Was bedeutet .NET für die Leit- und Automatisierungstechnik?

von Georg Harnischmacher, Kai Luppa und Jan Arph

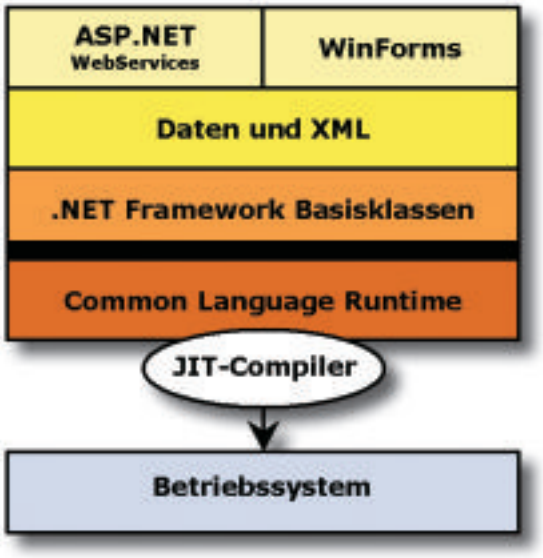

Bild 1: Grobstruktur der .NET-Architektur

Noch vor einigen Jahren Tabuthema, haben sich heute Produkte, die auf MS-Windows Betriebssystemen laufen, in der Leit- und Automatisierungstechnik etabliert. OPC und sogar um Plug-Ins und mit speziellen ActiveX-Controls hochgerüstetes Visual Basic zur Anlagendarstellung gelten als innovativ und modern. Was aber bedeutet es, wenn Microsoft die technologische Basis dieser Produkte ändert? Mit Microsofts neuer Plattform .NET (gesprochen "dot net") ist dies geschehen und es wird sich zukünftig wohl einiges ändern. Der Artikel beschreibt erste Eindrücke aus der Beta 1 Version der neuen Entwicklungsumgebung.

Viele Anwendungen der Leit- und Automatisierungstechnik verwenden heute Microsofts Component Object Model (COM). Das Objektmodell des Defacto-Standards OPC (OLE for Process Conrol) basiert z.Zt. vollständig auf COM und trägt sogar die COM-Technologie OLE im Namen. Noch im August letzten Jahres stellt die Profibus Nutzerorganisation PNO das auf COM aufbauende Profinet als vertikale Unternehmenslösung vor. Von den in der HMI-Special-Ausgabe des SPS-Magazins vorgestellten dreizehn Webanwendungen basieren sechs auf ASP und fünf verwenden ActiveX-Controls. Active Server Pages (ASP) verwenden von Haus aus COM- und einen bekannteren COM-Vertreter als die visuellen ActiveX-Controls kann man sich wohl kaum vorstellen. In der Leit- und Automatisierungstechnik, aber auch in vielen anderen Branchen, ist mit dem Ziel langlebiger und wiederverwendbarer Komponenten in den letzten Jahren viel Geld in die Entwicklung COM-basierter Anwendungen investiert worden.

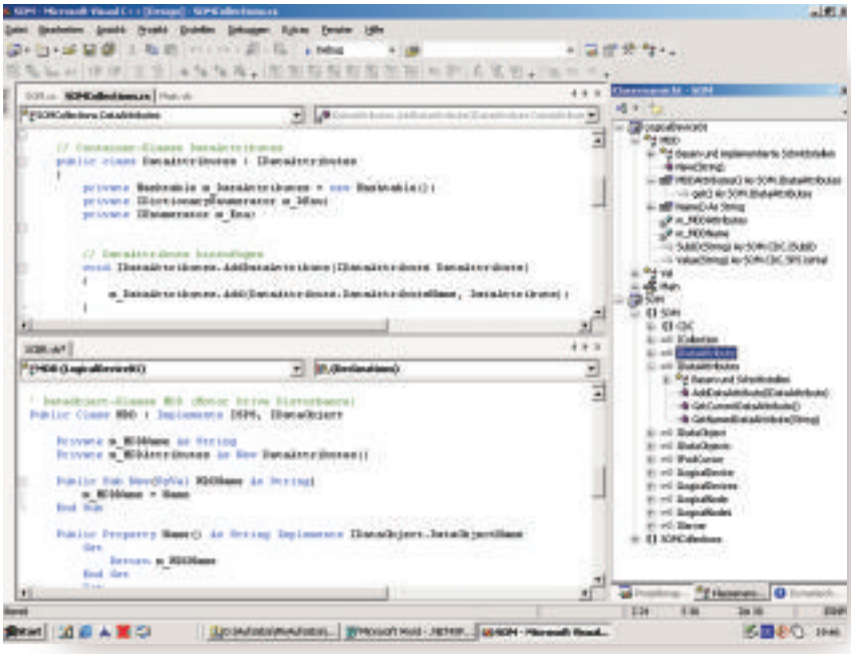

Bild 2: Visual Studio .NET, Sprachintegration auf Codeebene

.NET basiert nicht auf COM

Im Juli vergangenen Jahres hat Microsoft auf der Professional Developers Conference (PDC) in Orlando, Florida, seine .NET-Pläne vorgestellt und damit faktisch das Ende der COM-Ära eingeläutet. .NET ist eine völlig neue Komponentenarchitektur, die sich neuer Transportmechanismen und Protokolle bedient. Der Schritt von COM zu .NET wird offensichtlich ebenso hart wie der Übergang von DOS zu Windows. Inwieweit vorhandene COM-Komponenten im vollen Funktionsumfang im .NET wiederverwendet werden können, muss sich erst noch zeigen. Es scheint jedoch bereits heute klar zu sein, dass COM-Komponenten mit den neuen Visual-Studio-.NET-Entwicklungswerkzeugen nicht mehr wartbar sein werden. Erste Erfahrungen mit dem Visual Basic Importwerkzeug lassen eine aufwändige Portierung auf Codeebene erwarten und machen klar, dass .NET kein neues Feature einer Weiterentwicklung von COM ist. Die .NET-Architektur soll Einzug in Microsofts komplette Produktlandschaft halten, wird also früher oder später auch diejenigen betreffen, die heute noch nichts davon wissen wollen. Wir befinden uns derzeit in der Beta-1-Phase, gegen Ende des Jahres ist mit einer ersten Release-Version zu rechnen. Die ersten Bücher zu .NET sind erhältlich und viele weitere angekündigt. Weder ein einzelner Artikel noch ein einziges Buch kann alle Neuerungen im .NET beschreiben. Wir haben einige wichtige Konzepte der neuen Plattform herausgegriffen.

Gemeinsame Laufzeitschicht CLR

Den Kern der .NET-Architektur, welche in Bild 1 in ihrer Grobstruktur dargestellt ist, bildet die gemeinsame objektorientierte Laufzeitschicht CLR (Common Language Runtime). Das auf der CLR aufsetzende Framework enthält sowohl ein für alle Programmiersprachen gemeinsames Typsystem als auch eine gemeinsame Klassenbibliothek für die Anwendungsentwicklung. Code, der unter der Regie der CLR ausgeführt wird, bezeichnet man als verwalteten Code (Managed Code). Verwalteter Code wird von .NET kompatiblen Sprachcompilern in einen prozessorunabhängigen Zwischencode (MSIL, Microsoft Intermediate Language) übersetzt, welcher dann zur Laufzeit von dem Just-In-Time (JIT) Compiler der CLR in den entsprechenden Nativecode des Zielsystems aufgelöst wird. Dadurch erreicht .NETs CLR eine Entkopplung vom darunterliegenden Betriebssystem und erinnert vom Konzept her stark an Javas virtuelle Maschine. Zur Zeit stehen unter .NET vier Sprachcompiler, alle von Microsoft, zur Verfügung: C# (s.u.), Visual Basic, JScript und C++. Ausschließlich Visual C++ ist noch in der Lage, nichtverwalteten Code (Unmanaged Code) zu erzeugen, alle anderen Compiler produzieren MSIL und setzen damit direkt auf die CLR auf. MSIL ist von Microsoft dokumentiert und offengelegt. Damit besteht für Compilerhersteller die Chance, weitere Programmiersprachen für die .NET-Plattform anzubieten. Die Firma Rational soll bereits einen Java-Compiler für .NET planen.

Sprachintegration

Schon unter COM ist es möglich, Anwendungen unter verschiedenen Programmiersprachen zu entwickeln. Die Einbindung von unter Visual Basic, Delphi oder C++ entwickelten (Dll-) Komponenten in serverseitige ASP-Scripts, die zur Prozessvisualisierung in der webbasierten Leittechnik eingesetzt werden, ist ein typisches Beispiel für die mehrsprachige Zusammenarbeit unter COM. Die Sprachintegration findet bei COM jedoch auf Binärebene statt,

d.h., die kommunizierenden Komponenten liegen in compilierter Form vor. Im .NET findet die Sprachintegration auf Codeebene statt. Eine C#- oder C++-Klasse kann von einer Visual Basic-Basisklasse erben und das ganze auch umgekehrt. Bild 2 zeigt eine von einer C#-Klasse abgeleitete Visual Basic-Klasse. Im Gegensatz zu Visual Studio 6.0, das aus verschiedenen Entwicklungsumgebungen und Compilern besteht, ist Visual Studio .NET eine einheitliche Entwicklungsumgebung, die neben den verschiedenen Programmiersprachen mehrere Editoren, insbesondere für die XML-Technologien, unterstützt.

C# und Visual Basic .NET

C#, gesprochen "C sharp", ist eine von Microsoft neu entwickelte Programmiersprache. Sie ist die Systemsprache von .NET und folgt der Syntax von C++ und Java. Ein Entwurfziel war die einfache Anwendung und so ist das auffallendste Merkmal beim Vergleich mit C++, dass es, wie in Java, keine Zeiger gibt. Zum C#-Sprachumfang gehören, für Visual Basic Programmierer nichts unbekanntes, Eigenschaften und Ereignisse. Visual Basic hat es unter .NET endlich zu einer wirklich objektorientierten Sprache geschafft und stellt jetzt auch echte Vererbung, Überladung, virtuelle Funktionen und dergleichen zur Verfügung. Seine einfache Syntax hat Visual Basic weitestgehend behalten, die Neuerungen des .NET-Unterbaus halten aber auch für die VBler jede Menge Hausaufgaben bereit.

ASP.NET

Mit ASP.NET hat Microsoft das Konzept der bestehenden serverseitigen ASP-Technologie, die zum Einsatz dynamischer Webapplikationen eingesetzt wird, konsequent weiter entwickelt sowie vorhandene Schwächen beseitig. Bei ASP existiert keine formale Trennung von Seiteninhalt und Verhalten. Das Ergebnis einer ASP-Seite ist oft ein buntes Treiben von HTML, Style Sheets, clientseitigem JavaScript-Code sowie serverseitigem VBScript-Code, der sich oft zusätzlicher COM-Objekte bedient. Design und Funktionalität einer ASP.NET-Webanwendung werden dagegen nicht mehr zusammen, sondern durch Design- und Codeeditor getrennt von einander entwickelt. Dadurch erhöht sich die Übersichtlichkeit und oben beschriebener "Spaghetti-Code" wird vermieden. Im Gegensatz zu herkömmlichen ASP-Scripten wird ASP.NET-Code vorcompiliert, wodurch die Performance der Anwendung erheblich gesteigert wird. Die Entwicklung von ASP.NET-Funktionalität kann u. a. in C# oder Visual Basic .NET erfolgen, wodurch die Nachteile der typlosen ASP-Scriptprogrammierung (i.d.R. VBScript) vermieden werden. Als integraler Bestandteil des .NET-Frameworks bietet ASP.NET, neben dem Grundprinzip der Entwicklung dynamischer Webapplikationen, den Einsatz einer vollkommen neuen Technologie: den WebServices.

WebServices, XML und SOAP

Auch OPC hat das Problem, dass Distibuted COM nicht für den Einsatz verteilter Anwendungen im Internet geeignet ist und verteilte OPC-Anwendungen auch in Intranets auf Systeme mit DCOM-Unterstützung eingeschränkt werden. Aus diesem Grund wird bei OPC an einer XML-Spezifikation gearbeitet, die es ermöglicht, Systemgrenzen zu überwinden. Wie bestellt bietet .NET mit den WebServices eine XML- und SOAP-basierende Technologie zum entfernten Methodenaufruf. WebServices besitzen im Gegensatz zu Webapplikationen kein visuelles Frontend und stellen ihre Information auch nicht mittels HTML dar. Bild 3 zeigt die Auswahl eines WebService Projektes unter Visual Basic .NET. Der Client erhält als Ergebnis eines entfernten Methodenaufrufes via WebService reines XML. Die XML-Nutzlast der WebServices wird mittels SOAP zwischen den kommunizierenden Komponenten transferiert. Der SOAP-Standard setzt dabei auf bestehende Transportprotokolle, beispielsweise HTTP, auf. Die Verwendung bestehender, insbesondere durch die Infrastruktur des Internets geprägter Transportprotokolle, lassen die WebServices als ein sehr effektives Werkzeug für die Entwicklung verteilter Anwendungen erscheinen.

Fazit

Ziel der .NET-Plattform ist erstens eine verbesserte Softwareentwicklung auf den Windows-Betriebssystemen, die durch die gemeinsame Laufzeitschicht CLR und der von ihr zur Verfügung gestellten Dienste erreicht wird. Die im Laufe der Jahre unter COM zu einem immer größeren Problem aufgelaufenen Themen wie Registierung und Versionskonflikte ("Dll-Hell") sollen mit .NET der Vergangenheit angehören. Das zweite, vielleicht noch wichtigere Ziel der .NET-Plattform, ist die Entwicklung verteilter Internet-Anwendungen. Anstelle proprietärer Kommunikationsmechanismen, was wir eigentlich von Microsoft erwartet hätten, setzt .NET auf die Stan-

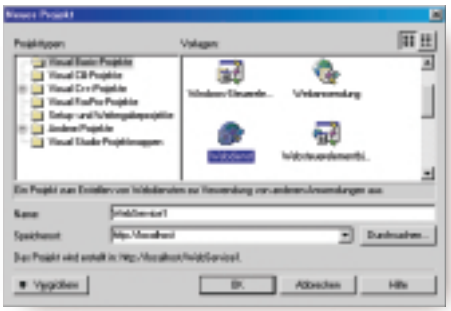

Bild 3: Projektauswahl Visual Basic .NET WebService

dards XML, SOAP, HTML und HTTP, um Dienste und Anwendungen über das Internet anzubieten. Offensichtlich kann .NET praktikable Lösungsansätze für viele aktuelle und zukünftig wichtige Aufgaben der Leit- und Automatisierungstechnik anbieten. Denn im Web verteilte Leittechnik-Anwendungen sowie ein standardisierter, vielleicht durch ein XML-Schema beschriebener Datenaustausch zwischen diesen Leitsystemen und den Geräten der Automatisierungstechnik, sind beispielhaft für technologische Entwicklungen, die gerade erst am Anfang stehen. Es wird also spannend, "ob überhaupt?", wenn "ja" dann "wann und wie?" .NET als Ablösung von COM in der Leit- und Automatisierungstechnik Fuß fassen kann. ■

5036

www.energietechnik.fh-dortmund.de
www.microsoft.com/net
www.wrox.com/dotnet.asp

Prof. Dr.-Ing. Georg Harnischmacher (40), VDE, vertritt im Fachbereich Elektrische Energietechnik der Fachhochschule Dortmund das Lehrgebiet Elektrische Energieerzeugung und -verteilung. Forschungsschwerpunkt ist u.a. die Informationstechnik in der Energieversorgung, insbesondere die Schutz- und Stationsleittechnik.
Dipl.-Ing. Kai Luppa (33), VDE, ist neben seiner Industrietätigkeit in der Leit- und Automatisierungstechnik Lehrbeauftragter an der Fachhochschule Dortmund. Im Fachbereich Elektrische Energietechnik lehrt er das Fach Datenverarbeitung und Programmiersprachen mit Schwerpunkt Visual Basic.
Cand.-Ing. Jan Arph (24) studiert Elektrotechnik an der Fachhochschule Dortmund, Studienrichtung Elektrische Energie- und Umwelttechnik, und arbeitet als Werkstudent mit ASP und Visual Basic an der Softwareentwicklung webbasierter Leitsysteme.

Was bedeutet .NET für die Leit- und Automatisierungstechnik?

Internet als Automation-Middleware

Teil 2

von Georg Harnischmacher, Kai Luppa und Jan Arph

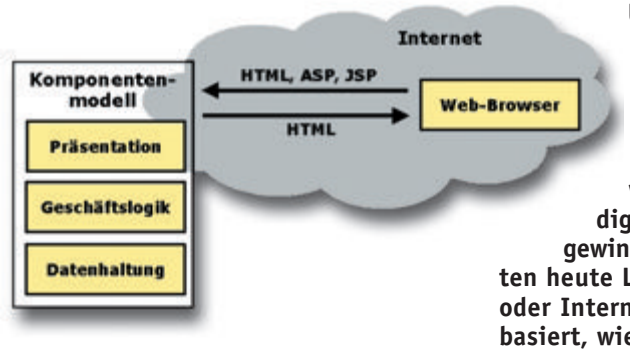

Bild 1: HTML zentriertes Internet

In der Ausgabe 8+9 2001 des SPS Magazins haben wir einen Überblick über die neue Microsoft Plattform .NET gegeben. Dieser Beitrag befasst sich intensiver mit den schon angesprochenen WebServices, welche einen neuen Ansatz verteilter Softwarekomponenten mit sich bringen. Verteilte Softwarekomponenten auf Basis der Middleware-Plattform DCOM spielen in der Leit- und Automatisierungstechnik eine wichtige Rolle, da Defacto-Standards wie OPC derzeit vollständig auf diese Technologie aufsetzen. Web-basierte Leitsysteme gewinnen zunehmend an Bedeutung und die meisten Hersteller bieten heute Lösungen mit Browser-Frontend zur Visualisierung via Intra- oder Internet an. Oft existiert eine Arbeitsteilung: Die Geschäftslogik basiert, wie im Fall von OPC, auf DCOM und die Darstellung erfolgt über das Internet als Mensch-Maschine-Schnittstelle. WebServices sind dagegen programmierbare Komponenten, welche ihre Dienste direkt über das Internet anbieten und als Ergebnis XML-Daten an den Aufrufer zurückgeben. Damit könnte das Internet zur Middleware der Geschäftslogik anwachsen und sich eine völlig neue Perspektive der Gerätekommunikation ergeben. Diese junge Technologie ist sicherlich noch nicht ausgereift, sie lässt jedoch schon heute viele Chancen von morgen erkennen.

Seit Anfang der 90er Jahre hat sich das Konzept der Softwarekomponenten sowie die Entwicklung drei- oder mehrschichtiger Anwendungen durchgesetzt. Die Grundidee der komponentenbasierten Softwareentwicklung ist, dass eine Anwendung nicht aus einem einzigen monolithischen Block besteht, sondern aus vielen kleinen und in sich logisch abgeschlossenen binären Funktionseinheiten. Diese nach dem Baukasten-System zu einem Gesamtsystem skalierbaren Komponenten kommunizieren untereinander nach dem Client/Server-Prinzip. Das Konzept der Komponentenmodelle wurde dahingehend erweitert, dass der Zugriff auf einzelne Komponenten nicht mehr ausschließlich auf lokale Systeme beschränkt blieb, sondern die Verfügbarkeit von Server-Komponenten für Client-Anwendungen, welche sich auf anderen Maschinen im Netzwerk befinden, erweitert wurde. Damit Client- und Server-Komponenten miteinander kommunizieren können ist jedoch eine Infrastruktur in Form einer Middleware-Plattform notwendig. Dies gilt sowohl für lokale als auch für im Netzwerk verteilte Komponenten. Als Middleware-Plattformen haben sich heute verschiedene und im allgemeinen nicht kompatible Binärformate etabliert. Die bekanntesten davon sind DCOM (Distributed Component Object Model) und CORBA (Common Object Request Broker Architecture). In der Leit- und Automatisierungstechnik hat sich DCOM weitestgehend durchgesetzt und Defacto-Standards wie OPC (OLE for Process Control) setzten heute auf dieses Komponentenmodell auf. Im Falle von OPC wird jedoch eine homogene DCOM- und damit in der Regel Windows-basierte Netzwerkumgebung vorausgesetzt.

Voraussetzungen im WWW

Mit der zunehmenden Verfügbarkeit des Internets besteht der Wunsch, das WWW (World Wide Web) als Basis für ein weltweit verteiltes und verfügbares Komponentenmodell-Konzept zu nutzen. Die organisationsspezifischen und proprietären Kommunikationsstandards der heute verfügbaren Komponentenmodelle machen ihren Einsatz im Internet jedoch nahezu unmöglich. Die Nutzung des heterogenen Internets als verteiltes Komponentenmodell setzt einen einheitlichen und offenen Kommunikationsstandard zwingend voraus, welcher durch übergeordnete und unabhängige Institutionen, wie dem W3C (WWW-Consortium), getragen wird. Heute existieren im Internet in erster Linie Anwendungen, deren Endverbraucher der Mensch ist. Der Mensch nutzt als Client über die Browser-Schnittstelle Dienste, welche auf entfernten Web-Servern zur Verfü-

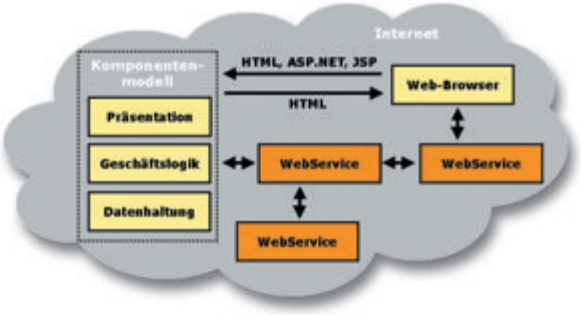

Bild 2: Im Internet verteilte Geschäftslogik

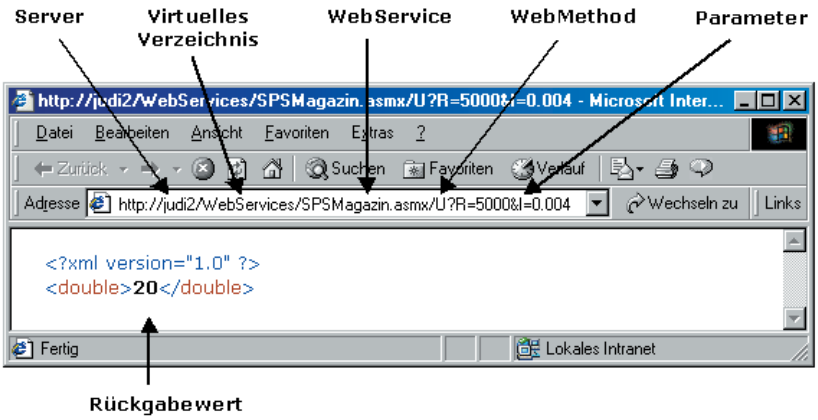

Bild 3: Aufrufparameter und Rückgabewerte

gung gestellt werden. Dementsprechend liefert die angefragte Maschine eine durch präsentationsbezogene Tags ausgezeichnete Antwort in HTML (Hypertext Markup Language), die dann mit Hilfe des Browsers vom Menschen gelesen und interpretiert wird. Bild 1 zeigt die heutige, auf HTML zentrierte Internet-Kommunikation. Eine semantische Auszeichnung der übertragenen Daten findet in HTML nicht statt. Diese ist jedoch Voraussetzung für eine, anstelle des Menschen, clientseitig informationsverarbeitende Softwarekomponente, z.B. eines intelligenten Automatisierungsgeräts. Ein einheitlicher Kommunikationsstandard und durch Metadaten beschriebene Informationen können als Grundvoraussetzung für ein internetbasiertes Komponentenmodell angesehen werden.

Software als Service

WebServices basieren auf dem im Internet kleinsten verfügbaren gemeinsamen Nenner, dem Übertragungsprotokoll HTTP (Hypertext Transfer Protocol) und der Auszeichnungssprache XML (eXtensible Markup Language). Dabei sind WebServices programmierbare URIs (Uniform Resource Identifier) und werden somit über das im Internet übliche Adressierungsschema aufgerufen. Nach Dienstanforderung und Parameterübergabe berechnen sie entsprechend ihrer Implementierung Ausgabedaten und senden diese an den aufrufenden Client in XML zurück. Wie in Bild 2 dargestellt, kann ein WebService also als eine Art Geschäftslogik-Komponente, die ihre Dienstleistung im Netz anbietet, angesehen werden. Damit ein Client einen WebService nutzten kann, muss er den Ort, d.h. die Adresse -genauer den URI- des Services im Netz kennen. Weiterhin benötigt der Client Informationen über die einzelnen Methoden des WebServices, wie deren Syntax, Aufrufparameter und Rückgabewerte. Diese Metadaten sind in der sogenannten WSDL-Datei (Web Service Description Language) in XML beschrieben. In der Beta 1 Version des .NET-Frameworks wird zu diesem Zweck noch das SDL-Format verwendet, Beta 2 soll aber bereits den zukünftigen Defacto-Standard WSDL unterstützen. Die WSDL beschreibt die Schnittstellen eines WebServices auf ähnliche Weise wie eine Typbibliothek die Schnittstellen einer COM-Kompo-

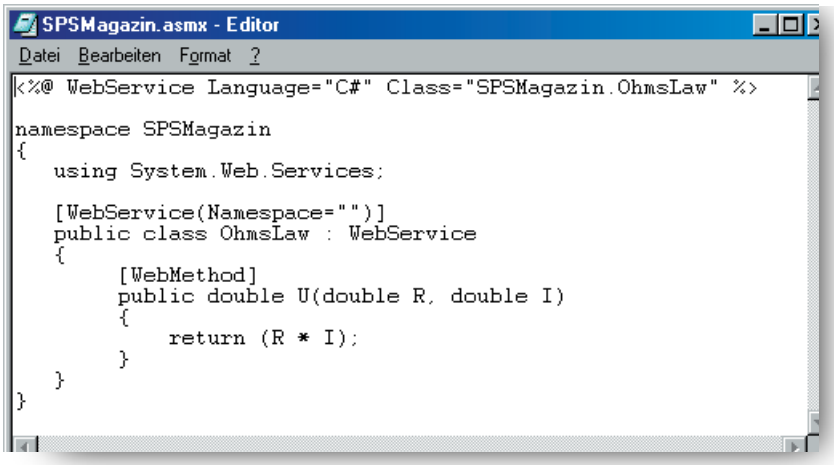

Bild 4: Eine Beispiel WebService in C#

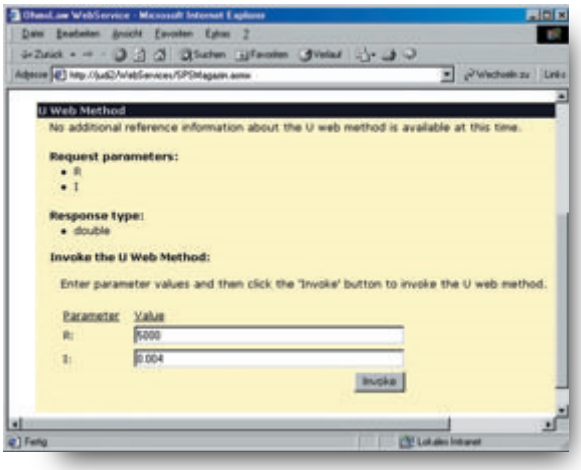

Bild 5: Automatisch generierte Browser Schnittstelle

nente oder die IDL (Interface Definition Language) eine CORBA-Schnittstelle. Jeder WebService-Nutzer muss diese Beschreibung des Dienstes zunächst auswerten so wie heute auch ein COM-Client Erkundigungen über die Schnittstellen seines Servers einholen muss. Diese vertragsmässig und über ein gesamtes Objektmodell festgelegten Schnittstellendefinitionen ermöglichen ja gerade erst die unabhängige Entwicklung von Client- und Server-Komponenten – OPC ist eines der besten Beispiele. Mit den WebServices entsteht eine neue Sichtweise von Dienstleistungen in Form einer "Black-Box", die ähnlich der zugekauften Funktionalität einer COM-Komponenteverwendet werden. Wie eine als WebService offengelegte Dienstleistung angefordert werden kann zeigt Bild 3. Im einfachsten Fall erfolgt der Methoden-Aufruf des WebServices wie dargestellt über HTTP-GET.Jedem Anwender von ASP (Active Server Pages) wird das bekannt vorkommen. Beim HPPT-POST Aufruf liegen die Methodenargumente im Körper der POST-Nachricht, in der Regel innerhalb eines HTML-Formulars.

WebServices unterstützen SOAP

Neben HTTP-GET und HTTP-POST wird der zukünftige Kommunikationsstandard SOAP (Simple Object Access Protocol) von den WebServices unterstützt. SOAP wurde unter anderem von Firmen wie UserLand, DevelopMentor, IBM und Microsoft sowie weiterer führender Firmen der IT-Branche entwickelt und stellt ein Protokoll für entfernte Methodenaufrufe dar, welches auf XML basiert. Zur Datenübertragung im Internet setzt SOAP auf das hier im allgemeinen verfügbare HTTP auf, wobei jedoch auch andere Transportprotokolle, wie zum Beispiel TCP (Transmission Control Protocol) oder FTP (File Transfer Protocol), eingesetzt werden können. SOAP hat die Aufgabe, den Aufbau der XML-Nachricht zu definieren und die Aufrufreihenfolge zwischen den miteinander kommunizierenden Partnern zu regeln (Marshaling). Ein auf HTTP basierender SOAP-Methodenaufruf passiert, im Gegensatz zu einer DCOM- oder CORBA-Methode, die im Internet aufgestellten Firewalls, da diese den entsprechenden HTTP-Port (Port 80) nicht blockieren. Ein für die Leit- und Automatisierungstechnik wichtiger Aspekt ist, dass SOAP sowohl synchrone als auch asynchrone Kommunikation unterstützt und damit die Möglichkeit besteht parallele Anfragen zu bearbeiten.

.NET Remoting

Das .NET Remoting-Framework stellt neben den WebServices ein weiteres Konzept einer horizontalen Middleware zur Verfügung und ist wohl eher als Nachfolger von DCOM anzusehen. Ebenso wie bei den WebServices wird bei der Remoting-Architektur der Objektaufruf transparent gestaltet und von dem zu Grunde liegendem Transportmechanismus getrennt. Anders als bei den WebServices bietet das Remoting-Framework jedoch für den Transport von Nachrichten zwischen kommunizierenden Komponenten Kommunikationskanäle (Channels) an, für deren Kodierung zur Zeit SOAP, HTTP sowie für Anwendungen mit zeitkritischem Datentransfer direkte TCP-Sockets zur Verfügung stehen. Die Remoting-Architektur erlaubt es hierbei, das Transportprotokoll des Verbindungsmechanismus mittels einer Konfigurationsdatei (ab der Beta 2 in XML) zu bestimmen. Das bedeutet, dass das Kommunikationsprotokoll, ohne den Aufwand die gesamte Anwendung neu erstellen zu müssen, gewechselt werden kann.

Von der Theorie zur Praxis

Nach soviel Theorie noch ein kurzes Beispiel einer WebService-Implementierung. Wir verzichten auf "Hello SPS Magazin" als das übliche von "Hello World" abgeleitete Beispiel und stellen an Stelle dessen das Ohm'sche Gesetz als WebService zur Verfügung. Für dieses Beispiel wird der IIS (Internet Information Server), das .NET SDK (Software Development Kit) und ein Texteditor benötigt. Eine Beta Version des Visual Studio .NET ist nicht erforderlich. Bild 4 zeigt eine vollständige Implementierung des .NET WebServices in C#. Ein WebService trägt die Dateiendung asmx und beginnt mit der Direktive <%@ WebService ...%>, die der ASP.NET Runtime anzeigt, dass es sich hier um einen WebService handelt. Bei dem gezeigten Quellcode ist zu erkennen, dass der Unterschied zu einer gewöhnlichen Klasse lediglich darin besteht, dass die Klasse OhmsLaw, welche unseren WebService zur Verfügung stellt, von System.Web.Services abgeleitet und mit [WebService] gekennzeichnet wird, (Namespace="") blendet den Standard-Namensraum aus. Um eine Methode im Web anzubieten, wird ihr [WebMethod] vorangestellt. Diese Schlüsselwörter sind bereits ausreichend, damit das .NET-Framework aus diesem Code einen funktionierenden WebService samt WSDL-Schnittstellenbeschreibung generiert. Zudem liefert das .NET- Framework automatisch eine Browser-Schnittstelle in HTML, diese ist in Bild 5 dargestellt. n

6951

www.energietechnik.fh-dortmund.de
www.gotdotnet.com

Prof. Dr.-Ing Georg Harnischmacher (40), VDE, vertritt im Fachbereich Elektrische Energietechnik der Fachhochschule Dortmund das Lehrgebiet Elektrische Energieerzeugung und ?verteilung. Forschungsschwerpunkt ist u.a. die Informationstechnik in der Energieversorgung, insbesondere die Schutz- und Stationsleittechnik.
Dipl.-Ing. Kai Luppa (33), VDE, ist neben seiner Industrietätigkeit in der Leit- und Automatisierungstechnik Lehrbeauftragter an der Fachhochschule Dortmund. Im Fachbereich Elektrische Energietechnik lehrt er das Fach Datenverarbeitung und Programmiersprachen mit Schwerpunkt Visual Basic.
Cand.-Ing. Jan Arph (24) studiert Elektrotechnik an der Fachhochschule Dortmund, Studienrichtung Elektrische Energie- und Umwelttechnik, und arbeitet als Werkstudent mit ASP und Visual Basic an der Softwareentwicklung web-basierter Leitsysteme.