls in a large number of different applications. The management functions have to be mapped to the user interface provided by the control. **Figure 8** shows a screenshot of an ActiveX based management solution for SDS systems. The treeview control on the left side represents the logical structure of the SDS installation. The access to data in the SDS components is possible by means of the text boxes and buttons on the right side. This data is manipulated by the control's internal functions and then transferred to the DCOM server that passes the data to the OLE Automation server. It is possible that the OLE Automation server is implemented as a part of the DCOM server. However, it has to be considered that the OLE Automation server has to access the interface card with its specific interface. The major advantage of the described installation is that the data is transmitted between the controls and the server in a binary format, so that security problems are reduced.

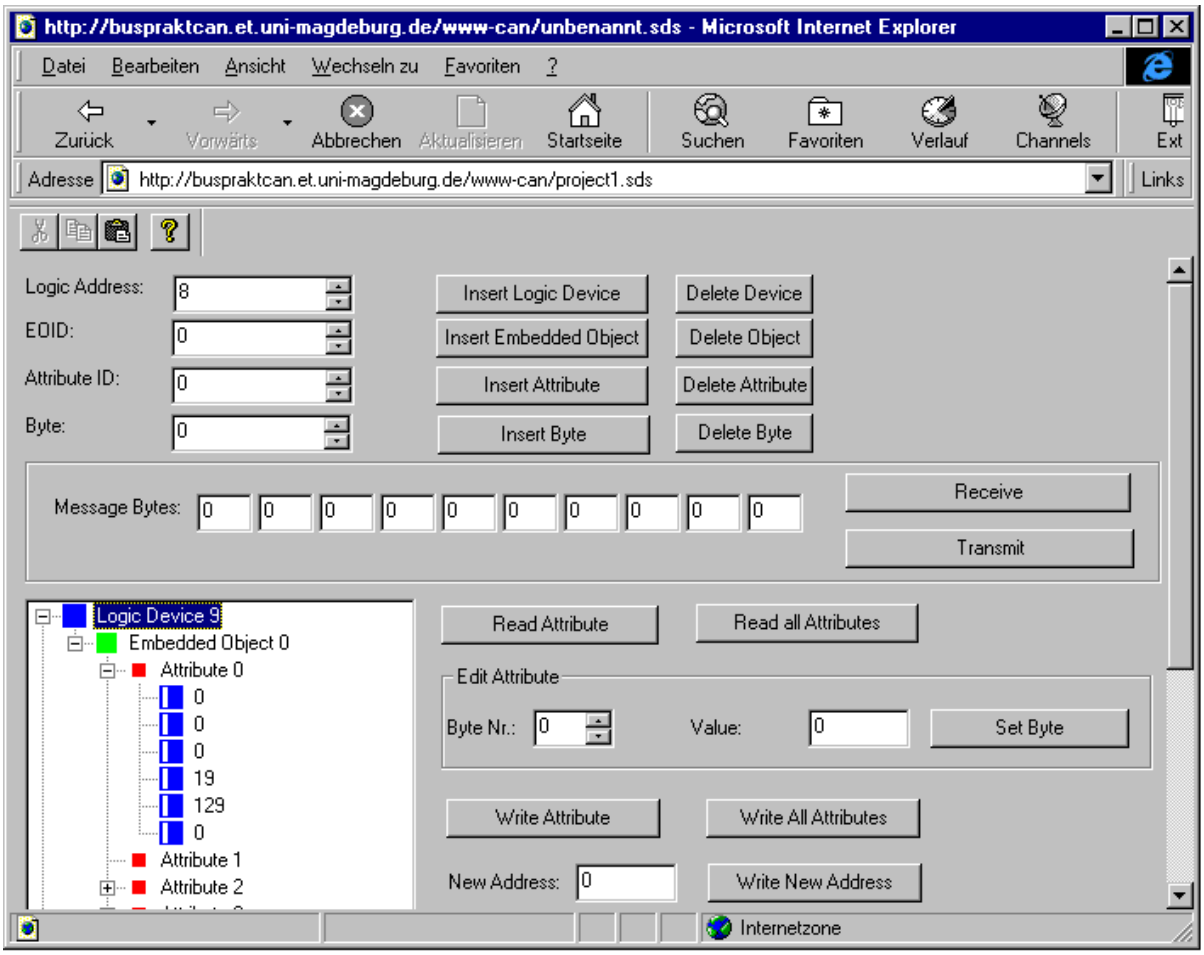

Figure 8: Screenshot of an ActiveX-based SDS management solution

5. Conclusion and future trends

An intranet based management solution enables an easy handling of a fieldbus system as a part of a process information system with high complexity. It requires the integration of a fieldbus into a local area network. This integration is possible by implementing appropriate software objects, that provide unique interfaces for applications. The integrative aspects of such implementations are considered as a new feature of modern software. The growing success of OPC (OLE for Process Control) [5], that also uses a COM based concept, is an example for this. Future management solutions can be implemented on top of OPC in order to use fieldbus independent data access methods.

The installation with ActiveX-Controls can be integrated into Java based management systems. Java can instantiate and access COM objects, so that data exchange between Java applications and fieldbus data can be performed using the OLE Automation server described above. On the other hand, ActiveX-Controls can be integrated into Java applets [6], that can be used in web browsers or Java applications. Furthermore, applets or controls with component-specific management functions can be implemented directly in the fieldbus components. They can be uploaded via the fieldbus into a network-centric application framework.

The introduction of Intranet based management solutions allows fieldbus handling with a new quality in user support. It offers a platform and a starting point for further developments in information technology, as well as in electronics and automation and control.

References

[1] Chappel, D.:
Understanding ActiveX and OLE.
Microsoft Press, 1996.

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

[3] Denning, A.:
ActiveX Controls Inside Out.
Microsoft Press, 1997.

[4] n.n.:
SDS specification.
Honeywell Inc. Microswitch Division,
GS 052 103 ... GS 052 107, 1995.

[5] n.n.:
OLE for Process Control.
Industry Standard Specification,
Version 1.0
OPC Task Force, August 29th 1996.

[6] Ladd, S.R.:
Active Visual J++.
Microsoft Press, 1997.

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

Dipl.-Ing. Stefan Wehrmann
3SOFT GmbH
Wetterkreuz 19a, D-91058 Erlangen, Germany
Phone: +49 (9131) 7701-159
Fax: +49 (9131) 7701-333
e-mail: wehrmann@3SOFT.de
http://www.3soft.de/