Diplomarbeit

Integration von Webtechnologien in LabVIEW™ / ObjectVIEW™

eingereicht von Andreas Himmelreich

geb. am 16.10.1973 in Jena

Matrikel-Nr.:	209846
Seminargruppe:	982 ET / AT

Hochschulbetreuer:	Prof. Dr.-Ing. J. Müller, Fachhochschule Jena
Mentor:	Dipl.-Ing. J. Vogel, Vogel Automatisierungstechnik GmbH

Themenausgabe:	26.08.2002
Abgabedatum:	21.02.2003

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

A2A	Application to Application
AConML	Automatic Control Markup Language
ARPA	Advanced Research Projects Agency
ASP	Active Server Pages
CERN	Conseil Européenne pour la Recherche Nucléaire
CGI	Common Gateway Interface
CLS	Common Language Specification
COM/DCOM	Component Object Model/ Distributed Component Object Model
CORBA	Common Object Request Broker Architecture
CRL	Common Language Runtime
CSS	Cascading Style Sheets
DLL	Dynamic Link Library
FTP	File Transfer Protocol
GML	Generalized Markup Language
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IPC	Industrie PC
IT	Informationstechnologie
JIT	Just-in-Time
JPEG	Joint Photographic Expert Group
JSP	Java Server Pages
LabVIEW	Laboratory Virtual Instrument Engineering Workbench
M2M	Machine to Machine
MSIL	Microsoft Intermediate Language
NI	National Instruments
OET	Object Event Toolset
OPC	OLE for Process Control
PC	Personal Computer
PDA	Personal Digital Assistant
RMI	Remote Method Invocation

RPC	Remote Procedure Call
SGML	Standard Generalized Markup Language
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
TCP/IP	Transmission Control Protocol over Internet Protocol
UDDI	Universal Description, Discovery and Integration
UML	Unified Modelling Language
URL	Uniform Resource Locator
VAT	Vogel Automatisierungstechnik GmbH
VI	Virtual Instrument
VM	Virtual Machine
W3C	Word Wide Web Consortium
WSDL	Web Services Description Language
WWW	World Wide Web
XML	Extensible Markup Language
XSL	Extensible Style Language

1 Einleitung

1.1 Vorstellung der Firma

Die Vogel Automatisierungstechnik GmbH (vat) wurde 1993 gegründet und bietet die Lieferung von autonomen Steuerungen bis zur Automation von komplexen Produktionsanlagen. Vat setzt auf die Vereinigung von Automation und Informationstechnologie (IT), so dass eine Reihe innovativer Produkte entstehen.

Das gesamte Leistungsspektrum kann wie folgt zusammen gefasst werden:

- Ausführliche Prozessanalyse,
- Konzeption und Beratung,
- Planung und Umsetzung von Automations- und IT-Systemen,
- Integration lokaler Systeme,
- Inbetriebnahme,
- Produktschulung, Service und Support.

Bei der Softwareerstellung geht die Firma objektorientiert vor und verwendet dabei Standardsoftware, sowie selbst entwickelte Tools.

Nach dem nun fast 10-jährigem Bestehen kann vat auf eine Reihe erfolgreicher Projekte für namhafte Unternehmen verweisen. Mehr Information zu vat unter der URL: http://www.vat.de.

1.2 Motivation

Die Grenzen zwischen der Automatisierungstechnik und Informationstechnik verwischen zunehmend. Der Grund dafür liegt in der Verwendung der Technologien der IT-Welt in der Automation. Derzeit wird vielerorts speziell der Einsatz der Webtechnologie vorangetrieben. Webtechnologie heißt für den Automatisierer, auf eine Palette von bewährten, allgemein akzeptierten und weit verbreiteten Techniken zurück greifen zu können. Ziel des Einsatzes dieser Technologie soll es sein, den industriellen Kommunikationsstrukturen mehr Transparenz zu verleihen und derzeit vorhandene Kommunikationsbarrieren verschwinden zu lassen. Automatisierungsaufgaben können schnell und mit minimalen Kostenaufwand auf andere Plattformen portiert und quasi weltweit verteilt werden. Sämtliche Clients, die eine Unterstützung der Webtechnologien bieten, sind einsetzbar, so z.B. auch Pocket-PC´s. Geräte können eine eigene Homepage erhalten, über welche die Bedienung oder im Fehlerfall eine Ferndiagnose möglich ist.

Der Vorteil für den Anwender vergrößert sich, wenn dieser mit Hilfe einer grafischen Programmierumgebung seine Applikation webfähig gestalten kann. Eine solche grafische Programmierumgebung findet man in LabVIEW. Mit der Kombination der Vorteile der grafischen Programmierung und der Webtechnologie eröffnet sich nicht nur das weite Feld des WWW, sondern es tun sich neue Anwendungsfelder auf und es erweitert sich so das Dienstleistungsspektrum des LabVIEW-Programmierers.

1.3 Aufgabe der Diplomarbeit

Die Diplomarbeit befasst sich mit der Kopplung von LabVIEW mit der Technologie des WWW. Zu diesen Zweck soll zunächst festgestellt werden, inwieweit Webtechnologie bereits Bestandteil von LabVIEW ist. Vorhandene Lösungen sollen analysiert und bewertet werden. Im Ergebnis dieser Untersuchung steht, welcher Integrationsgrad erreicht ist und welche zusätzliche Funktionalität abgeleitet werden kann. Darauf aufbauend soll entschieden werden, ob eine Anpassung bzw. Erweiterung auf Basis der vorhandenen Lösungen erfolgt oder ob im Rahmen einer neuen Applikation die gewünschte Funktionalität umgesetzt wird.

Ziel ist es, dem Anwender ein Webinterface bereitzustellen, mit dessen Hilfe er in der Lage ist, seine LabVIEW-Applikation komfortabel via Internet/Intranet von entfernten Rechnern aus fernzusteuern.

Die erreichte Funktion des zu diesem Zeitpunkt vorliegenden Webinterface wird in einem weiterem Schritt in einem praktischen Anwendungsfall nachgewiesen. Als Beispielanwendung dient hierbei die Etagensteuerung der Räumlichkeiten der Vogel Automatisierungstechnik GmbH. Diese basiert auf einen Feldbuscontroller mit entsprechend angeschlossenen I/O´s und einem Steuerungsprogramm. Als übergeordnete Steuerung soll ein LabVIEW- Programm entstehen, mit dessen Hilfe man den Raum entsprechend seiner Funktionen bedienen kann. Damit ergibt sich mit der Kopplung zur Hardware eine weitere Teilaufgabe.

Als Ergebnis soll die Steuerung der Etage mittels Webbrowser vom Arbeitsplatz der Mitarbeiter aus, sowie drahtlos per Pocket PC möglich sein.

Als Abschluss erfolgt ein Ausblick hin zu zukunftweisenden Technologien und deren Erörterung.

2 Grundlagen

2.1 System- und Softwarearchitekturen

2.1.1 Allgemein

Ausgangspunkt für die folgenden Betrachtungen sind softwarebasierte, technische Systeme. Im allgemeinen verarbeitet ein System anstehende Eingangssignale und generiert daraus eine Anzahl Ausgangssignale. Neben entsprechender Hardware und Systemsoftware (Betriebssystem, Treiber) ist vor allem die Anwendungssoftware für die Ausführung der gewünschten Verarbeitungsfunktion verantwortlich. Hardware und Systemsoftware stehen heute als standardisierte Komponenten von verschiedenen Herstellern zur Verfügung. Dagegen wird die Anwendungssoftware vom Systemanbieter aufgrund der speziellen Anforderungen des künftigen Systemnutzers erst erstellt.

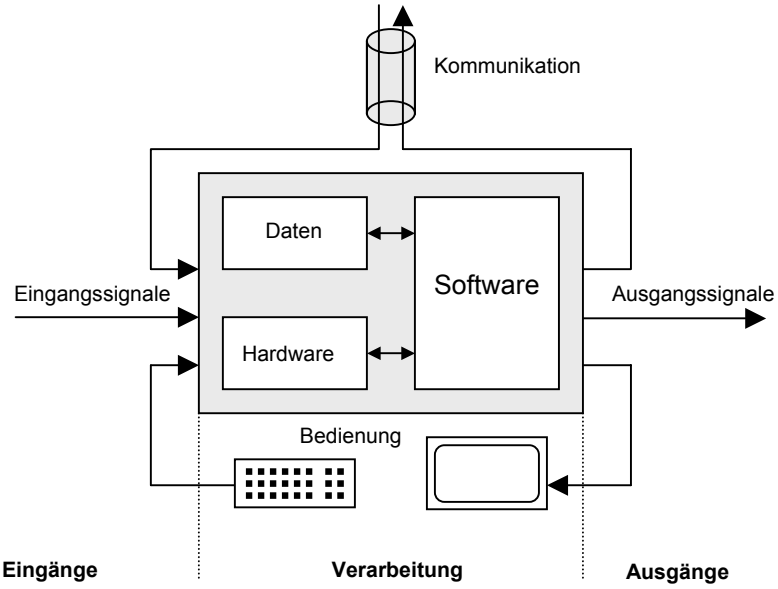

Abb. 2-1: Softwarebasiertes System allgemein

Bei heutigen, meist sehr komplexen Systemen nimmt das Design und die Erstellung der Anwendungssoftware oft die meiste Zeit in Anspruch und beeinflusst somit die Gesamtkosten entscheidend. Zur Erstellung der Anwendungssoftware kann der Entwickler mittlerweile auf eine Reihe verschiedener Werkzeuge mit den unterschiedlichsten Philosophien zurückgreifen.

2.1.2 Dezentrale Systeme

Bis Mitte der siebziger Jahre dominierten die zentralen Systeme. Auf einem Zentralrechner waren sämtliche Anwendungsfunktionen implementiert. Der Zugriff erfolgte über Terminals. Mit der Zeit wurden Anwendungen komplexer und Datenmengen größer. Die Ressourcen der Zentralrechner stießen an ihre Grenzen und die Anwendungssoftware wurde unübersichtlich und kaum wartbar.

Durch Fortschritte in digitalen Datenübertragung, Entstehung erster digitalen Netze und nicht zuletzt wegen fallender Preise für die Hardware wurde eine Dezentralisierung möglich. In einer dezentralen Anwendung konnten nun mehrere Rechner und Arbeitsstationen miteinander kommunizieren. Vorhandene Übertragungsprotokolle wurden in das Betriebssystem integriert und die Anwendungssoftware konnte auf Daten und Funktionen von Rechnern innerhalb des Systems zugreifen.

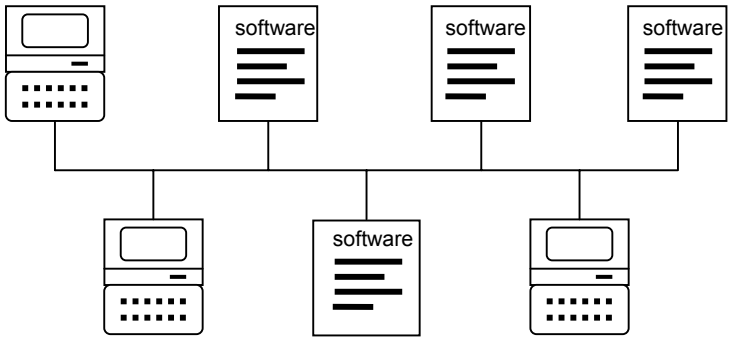

Abb. 2-2: Dezentrale Systemarchitektur

Die Hersteller erarbeiteten in diesem Zeitraum jeweils eigene Lösungen für dezentrale Systemarchitekturen mit jeweils eigenen Übertragungsprotokollen und selbstgewählter Funktionsaufteilung. Eine herstellerunabhängige Verbreitung von dezentralen Anwendungen war kaum möglich, der Ruf nach Standardisierung war gerechtfertigt.

2.1.3 Client/Server-Architektur

Bei der Client/Server-Architektur wird die Anwendung in einen Client- und einen Serverteil aufgetrennt. Im Allgemeinen werden die interaktive, grafische Benutzeroberfläche, die Darstellung der Resultate sowie sämtliche sichtbare Funktionen der Client-Seite zugeordnet. Die Verarbeitungsprozeduren und die Verwaltung der Daten ist Aufgabe spezieller Server. Der Client nutzt die Dienste einer oder mehrerer Server. Die Implementierung erfolgt i.a. auf verschiedenen vernetzten Rechnern.

Kommunikationsprotokoll und Form des Zugriffs sind entsprechend vereinbart und von verschiedenen Herstellern definiert und implementiert. Derartige Definitionen bezeichnet man häufig als Middleware. Clients fordern Daten oder Dienstleistungen über Request Messages vom Server an und dieser liefert die Ergebnisse in Form von Reply Messages. Dabei bilden die Clients die eigentliche Anwendung, während der Server sich als selbstständige Funktionseinheit passiv verhält, bis die Anwendung eine Anfrage an den Server vorsieht.

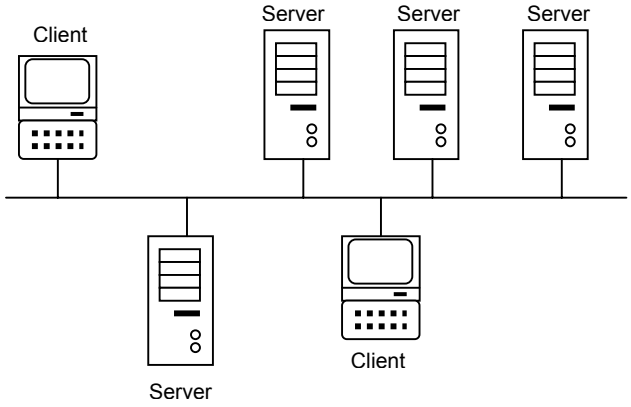

Abb. 2-3: Client/Server-Architektur

Die Client/Server-Architektur stellt die moderne Form der dezentralen Systeme mit definierten Kommunikationsprotokollen, Zugriffsformen und Funktionsaufteilungen dar.

2.1.4 Daten-/Funktionsmodell

Der Ansatz der klassischen Softwaretechnologie besteht darin die Daten und die Verarbeitungsprozeduren getrennt zu betrachten. Daten werden in speziellen Datenstrukturen (Datenbanken) gespeichert. Verarbeitungsstrukturen sind in sinnvoller, funktionaler Aufteilung in Form von Programmen implementiert. Diese lesen bzw. schreiben die Daten innerhalb der vorgegebenen Struktur. Diese Vorgehensweise beim Programmentwurf ist historisch so gewachsen.

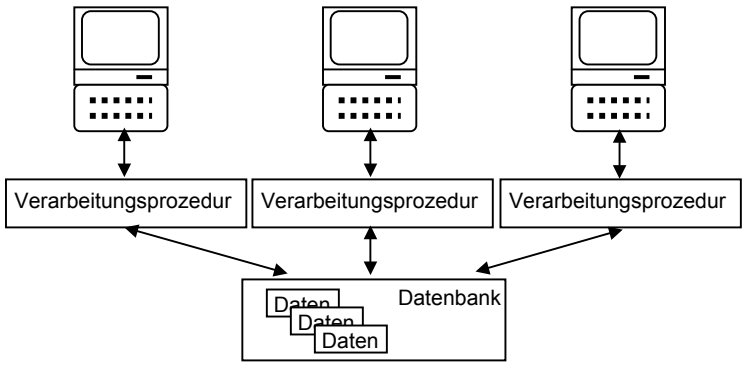

Abb. 2-4: Daten-/Funktionsmodell

Der Entwurf kann daher in zwei Schritte gegliedert werden:

1. Entwurf einer geeigneten, anwendungsgerechten Datenstruktur
2. Entwurf einer geeigneten Programmstruktur, d.h. Aufteilung der Funktionen auf modulare Verarbeitungsstrukturen

Um die Daten möglichst vielen Programmen zur Verfügung stellen zu können, bedarf es einer universellen Datenstruktur mit dem Effekt, dass aufwendige und komplexe Datenverwaltungssysteme mit hohem Ressourcenbedarf entstehen. Der entscheidende Nachteil des klassischen Daten-/Funktionsmodell besteht in der Abhängigkeit der Prozeduren von der Datenorganisation. Dadurch sind die verschiedenen Verarbeitungsprozeduren i.d.R. nicht wiederverwendbar. Eine Änderung der Datenorganisation ist kaum möglich, da in diesem Falle zu viele Verarbeitungsprozeduren geändert werden müssen. Die Folge ist, die Softwaresysteme sind starr, unbeweglich und lassen sich kaum dynamisch den sich ändernden Anforderungen anpassen.

2.1.5 Objekttechnologie

Eine mögliche Lösung der Probleme des Daten-/Funktionsmodell bietet die Objekttechnologie. Deren Grundprinzip ist die Zusammenfassung aller Funktionen und Daten für eine definierten Teilprozess zu einem Objekt. Der Zugriff auf das Objekt (von anderen Objekten) erfolgt ausschließlich über die vom Objekt zur Verfügung gestellten Schnittstellen. Die Objekte bilden somit selbstständige, selbstverwaltete und robuste Komponenten in der objektorientierten Architektur (→ Kapselung).

Alle Verarbeitungsprozeduren (Methoden) und Daten, bzw. Attribute, befinden sich in einer Kapsel, welche nur für bestimmte Operationen durchlässig ist.

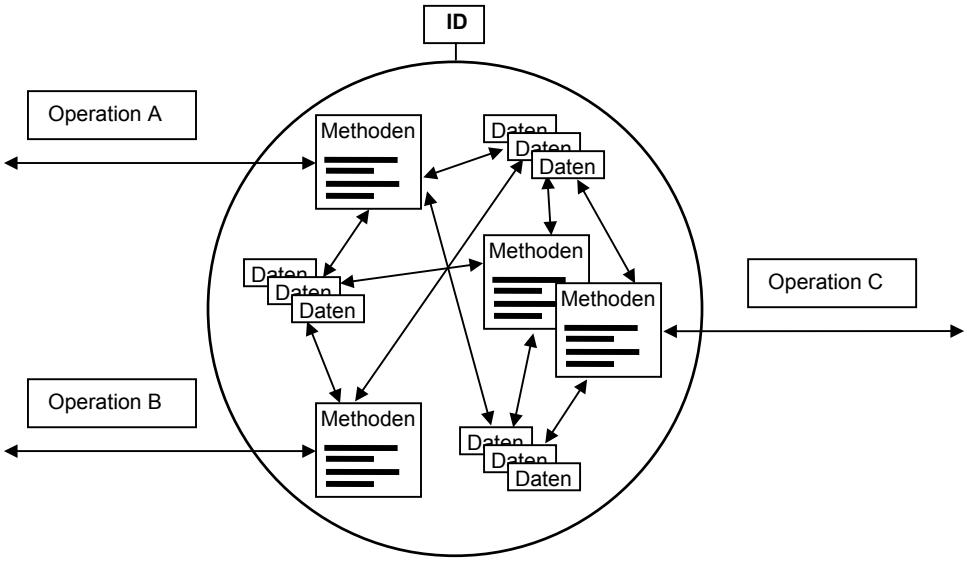

Abb. 2-5: Objekt = Kapselung von Verarbeitungsprozeduren (Methoden) und Daten

Zur vollständigen Beschreibung eines Objektes gehört seine Objekt-ID. Das Objekt kann so seiner Umwelt über die definierten Operationen (Methoden) Dienste anbieten. Im Allgemeinen werden bei der Ausführung der Operation Parameter an das Objekt übergeben und zugehörige Methode ausgeführt. Es erfolgt eine Modifikation der Daten im Objekt und die Rückgabe des Verarbeitungsresultates. Der Zustand (State) eines Objekts wird durch seine Attributwerte bzw. Daten und den jeweiligen Verbindungen zu anderen Objekten bestimmt. Operationen auf Objekte erfolgen über Messages. So wird die Verarbeitung über eine Request-Message angestoßen und die Resultate über Response-Messages zurückgegeben. Dieser Umstand ermöglicht eine Definition der Objekte als Server, Client oder auch beides. Man erhält ein System mit dezentralen Objekten, welche über Messages kommunizieren.

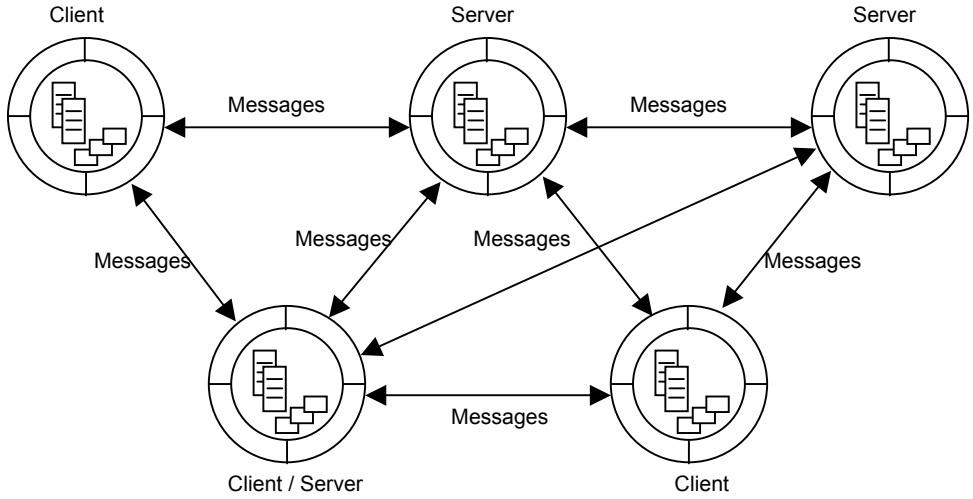

Abb. 2-6: Dezentrale Client/Server-Objekte

Neben der Kapselung und dem Bereitstellen von Schnittstellen ist die Abstraktion eine weitere Eigenschaft von Objekten. Aus einem komplexen Gebilde (aus der Realität) werden diejenigen Eigenschaften und Verhaltensweisen herauskristallisiert, welche für die Realisierung in der Anwendung notwendig sind. Man erhält das einfachstmögliche, aber dennoch vollständige Modell zur korrekten Programmierung der gewünschten Anwendung. Objekte mit den selben Eigenschaften (Attributen), den selben Verhalten (Operationen) und den selben Beziehungen werden zu Klassen zusammengefasst. Eine Klasse dient sozusagen als Vorlage bzw. Schablone. Sie besitzt Mechanismen zum Erzeugen von konkreten Objekten. Als Synonym für Objekte wird häufig auch der Begriff Instanz verwendet. Eigenschaften einer Klasse können an andere Klassen weitergegeben werden (Vererbung) oder mehrere Klassen können zu einer Klasse zusammen gefasst werden (Generalisierung).

In das weite Feld der Objekttechnologie an sich soll an dieser Stelle nicht weiter eingedrungen werden. Der Vorteil der objektorientierten Analyse-, Entwurfs-, und Implementierungsmethoden sind heute in unzähligen, erfolgreichen Projekten nachgewiesen worden. Die Objekttechnologie ermöglicht dank durchsetzungsfähigen De-facto-Standards die herstellerübergreifende Produktion von wiederverwertbaren Softwarekomponenten, was durchaus mit den Übergang vom softwaremäßigen Handwerks- in das softwaremäßige Industriezeitalter verglichen werden kann [Furr99].

Auf dieser Basis ist es dem Entwickler möglich, seine Anwendung wesentlich schneller zu entwickeln bzw. komplexe Anwendungen überhaupt erst beherrschbar zu machen. Mit der Verwendung eines vorhandenen und bereits bewährten Objektes greift der Entwickler auf konzentriertes Wissen und Know-how zu, ohne sich selbst mit der Programmierproblematik auseinander setzen zu müssen. Er muss lediglich wissen, welche Dienste das Objekt bereitstellt und kann diese dann einfach nutzen.

Aufgrund dieser Vorgehensweise bildet die Objekttechnologie die Basis moderner, verteilter Softwaresysteme.

Vorzüge der Objekttechnologie im Überblick [Furr99]:

- Adaptives, agiles Softwaresystem: Objektorientierte Systeme lassen sich schnell, planbar, risikoarm und kostengünstig an wechselnde und an neue Bedürfnisse anpassen, wobei sich diese schnell und exakt mit objektorientierten Analyse- und Modellierungswerkzeugen erfassen lassen.
- Beherrschung der Komplexität: Die objektorientierten Analyse-, Design- und Implementierungsverfahren erlauben die Aufteilung komplexer Systeme in beherrschbare Teile.
- Integrationsfähigkeit von Fremdkomponenten: Die standardisierte Infrastruktur ermöglicht die direkte Integration von Fremdkomponenten. Hochwertige, erprobte Funktionalität kann zeitgerecht eingekauft werden, ohne das entsprechendes Spezialwissen vorhanden sein muss.
- Software-Wiederverwendung: Die Wiederverwendung von Softwarekomponenten ist grundlegend bei der objektorientierten Denk- und Bauweise.
- Skalierbarkeit: Systeme auf Basis der objektorientierten Systemarchitektur können kurzfristig und ohne Softwaremodifikation auf ein höheres Verarbeitungsvolumen ausgebaut werden.

2.2 Das Netz der Netze

2.2.1 Die Entstehung des Internets

Das wohl größte, auf der Client/Server-Architektur beruhende, dezentrale System ist das Internet. Die Anfänge des heutigen Internets liegen in den 60er Jahren. Wie so oft bei technischen Neuerungen war es das Militär, was die Entwicklung vorantrieb. Die Aufgabe war, die Rechner so zu vernetzen, dass auch bei Ausfällen von Teilen dieses Netzes die militärischen Daten synchronisiert werden konnten. Die Advanced Research Projects Agency (ARPA), Teil der US-Militärs, realisierte das geplante Projekt. In den ersten Jahren wurde das Netz deshalb ARPA-Net genannt. Bei der Suche nach fehlertoleranten Übertragungsmethoden wurde von den ARPA-Ingenieuren eine völlig neuartige Datenkommunikationstechnik entwickelt: die paketorientierte Datenübertragung. Diese Übertragungsmethode beruht auf der Aufteilung der zu übertragenden Daten in Pakete, welche anschließend jedes für sich über das Netzwerk versandt werden. In Abhängigkeit von Verfügbarkeit, Verkehrsbelastung und Übertragungszeit der verschiedenen Netzwerkstrecken wählen Router die für die jeweiligen Pakete optimalen Übertragungsstrecken.

Ende 1969 waren die ersten vier Rechner an das ARPA-Net angeschlossen. Drei Jahre später waren es bereits 40 Rechner. In den frühen 70er Jahren entdeckten auch akademische Einrichtungen den Nutzen eines solchen Rechnernetzes, allerdings mehr zum Austausch der Daten untereinander. 1973 begann die zwischenzeitlich in DARPA umbenannte Forschungsanstalt ein Projekt, dessen Ziel es war , die unterschiedliche Implementierungen von paketorientierten Mechanismen , welche parallel entstanden waren, miteinander zu verbinden. Es nannte sich: das Internet-Projekt. Vier Jahre später arbeiteten vier unterschiedliche, paketorientierte Netzwerke zusammen: ARPANET, ein Satelliten-Netzwerk, ein Funk-Netzwerk und das von XEROX-PARC entwickelte Ethernet. Basis war das TCP/IP-Kommunikationsprotokoll, das zusammen mit der Stanford University eigens dafür entwickelt wurde.

Die Anzahl der an das ARPANET angeschlossenen Rechner stieg weiter an. Parallel dazu entstanden eine Reihe kleinerer Universitätsnetze, die über Rechenzentren miteinander verbunden wurden. Das Netz der Netze entstand und bald bürgerte sich der Name „Internet“ ein. 1983 hatte das ARPANET eine solche Ausdehnung erreicht, dass es in einen wissenschaftlichen, forschungsorientierten Teil (ARPANET) und einem militärischen Teil (MILNET) aufgeteilt wurde. Auf anderen Kontinenten gab es ähnliche Entwicklungen, so dass das Internet bald den gesamten Globus umspannte.

Was wir also heute unter "Internet" verstehen, ist nicht ein einziges homogenes Netz, sondern ein Verbund aus vielen kleinen, territorial oder organisatorisch begrenzten Netzen, welche über standardisierte Protokolle miteinander kommunizieren. Die ursprüngliche Zielsetzung des Internets ist die Ursache dafür, dass sich dieses Netzwerk seit Beginn der achtziger Jahre bis heute praktisch ohne zentrale Steuerung und Planung mit rasanter Geschwindigkeit ausbreitet. Das Wachstum findet an unzähligen Stellen gleichzeitig statt und ist kaum kontrollierbar. Trotzdem stellt das Internet, wie ursprünglich von den Militärs gedacht, ein außerordentlich zuverlässiges Kommunikationsmedium dar [Kyas01].

2.2.2 Bedeutung des WWW

Der sicherlich bekannteste Dienst, der auf dem Internet aufbaut, ist das World Wide Web (WWW). Die Entstehung begann 1990 und geht auf eine Initiative von Wissenschaftlern des Genfer Hochenergieforschungszentrum CERN, insbesondere Tim Berneers-Lee, zurück. Während das Internet recht umständlich zu bedienen war, sollte der neue Dienst wissenschaftliche Dokumente online und formatiert sichtbar machen und die Möglichkeit des Einbindens von Grafik bieten. Des weiteren bestand die Idee, Hypertextfunktionalität einzubauen, so dass Dokumente Verweise auf beliebige andere Dokumente enthalten können, auch wenn diese auf ganz anderen Internet-Servern liegen.

Entscheidend für die Umsetzung war die Entwicklung von HTML, der Hypertext Markup Language und ein dafür geeignetes Übertragungsprotokoll, dem Hypertext Transfer Protocol, kurz HTTP, sowie die Entwicklung von Browsern als Clientprogramme zur Darstellung der HTML-Dokumente. Im Mai 1991 wurden die ersten Dokumente im WWW-Format auf Server des CERN geladen. Einen Browser gab es jedoch ausschließlich für die ASCII-basierende Darstellung. Im Februar 1993 wurde dann das erste WWW- Browserprogramm mit grafischer Bedienoberfläche vorgestellt.

Die Freigabe der WWW-Technologie durch das CERN im April 1993, sowie die Möglichkeiten und die Einfachheit von HTML bildeten die Voraussetzung für die ab etwa 1993 einsetzende explosionsartige Ausweitung des WWW. Jedermann konnte die Technologie benutzen, ohne irgendwelche Patentrechte erwerben oder Copyrightgebühren entrichten zu müssen. Auch ungeübteren Anwendern war es möglich, sich im Informationsangebot zu bewegen ohne sich um Verzeichnisstrukturen, Dateinamen oder komplizierte Eingabebefehle zu kümmern. Das WWW war so vollständig der Öffentlichkeit zugänglich gemacht worden und entwickelte sich zur bevorzugten Bedienoberfläche im Internet.

Der Boom des Web war nicht mehr zu bremsen. Während 1993 weltweit 50 HTTP-Server existierten, waren es Ende 1994 schon 100 000. Bereits 1995 war WWW der führende Dienst innerhalb des Internets und zum Massenmedium geworden.

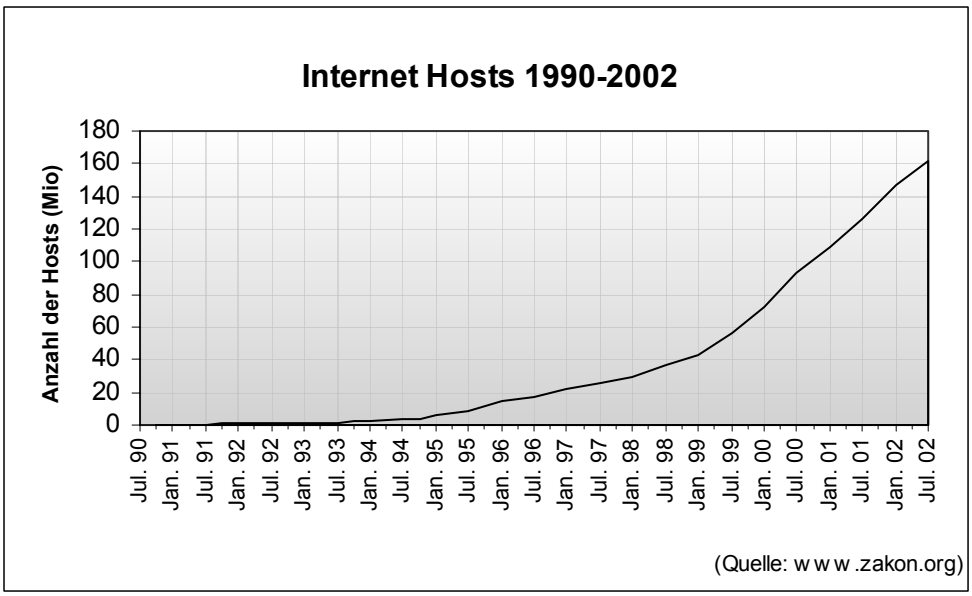

Abb. 2-7: Entwicklung der Anzahl von Hosts weltweit

Mittlerweile ist das Web fester Bestandteil in der Geschäftswelt (eBusiness) und dringt immer weiter in neue Bereiche vor. Beispiele sind: mobiles Internet auf Kleingeräten oder neue Schlagwörter, wie „eProduction“ oder „vom selling Internet zum working Internet“.

2.2.3 Funktionsweise und Architektur des WWW

Die Architektur des WWW basiert vollständig auf der dem Client/Server-Prinzip. Innerhalb der Web-Architektur spricht man jedoch vom Web-Client und Web-Server. Der Client präsentiert die Benutzeroberfläche in Form des Web-Browsers. Der Server übernimmt die Verteilung der HTML-Dokumente und übernimmt gegebenenfalls die Kommunikation des Clients mit der Anwendungslogik. Die Arbeitsweise beruht auf den drei Standards:

- HTML (Hyper Text Markup Language),
- HTTP (Hyper Text Transport Protocol),
- URL (Uniform Resource Locator).

HTML ist eine Seitenbeschreibungssprache, die es ermöglicht, die für das WWW charakteristischen Web-Dokumente zu verfassen, um diese im Netz verfügbar zu machen. Dabei macht HTML eigentlich nichts weiter, als die ASCII-Zeichen eines Dokumentes mit Hilfe von Formatierungsanweisungen (Markups) zu gliedern. Für die endgültige Darstellung einer Seite ist jedoch der Browser verantwortlich.

URL ist ein Adressierungsschema, mit dem der Ort jeder Datei im Internet, sowie das für den Zugriff erforderliche Übertragungsprotokoll angegeben werden kann.

HTTP ist schließlich das Protokoll, auf dem die Kommunikation zwischen Web-Client und Web-Server basiert und zur Übertragung der Web-Dokumente dient. HTTP ist ein zustandloses Protokoll, d.h. es gibt nur Transaktionen.

Eine Transaktion läuft nach folgendem Schema ab:

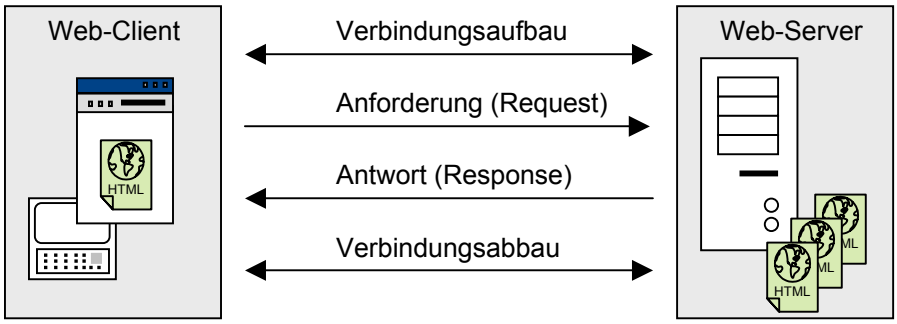

Abb. 2-8: Transaktion zwischen Web-Client und Web-Server

Die folgende Abbildung soll zusammenfassend das Funktionsprinzip des WWW als Dienst des Internets verdeutlichen.

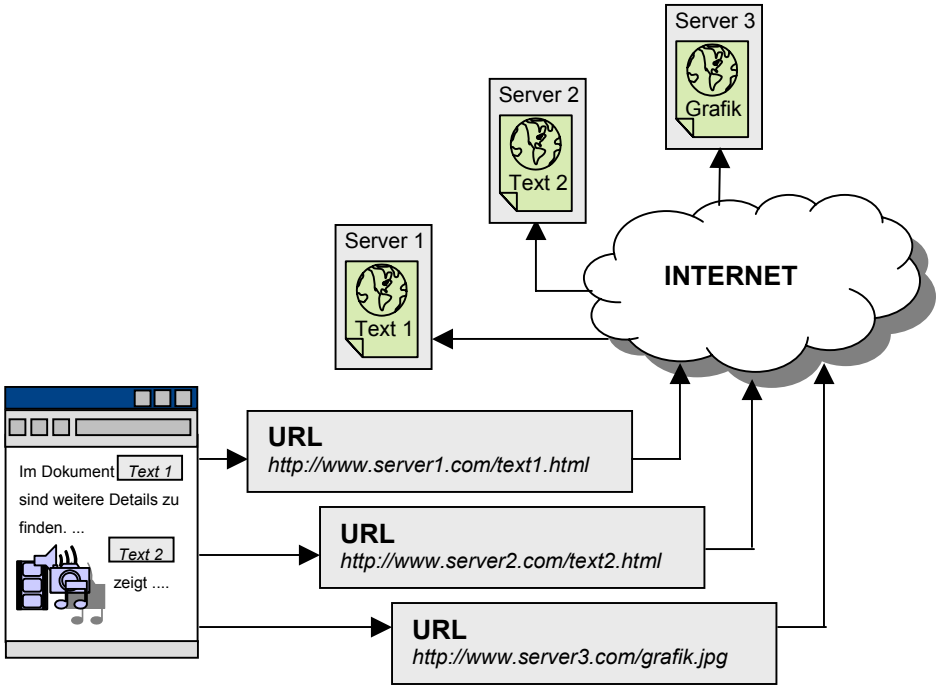

Abb. 2-9: Funktionsprinzip des WWW mit HTML und URL

2.3 LabVIEW™

2.3.1 LabVIEW™ und die Programmiersprache G

LabVIEW ist eine professionelle Programmierumgebung der Firma National Instruments aus Austin in Texas, USA. Ein in LabVIEW erstelltes Programm wird als Virtuelles Instrument (VI) bezeichnet. Diese Bezeichnung verrät den Ursprung von LabVIEW in der Messtechnik. Die Funktionalität und das Aussehen sind an reale Instrumente angelehnt. Ein VI besteht prinzipiell aus zwei Teilen:

- Frontpanel: die grafische Benutzeroberfläche (GUI). Per Drag and Drop kann der Programmierer die Ein- und Ausgabeelemente platzieren.
- Blockdiagramm: ist direkt mit den Elementen des Frontpanels verknüpft und enthält den Programmcode.

Programmiert wird in LabVIEW mit der grafischen Programmiersprache G und nach dem Prinzip des Datenflusses. Der Programmcode im Blockdiagramm besteht aus einer Auswahl verschiedenster Elemente, welche jeweils für eine bestimmte Funktion stehen. Bestandteile des Programmcodes können sein:

- Ein- und Ausgabeelemente (Controls und Indikators), sowie Konstanten,
- Funktionen und Operatoren,
- Ablaufstrukturen (For- und While-Schleife, Case-Anweisung, ...),
- Unterprogramme (SubVI´s).

Diese werden miteinander verbunden und bestimmen so den Datenfluss und somit die Programmfunktion.

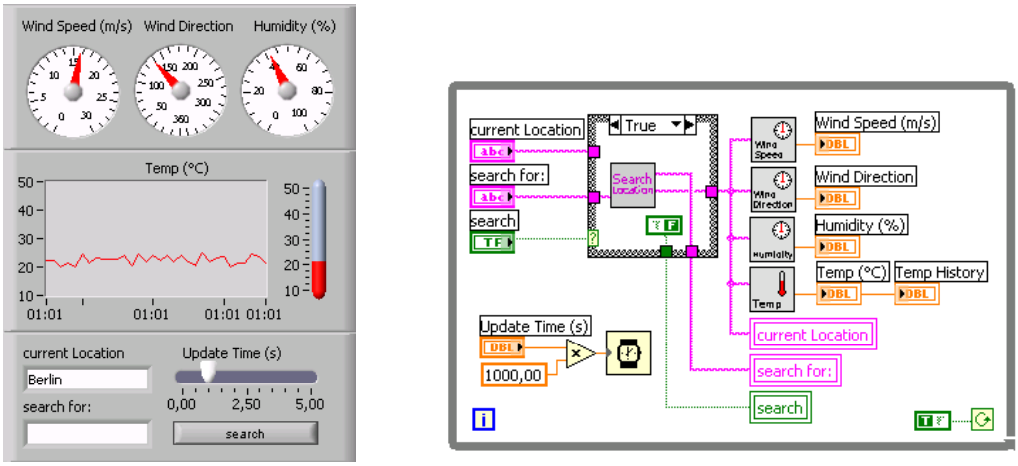

Abb. 2-10: Frontpanel und Blockdiagramm in LabVIEW

Die in LabVIEW erstellten Blockdiagramme werden von einem grafischen Compiler in optimierten Maschinencode übersetzt. Die Ablaufgeschwindigkeit der LabVIEW-Programme ist vergleichbar mit kompilierten C- oder Pascal-Programmen.
Die erste Version von LabVIEW entstand 1986. Seit dieser Zeit wurde permanent an der Verbesserung und Erweiterung von LabVIEW gearbeitet. Mittlerweile liegt es in der Version 6.1 vor und deckt zusammen mit den optional erhältlichen Toolsets nahezu das gesamte Spektrum der Mess- und Automatisierungstechnik ab.
Diese umfangreiche Funktionspalette gepaart mit der grafischen Programmierung machen LabVIEW zu einer mächtigen und effektiven IT-Entwicklungsumgebung. Sie bietet die volle Flexibilität einer modernen Hochsprache, ohne das sich der Entwickler mit aufwendiger Syntax befassen muss. Die Vorteile von LabVIEW sind:

- grafische Programmierung,
  - intuitiv zu bedienen,
  - kommunikativ, Programme können leichter erläutert und besprochen werden ,
- Debugging-Umgebung,
- einfache Handhabung von Unterprogrammen (Sub-VI),
- sehr gute Hardware-Anbindung,
- umfangreiche Bibliotheken,
- plattformübergreifend.

Das Ergebnis ist eine stark verkürzte Entwicklungszeit. Es ist möglich, innerhalb kürzester Zeit lauffähige Programmteile zu erstellen und zu testen. LabVIEW bietet damit ein erhebliches Potential an Zeit- und Kosteneinsparung für ein breites Spektrum von Anwendungen.

2.3.2 ObjectVIEW™

Trotz der im vorangegangenen Abschnitt herausgearbeiteten Vorteile von LabVIEW treten insbesondere bei sehr umfangreichen und komplexen Anwendungen Probleme auf. Um bei der Erstellung solcher Anwendungen nicht auf die Vorzüge der grafischen Programmierung verzichten zu müssen, wurde von der Vogel Automatisierungstechnik GmbH ObjectVIEW, ein Toolset für LabVIEW, entwickelt. Mit ObjectVIEW halten die Techniken und Vorzüge der in Abschnitt 2.1.5 beschriebenen Objekttechnologie in LabVIEW Einzug. Deren Anwendung ist der Schlüssel zur erfolgreichen Realisierung umfangreicher Softwareprojekte in verteilten Softwaresystemen.

Grundlage vom ObjectVIEW sind aktive Objekte (Objekte mit Prozessen), die untereinander ereignisgesteuert kommunizieren.

Die hierarchische Strukturierung des Problems in Teilprozesse ist direkt grafisch in ObjectVIEW umsetzbar. Objekte können zur Modellierung des Objektlebenszyklusses State-Charts oder Petri-Netze enthalten. Die Kommunikation, auch über Knotengrenzen hinweg, erfolgt durch Adressierung über URL´s (Middleware). Damit sind intra- und internetbasierte verteilte, intelligente Systeme einfach realisierbar. Die Kombination LabVIEW und ObjectVIEW sind nicht nur Entwicklungsbasis für einfach skalierbare und sehr gut wartbare Applikationen, die Rapid Prototyping und Rapid Application Development ermöglichen, sondern auch ideal für die Entwicklung neuartiger Entwurfskonzepte [ObjV02].

Ausführliche Produktbeschreibung, Beispielanwendungen und verschiedene Testversionen sind unter http://www.objectview.de zu finden.

3 Darstellung des Ausgangszustandes

3.1 Technologien des WWW

3.1.1 Allgemein

Im Kapitel Grundlagen wurde bereits die grundlegende Funktion und die Bedeutung des WWW herausgearbeitet. Doch seit den Anfängen des WWW unterlag dieses einer ständigen Weiterentwicklung. Einfache Websites sind zu Web-Anwendung ausgebaut worden.
Als Web-Anwendung wird hierbei eine interaktive Anwendung verstanden, bei der die Benutzeroberfläche, im Unterschied zu Client/Server-Anwendungen, in einem Webbrowser angezeigt wird. Die Web-Anwendung benutzt die Website als Front End [HeiBa00].
Die dabei entstandenen Technologien erlauben es nun auch, Anwendungen webfähig zu gestalten, die bisher losgelöst vom Web innerhalb einer homogenen Client/Server-Architektur laufen. Innerhalb der unterschiedlichsten Unternehmensbereiche geht der Trend dahin, Anwendungen komplett web-basiert zu gestalten. Daraus ergeben sich eine Reihe von Vorteilen:

- Spezielle, plattformabhängige Clientprogramme werden durch einen einfach zu bedienenden Web-Browser ersetzt.
- Es vereinfacht sich die Wartung, da das Clientsystem lediglich aus Betriebssystem und Web-Browser besteht und die Unternehmenslösung auf einem oder wenigen Servern liegt.
- Die Hardware-Anforderungen an das Clientsystem werden reduziert, da nur noch der Web-Browser ausgeführt wird.
- Ohne zusätzliche Infrastrukturen zu schaffen, kann der Zugriff auf das Anwendungssystem von nahezu jedem beliebigen Ort aus erfolgen.

Die Technologien, die dem Entwickler dabei zur Verfügung stehen, kann man grob in clientseitige und in serverseitige Techniken unterteilen.
Im folgendem soll nun ein kleiner Abriss dieser Technologien erfolgen , welche die fortschreitende Dynamisierung des Webs hervorgebracht hat.

3.1.2 Clientseitige Webtechnologien

Clientseitige Techniken sind für die Präsentation der Daten im Web-Browser (Web-Client) und die Interaktion mit dem Nutzer zuständig.

- **Plug-Ins:**

Plug-Ins sind Programmmodule, welche die Basisfunktonalität des Browsers erweitern. Solche Module müssen erst auf der Clientmaschine installiert werden, damit die Funktion genutzt werden kann. Sie sind meist browserspezifisch und plattformabhängig.

- **Skripte**

Clientseitige Scripte ermöglichen eine Interaktion mit dem Browser, die über das bloße Anklicken eines Links hinausgeht. So ist es z.B. möglich, Buttons und Eingabefelder innerhalb der Web-Seite bereitzustellen. Eingegebene Daten werden dann an den Webserver gesendet. Ein Script selbst kann nicht auf den Server zugreifen, sondern nur eine gewisse Vorverarbeitung der Eingabedaten vornehmen und damit den Server entlasten. Soll eine Weiterverarbeitung der Daten erfolgen, ist eine serverseitige Programmierung erforderlich. Der Quellcode eines Scripts wird direkt in den HTML-Code integriert und vom Browser interpretiert. Haupteinsatzgebiet von clientseitigen Scripten sind Formulare innerhalb von Web-Seiten. Der bekannteste Vertreter der Scriptsprachen ist sicherlich JavaScript.

- **Applets**

Applets stellen die Brücke zwischen Java und dem Internet dar. Ihnen ist hauptsächlich der Erfolg der Programmiersprache zu verdanken. Applets sind Java-Programme, die über das Web verbreitet und im Standardwebbrowser ausgeführt werden können. Durch sie steht nahezu der gesamte Sprachumfang einer modernen Programmiersprache im gesamten Web zur Verfügung. Die Nutzung aller Merkmale von Java ermöglicht eine Interaktion mit dem Webbrowser auf höchstem Niveau. Wesentlich für die Eignung von Java als Sprache für das Internet ist deren Plattformunabhängigkeit. Ein einmal kompiliertes Java-Programm läuft also auf verschiedenen Plattformen. Dazu gibt es für die unterschiedlichen Plattformen entsprechende Interpreter, die sogenannte Virtual Machine, kurz VM. Ohne explizit auszuführende Installationsroutinen lädt der Classloader der VM den Bytecode und führt das Programm aus. Ein Java-fähiger Browser enthält eine solche Virtual Machine und dazu die Laufzeitbibliothek, welche benötigt wird, die Ausführung des Applets zu unterstützen.

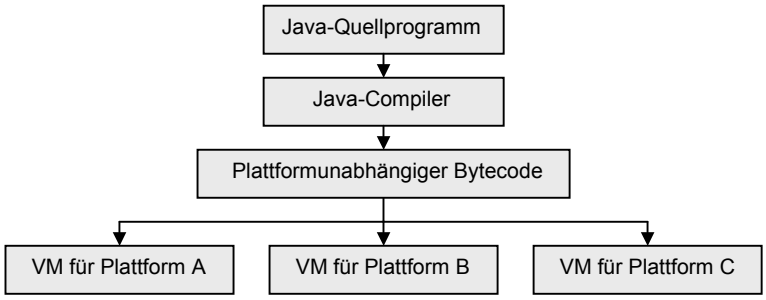

Abb. 3-1: Übersetzungsprinzip von Javaprogrammen

3.1.3 Serverseitige Webtechnologien

Serverseitige Techniken laufen meistens darauf hinaus, dass durch Anfordern einer bestimmten Web-Seite oder Betätigung eines Buttons eine Aktion auf dem Server ausgeführt wird. Das Ergebnis dieser Aktionen ist dann meist eine dynamisch erzeugte HTML-Seite, die als Ergebnis der Anforderung an den Browser zurück gesendet wird.

- **CGI**

Die Abkürzung CGI steht für Common Gateway Interface und spezifiziert eine Schnittstelle zwischen Webserver und einem externen Programm. Bei CGI wird eine HTTP-Anfrage des Web-Browsers an den Server als Kommandozeile genutzt. Die URL bezeichnet in diesem Fall keine HTML-Seite sondern ein ausführbares Programm und die notwendigen Parameter. Der Web-Server erkennt die Anfrage als CGI-Kommando und startet des entsprechende Programm in einem eigenen Prozess. Das Ergebnis der Programmabarbeitung ist dann eine dynamisch generierte HTML-Seite, welche zurück zum Browser geschickt wird.

- **Skripte**

Serverseitige Scripte sind von Webserver selbst interpretierte Programme und werden im RAM der Servermaschine ausgeführt. Ähnlich wie bei dem Aufruf eines externen Programms erhält das Script Parameter vom Browser und führt eine Aktion durch, z.B. eine Datenbankabfrage. Die Ausgabe ist dann wieder eine HTML-Seite.

- **Servlets**

Ein Servlet ist ein Java-Programm, welches auf dem Webserver läuft. Was Applets für den Browser sind, sind Servlets für den Server, nur ohne GUI. Auch Servlets erstellen dynamisch Web-Seiten bei entsprechender Anforderung. Der Vorteil der Servlets liegt in der Plattform-unabhängigkeit.

- **Java Server Pages (JSP)**

Java Server Pages sind eine Weiterentwicklung und Ergänzung zu Servlets. Sie enthalten neben den HTML-Tags spezielle JSP-Tags, die für die dynamischen Komponenten der Seite stehen. Greift der Client auf die Seite zu, wird diese in ein Servlet übersetzt. Die Ausgabe des Servlets wird dann an den Client geschickt.

- **Active Server Pages (ASP)**

Bei ASP handelt es sich um ein serverseitiges Skript-Konzept der Firma Microsoft und ist entsprechend auch nur lauffähig innerhalb einer Microsoftumgebung. Eine Active Server Page ist eine HTML-Datei die neben den üblichen Elementen, wie HTML, Bildern, Applets oder ActiveX Steuerelemente auch Scriptprogramme enthält.

Die Scriptprogramme werden auf dem Server ausgeführt, bevor die Seite an den Browser übertragen wird.

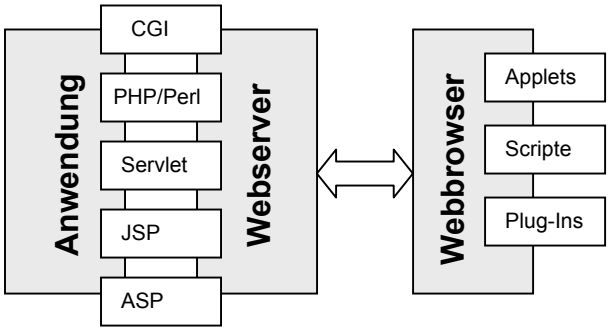

Abb. 3-2: Übersicht Webtechnologie

3.1.4 Web-Architektur vs. Client/Server-Architektur

In den vorhergehenden Abschnitten wurde herausgestellt, dass das Internet und damit auch das WWW auf der Client/Server-Architektur basiert. Für verteilte Anwendungen und deren Entwicklung ergeben sich jedoch z.T. erhebliche Unterschiede. Im folgenden sollen nun beide Architekturen kurz charakterisiert und verglichen werden.

- **Client/Server-Architektur**

In der Regel ist der auf dem Client befindliche Teil der Anwendung bis zur Abmeldung mit dem Server permanent verbunden. Die max. Anzahl der nebenläufigen Benutzer liegt zur Entwurfszeit fest. Die Umgebung ist meist bekannt und kann beeinflusst werden (Betriebssysteme, Datenbanken, ...). Bei der Entwicklung müssen Client und Server gleichermaßen Beachtung finden.

Es ergeben sich folgende Vor- und Nachteile [HelBa01]:

+ Da der Server pro Client nur eine Verbindung verwaltet, ist es einfach eine Benutzersitzung zu verfolgen,
+ Durch den Einfluss der Laufzeitumgebung kann die Systemkomplexität reduziert werden,
+ Die bekannte Anzahl der max. Systembenutzer erleichtert den System-Entwurf,
- Schlecht skalierbar, d.h. nur aufwendig auf andere Benutzerzahlen einstellbar,
- Aufwendige Verteilung der Clientsoftware, da sie auf jeden Client installiert werden muss,
- Änderung der Plattform führen u.U. zur Neuprogrammierung von Anwendungsteilen.

- **Web-Architektur**

Zwischen Web-Client und Web-Server gibt es keine permanente Verbindung. Aufgrund des HTTP wird bei jeder Anforderung einer Website eine TCP-Verbindung mit dem Web-Server aufgebaut. Die Anfrage wird gesendet, vom Server bearbeitet und nach Rücksendung der Antwort wird die Verbindung wieder abgebaut. Erst mit dem HTTP 1.1 wird die Verbindung für eine kurze Zeitperiode aufgebaut. Web-Anwendungen haben potentiell eine unbegrenzte Anzahl von Benutzern. Auf die Laufzeitumgebung des Clients hat der Entwickler keinen Einfluss. Es ergeben sich hier folgende Vor- und Nachteile:

+ es sind potentiell hohe Benutzerzahlen möglich,
+ gute Skalierbarkeit und Wartbarkeit,
+ keine Verteilungsprobleme, da keine anwendungsspezifische Software auf dem Client installiert werden muss,
- durch das verbindungslose und sitzungslose HTTP 1.0 ist es aufwendig, den Zustand während und zwischen den Sitzungen zu speichern und zu verfolgen,
- es müssen in der Regel mehrere unterschiedliche Web-Browser unterstützt werden.

Die Vor- und Nachteile der Architekturen zeigen, dass vor Beginn des Entwurfs entschieden werden muss, welche Architektur letztendlich verwendet werden soll. Es ist jedoch in der Praxis möglich, die Anwendung für beide physische Architekturen zu entwerfen.

3.2 LabVIEW™ und das WWW

3.2.1 Allgemein

Entsprechend der Aufgabenstellung sind zunächst die vorhandenen Möglichkeiten zur Fernsteuerung von LabVIEW-Applikationen über das Web zu untersuchen.

Seit der Version 6i wird LabVIEW von NI als internetoptimiert angepriesen, was auch durchaus seine Berechtigung hat. So ist es beispielsweise möglich, über Low-Level-Kommunika-tion (TCP/IP) Daten mit anderen Anwendungen auszutauschen oder HTML-Seiten über den integrierten Webserver aufzurufen. Passend dazu gibt es Funktionen, die eine HTML-Datei erzeugen, in der ein statisches Bild eines Frontpanels eingebettet ist.

Die bestehenden Möglichkeiten, die LabVIEW mit dem WWW verbindet, sollen in den folgenden Abschnitten aufgezeigt werden. Zu Beginn wird jedoch der Begriff „webfähige Applikation“ mit LabVIEW näher durchleuchtet.

3.2.2 Applikationen mit LabVIEW™ / ObjectVIEW™

Eine Applikation besteht aus mehreren einzelnen, untereinander abgegrenzten Softwarekomponenten. Besteht nun eine Applikation aus einer Vielzahl von Komponenten, dann ist eine stärkere Strukturierung sinnvoll. Eine häufig verwendete Strukturierungsform ist die Zuordnung der Komponenten in Schichten. Dabei hat sich eine 3-Schichtstruktur als besonders geeignet herausgestellt. Befindet man sich in der objektorientierten Welt von ObjectVIEW, sind es entsprechend unabhängige Objekte die den einzelnen Schichten zugeordnet werden.

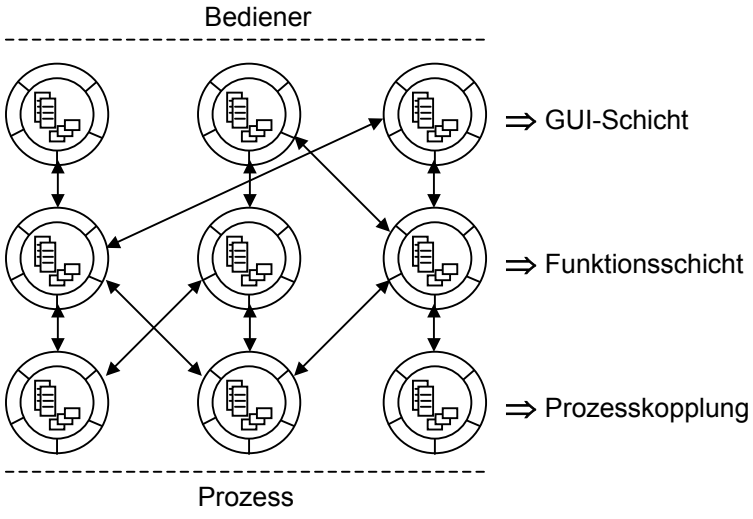

Abb. 3-3: Allgemeine Applikationsstruktur

Soll eine derart aufgebaute Applikation webfähig werden, tritt anstelle der GUI-Komponente bzw. des GUI-Objekts der Web-Browser auf einem entfernten Rechner. Die Struktur der Applikation bleibt unbeeinflusst. Im konkreten Fall bildet meist das Frontpanel eines speziellen VI´s in der obersten Applikationsschicht das GUI der gesamten Applikation. Die Funktionen dieses Frontpanels muss nun der Web-Browser übernehmen. Mit diesem kann der Benutzer die GUI-Schicht nachbilden und steuert so den Prozess über das Web.
Ist daher im Rahmen dieser Arbeit von Anwendungs-VI die Rede, bezieht sich dies genau auf dieses VI der obersten Applikationsschicht. Die darunter liegenden Schichten bleiben unbeachtet.

3.2.3 Das Internet Toolkit

Bei dem Internet Toolkit der Firma NI handelt sich um ein Add-On-Produkt zu LabVIEW. Es erweitert dessen Internetfähigkeiten durch folgende Funktionen:

- HTML-VI´s zu Generierung von HTML-Seiten,
- CGI-Funktionalität,
- E-Mail Funktionen,
- FTP Funktionen,
- Telnet,
- HTTP-Server mit größeren Funktionsumfang als der Standard-Webserver.

Mit der umfangreichen Funktionspalette ist es dem Entwickler auf Programmierebene möglich die Dienste des Internets in seiner Anwendung zu integrieren.

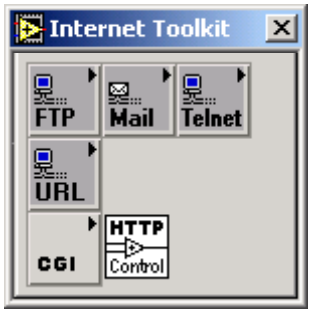

Abb. 3-4: Funktionspalette des Internettoolkits für LabVIEW

3.2.4 Remote Panel

Remote Panel ist eine Technologie von National Instruments und in beschränkter Form Bestandteil von LabVIEW 6.1, der z.Z. aktuellsten Version. Diese Technologie stellt die Fortsetzung der Integration Web-Funktionalität mittels Tools in LabVIEW dar. Mit wenigen Mausklicks ist der Entwickler in der Lage Frontpanels im WWW zu publizieren. Dem Anwender wird damit die Möglichkeit gegeben, Applikationen über das Netzwerk fernzusteuern. Dabei kann der Client LabVIEW oder ein Standardwebbrowser (Internetexplorer, Netscape, ...) sein.

Arbeitsweise von Remote Panel

Bei der Verwendung von Remote Panel wird serverseitig die komplette Funktion, d.h. der vollständige Programmcode inkl. aller Sub-VI´s, realisiert. Zum Client wird lediglich das Front Panel der Applikation übertragen. Dazu ist es zwingend erforderlich, dass clientseitig eine LabVIEW-Installation bzw. eine LabVIEW-Runtime Engine vorhanden ist. Über ein Kontextmenü (rechter Mausklick auf das übertragene Frontpanel) kann Control-Recht beantragt werden, welches letztendlich das Bedienen des VI´s erlaubt. Prinzipiell sind mehrere Verbindungen gleichzeitig möglich, jedoch nur ein Client kann Control-Recht besitzen. Serverseitig können sämtliche Verbindungen mit dem Remote Panel Connection Manager überwacht werden.

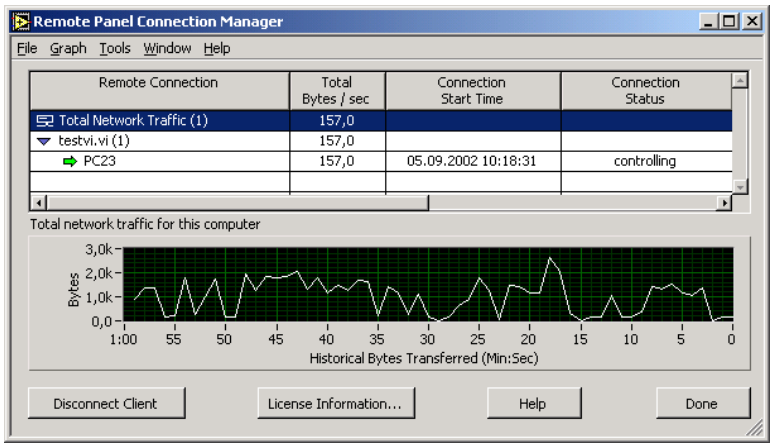

Abb. 3-5: Remote Panel Connection Manager

Beispielapplikation

- Client ist LabVIEW:

Über die Operation „Connect to Remote Panel“ kann aus LabVIEW heraus, durch Angabe der IP-Adresse, VI-Namen und Port-Nummer, direkt eine Verbindung zu einem entfernten Rechner mit laufenden VI hergestellt werden. Bei erfolgreicher Verbindung erscheint dann das Front Panel auf der Client-Maschine in einem neuen Fenster der Entwicklungsumgebung.

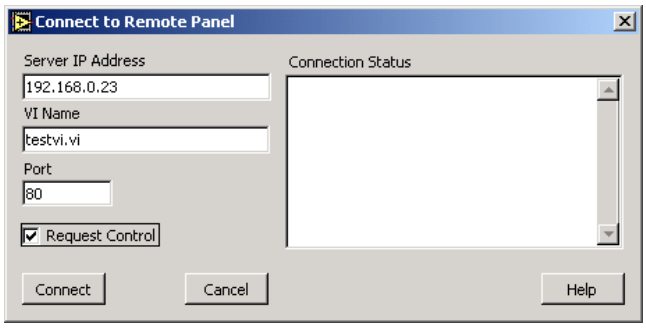

Abb. 3-6: „Connect to Remote Panel“ in LabVIEW

- Client ist ein Webbrowser:

Über das „Web Publishing Tool“ kann der Entwickler, einfach und schnell die HTML-Seite erstellen, in welcher das fernzusteuernde Frontpanel eingebettet wird. Es können verschiedene Optionen gewählt und zusätzlicher Text editiert werden. Nach einer möglichen Vorschau im Browser erfolgt die Speicherung im gewünschten Verzeichnis des LabVIEW-Webservers. Beim Aufruf dieser Seite wird dem Browser mit dem <Object> Tag die Information über das benötige Plug-In geliefert, mit dessen Hilfe dann das Front Panel auf der Client-Maschine läuft.

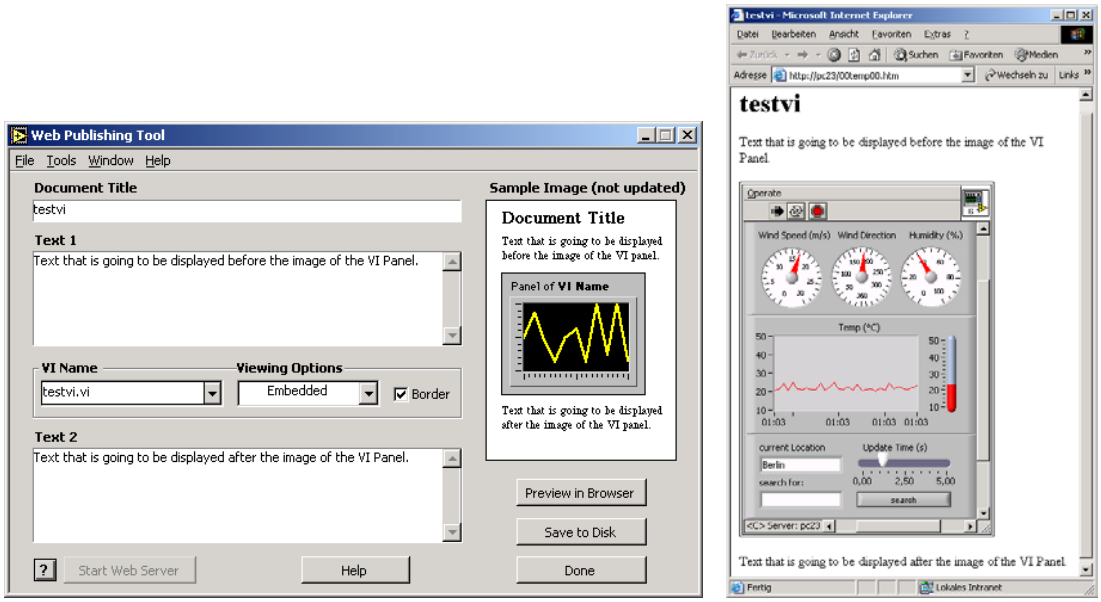

Abb. 3-7: “Web Publishing Tool” in LabVIEW und Resultat im Browserfenster

3.2.5 AppletVIEW™

AppletVIEW ist ein Toolkit der Firma Nacimiento aus Austin, USA. Es ermöglicht die Anzeige und Bedienung von LabVIEW-Frontpanels mittel Standard Webbrowser (z.B. Internet-explorer, Netscape) über das Intranet/Internet. Die Funktion von AppletVIEW basiert auf Java und TCP/IP. Es liegt derzeit in der Version 2.0 vor und ist kompatibel mit LabVIEW 5 und 6, sowie Java 1.1 bis 1.3. Enthalten sind sämtliche Komponenten die zur Veröffentlichung von VI´s im Web erforderlich sind. Dazu gehören die Libraries mit den speziellen VI´s, welche die Verbindung des zu veröffentlichen Instrumentes zu AppletVIEW herstellt, die Serverkomponenten und die eigentlichen Tools. Diese Tools sind im einzelnen:

- Server Monitor,
  zeigt den Status des AppletVIEW Servers an und ermöglicht den Zugang zu den anderen Tools über eine Toolbar,
- Applet Builder,
  dient zur Erstellung des erforderlichen Java Applets, welches das Frontpanel des zu veröffentlichen VI´s nachbildet,
- Channel Organizer,
  erstellt und konfiguriert die Kommunikationskanäle zwischen Applet und VI
- Server Manager,
  erlaubt das Managen der drei AppletVIEW Server: http, instruments und proxy,
- Log-Tool,
  zeigt empfangene und von AppletVIEW verarbeitete Events an,
- Preferences-Tool,
  ermöglicht das Einstellen der Level des Log-Tools.

Das Toolkit kann in der Evaluation-Version unter *www.Nacimiento.com* heruntergeladen werden.

Arbeitsweise von AppletVIEW™

Die Philosophie von AppletVIEW ist, das Frontpanel des Anwendungs-VI´s als Java Applet nachzubauen. Dieses Applet wird dann in eine Internetseite eingebettet und beim Aufruf der Seite geladen und im Browser ausgeführt. AppletVIEW übernimmt dabei das Handeln der Verbindung zwischen Applet und LabVIEW, so dass die eigentliche Kommunikation auf TCP/IP-Ebene für den Entwickler transparent bleibt.

Zunächst muss das Diagramm des entsprechenden VI´s um die Funktionen zur Kommunikation mit dem Applet erweitert werden. Diese Funktionen sind von Nacimiento in verschiedenen Sub-VI´s gekapselt. Danach wird die späterer Bedienoberfläche des Applets mit dem Applet Builder erstellt. Mit vorgefertigten Elementen ist es möglich, diese Bedienoberfläche dem LabVIEW-Frontpanel nachzuempfinden. Die Anzeige- und Bedienelemente werden dann mit dem LabVIEW-VI synchronisiert und das Ganze als Konfigurationsdatei im viml-Format gespeichert. Das viml-Format ist eine XML-Spezifikation und erlaubt somit das Betrachten der Dateien in einem Texteditor.

Die aufzurufende HTML-Seite muss mit dem <APPLET> Tag und dem entsprechenden Code erweitert werden. Beim Aufruf der HTML-Seite wird dem Browser dann mit dem <APPLET> Tag signalisiert, dass ein Java Applet zu laden ist. Der Browser startet die Java Virtual Machine und fordert die Java Dateien an. Diese sind in der Form „AppletVIEW.jar" auf dem Server. Ist das Applet gestartet, wird als erstes die mit dem Applet Builder erstellte viml-Datei angefordert und ebenfalls geladen. Das Applet versucht nun über den angegebenen Port (Standard: 4747) die Verbindung zur LabVIEW-Anwendung aufzunehmen. Das entsprechende VI muss vollständig geladen und gestartet sein. Eine LED in der linken, oberen Ecke des Applets informiert über den aktuellen Status der Verbindung. Ist diese erfolgreich, kann der Webserver nun am HTTP-Port (Standard: 80) weitere Verbindungen entgegennehmen.

Kommunikation mit LabVIEW™

Die Low-Level-Kommunikation übernimmt vollständig AppletVIEW. Dafür muss vom Programmierer spezifiziert werden, welche Daten gelesen/geschrieben werden und welche Komponenten des Applets mit welchen des Server-VI´s kommunizieren. Die Kommunikation von AppletVIEW mit LabVIEW bedeutet somit, Daten von einer Komponente zu lesen und Daten auf eine Komponente zu schreiben.

Um Daten auf eine Komponente des Applets zu schreiben, stehen zwei Möglichkeiten zur Auswahl. Zum einem das einzelne Schreiben auf eine Komponente mit „Write Applet.vi"
oder das Schreiben auf mehrere Komponenten gleichzeitig mit „Write Multiple Components to Applet.vi".

Beim Lesen von Appletkomponenten gibt es ebenfalls verschiedene Varianten. Prinzipiell stehen für das Applet zwei Modi zur Verfügung: der Event-Modus und der Poll-Modus. Der Standardfall für alle Applets ist der Event-Modus.

Sobald sich der Wert einer Bedienkomponente des Applets ändert, z.B. durch Drehen eines Bedienknopfes, wird der aktuelle Wert an den Datenport gesendet. Das AppletVIEW-VI „Read Applet.vi“ liest diesen Wert, sobald dieser über das Netz empfangen wurde. Innerhalb der Anwendung muss nun sichergestellt werden, dass das VI regelmäßig (in einer Schleife) aufgerufen wird, um diesen Wert der Anwendung zu übergeben und den TCP/IP Puffer zu leeren (serverseitiges Polling).
Im Pollmodus werden die Daten bei Änderung in eine Datenstruktur übernommen und nicht automatisch gesendet. Die serverseitige LabVIEW-Anwendung entscheidet, wann und welche Komponente gepollt wird.

Beispielapplikation

Bei der Erstellung einer Applikation mit AppletVIEW muss zunächst entschieden werden ob Serverfunktionalität enthalten sein soll. Serverfunktionalität heißt, dass die erstellte Anwendung selbst am eingestellten Port hört und auf eingehende Appletverbindungen wartet. Die dafür nötigen AppletVIEW-VI´s sind „Create Applet Listener.vi“ und „Wait on Applet Connection.vi“.
Die Alternative dazu ist die Kommunikation über den AppletVIEW-Server abzuwickeln. Dieser Anwendungsfall soll kurz in einer kleinen Beispielapplikation näher betrachtet werden. Dazu wird wieder das kleine LabVIEW Programm (testvi.vi) mit verschiedenen Anzeige- und Bedienelementen herangezogen. Dieses muss nun mit den AppletVIEW-VI´s erweitert werden, welche später die Kommunikation ermöglichen. Benötigt werden folgende Funktionen:

- Eröffnen einer Verbindung zum AppletVIEW-Server,
- Lesen und Schreiben der Daten,
- Beenden der Verbindung.

Analog dazu gibt es folgende AppletVIEW-VI´s:

- Connect To Server.vi,
- Read Write Refnums.vi,
- Close Applet Connection.vi.

Das “Connect To Server.vi” erhält als Parameter die Applet-ID (3) und übernimmt die für die verbleibenden Parameter die Standardwerte. Für das „Read Write Refnums.vi“ werden die Referenzen der Anzeige- und Bedienelementen, sowie die Komponenten-ID in äquivalenter Reihenfolge benötigt.

Das Diagramm des “testvi.vi” ändert sich wie folgt:

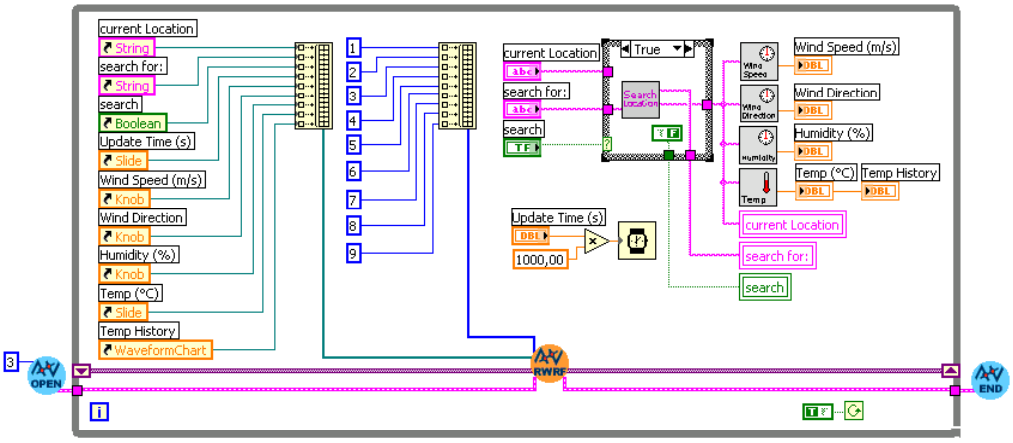

Abb. 3-8: Erweiterung des LabVIEW-Programmcodes bei Verwendung von AppletVIEW

Mit Hilfe des Applet Builder wird im nächsten Schritt die Bedienoberfläche des Applets erstellt. Jede Komponente erhält eine Komponenten-ID, welche mit den ID´s im LabVIEW-Diagramm übereinstimmen muss. Darüber hinaus muss der gesamten Datei eine Applet-ID (3) zugewiesen werden.

Um das Ergebnis im Browser betrachten zu können, muss am Schluss noch die HTML-Datei erstellt werden.

```
<html>
  <head>
    <title>My Applet</title>
  </head>
  <body>
    <applet
     code="com.nacimiento.appletview.applet"
     archive="AppletVIEW.jar,AppletVIEWui.jar"
     width="346" height="385">
     <param name="DataPort"   value="4749">
     <param name="ConfigFile" value="testvi.viml">
    </applet>
  </body>
</html>
```

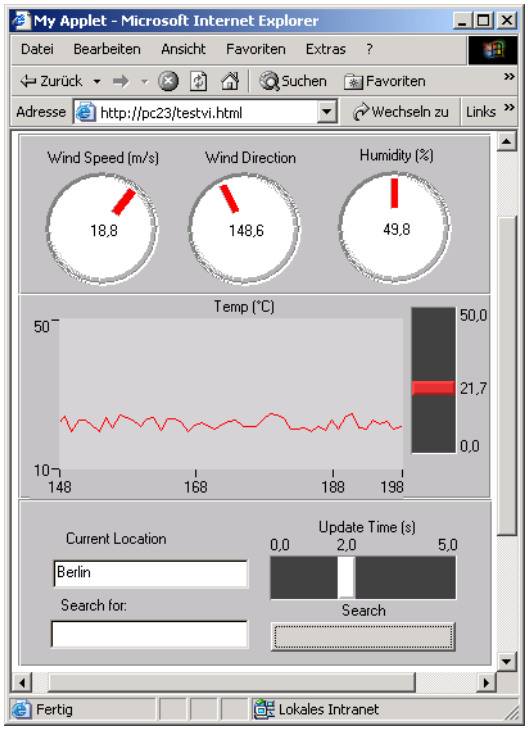

Abb. 3-9: HTML-Code und Client-“Frontpanel” mit AppletVIEW

4 Bewertung und Schlussfolgerungen

4.1 Beurteilung des Ausgangszustandes

4.1.1 Das Internet Toolkit

Mit dem Internet Toolkit erhält der Entwickler eine umfangreiche Erweiterung der Basisfunktionalität von LabVIEW in Richtung Dienste des Internets. Das WWW wird dabei mit der CGI-Schnittstelle und einer Reihe von VI´s zu dynamischen Generierung von HTML-Seiten berücksichtigt. Obwohl CGI eher zu den veralteten Web-Technologien zählt, bietet es zusammen mit der grafischen Programmierung unter LabVIEW eine gute Möglichkeit zur Erstellung von Web-Anwendungen. Eine solche Web-Anwendung erhält damit automatisch sämtliche Schnittstellen, die LabVIEW bietet. Da die Steuerelemente beim Einsatz von CGI jedoch sehr begrenzt sind, geht dessen Anwendung eher in Richtung Formulare auf der Web-Seite, Datenbank- oder Prozesswert-Abfrage oder der gelegentlichen Übertragung eines Frontpanel-Abbildes auf Anforderung des Benutzers.
Der Preis des Internet Toolkits beträgt 655,- Euro.

4.1.2 LabVIEW™ Remote Panel

Die Lösung von NI stellt eine recht komfortable Möglichkeit dar, Frontpanels im Netz verfügbar zu machen. Der Anforderung, VI´s einfach und schnell zu publizieren und damit Anwendungen webfähig zu gestalten, wird diese Technik gerecht.
Vorteile:

- Einfache, intuitive Bedienung,
- Keine Erweiterung des Programmcodes erforderlich,
- clientseitig gute Grafik, sowie verzögerungsarme Anzeige der Controls bei Bedienung.

Nachteile:

- Clientseitig muss eine LabVIEW-Installation oder eine LabVIEW-Laufzeitumgebung vorhanden sein, welche das benötigte Plug-In für den Browser bereitstellt. Dadurch ist eine Portierung nur auf Plattformen möglich, für die eine LabVIEW Runtime erhältlich ist. Geräteklassen, wie z.B. Pocket PC sind daher von der Nutzung ausgeschlossen.
- Serverseitig muss sich das VI im Speicher befinden und wird nach Beendigung der Verbindung nicht aus dem Speicher entfernt

- Standardmäßig ist nur eine Verbindung möglich, für mehr Verbindungen werden Lizenzgebühren fällig:
  - 5 Verbindungen 410,- Euro,
  - 20 Verbindungen 1.645,- Euro,
  - 50 Verbindungen 4.145,- Euro.
- Keine Instanzierung des Anwendungs-VI zur Laufzeit möglich,
- Keine Open-Source-Anwendung, daher sind keine Anpassungen möglich.

Optionen:

In Kombination mit dem Internet Toolkit von National Instruments (695,- Euro), erhält man einen Open-Source HTTP-Server und CGI-Funktionalität, so dass entsprechende Erweiterungen möglich werden. Die nicht unerheblichen Lizenzgebühren müssen jedoch auch hier einkalkuliert werden.

4.1.3 AppletVIEW™

Das untersuchte Toolkit AppletVIEW lieferte im Ergebnis die von der Firma Nacimiento versprochenen Resultate. Es ergibt sich folgende Bewertung:

Vorteile:

- Clientseitig ist kein Plug-In erforderlich, es wird lediglich eine Unterstützung des Browsers von Java vorausgesetzt, sämtliche Webplattformen sind erreichbar,
- Es sind keine zusätzlichen Lizenzgebühren für Verbindungen fällig,
- Verzögerungsarme Rückkopplung bei Bedienung der Controls,
- Umfangreiche Library, die es ermöglicht Anwendungen unter LabVIEW webfähig zu gestalten.

Nachteile:

- Erhöhter Programmier- und Konfigurationsaufwand durch Erstellen des Applets und umfangreiche Erweiterungen des LabVIEW-Codes,
- Nicht alle in LabVIEW vorhandenen Kontroll-Typen werden unterstützt,
- Ingesamt ungenügende Optik der Bedienoberfläche im Browser,
- Applet muss für jedes VI neu erstellt und abgespeichert werden,
- Differenzierung der Nutzerrechte muss über unterschiedliche Bedienoberflächen erfolgen,
- Datenaustausch durch Polling,
- Anschaffungskosten von 895 US$.

4.2 Schlussfolgerung

Für eine Verwendung der Technologie von Remote Panel spricht die einfache und komfortable Handhabung serverseitig und die gute Grafik, sowie der gute Bedienkomfort clientseitig. Deutlich dagegen sprechen die recht hohen Lizenzgebühren in Abhängigkeit der max. möglichen Anzahl der Verbindungen. Erheblich sind die Nachteile, die sich durch das zwingend erforderlichen Browser-Plug-In ergeben. Dieses wird nur bereitgestellt, wenn mindestens eine LabVIEW-Laufzeitumgebung auf dem Client installiert ist. Damit ist der Einsatz dieser Technologie auf Plattformen beschränkt, auf denen diese Installation möglich ist. Ein wesentlicher Vorteil den der Einsatz der Webtechnologie bietet, geht damit verloren.
In diesem Punkt überzeugt AppletVIEW. Hier ist clientseitig keine zusätzliche Installation nötig. Die Unterstützung zur Ausführung von Java-Applets ist mit der Verwendung eines Standardwebbrowsers gesichert. Die Nachteile bei der Verwendung von AppletVIEW sind bei der Erstellung der Anwendung zu sehen. Der Anwender muss ein sehr spezielles VI programmieren und sich zusätzlich um das clientseitige Userinterface kümmern, dessen Erscheinung völlig losgelöst vom LabVIEW Front Panel ist. Eine Webanwendung mit AppletVIEW bedeutet, automatisch doppelten Aufwand betreiben zu müssen.
Zusammenfassend kann man also feststellen, dass Remote Panel die Webtechnologie nicht vollständig durchdringt und AppletVIEW nicht vollständig LabVIEW durchdringt.
Vorteilhaft wäre also eine Kombination der Vorteile aus beiden Varianten. Diese kann jedoch nur innerhalb eines neuen Webinterfaces erfolgen.

4.3 Ableitung der zu realisierenden Funktionalität

4.3.1 Allgemein

Die Kombination der Vorteile aus den bestehenden Varianten ergibt sich im wesentlichen aus der einfachen Anwendung und der Offenheit für sämtliche Webplattformen.
Daher soll sich die Realisierung des neuen Webinterfaces im Punkt Anwendung an der Lösung von NI orientieren, d.h. es soll ein Tool entstehen, mit dessen Hilfe der Anwender in wenigen Schritten seine Anwendung webfähig gestalten kann. Gleichzeitig soll es aber keine Einschränkung hinsichtlich der Web-Plattform geben. Ein weiterer wichtiger Punkt ist, dass keine Programmierarbeit für den Client notwendig ist und das kein bzw. nur ein geringer Eingriff in den Programmcode des Anwendungs-VI erfolgen muss.

4.3.2 Anforderungen an den Client

Der entfernte Zugriff auf das Server-VI soll mittels Standardwebbrowser und ohne zusätzliches Plug-In erfolgen. Nur so hält man sich sämtliche Webplattformen offen und reduziert den Installations- und Verteilungsaufwand. Als GUI im Browser soll der Nutzer das Frontpanel seiner Anwendung 1:1 wiederfinden. Für den Veröffentlichungsvorgang ist es wichtig, dass für die Client-Seite kein zusätzlicher Programmieraufwand entsteht, d.h. die Client-Seite bleibt bei der Benutzung des neuen Webinterfaces transparent.

4.3.3 Anforderungen an das Anwendungs-VI

Auf der Server-Seite befindet sich zunächst das Anwendungs-VI, auf welches der entfernte Zugriff über das Web erfolgen soll. Mit dem Webinterface wird diese Anwendung nun zur Webanwendung. Die Verzahnung der Anwendung mit dem Webinterface soll so gering wie möglich gehalten werden, d.h. es soll keine bzw. nur eine geringe Erweiterung bzw. Änderung des Programmcodes der Anwendung erfolgen. So ist es möglich, dass Anwendungen bei denen kein Zugriff auf den Programmcode existiert, ebenfalls über das Web bedienbar sind. Zudem reduziert sich der Gesamtaufwand zur Generierung einer webfähigen Anwendung. Sofern das angeforderte VI noch nicht läuft, soll es geladen, ausgeführt und nach Beendigung der Verbindung wieder geschlossen werden.

4.3.4 Benutzerspezifische Instanzen

Benutzerspezifische Instanzen heißt, dass jeder Benutzer der mittels Browser auf die Anwendung zugreift, seine eigene Instanz dieser Anwendung erhält. Es ist damit möglich dem Nutzer die Funktion einer Anwendung unabhängig von anderen Benutzern bereitzustellen. Damit wird erreicht, dass eine Anwendung mit benutzerspezifischen Parametern ausgeführt werden kann und jeder Nutzer das für sich gewünschte Ergebnis erhält.
Die Anwendung selbst ist als Klasse auf dem Server abgelegt, von der zur Laufzeit Instanzen gebildet werden.

4.3.5 Ereignissteuerung

Ereignisgesteuerte Kommunikation bietet eine Reihe von Vorteilen gegenüber der zyklischen Programmabarbeitung und Kommunikation.

Daten werden nur übertragen, wenn tatsächlich neue Informationen anstehen. In erster Konsequenz ergibt sich daraus eine deutliche Schonung der Netzwerkressourcen.
Da nun auch die Verarbeitung der Daten auf ein Minimum beschränkt wird, ergibt sich weiterhin eine erheblich verringerte Rechnerlast.
Für das zu realisierende Webinterface heißt das konkret, das Frontepanel in Browser wird nur dann aktualisiert, wenn sich auch Änderungen auf dem Frontpanel der Anwendung ergeben haben. Das gleiche soll für Bedienhandlungen im Browser gelten. Daten werden nur bei tatsächlich stattfindenden Maus- und Tastaturereignissen an das Webinterface übermittelt und von diesem an das Anwendungs-VI weitergeleitet. So wird sichergestellt, dass die vorhandenen Ressourcen hauptsächlich der eigentlichen Anwendung zu Verfügung stehen.

5 Entwurf und Entwicklung

Das Kapitel „Entwurf und Entwicklung“ ist nur Bestandteil der kostenpflichtigen Version der Diplomarbeit.

e-Mail: info@vat.de

6 Umsetzung

Das Kapitel „Umsetzung“ ist nur Bestandteil der kostenpflichtigen Version der Diplomarbeit.

e-Mail: info@vat.de

7 Beispielanwendung

7.1 Aufgabenstellung

Mit einem Beispiel soll nun nachgewiesen werden, wie man seine Applikation mit dem neu entstandenen Webinterface webfähig gestalten kann. Als Anwendungsbeispiel wurde ein Fall aus der Gebäudeautomation gewählt, speziell die Steuerung der Büroetage der vat. Die konkrete Aufgabe besteht darin, es den Mitarbeitern mittels Webbrowser zu ermöglichen, jeden einzelnen Raum entsprechend seinen Möglichkeiten über das Intranet zu bedienen. Es soll weiterhin die Möglichkeit geben, die Bedienung losgelöst von einem bestimmten Arbeitsplatz mit dem PDA vornehmen zu können.

7.2 Ausgangszustand

7.2.1 Hardware

Den zentralen Teil der Hardware bildet ein I/O-System mit einem programmierbaren Feldbuscontroller der Firma Beckhoff. Die Steuerungselemente des Raumes (Taster, Lampen, Fensterkontakte, Temperatursensoren, ...) sind auf I/O-Klemmen geführt, die wiederum über einen Feldbus mit dem Feldbuscontroller verbunden sind, auf dem ein Steuerungsprogramm läuft. Zur Unterbringung dient ein Schaltschrank der darüber hinaus einen Panel-PC beherbergt, welcher jedoch zu diesem Zeitpunkt noch keine Funktion übernimmt.

7.2.2 Software

Der Anteil der Software ist derzeit auf das Steuerungsprogramm im Feldbuscontroller begrenzt. Die Programmierung erfolgt nach IEC 61131-3. Als Programmierumgebung steht TwinCat der Firma Beckhoff zur Verfügung.
Mit Hilfe dieses Programms werden die einzelnen Räume entsprechend ihrer Möglichkeiten gesteuert. Dazu gehört z.B. das Ein- und Ausschalten der Beleuchtung bei Betätigung der Taster und die Regelung der Raumtemperatur.

7.3 Umsetzung

7.3.1 Allgemein

Die Umsetzung gliedert sich in drei Teilschritte:

1. Erstellen des Anwendungs-VI als übergeordnete Steuerung,
2. Kopplung des Anwendungs-VI mit der Hardware,
3. Kopplung des Anwendungs-VI mit dem Webinterface.

Die Umsetzung der Beispielanwendung erfolgt nach folgender Struktur:

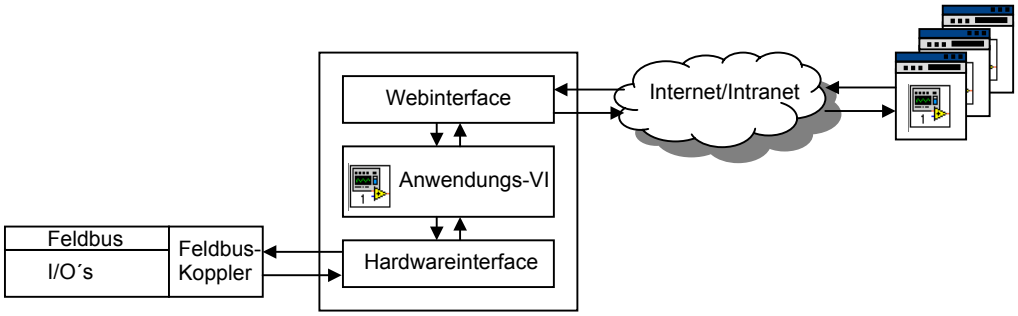

Abb. 7-1: Struktur der Beispielanwendung

7.3.2 Das Anwendungs-VI

Das Anwendungs-VI bildet die Schnittstelle zwischen Mensch und Steuerung. Um die Bedienung eines Raumes durch den Mensch zu ermöglichen, muss das VI die Funktionen des Raumes abbilden. Dazu gehört:

- Schalten der Beleuchtung,
- Einstellen der Heizungsbetriebsart (Aus, Nacht, Standby, Komfort, Hand, Auto),
- Einstellen der Solltemperatur für den Handbetrieb,
- Anzeige der Ist-Temperatur im Raum.

Darüber hinaus muss eine geeignete Möglichkeit zur Auswahl des zu bedienenden Raumes geschaffen werden. Die Möglichkeit für die Bedienung mit dem Pocket-PC wird in Form eines kompakten Front Panel´s berücksichtigt.

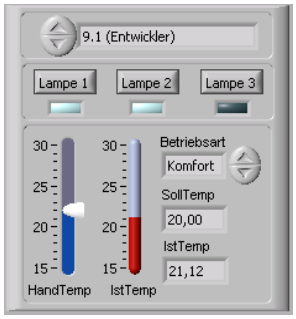

Abb. 7-2: Frontpanel des Anwendungs-VI für die Etagensteuerung

7.3.3 Kopplung mit der Hardware

Bei der Kopplung mit der Hardware kann auf Standardlösungen zurückgegriffen werden. Der Feldbuscontroller bietet eine Ethernet-Schnittstelle, so dass eine Kommunikation per TCP/IP mit dem Panel-PC möglich ist. Für den eigentlichen Datenaustausch bietet die Firma Beckhoff einen OPC-Server an. OPC ist die Spezifikation einer einheitlichen und herstellerunabhängigen Software-Schnittstelle. Diese eröffnet die Möglichkeit auf Prozessdaten verschiedener Systeme von beliebigen Herstellern in einheitlicher Art und Weise zuzugreifen. Da diese Schnittstelle vollständig und recht komfortabel von LabVIEW (auf Windows-Plattformen) unterstützt wird, kommt dieser Standard zum Einsatz.

Entsprechend der Konfiguration des OPC-Servers können sämtliche Items des Steuerungsprogrammes einer übergeordneten Anwendung verfügbar gemacht werden. Je nach Item kann es sich dabei um eine Hardwareadresse (I/O-Klemme) oder eine Merker-Adresse im Steuerungsprogramm handeln.

Die folgende Abbildung verdeutlicht den Kommunikationsweg:

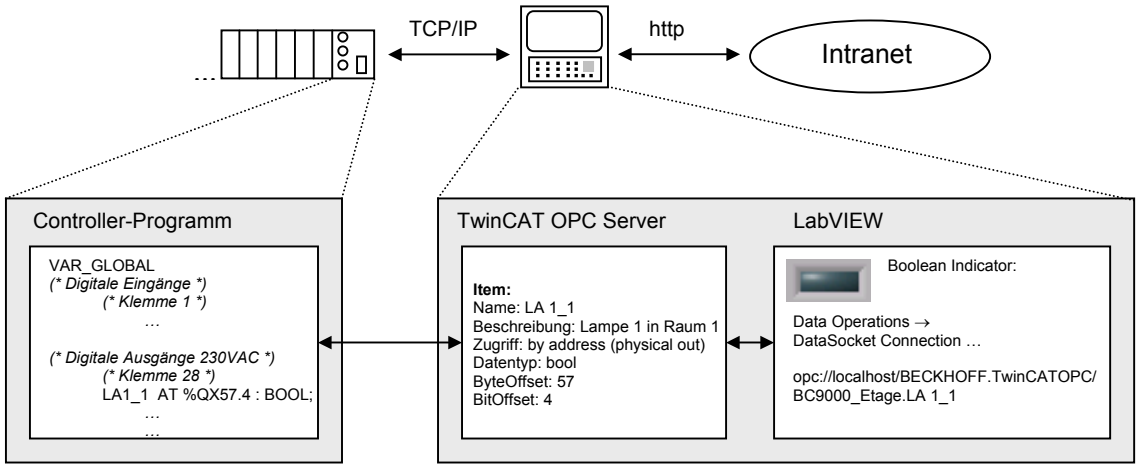

Abb. 7-3: Zugriff auf die Prozessvariablen mittels OPC-Server

7.3.4 Kopplung mit dem Webinterface

Grundsätzlich ist eine Kopplung mit dem Webinterface ohne Änderungen im Programmcode des Anwendungs-VI´s möglich. Dazu kann im Auto-Modus des Webinterfaces eine Updaterate einstellen, welche die Aktualisierung des Frontpanels im Browser übernimmt. Im Abschnitt 4.3.5 wurden jedoch die Vorteile der Ereignissteuerung herausgestellt. Zudem kommt, dass stets das aktuelle Frontpanel im Browser zu sehen ist.

Um jedoch die Möglichkeiten der Ereignissteuerung zu nutzen, muss das Anwendungs-VI die Fähigkeit besitzen, Ereignisse auszulösen. Dazu ist eine geringfügige Erweiterung des Programmcodes nötig.

Diese Erweiterung erfolgt durch das Einfügen eines bestimmten VI´s, welches bei Aufruf das gewünschte Ereignis auslöst. Ein solcher Aufruf muss also immer dann erfolgen, wenn sich Änderungen auf dem Frontpanel ergeben. Seit der Version 6.1 von LabVIEW gibt es eine Ereignisstruktur für Bedienhandlungen auf dem Frontpanel. Es ist damit möglich, für jedes Control auf dem Frontpanel einen korrespondierenden CASE in dieser Ereignisstruktur anzulegen. Ändert sich beispielsweise der Wert des Controls arbeitet das Programm den entsprechenden CASE ab. Somit bedeutet jede Änderung des Frontpanels einen Schleifendurchlauf. Folglich muss bei jedem Durchlauf das Ereignis für das Webinterface ausgelöst werden. Der Aufruf des dafür vorgesehenen VI´s erfolgt daher einfach nach der Ereignisstruktur innerhalb der Hauptschleife. Die folgende Abbildung zeigt die Funktion, die für das Ein- und Ausschalten der Lampen zuständig ist. Entsprechend des betätigten Controls wird eine bestimmte Bitfolge in ein Bytefeld geschrieben und dadurch die Flanke ausgelöst, welche schließlich das Schalten der Lampe bewirkt. Da diese Funktion für die vorhandenen Taster auf dem Frontpanel im wesentlichen gleich ist, sind diese in einem gemeinsamen CASE zusammengefasst. Im Anschluss wird das Ereignis „Frontpanel-Änderung“ ausgelöst und so das aktuelle Frontpanel an den Client übertragen.

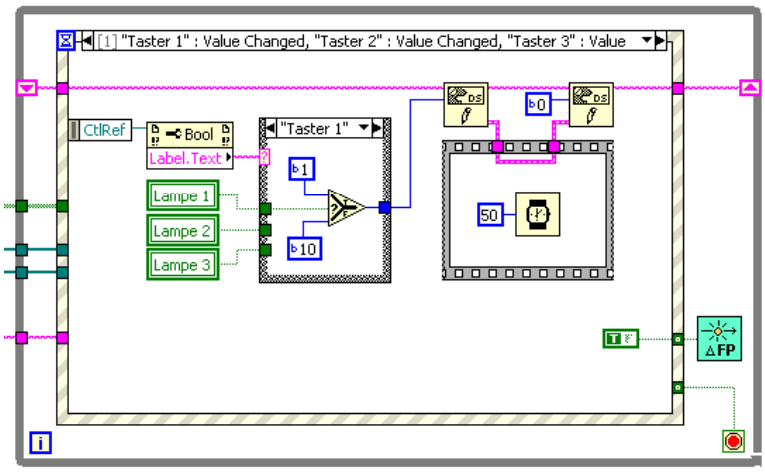

Abb. 7-4: Ereignisstruktur im VI für die Etagensteuerung

Die Ereignisstruktur von LabVIEW arbeitet ausgezeichnet bei der Verwendung von Controls. Ereignisse von Indikatoren werden jedoch nicht berücksichtigt. Nun können sich die Werte von Indikatoren unabhängig von Bedienhandlungen auf dem Frontpanel ändern z.B. ein Mitarbeiter betätigt einen Taster im Raum und schaltet dadurch eine Lampe an.

Bei einer solchen Änderung muss natürlich ebenfalls des Ereignis „Frontpanel-Änderung“ ausgelöst werden. Mit Standardmitteln unter LabVIEW ist es daher unumgänglich, sämtliche lesende OPC-Werte zyklisch abzufragen. Es ist daher eine zweite Schleife notwendig, welche die Abfrage übernimmt. Da sich der Umfang der Verarbeitung innerhalb dieser Schleife in Grenzen hält und die Verarbeitungsgeschwindigkeit nicht maximal sein muss, ist diese Vorgehensweise durchaus berechtigt.

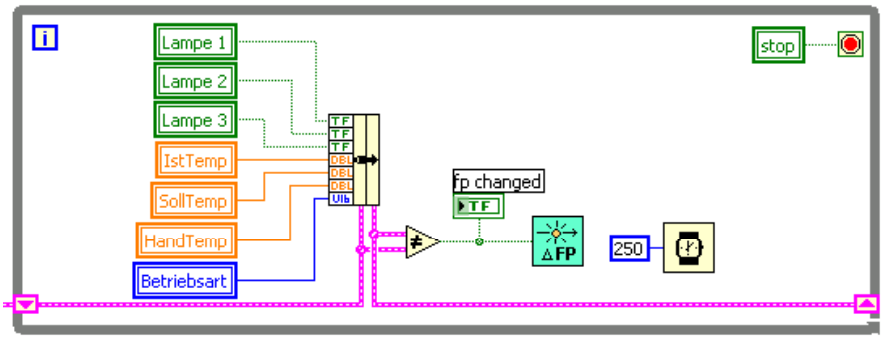

Abb. 7-5: Schleife zu Abfrage der lesenden OPC-Variablen im VI für die Etagensteuerung

Dieser Schritt entfällt, wenn eine Anwendung vollständig auf ObjectVIEW aufbaut. ObjectVIEW arbeitet durchgängig ereignisgesteuert und unterstütz u.a. auch OPC-Ereignisse.

7.3.5 Veröffentlichungsvorgang

Wenn sichergestellt ist, dass alle Möglichkeiten der Änderung des Frontpanels durch Ereignisse erfasst sind, kann die Veröffentlichung des VI zur Etagensteuerung erfolgen.
Im Konfigurationswerkzeug ist zunächst „VI Publishing” auszuwählen. Durch Betätigung des „NEW“-Button aktiviert man den Dateibrowser. Mit diesem ist es nun möglich, das VI im Dateiverzeichnis zu selektieren. Das Betätigen des „ADD“-Buttons erzeugt einen Eintrag in der Liste der veröffentlichten VI´s und zeigt den Name der VI-spezifischen Web-Seite an. Durch Bestätigung der Aktion mit „SAVE“ wird nun automatisch die HTML-Seite mit den entsprechenden Parametern generiert und im eingestellten Verzeichnis abgelegt. Wichtig dabei ist, dass dieses Verzeichnis mit dem LabVIEW-Webserver synchronisiert worden ist.
Mit diesen wenigen Schritten ist das VI nun veröffentlicht und kann durch Aufruf der angegebenen Web-Seite mittels Browser bedient werden. Die Anwendung ist webfähig geworden.

In Abbildung 7-6 ist das Konfigurationswerkzeug zum Veröffentlichen von VI´s zu sehen.

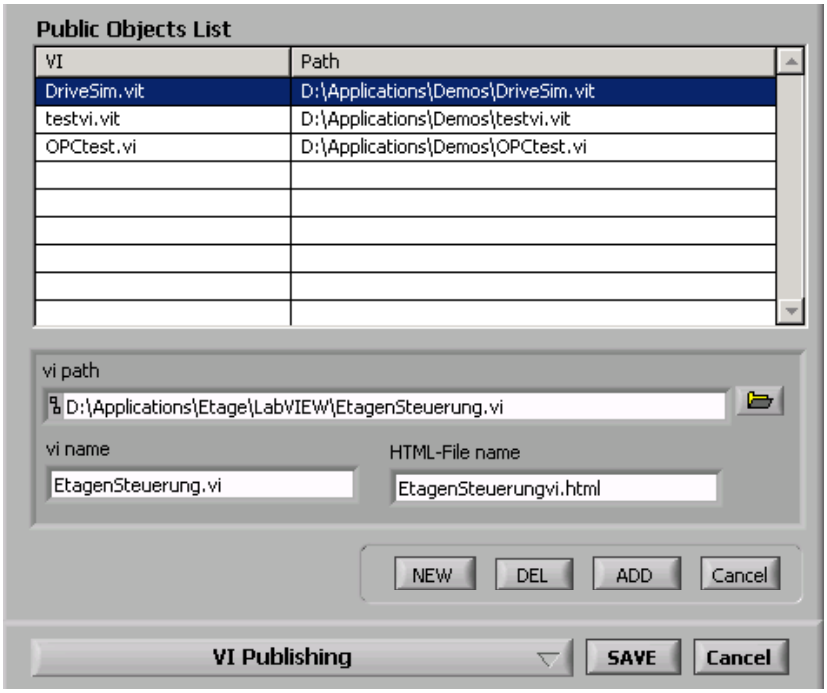

Abb. 7-6: Konfigurationswerkzeug zum Veröffentlichen eines VI´s

7.3.6 Benutzen

Mit Abschluss des Veröffentlichungsvorganges ist die Applikation sofort einsatzbereit. Durch Angabe der URL kann nun die Bedienung des Raumes mittels Web-Browser von jedem beliebigen Rechner im Netzwerk aus erfolgen. Die URL setzt sich aus dem Rechnernamen und der VI-spezifischen HTML-Seite zusammen.

Weiterhin wurde ein Zugang zum Netzwerk für einen PDA mittels Bluetooth eingerichtet. Damit ist es möglich, die Bedienung eines beliebigen Raumes der Etage drahtlos vorzunehmen.

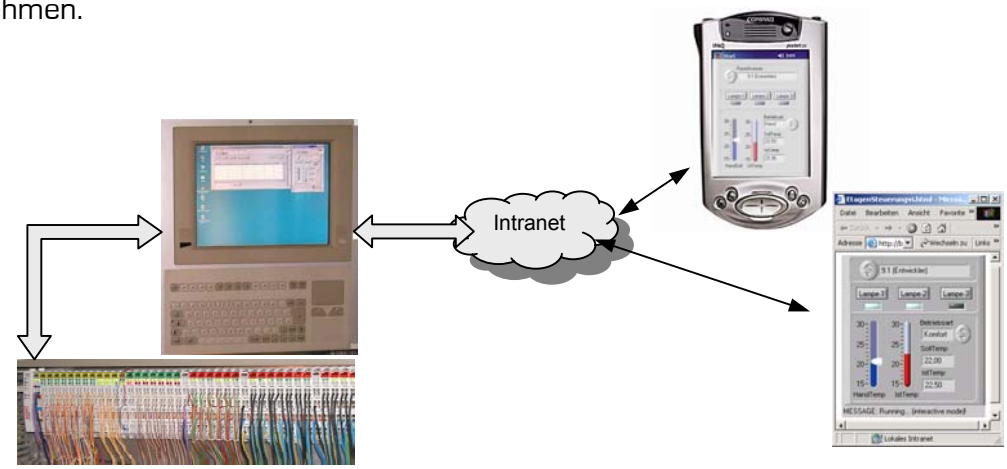

Abb. 7-7: Etagensteuerung mit einem PDA und einem Web-Browser

7.4 Beurteilung

7.4.1 Handhabung

Die Forderung nach einfacher Handhabung des Webinterfaces wurde erreicht. Es bedarf nur weniger Schritte, um ein Anwendungs-VI zu veröffentlichen und damit eine Applikation über das Web zu steuern. Auf einen Eingriff in den Programmcode kann man im Auto-Modus gänzlich verzichten. Allerdings entsteht dabei Netzwerktraffic, auch wenn sich das Frontpanel nicht geändert hat. Wenn man dafür eine hohe Updaterate wählt, ist nicht gewährleistet, dass das Frontpanel im Browser aktuell ist. Daher ist es besser, ereignisgesteuert zu arbeiten. Die dafür notwendige Erweiterung des Programmcodes des Anwendungs-VI´s fällt vergleichsweise gering aus und vereinfacht sich weiter bei der Verwendung von ObjectVIEW.
Für die einfache Handhabung spricht weiterhin, dass keine Programmierarbeit für den Client anfällt.

7.4.2 Darstellung des Frontpanels

Als Bedienoberfläche im Browser findet man das Frontpanel des Anwendungs-VI wieder.
Die Menge der zur Darstellung erforderlichen Daten hängt jedoch von der gewünschten Qualität des Bildes ab. LabVIEW bietet für die Generierung der Bilddaten eines Frontpanels die Farbtiefen 1, 4, 8 oder 24 Bit an. Während die beiden ersten Werte eher unbrauchbar sind, ergibt sich bei 8 Bit Farbtiefe bereits eine ausreichende Bildqualität. Auffällige Abweichungen ergeben sich nur bei aufwendigen 3-D-Elementen auf dem Frontpanel. Dem Original am nächsten kommt man natürlich mit einem Wert für die Farbtiefe von 24 Bit. Damit steigt aber die Datenmenge und der Ressourcenverbrauch an. Die maximale Ereignisrate verringert sich dadurch. Als Standardwert wird daher 8 Bit vorgeschlagen.
Als weitaus größerer Nachteil erweist sich, dass Popup- und Pulldown-Menüs nicht unterstützt werden. Derartige Elemente finden bei der Generierung der Bilddaten mit LabVIEW-Bordmitteln keine Berücksichtigung und sind daher nicht in den Bilddaten enthalten.

7.4.3 Betrachtung der Reaktionszeiten

Ein wichtiger Punkt für die Beurteilung des neuen Webinterfaces sind die verschiedenen Reaktionszeiten die sich zwangläufig ergeben. Diese Zeiten hängen von den verschiedensten Faktoren ab.

Das Medium und deren Bandbreite stellt den größten Einflussfaktor dar. Ein ausführlicher Vergleich der Reaktionszeiten in Abhängigkeit der verschiedenen Einflussfaktoren ist innerhalb dieser Arbeit jedoch nicht möglich. Dennoch sollen zu diesem Punkt einige Betrachtungen durchgeführt werden. Diese basieren ausschließlich auf die Erprobung im Intranet. Weiterhin werden Reaktionszeiten, welche durch untergeordnete Systeme bzw. dem Prozess entstehen, nicht berücksichtigt. Um Aussagen zu machen, muss man zunächst zwischen den einzelnen Zeiten unterscheiden.

Zum einem existiert eine Reaktionszeit, welche zwischen der Bedienhandlung im Browser und der Reaktion des korrespondierenden Controls vergeht, d.h. bis die gewünschte Funktion im Anwendungs-VI ausgelöst wird.

An dieser Stelle entstehen relativ geringe Verzögerungszeiten und liegen bei wenigen hundert Millisekunden. Im Vergleich zu den bestehenden Lösungen (Remote Panel, Applet-VIEW) ist das kaum mehr.

Für den Benutzer ist aber weiterhin wichtig, welche Zeit vergeht, bis der neue Zustand des Controls im Browser aktualisiert ist. Bei den bestehenden Lösungen sind die Bedienelemente tatsächlich im Browser vorhanden. Die Interaktion mit diesen ist nahezu verzögerungsfrei. Bei dem neu entstandenen Webinterface muss dies Bedienereignis erst auf der Serverseite angekommen sein. Erst dann wird ein neues Frontpanelabbild generiert und die geänderten Bilddaten an das Applet gesendet und dort schließlich das Frontpanel aktualisiert. Der Bediener erhält eine Rückkopplung seiner Bedienhandlung entsprechend verzögert. Die Zeit, die dafür benötigt wird, ist nun wieder abhängig von den sich ergebenden Änderungen. Es können im Intranet aber kaum Zeiten unter einer Sekunde erreicht werden.

Die Ursache liegt insbesondere in der nötigen Bildverarbeitung und dem erhöhten Datenvolumen. Das Update eines einzelnen Punktes in einem Diagramm verursacht Bilddaten von ca. 50 Byte. Das Update eines Booleschen Controls dagegen verursacht bereits 1000 bis 3000 Byte Bilddaten bei der Standardgröße, obwohl nur 1 Bit Nutzdaten zu repräsentieren sind.

Diese ungünstigen Verhältnisse schmälern den Bedienkomfort und begrenzen in gleichem Maße die max. Eventrate. Unter Eventrate wird in diesem Zusammenhange die Anzahl der Änderungen auf dem Frontpanel pro Zeiteinheit verstanden.

Obwohl der Einsatz der Webtechnologie bei Echtzeit- bzw. zeitkritische Anwendungen kaum sinnvoll erscheint, bilden die derzeit vorhandenen Verzögerungszeiten des neuen Webinterfaces einem Nachteil gegenüber den Lösungen von NI und Nacimiento.

7.4.4 Resümee

Die Beispielanwendung zeigt, dass die entstandene Lösung die gesetzten Ziele erreicht hat. Durch die Kombination der Vorteile von AppletVIEW und Remote Panel ist es gelungen, dem Nutzer ein einfach zu bedienendes und auf allgemeinen Standards aufbauendes Webinterface bereitzustellen. Bestehende Einschränkungen konnten überwunden werden. Als Ergebnis besteht nun die Möglichkeit, LabVIEW-Frontpanels losgelöst von einer LabVIEW-Laufzeitumgebung über das Web zu bedienen und zu beobachten. So kann man nun u.a. auch PDA´s als Plattform für LabVIEW-Anwendungen einsetzten.

Durch den verwendeten Lösungsansatz entstehen aber neue Nachteile, so dass man die beiden bestehenden Lösungen sicher nicht ersetzen kann. Der Punkt Verzögerungszeiten wird bei entsprechend steigenden Ressourcen jedoch weniger ins Gewicht fallen. Der verwendete Panel-PC entspricht mit einer Taktrate von 233 MHz bereits jetzt nicht mehr dem Standard. Ein leistungsfähigerer Rechner könnte die benötigte Zeit für die Bildverarbeitung erheblich verkürzen und damit einen bestehenden Nachteil ausgleichen.

In jedem Falle wird die Palette der Webtools für LabVIEW erweitert. Unter Abwägung der einzelnen Vor- und Nachteile erhält der Nutzer nun eine Auswahlmöglichkeit mehr, um seine LabVIEW-Applikation webfähig zu gestalten.

8 Künftige Webtechnologien

8.1 Allgemein

Im Abschnitt 2.2.2 wurde die enorme Bedeutung des WWW als Medium zum Austausch von Informationen zwischen den Menschen erörtert. Die Webseite dient dabei zur Bereitstellung und Präsentation der Informationen. Der Datenaustausch findet nur in eine Richtung statt, d.h. der Web-Server schickt HTML-Seiten an den Web-Browser.

Zunehmend werden jedoch Geschäfts-Anwendungen über das Web abgewickelt. Der Benutzer verändert Daten auf dem Server. Auch Bereiche, wie die Mess- und Automatisierungstechnik haben sich das WWW zu Nutze gemacht. So ist es möglich geworden ganze Anwendungen über das Web zu bedienen und zu beobachten.

Trotz der Vorteile, die der Einsatz der Webtechnologie bringt, geht deren Integration mit der Verwendung der bis zu diesen Zeitpunkt vorgestellten Technologien nicht über die Funktion des Clients als GUI hinaus. Es ist also stets der Mensch, der mittels Webbrowser in Interaktion mit der Anwendung tritt.

Mittlerweile haben sich Technologien herausgebildet, die diese Schranke durchbrechen und es ermöglichen auf Basis des WWW anwendungsorientiert Daten auszutauschen oder entfernte Prozeduraufrufe durchzuführen. Diese Technologien und deren Bedeutung für LabVIEW sollen im folgenden kurz erörtert werden.

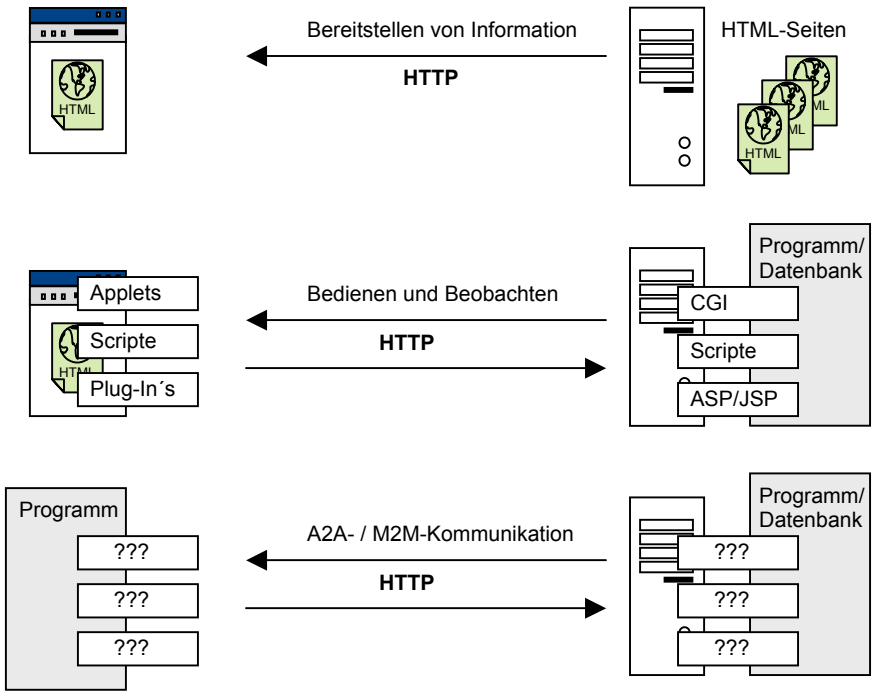

Abb. 8-1: Evolution der Webtechnologie

Innerhalb verteilter Anwendungen gibt es längst Möglichkeiten des Datenaustauschs und des entfernten Methodenaufrufes über lokale Netzwerke und das Internet. Als die wichtigsten Technologien sind hierbei folgende zu nennen:

- CORBA,
- COM/DCOM,
- Java RMI.

Die Entwicklung dieser Technologien fand unabhängig voneinander statt und bauen daher auf jeweils eigenen Standards auf, welche allesamt losgelöst von denen des WWW sind.
CORBA nimmt zwar für sich in Anspruch plattform- und programmiersprachenunabhängig zu sein, doch steht nicht einfach nur eine Schnittstelle, sondern eine gesamte Architektur hinter dieser Technologie. Innerhalb dieser Architektur werden wiederum eigene Transportprotokolle verwendet.
COM/DCOM ist eine Entwicklung der Firma Microsoft und daher weitestgehend auf Microsoft-Umgebungen beschränkt. Dort bildet diese Technologie eine Schlüsselrolle bei der Kommunikation verteilter Komponenten, so z.B. auch beim OPC-Standard.
Java-RMI stellt einen Mechanismus dar, der es ermöglicht, Operationen für Java-Objekte auszuführen, die sich irgendwo im Netzwerk befinden. Java-RMI bietet den Vorteil der Plattformunabhängigkeit, doch über die Programmierumgebung Java kommt man nicht hinaus.
Die bestehenden Technologien für herkömmliche verteilte Anwendungen bilden somit keinen Ansatz, um auf Basis des WWW anwendungsorientiert zu kommunizieren.
Verteilte Anwendungen webbasiert realisieren heißt:

- unabhängig von Plattform und Programmiersprache zu sein,
- keine Bindung an Hersteller, Lizenzen und spezielle Architekturen bzw. Systeme;
- eine quasi weltweit vorhandene Infrastruktur,
- robuste, erprobte, auf allgemein akzeptierte Standards beruhende Technologie,
- durchgängige Transparenz bei der Kommunikation.

8.2 XML in verteilten Systemen

8.2.1 Einordnung

Eine mögliche Lösung existiert bereits seit 1969. In diesem Jahr wurde in der Forschungsabteilung von IBM die erste moderne Auszeichnungssprache mit dem Namen Generalized Markup Language (GML) entwickelt. GML diente der Beschreibung von Strukturen beliebiger Datenmengen. Man war somit auch in der Lage, andere Programmiersprachen, ihre Syntax und Grammatik zu beschreiben.

Später wurde aus GML schließlich SGML (Standard Generalized Markup Language) und wurde 1986 von der ISO als Standard zum Austausch und Speicherung von Daten festgelegt.

SGML ist eine sehr mächtige aber auch komplizierte Auszeichnungssprache. Sie findet breite Anwendung beispielsweise bei Regierungen, großen Firmen, Forschungseinrichtungen und Verlagen. Die hohe Komplexität dieser nützlichen Technologie hat eine weitere Verbreitung verhindert. Auch HTML hat seinen Ursprung in SGML, jedoch in erheblich vereinfachter Form, was ausschlaggebend für den großen Erfolg von HTML war.

1996 begann das World Wide Web Consortium (W3C) mit der Entwicklung einer erweiterbaren Auszeichnungssprache, mit dem Ziel, die Mächtigkeit und Flexibilität von SGML mit der breiten Akzeptanz von HTML zu verbinden. Es entstand die Extensible Markup Language, kurz XML. Der Grundlegende Unterschied zu HTML besteht in der Trennung der Daten von deren Darstellung, sowie die Möglichkeit eigene, beliebige Elemente zu definieren. Mit XML ist man, genau wie mit SGML, in der Lage beliebige Daten mit beliebigen Strukturen auszutauschen und zu speichern, wobei bestehende Internetstandards benutzt werden. Man ist jedoch nicht mehr an einen Browser als Client gebunden, sondern die Daten eines XML-Dokumentes können leicht von Programmen erfasst und manipuliert werden. Wird eine Darstellung der Daten im Browser gewünscht, ist dazu ein weiteres Dokument erforderlich. Im einfachsten Fall benutzt man ein von der HTML-Technologie her bekanntes Stylesheet (CSS). Darüber hinaus gibt es weitere und wesentlich mächtigere Methoden zur Darstellung von XML-Dokumenten (z.B. XSL).

Die Tatsache, dass Informationen bzw. Daten in Form von XML-Dokumenten und damit von anderen Anwendungen lesbar und manipulierbar sind, im WWW veröffentlicht werden, eröffnet eine Reihe neuer, attraktiver Möglichkeiten.

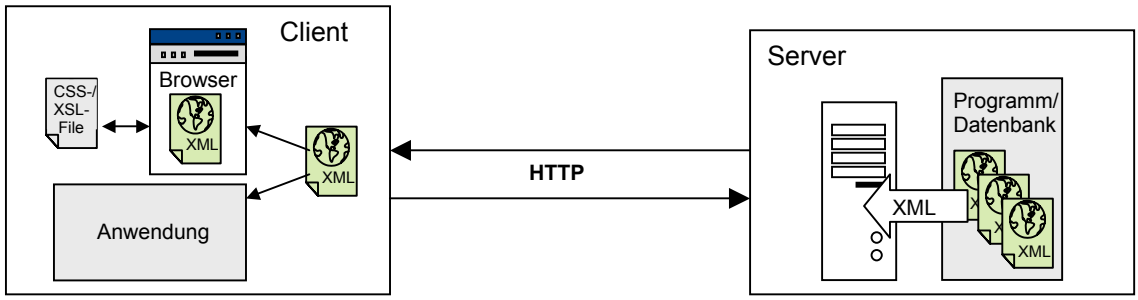

Abb. 8-2: Web-Architektur mit XML

8.2.2 XML-RPC

Auf der Basis von XML wurde mit XML-RPC ein Standard für entfernte Funktionsaufrufe (Remote Procedure Call, kurz RPC) geschaffen. XML-RPC ist eine Spezifikation, welche vollkommen unabhängig von der Programmiersprache ist, da sie auf purem HTTP aufsetzt und XML (also einfachen Text) zum verschicken der Nachrichten bzw. zum Funktionsaufruf verwendet. In der XML-RPC Spezifikation wird festgelegt, wie man den Rückgabewert einer speziellen Funktion eines Servers erhält. Der Funktionsaufruf ist nicht nur auf einen Client beschränkt, sondern kann auch von einem anderen Server aus erfolgen. Da für den Transport das HTTP verwendet wird, gibt es keine Probleme mit Firewalls.

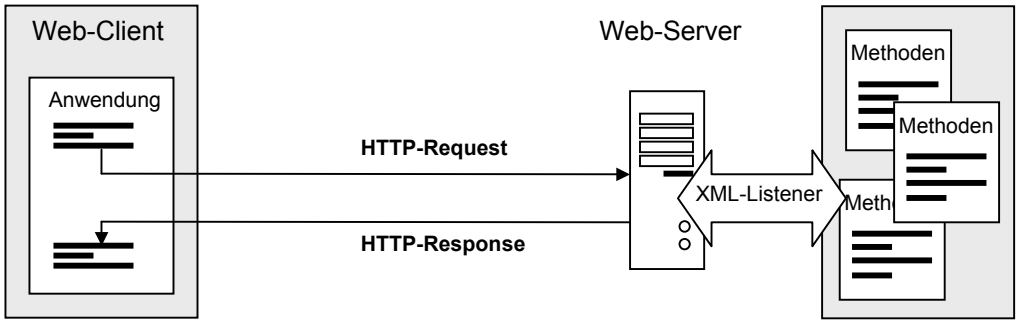

Abb. 8-3: Funktionsweise XML-RPC

Funktionsbeschreibung:

Ein XML-RPC Request enthält eine Anforderung an einen Server, eine bestimmte Methode mit entsprechenden Parametern aufzurufen und auszuführen. Die Parameter können sein:

- Ganze Zahl (Integer),
- Wahrheitswert (Boolean),
- Zeichenkette (String),
- Fließkommazahl (Double),
- Datums-, Zeitangabe (Date/Time),
- Binärcodierung (Base 64),
- Felder (Arrays),
- Strukturen (Struct).

Der HTTP-Server nimmt den Request an und leitet den XML-Inhalt an einen XML-Listener. Dieser parst den XML-Inhalt und extrahiert die Methoden und deren Parameter.
Nach der Bekanntgabe der Version kann direkt eine Methode mit Parametern aufgerufen werden.

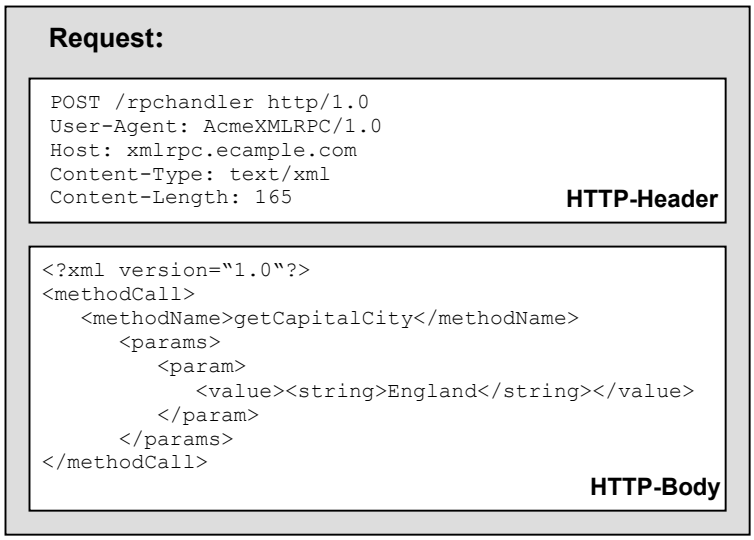

Abb. 8-4: XML-RPC Request

Die betroffene Methode wird ausgeführt und antwortet mit einem XML-RPC Response.
Die XML-RPC Response enthält eine begrenzte Anzahl von Parametern, die das Ergebnis repräsentieren (siehe Abb. 8-5) oder ein XML-Fehlerdokument mit Fehlercodes. Der XML-RPC-Client parst das Ergebnisdokument und extrahiert den Rückgabewert. Im Anschluss kann das Clientprogramm seinen Prozess fortsetzen.

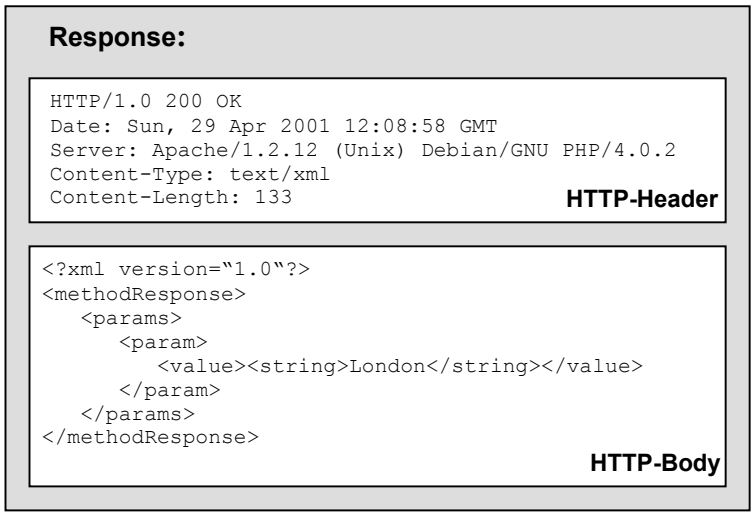

Abb. 8-5: XML-RPC Response

8.2.3 SOAP

Trotz der Möglichkeiten von XML/XML-RPC und der breiten Unterstützung durch die Entwickler gibt es stets Möglichkeiten der Verbesserung und Erweiterung eines Standards. Insbesondere die Firmen DevelopMentor, IBM, Microsoft, Lotus und UserLand begannen Ende 1998 mit Überlegungen, bezüglich eines XML-basierten Lightweight-RPC-Protokolls, dem Simple Object Access Protocol, kurz SOAP. Noch in der Entwicklungsphase schlugen die beteiligten Firmen SOAP als neuen RPC-Standard beim W3C vor. Im Dezember 1999 wurde Version 1.0 der SOAP Spezifikation ratifiziert.

XML-RPC ist ausschließlich als Protokoll zum Aufrufen entfernter Methoden designed worden. Es bedient sich einer einfachen XML-Struktur (<methodCall>, <methodResponse>, <params>) um RPC´s durchzuführen. Beim SOAP ist RPC nur ein Teilbereich. Alle Funktionen die XML-RPC bietet, können auch in SOAP implementiert werden, umgekehrt jedoch nicht. SOAP stellt einen Mechanismus zum Austausch von strukturierter Information zwischen Rechnern in verteilten Systemen zur Verfügung. Das können RPC´s sein, aber auch Daten und Nachrichten (one-way, multicast, ...). Folgenden Punkten schenkten die Entwickler besondere Beachtung:

- Heterogenität,
- Offenheit,
- Erweiterbarkeit.

SOAP bildet als XML-basiertes, herstellerunabhängiges und plattformübergreifendes Netzwerkprotokoll die Basis für die Datenkommunikation zwischen verteilten Objekte im Internet.

Das am meisten benutzte Transportprotokoll für eine SOAP-Nachricht ist HTTP. Seit der Version 1.1 sind jedoch auch andere Protokolle, wie z.B. SMTP vorgesehen. Jede SOAP-Nachricht ist ein XML-Dokument, welches gleichzeitig die Nutzlast des verwendeten Protokolls (bevorzugt HTTP) darstellt. Eine SOAP-Nachricht hat prinzipiell drei Bestandteile:

- den SOAP-Envelope,
- den SOAP-Header,
- den SOAP-Body.

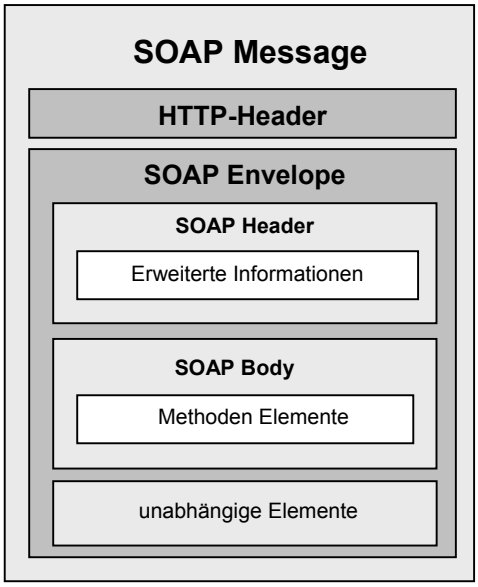

Abb. 8-6: Grundstruktur einer SOAP-Nachricht

8.2.4 AConML

Die Automatic Control Markup Language (AConML) stellt einen möglichen Ansatz für die Automatisierungs- und Prozessleittechnik dar, Daten XML-basiert auszutauschen. AConML ist das Ergebnis eines Forschungsprojekts, in dem drei Hochschulen involviert sind (Aachen, Stuttgart, Kaiserslautern). Ein Standard soll bis Mitte 2003 spezifiziert sein.

Das grundsätzliche Ziel besteht darin, die Daten innerhalb einer heterogenen Umgebung möglichst effizient zu definieren und allen beteiligten Geräten einheitlich zur Verfügung zu stellen. Im Gegensatz zu OPC ist man nicht mehr auf die Windows-Plattform und bestimmte Geräteklassen beschränkt, sondern erhält einen offenen, für die Bedürfnisse der Automatisierungs- und Prozessleittechnik zugeschnittenen Standard für den Datenaustausch.

In Abbildung 8-7 ist die grundlegende Architektur dargestellt, wie sie sich bei der Verwendung von AConML ergibt. Ausgangspunkt ist ein embedded Web-Server, welcher direkt mit der prozessnahen Komponente (PNK) kommuniziert. Über die Standard Internet-Protokolle (HTTP, SMTP, FTP) kann sich nun ein Client nach entsprechender Autorisierung mit diesem Web-Server verbinden. Die Art und Funktion des Clients kann jedoch unterschiedlich sein (Browser, SCADA-System, Exel, ...). Für jede Client-Funktion steht ein Modul zum download bereit. Bei der Verwendung eines embedded Web-Server sind jedoch die Ressourcen sehr begrenzt, so dass bei Bedarf ein weiterer, zentraler Server für größere Softwaremodule zum Einsatz kommt.

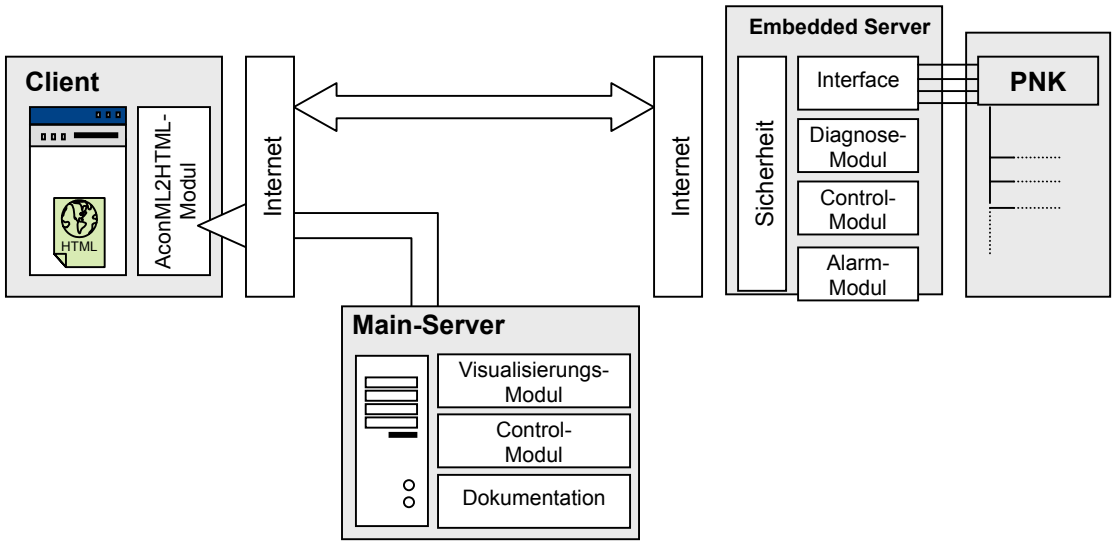

Abb. 8-7: Architektur bei der Verwendung von AConML

Die Prozessdaten selbst sind ausschließlich in AConML codiert. Der Informationsaustausch zwischen der Client-Anwendung und dem Prozess über das Web findet in jedem Falle mit dem Web-Server der PNK satt.

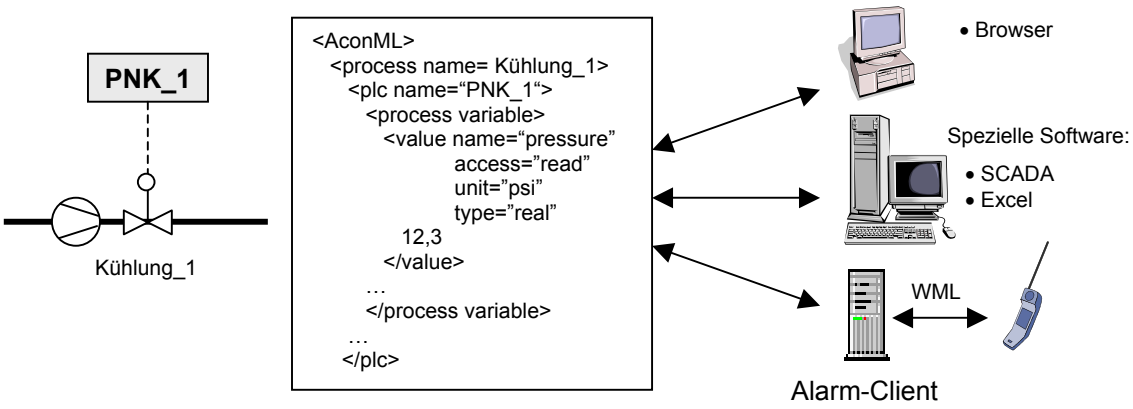

Abb. 8-8: Datenaustausch mit AConML

8.3 Web Services

8.3.1 Allgemein

Web Services sind eine Integrationstechnologie, die eine dynamische Zusammenarbeit von Applikationen im Web unter Nutzung von standardisierten Internetprotokollen ermöglicht. Im Gegensatz zu webbasierten verteilten Anwendungen bilden Web Services den nächsten Evolutionsschritt.

Innerhalb der klassischen Web-Architektur findet man ausschließlich Informations- oder Shoppingsysteme. Jeder kann sein spezielles Angebot in Form von Web-Seiten im WWW bereitstellen. Mittels Suchmaschinen ist der Benutzer in der Lage das gewünschte Angebot zu finden und zu nutzen. Es ist also der Mensch, der als Bediener des Web-Browsers das vielfältige Angebot des WWW in Form von GUI-basierten Anwendungen nutzt.

Die Idee der Web Services geht einen Schritt weiter. Der Web Service selbst ist ein Anwendungsprogramm, welches von anderen Anwendungen gefunden und genutzt werden kann. Web Services stellen somit eine neue Variante der alten Idee der komponentenorientierten Entwicklung dar, jedoch mit einer Ausweitung auf das WWW.

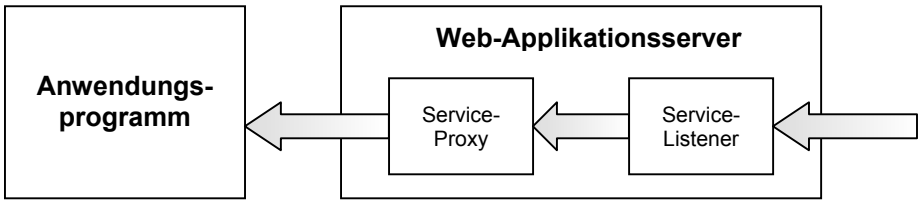

Abb. 8-9: Zentrale Bestandteile eines Web Services

8.3.2 Die Technologie der Webservices

- **XML**

Die Grundlage der Web Service-Standards bildet XML, da es sich besonders gut zum plattformunabhängigen Datenaustausch eignet. Mit Hilfe von XML lassen sich Anwendungen über Rechnergrenzen zu einem Gesamtsystem integrieren.

- **SOAP**

Das Simple Object Accsess Protocol (SOAP) legt fest, wie ein Funktionsaufruf (Remote Procedure Call, kurz RPC) mit XML-Daten über Rechnerplattformen erfolgt. Als Transportprotokoll ist in erster Linie HTTP vorgesehen, wobei die SOAP-Spezifikation durchaus weitere Transportprotokolle zu lässt.

- **WSDL**

Die Web Services Description Language (WSDL) ist ein XML-basierter Rahmen, in dem ein Webservice beschrieben wird: welche Funktionen er bietet, wie diese aufzurufen und welche Parameter dabei zu übergeben sind. Ehe solche Aufrufe von Applikationen unterschiedlicher Anbieter zur Laufzeit quasi selbsttätig vonstatten gehen, ist noch einiges zu tun: Erforderlich sind weitergehende Standardisierungen für die Schnittstellen und Übereinkünfte zur Bedeutung der Parameter. Erst dann könnten die Geschäftsprozesse so automatisiert werden, wie es der Vision der Webservices entspricht.

- **UDDI**

Der entstehende Standard Universal Description, Discovery and Integration (UDDI) spezifiziert, wie Verzeichnisse aufzubauen sind, in denen Webservices registriert und von anderen Programmen gefunden und aufgerufen werden können. Die so genannten Weißen Seiten enthalten Namen und Adressen der Unternehmen, die Gelben Seiten ergänzen Branchen- und Produktinformationen sowie Kontaktmöglichkeiten. Die Grünen Seiten schließlich enthalten, dargestellt in der WSDL, Informationen über die Webservices, die ein Unternehmen anbietet.

8.3.3 Generierung eines Web Service

Durch folgende Schritte werden Web Services generiert:

- Ein Anbieter erstellt, kombiniert und verteilt einen Dienst (wie zum Beispiel Aktienkurse, Terminkalender oder die Datensynchronisierung) auf Basis seiner bevorzugten Programmiersprache, Infrastruktur und Plattform über das World Wide Web.
- Der Anbieter definiert seinen Service in der hierfür relevanten Sprache WSDL (Web Service Description Language). Ein WSDL-Dokument beschreibt hierbei den Web Service und alle anderen Nutzungsbedingungen (beispielsweise die Zahlungsmodalitäten für die Nutzung).
- Der Anbieter registriert seinen Web Service in sog. UDDI-Registries (UDDI: Universal Description, Discovery and Integration). Dies sind Registrierungsstellen, die - ähnlich den „Gelben Seiten“ - Dienste erfassen und veröffentlichen. Die weltweite Verwaltung der UDDI-Registries ist derzeit noch nicht endgültig definiert.
- Der Anbieter ermöglicht es dem Konsumenten auf den relevanten Web Service zuzugreifen und die Funktionen des Web Service unter Verwendung von SOAP (Simple Object Access Protocol) aufzurufen.

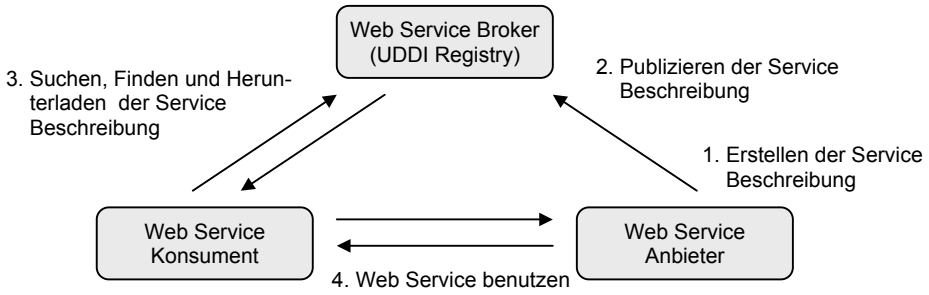

Abb. 8-10: Die Web Service-Architektur

8.3.4 Nutzen der Web Services

Bereits die komponentenbasierte Programmierung erwies sich als Segen für die Produktivität der Entwickler. Mit Web Services ergeben sich neue und entscheidende Vorteile. Web Services ermöglichen eine dynamische, zeitnahe und abrechenbare Abwicklung von komplexen Geschäftsprozessen über Plattform- und Unternehmensgrenzen hinweg. Dies ermöglicht dynamisch aus einem weltweit angebotenen Pool von speziellen und bewährten Web Services innerhalb kürzester Zeit neue Anwendungen zu kreieren, wobei Plattform und Kommunikation transparent werden. Die Vision von dezentralisierten Anwendungen und Services, die unabhängig von Plattformen, Hardware und Lokalität funktionieren, rückt in greifbare Nähe.

Jedes Unternehmen kann gleichzeitig Anbieter und Konsument von Web Services sein. Als Anbieter stellt ein Unternehmen seine Informationen und Dienste kostenpflichtig anderen zur Verfügung. Als Konsument hat man Zugriff auf Informationen und Dienste und kann diese beispielsweise für seine Anwendungen nutzen. All dies geschieht ohne großen Aufwand über ein einheitliches Protokoll. Neue Geschäftsmodelle, wie die kostenpflichtige Bereitstellung und Nutzung von Content werden möglich. Beispielsweise werden einzelne Dienste, wie z.B. Aktienkurse, Terminkalender Bonitätsabfragen, Wetterinformationen, ... zu webbasierenden Komponenten.

Die Integration verschiedenartigster Anwendungen erwies sich in der Vergangenheit oftmals als komplex und schwierig, weil verschiedene Programmiersprachen und Infrastrukturen koordiniert werden mussten. Wenn CORBA, DCOM oder Java RMI verwendet werden soll, muss die passende Laufzeitumgebung installiert werden. Die Anwender müssen dazu gebracht werden auf ihren Systemen die dazugehörige verteilte Infrastruktur einzurichten.

Firewalls müssen neu konfiguriert werden, so dass systemspezifische Pakete in das lokale Netzwerk eingelassen werden und es ebenso verlassen können.
Die neue Generation von Web Services verwendet stattdessen Protokolle und Datenformate wie HTTP und XML, so dass quasi jede Programmiersprache, jede Infrastruktur und jede Plattform mit allen anderen kommunizieren und interagieren kann.

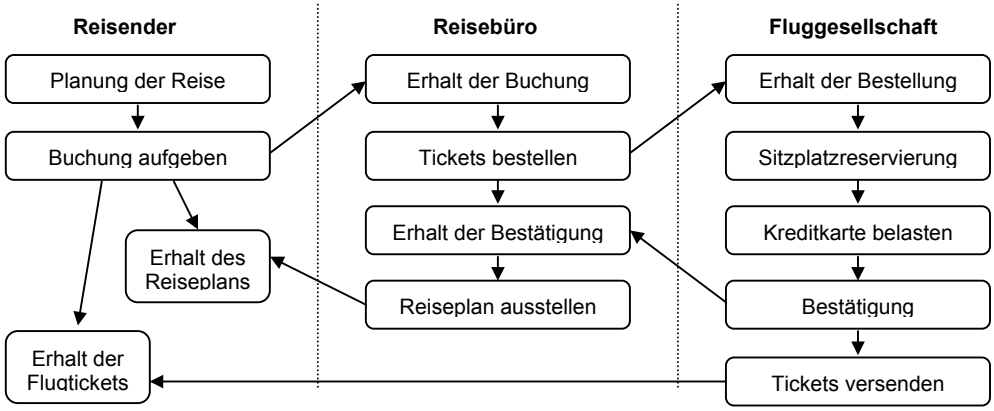

Abb. 8-11: Web Service Szenario

8.4 .NET

8.4.1 Allgemein

.NET stellt die neuste Entwicklung aus dem Hause Microsoft dar und wurde das erste mal im Sommer 2000 vorgestellt. Hinter dem Begriff steht nicht nur eine neue Technologie, sondern vielmehr eine neue Plattform, die auf einer Mehrzahl von neuen Technologien basiert. Es ist Microsofts Antwort auf die aktuellen Entwicklungen innerhalb der Branche, die eindeutig auf das Internet und das WWW, sowie neue Hardwaresegmente setzt und daher offene Systeme verlangt. Mit .NET verlässt Microsoft selbst geschaffene, proprietäre Standards (wie z.B. COM/DCOM) und setzt auf neue allgemeine Standards und Technologien. Die überaus erfolgreiche und von Microsoft vorangetriebene Komponentenarchitektur soll sich anderen Plattformen, inkl. anderen Geräteklassen erschließen. Da die Idee des weltweit verteilten und verfügbaren Komponenten-Modells bereits mit den Web Services verwirklicht ist, findet man im Kern von .NET diese Idee mit den damit verbundenen Technologien wieder.

Man kann also zwei grundsätzliche Ziele von .NET nennen:

- Erweiterung der produktorientierten Desktop-Softwarewelt um die dienstleistungsorientierte Internet-Komponentenwelt,
- Öffnung gegenüber einer Vielzahl von Hardware- und Betriebssystemplattformen.

Damit bildet .NET eine vielversprechende Plattform zum Umgang mit Web Services, d.h. sie können genauso einfach definiert und konsumiert werden, wie lokale Software-Komponenten.

Die vier Säulen der .NET-Plattform sind:

- .NET Framework,
- .NET Enterprise Servers,
- .NET My Services,
- .NET Devices.

8.4.2 Architektur von .NET

Für Kommunikationspartner, welche losgelöst von der .NET-Plattform sind, ist SOAP der kleinste gemeinsame Nenner. Innerhalb der .NET-Plattform sorgt eine ganz bestimmte Architektur für Interoperabilität. Die wohl wichtigste Komponente dieser Architektur ist Common Language Runtime (CLR). Die CLR ist die Laufzeitumgebung für .NET-Anwendungen. Deren Hauptaufgaben sind die Übersetzung des Zwischensprachencodes in Maschinencode, Verwaltung von Speicher und Threads, Durchsetzung von Sicherheitsmechanismen und Laden von Komponenten.

Der erwähnte Zwischensprachencode wird als Microsoft Intermediate Language (MSIL oder kurz: IL) bezeichnet. Alle .NET-Programmiersprachen erzeugen diesen Code anstelle von Maschinencode. Dieser wird zusammen mit Metadaten in Assemblys abgelegt. IL-Code wird nicht wie bei Java-Bytecode interpretiert, sondern von der CLR auf dem Zielsystem kompiliert. Dies geschieht Just-in-Time (JIT), d.h. während der Ausführung der Anwendung. Dabei wird nicht die gesamte Anwendung beim Programmstart, sondern nur gerade die Teile der Anwendung kompiliert, die gerade benötigt werden.

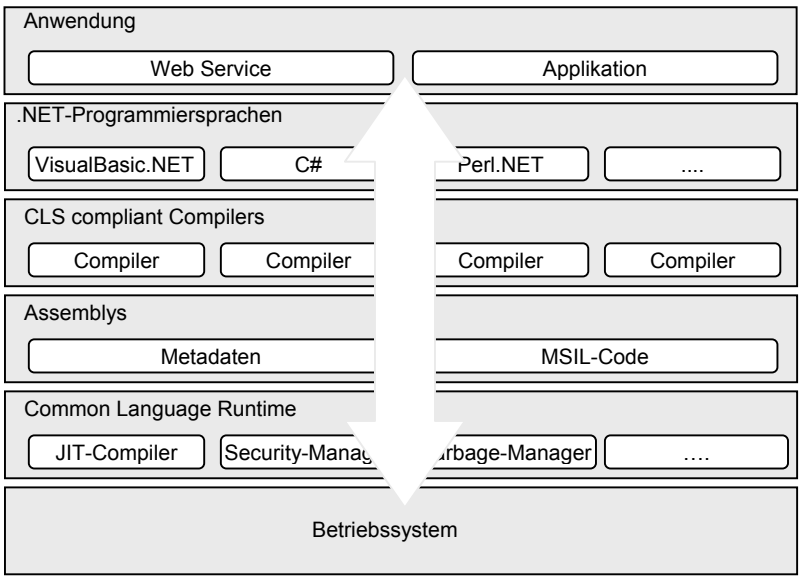

Abb. 8-12: Die .NET-Architektur

8.4.3 .NET Framework

Das .NET Framework stellt die Infrastruktur, Funktionalität und Dienste zum Erstellen, Bereitstellen und Ausführen von Desktop- und Web- und Smart Client-Anwendungen bereit. Die Basisklassen-Bibliothek stellt Standardfunktionen beispielsweise für die Ein-/Ausgabe, die Threadverwaltung oder für das Benutzeroberflächendesign bereit.

Die Datenklassen von Microsoft ADO.NET unterstützen die Verwaltung dauerhafter Daten und umfassen SQL-Klassen, um dauerhafte Datenspeicher über eine standardmäßige SQL-Schnittstelle zu bearbeiten. XML-Klassen ermöglichen die XML-Datenbearbeitung. XML spielt für die Zwischenspeicherung und den Austausch von Daten eine zentrale Rolle.

Die nächste Ebene unterstützt zwei unterschiedliche Anwendungstypen. Die Windows Forms-Klassen dienen der Entwicklung von Windows-basierenden Clientanwendungen und die Microsoft ASP.NET-Klassen dienen der Entwicklung von Webanwendungen und von XML-Webdiensten. Web Forms sind Steuerelemente, die für die Erzeugung der Benutzerschnittstelle zuständig sind. Sie bilden typische HTML Elemente, wie Textboxen oder Schaltflächen, nach. Web Services stellen ein Modell für die Verbindung verschiedener Anwendungen über das Internet zur Verfügung. Dieses Modell setzt auf bestehende Infrastrukturen und bereits existierende Applikationen auf.

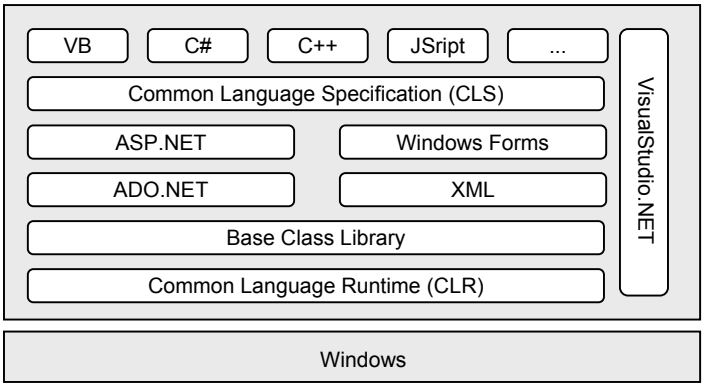

Abb. 8-13: Das .NET Framework

Wie in der obigen Abbildung zu sehen, gibt es verschiedene Programmiersprachen zur Entwicklung von .NET Framework-Anwendungen. Sie müssen den Anforderungen der Common Language Specification (CLS) entsprechen. Die Programmiersprache C# basiert zum Beispiel nur auf den Klassenbibliotheken des .NET Framework.
Zusammen bieten die Klassenbibliotheken eine gemeinsame, konsistente Entwicklungsschnittstelle für alle Sprachen, die vom .NET Framework unterstützt werden [.NET03].

8.4.4 .NET Enterprise Servers

Unter .NET Enterprise Servers ist eine Palette von bestehenden Server-Produkten zusammen gefasst worden. Dabei handelt es sich nicht um speziell an .NET angepasste Server, sondern viel mehr um etablierte Serverlösungen. Sie sind jedoch von zentraler Bedeutung für .NET-Anwendungen und somit eine wichtige Säule von .NET. Die vielleicht wichtigsten sind dabei SQL-, Exchange- und BizTalk Server, die jeweils umfangreiche XML-Schnittstellen bieten.

8.4.5 .NET My Services

.NET My Services sind konkrete, von Microsoft angebotene Web Services für Standardprobleme. Derzeit handelt es sich dabei vorwiegend um Verwaltung von Daten einzelner Personen. Beispiele sind:

- .NET Contacts: Kontaktinformationen, Adressen,
- .NET Wallet: Kreditkarten und Kontoinformationen,
- .NET Documents: Dokumente und Dateien,
- .NET Application Settings: Einstellungen für verschiedene Anwendungen,
- u.v.m.

So ist es dem Nutzer beispielsweise möglich, seine Daten mit den verschiedensten Anwendungen auf verschiedenen Geräteklassen zu synchronisieren.

8.4.6 .NET Devices

.NET Devices ist der Teil von .NET, der sich mit den Geräten jenseits des Desktop-PC´s befasst. Insbesondere handelt es sich hierbei um die sich permanent erweiternde Palette von PDA´s, Handys oder Auto-PC . Ziel dabei ist es, diese Geräte mit der .NET-Plattform zu unterstützen. Darüber hinaus arbeitet Microsoft an dem sogenannten Tablet-PC. Dieser garantiert eine hohe Mobilität durch wireless Connectivity und eine besonders intuitive Handhabung. Er soll die Lücke zwischen Laptop und PDA schließen .

8.5 Künftige Webtechnologien und LabVIEW™

8.5.1 Trends in der Mess- und Automatisierungstechnik

Will man Aussagen zur Bedeutung der genannten Technologien für LabVIEW machen, muss man zunächst auf die aktuellen Trends in der Mess- und Automatisierungstechnik eingehen.

Der seit längeren bestehende Trend zu dezentralen Steuerungslösungen wird aktuell durch den Einsatz von Ethernet weiter verstärkt. Ethernet ist mittlerweile zum Industriestandard geworden und bietet weitaus mehr Optionen als klassische Kommunikationslösungen.

Ein weiterer Trend ist das immer weitere Vordringen des Industrie-PC (IPC). Im Gegensatz zur klassischen SPS bietet dieser weitaus mehr Ressourcen und einen einheitlichen Hardwarestandard. Im gleichem Maße nimmt der Anteil an intelligenten Feldgeräten zu.

Das Resultat dieser Entwicklungen ist eine durchgängig einheitliche und standardisierte Kommunikationsarchitektur und eine zunehmend leistungsfähigere Automatisierungsplattform.

Eine einheitliche Kommunikationsarchitektur macht jedoch nur dann Sinn, wenn einheitliche Kommunikationsprotokolle verwendet werden. Da liegt es nahe bereits erprobte, robuste und allgemein akzeptierte Standards zu verwenden. An dieser Stelle eignen sich die Technologien des WWW. Mit den eingangs der Arbeit erwähnten Eigenschaften bieten diese Technologien das größte Potential. Mit einer durchgängigen Transparenz der Kommunikationsstruktur steigen gleichzeitig die Erwartungen. Man stößt an die gleichen Probleme, wie sie aus der IT-Welt her bekannt sind: einheitlicher Datenaustausch der beteiligten Systeme untereinander. Die logische Konsequenz scheint eine Etablierung der zukunftweisenden Webtechnologien in der Mess- und Automatisierungstechnik zu sein. Während die IT-Welt bereits vereinzelt von diesen Technologien profitiert, wird bis zur Integration in der Mess- und Automatisierungswelt noch einige Zeit vergehen. Erste Ansätze sind durch den vermehrten Einsatz von XML jedoch schon vorhanden. Als Beispiel sei hier AConML und OPC XML DA genannt.

8.5.2 Die Bedeutung und der Nutzen für LabVIEW™

Der bestehende Integrationsgrad der Webtechnologie spiegelt die aktuellen Tendenzen in der Branche und den Stand der Technologie ausreichend wieder. Mit der nun erweiterten Palette der Webtools, können LabVIEW-Applikationen von der Möglichkeiten der Steuerung über das Web profitieren. Das Potential von LabVIEW geht aber über das reine Bedienen und Beobachten von Anwendungen weit hinaus.

Um dieses Potential in Zukunft voll ausschöpfen zu können, muss mittelfristig der Zugang zu den künftigen Webtechnologien möglich sein. Dies sichert nicht nur eine gute Position in der Mess- und Automatisierungstechnik, sondern schlägt eine weitere Brücke zur IT-Welt. LabVIEW-Entwickler können dann beispielsweise am Komponenten-Markt in Form von Web-Services teilhaben und dort den Vorteil der grafischen Programmierung ausnutzen.

8.5.3 Fazit

Die aktuellen Entwicklungen und Tendenzen in der Mess- und Automatisierungstechnik lassen den Schluss zu, dass XML-basierte Webtechnologien mittel- und langfristig in der Branche Einzug halten.

Als bewährte und vielerorts eingesetzte Entwicklungsumgebung wird LabVIEW sich dieser Entwicklungen nicht verschließen. So gibt es bereits mit der aktuellen Version (6.1) Möglichkeiten der Verarbeitung von XML-Daten. Für die Folgeversion wird bereits eine Integration von .NET angekündigt.

Der Zeitpunkt und Umfang weiterer Integrationen wird sich an dem jeweiligen Entwicklungsstand in der Branche orientieren. Die Frage nach den wie lässt sich da schon schwieriger beantworten. Fest steht jedoch, dass XML als Datenformat und das darauf aufbauende Kommunikationsprotokoll SOAP den kleinsten gemeinsamen Nenner bei der Vielzahl von Möglichkeiten darstellt. In kürze ist daher auch mit konkreten Lösungen und Produkten im Bereich Automatisierungstechnik zu rechnen.

Für den LabVIEW-Entwickler lohnt es in jedem Falle die Entwicklungen in der Branche zu beobachten und möglichen Integrationslösungen offen gegenüber zu stehen.

9 Zusammenfassung und Ausblick

Es ist durchaus gelungen die Integration der Webtechnologie in LabVIEW / ObjectVIEW voranzutreiben. So ist es nun grundsätzlich möglich, Frontpanels auf Web-Plattformen anzuzeigen und zu bedienen für die es keine LabVIEW-Laufzeitumgebung gibt. Mittelfristig geht jedoch der Vorteil, der mit der Portierung auf andere webfähige Geräteklassen erreicht wurde, verloren. Die Firma NI plant die Einführung einer LabVIEW-Runtime für Palm bzw. Pocket PC. Auch wenn damit ein entscheidender Vorteil des neu entstandenen Webinterfaces verloren gehen würde, bereichert es dennoch die Palette der Web-Tools von LabVIEW.

Der derzeitige Entwicklungsstand reicht noch nicht aus, um einen kommerziellen Vertrieb anzustoßen. Es gibt allerdings noch Möglichkeiten der Verbesserung. Insbesondere bei der Bildverarbeitung könnte die Verarbeitungsgeschwindigkeit durch Auslagerung der Algorithmen in C-Code gesteigert werden. Ein entscheidender Nachteil könnte dadurch verringert werden.

Wenn man im Rahmen dieser Arbeit von Integration von Webtechnologien in LabVIEW / ObjectVIEW spricht, kann diese keinesfalls als vollständig angesehen werden. Das Kapitel Acht zeigt, dass weiterhin Bedarf besteht. Ein umfangreiches Potential an Technologien steht bereits bereit.

Im Moment fungiert die IT-Welt als Vorreiter bei der Anwendung XML-basierter Webtechnologien. Hier wird besonders den Web Services künftig eine wachsende Bedeutung zu gesprochen. Insbesondere die Initiativen von Sun und Microsoft zeigen, dass in der Tat die Softwarebranche ein erhebliches Potential in der Anwendung dieser Technologie sehen. Web Services basieren in besonderer Weise auf den Internetstandards. Wo sich andere Technologien auf die Nutzung von TCP/IP beschränken und für höhere Funktionen eigene Austauschprotokolle und Datenkodierungen nutzen, setzen Web Service-Standards auf die Nutzung allgemein akzeptierter und überall verfügbarer Web-Technologien wie HTTP und XML.

Während browserbasierte Anwendungen das Web zu dem gemacht hat, was es heute ist, werden die Webservices es zu dem machen, was in der Zukunft sein wird [Chap02].

Das Allheilmittel werden diese Technologien jedoch auch nicht sein. So sind z.B. viele Fragen hinsichtlich der Standardisierung, Sicherheit und Ausführungsgeschwindigkeit noch ungeklärt. Purer Aktionismus ist in jedem Falle verfrüht.

Fest steht jedoch, das XML und die darauf aufbauenden Technologien der Schlüssel zur plattform- und programmiersprachenunabhängigen Kommunikation darstellen und diese früher oder später dort Anwendung finden, wo diese Forderung besteht.

Literaturverzeichnis

[Albr02] Harald Albrecht, Dirk Meyer: XML in der Automatisierungstechnik, Rheinisch-Westfälische Technische Hochschule Aachen 2002, http://www.plt.rwth-aachen.de/mitarbeiter/publ/albrecht-meyer-xml-at-2002.pdf

[Arci01] Fabio Arciniegas: XML Developer´s Guide, Franzis Verlag, Poing 2001

[Chap02] David Chappell: .NET verstehen, Addison-Wesley Verlag, München 2002

[Fowl00] Martin Fowler with Kendall Scott: UML distilled, Addison-Wesley, Second Edition, 2000

[Furr99] Frank J. Furrer, Objekt-Technologie, Band 1: Grundlagen und Technologie, SYSLOGIC Press, Schweiz 1999

[Gabe01] Dipl.-Ing. O.Gabel, Prof. Dr. Litz: Automatic Control Markup Language – An Opportunity of Standardised Remote Control and Maintenance, Universität Kaiserslautern 2001, http://www.wbmt.tudelft.nl/pto/research/conferences/Proceedings/Sci2001/Paperspdf/P000444.PDF

[HeiBa00] Heide Balzert: Objektorientierung in 7 Tagen, Spektrum Akademischer Verlag, Heidelberg/Berlin 2000

[HeiBa01] Heide Balzert: UML kompakt, Spektrum Akademischer Verlag, Heidelberg/Berlin 2001

[HelBa00] Helmut Balzert, Lehrbuch der Software-Technik, Band 1: Softwareentwicklung, Spektrum Akademischer Verlag, Heidelberg/Berlin 2000

[HoltO2] Katrin Holt: Microsoft Intermediate Language (MSIL),
Ausarbeitung, Fachhochschule Wedel 2002,
http://www.fhwedel.de/~si/seminare/ws02/Ausarbeitung/5.msil/MSIL1.htm

[JamaO1] R. Jamal/A. Hagenstedt, LabVIEW-Das Grundlagenbuch,
Addison-Wesley Verlag, 3. Auflage, München 2001

[KnutO2] Michael Knuth: Web Services – Einführung und Übersicht,
Software & Support Verlag, Frankfurt 2002

[KrügO2] Guido Krüger: Handbuch der Java-Programmierung,
Addison-Wesley Verlag, 3. Auflage, München 2002

[KuscO2] Michael Kuschke, Ludger Wölfel: Web Services kompakt,
Spektrum Akademischer Verlag, Heidelberg/Berlin 2002

[KyasO1] Othmar Kyas, Markus a Campo: Internet Professionell,
MITP-Verlag, 2.Auflage, Bonn 2001

[LaurO1] Simon St.Laurent, Joe Johnston and Edd Dumbill:
Programming Web Services with XML-RPC,
O´Reilly & Associates, Sebastopol (USA) 2001

[Lema99] Laura Lemany/Rogers Cadenhead: Java 2 in 21 Tagen,
Markt und Technik Verlag, München 1999

[LitzO1] Prof. Dr. Litz, Dipl.-Ing. O.Gabel: Internet im Automatisierungssystem,
Powerpoint-Folien, Universität Kaiserslautern 2001.
http://www.eit.uni-kl.de/litz/hauptmenue/veroeffentlicht/vde/index.htm

[Mann97] Michelle M. Manning: Jbuilder in 21 Tagen,
SAMS, Haar bei München 1997

[ObjV02] Vogel Automatisierungstechnik GmbH, ObjectVIEW- Handbuch, http://www.lvot.de

[Rott02] Thilo Rottach, Sascha Groß: XML kompakt,
Spektrum Akademischer Verlag, Heidelberg/Berlin 2002

[Scrib01] Kennard Scribner, Mark C.Stiver: SOAP Developer´s Guide,
Markt und Technik Verlag, München 2001

[Snel02] James Snell, Doug Tidwell und Pavel Kuchenko:
Webservice-Programmierung mit SOAP,
O´Reilly Verlag, Köln 2002

[Stey99] Ralph Steyer: Java 2 Kompendium,
Markt und Technik Verlag, München 1999

[VAT02] Vogel Automatisierungstechnik GmbH, Firmenprofil, URL:
http://www.vat.de/

[West01] Ralf Westphal: .NET kompakt,
Spektrum Akademischer Verlag, Heidelberg/Berlin 2001

Selbstständigkeitserklärung

Ich versichere, dass ich die vorliegende Diplomarbeit selbstständig und ohne unerlaubte Hilfe Dritter verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Alle Stellen, die inhaltlich oder wörtlich aus Veröffentlichungen stammen, sind kenntlich gemacht. Diese Arbeit lag in gleicher oder ähnlicher Weise noch keiner Prüfungsbehörde vor und wurde bisher noch nicht veröffentlicht.

Jena, den 21.02.2003 *Andreas Himmelreich*

Danksagung

An dieser Stelle möchte ich mich zunächst bei all jenen bedanken, die mir während des gesamten Studiums Wissen vermittelten und bei Problemen mit Ratschlägen und Hinweisen behilflich waren.
Für die sehr gute Zusammenarbeit während der Bearbeitung der Diplomarbeit möchte ich mich bei Herrn Prof. Dr.-Ing. Jörg Müller von der Fachhochschule Jena bedanken. Besonderer Dank gilt meinem Mentor Herrn Dipl.-Ing. Jens Vogel von der Firma Vogel Automatisierungstechnik GmbH für die sehr gute technische und inhaltliche Unterstützung.
Für die angenehme Arbeitsatmosphäre und die vielen aufschlussreichen Gespräche möchte ich mich bei allen Mitarbeitern der Vogel Automatisierungstechnik GmbH bedanken.
Für die stetige Unterstützung im gesamten Studium möchte ich meiner Frau Diana Himmelreich herzlichst danken.

Thesen zur Diplomarbeit

Integration von Webtechnologien in LabVIEW™ / ObjectVIEW™

eingereicht von Andreas Himmelreich

geb. am 16.10.1973 in Jena

Matrikel-Nr.: 209846

Seminargruppe: 982 ET / AT

Hochschulbetreuer: Prof. Dr.-Ing. J. Müller, Fachhochschule Jena

Mentor: Dipl.-Ing. J. Vogel, Vogel Automatisierungstechnik GmbH

Themenausgabe: 26.08.2002

Abgabedatum: 21.02.2003

1. Software ist die Schlüsseltechnologie für alle modernen Geschäftssysteme und Produkte. Sie beeinflusst in zunehmenden Maße die Kosten.

2. LabVIEW verkürzt durch die grafische Programmierung die Entwicklungszeit von Programmen im Vergleich zu textorientierten Programmiersprachen erheblich.

3. Objekttechnologie ermöglicht überschaubare, modulare Software und macht komplexe Systeme beherrschbar. Sie ist ideal für dezentrale Systeme und die Widerverwendung von Softwarekomponenten geeignet.

4. Die UML ist eine grafische Notation zur Erstellung objektorientierter Modelle. Sie eignet sich besonders gut zur Analyse und zum Entwurf objektorientierter Software.

5. Der Einsatz von ObjectVIEW vereint die Vorteile der grafischen Programmierung unter LabVIEW mit den Vorteilen der Objekttechnologie.

6. Die Technologie des WWW hält Einzug in die Automatisierungstechnik. Der Einsatz dieser Technologien verleiht den Kommunikationsstrukturen mehr Transparenz und überwindet bestehende Kommunikationsbarrieren.

7. Webtechnologie baut auf allgemeine, herstellerunabhängige Standards auf und kostet keine Lizenzgebühren. Mit deren Anwendung lassen sich Automatisierungsaufgaben weltweit verteilen.

8. Plattform- und Programmiersprachenunabhängigkeit ist entscheidend für die Kommunikation im Internet.

9. Java ist wegen seiner Plattformunabhängigkeit und Unterstützung durch Standardbrowser besonders gut für Web-Anwendungen geeignet. Java ist die Sprache des WWW.

10. XML bietet die Möglichkeit des plattform- und programmiersprachenunabhängigen Datenaustauschs. XML-basierte Kommunikationsprotokolle werden sich daher in heterogenen Umgebungen durchsetzen.

11. Für die effiziente Verarbeitung von strukturiertem XML werden entsprechende Programme benötigt, welche die Wohlgeformtheit und Gültigkeit des XML-Inhalts prüfen.

12. Das Client/Server-Modell wird weiterhin die vorherrschende Architektur bei eBusiness- und verteilten Anwendungen in der Mess- und Automatisierungstechnik sein.

13. Der Einsatz von Komponenten in Form von Web Services ermöglicht der Gesamtanwendung eine sichere, erprobte und stets aktuelle Lösung einer Teilaufgabe.

14. Andere Geräteklassen, wie z.B. PDA, werden zunehmend an Bedeutung in der Automatisierungstechnik erlangen.