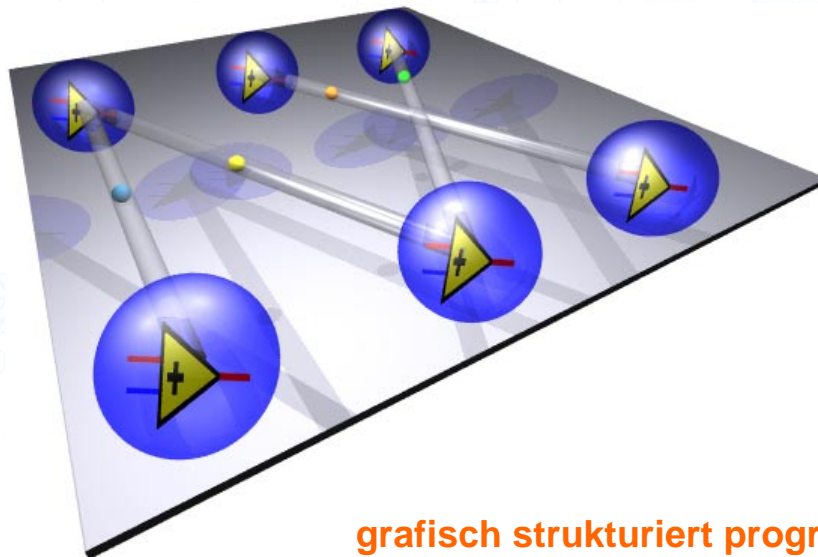


Graphical Object Technology ObjectVIEW™ für LabVIEW™

The future belongs to those who can find simplicity

Where everyone else sees only complexity

Peter Denning/Robert Metcalfe



grafisch strukturiert programmieren



Jenaer Straße 7
D- 07778 Dornburg

Phone:
+49 (0)36427-200 30

Fax:
+49 (0)36427-200 31

Email : info@vat.de
<http://www.vat.de>

Motivation

Die Objekt-Technologie ermöglicht den Übergang vom softwaremäßigen Handwerkszeitalter in das softwaremäßige Industriezeitalter. Der gesamte Prozeß vom Systementwurf bis zur Wartung wird revolutioniert. [Furr 99]

Meßbare Vorteile

1. Verkürzung der Time to Market
2. Verkürzung der Time to Market
3. Verkürzung der Time to Market

Das ist kein Druckfehler es unterstreicht die Wichtigkeit der Tatsache.

Time to Market :
Summe aller Aufwände (Zeit, Geld und Ressourcen) für eine erfolgreiche Implementation.

Das ObjectVIEW™ bietet die Grundlage für Systeme, die sich schnell an wechselnde Anforderungen adaptieren lassen, d.h. für agile und adaptive, zukunftsfähige Systeme.

ObjectVIEW™ ist die Voraussetzung, um mit LabVIEW™ komplexe Systeme klar strukturiert zu verwirklichen.

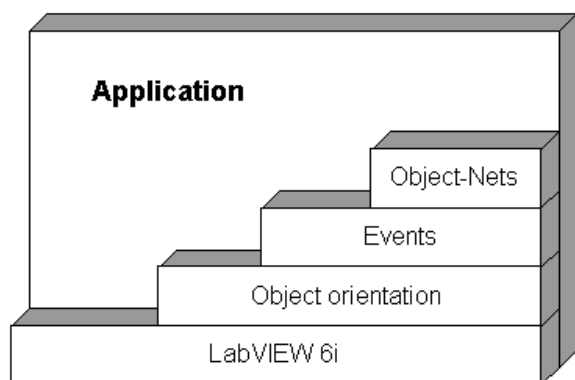
Deshalb arbeiten zur Zeit weltweit ca. 50% aller Programmierer objektorientiert [Balz 00].

ObjectVIEW™ für LabVIEW™

ist die Synthese der Konzepte:

- Datenflussprogrammierung LabVIEW™
- Objektorientierung Klassen, Vererbung, Prozess
- Ereignisorientierung Event, Message, Signale
- Petri-Netze Nebenläufigkeit im Objekt
- Hierarchische Objektnetze Aggregation, Ports

zur Erstellung von Softwaresystemen.

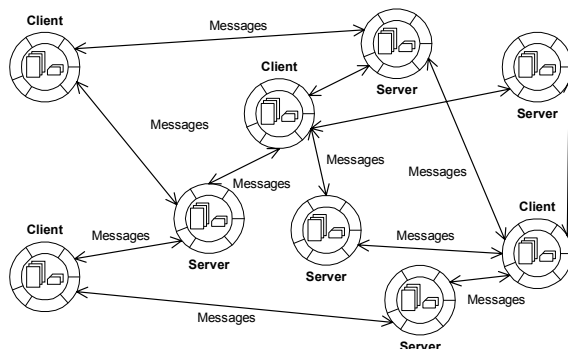


Alles in purem LabVIEW™ !!!

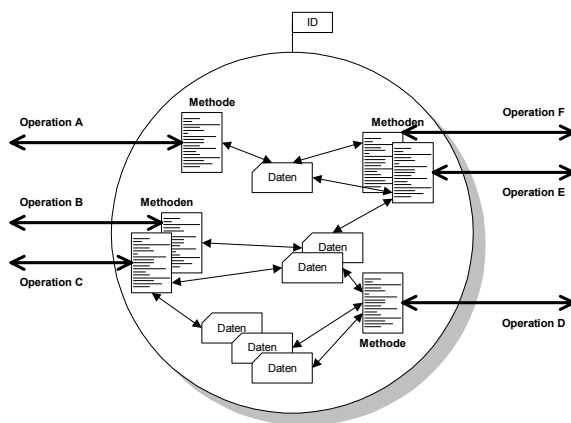
Objektorientierung - Beherrschung der Komplexität

Grundidee ist:

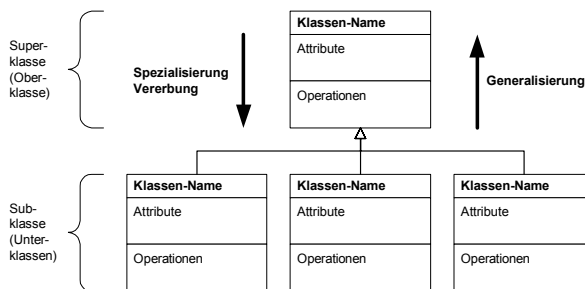
- der Aufbau von Anwendungen aus einer Menge von unabhängigen, gleichzeitig agierenden Objekten (verteilte nebenläufige dezentrale Objekte).



- Objekte kapseln Daten und Verarbeitungsprozeduren.



- Objekte abstrahieren reale Gebilde nur in ihren für das Projekt wesentlichen Eigenschaften und erben ihr Verhalten von ihren Oberklassen in der Klassenhierarchie und erweitern dieses durch Spezialisierung.



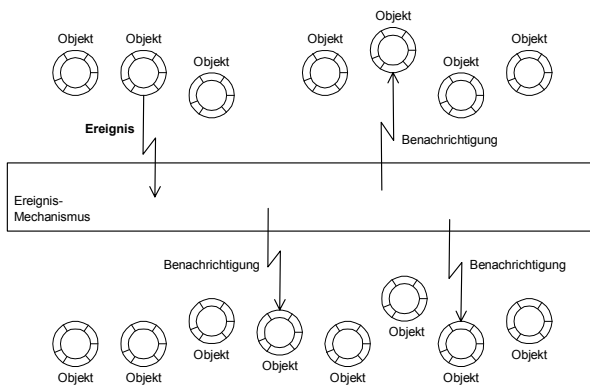
- Objekte verschiedener Klassen haben die Fähigkeit, auf gleiche Botschaften unterschiedlich zu reagieren (Polymorphismus).

Botschaften/Ereignisorientierung - Virtual Wire (Draht in LabVIEW)

Die Objekte kommunizieren untereinander über Botschaften, Kanäle und Signale.

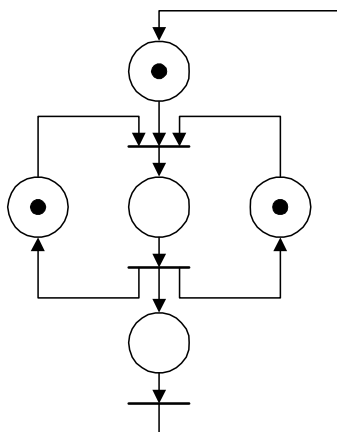
Synchrone (wartende) und asynchrone (nicht wartende) Botschaften lösen im Zielobjekt eine Unterbrechung (Event) aus und gestatten so die Reaktion des Objektes auf die Botschaft.

Durch die Ereignisorientierung werden Methoden nur dann aufgerufen, wenn neue Informationen im Objekt eingetroffen sind. Die Objekte (Prozesse) benötigen nur dann CPU-Zeit, wenn etwas getan werden muß. [Hüse 95]



Petri-Netze

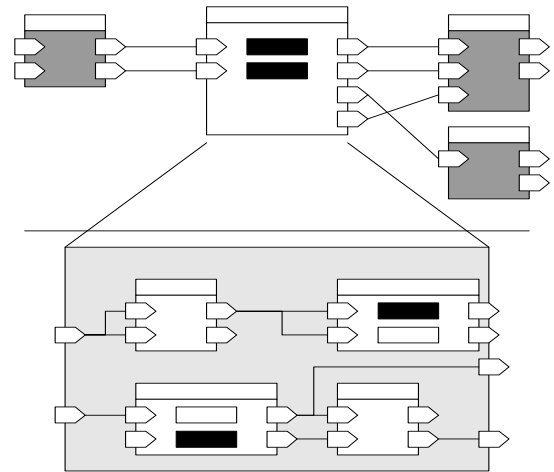
Petri-Netze ermöglichen die Beschreibung von dynamischen Systemen mit nebenläufigen und nichtdeterministischen Vorgängen. [Balz 00], [Baum 96], [Jens 97]



Objekt-Netze

Die Kommunikation der aktiven Objekte untereinander kann mit Objekt-Netzen beschrieben werden. Theoretische Grundlage ist die DEVS-Theorie und das darauf aufbauende Konzept der hierarchischen Objekt-Netze.

DEVS (discrete event system specification) University of Arizona (Tucson, USA) [ZeHaSa 97],



Objektorientierte Konzepte

Ein Objekt wird durch ein VI realisiert. Dieses wird zur Laufzeit mit „object.new“ erzeugt und erhält eine Identität (Name und Referenz).

Passive Objekte enthalten nur Daten (Attribute), auf die mit Methoden der Klasse zugegriffen wird und realisieren den wechselseitigen Ausschluß des Zugriffs paralleler Prozesse.

Aktive Objekte enthalten zusätzlich einen Prozess, d.h. das VI enthält ein Diagramm, das gestartet wird. Dieser Prozess ist meistens als Schleife ausgebildet. In der Zeit zwischen den Schleifendurchläufen wird passiv auf eintreffende Ereignisse gewartet. Am Ende des Objektlebenszyklus wird die Schleife verlassen und das Objekt aus dem Speicher entfernt.

Gemeinsame Daten aller Objekte (Klassenattribute) und Methoden einer Klasse sind möglich. Jede Klasse ist außerdem mit einer Klassenverwaltung ausgestattet, d.h. jede Klasse kennt ihre Instanzen.

Neue Klassen können durch Vererbung von bestehenden abgeleitet werden, auch die Mehrfachvererbung (d.h. eine Unterklasse ist Spezialisierung von mehreren Oberklassen) ist möglich. Es entstehen Klassenhierarchien.

Botschaften / Ereignissteuerung –Virtual Wire

(Draht in LabVIEW)

Botschaften können synchron über Methodenaufrufe übermittelt werden. Dabei werden Daten an das Objekt übergeben und/oder vom Objekt abgefragt. Ein direkter Zugriff auf die Attribute des Objektes ist nicht vorgesehen (Geheimhaltungsprinzip).

Eine weitere Möglichkeit ist die asynchrone Kommunikation über gepufferte Kanäle. Sie ermöglicht den Message-Empfang, auch wenn das Objekt nicht auf Botschaften wartet. Signale ermöglichen eine ereignisgesteuerte Verteilung von Informationen an beliebig viele Objekte (Broadcast, Multicast). Objekte bestimmen durch ihren Zustand (Daten) und ihr Verhalten (Methodenaufrufe), welche Botschaften sie empfangen können.

Zustandsautomaten / Petri - Netze / Objekt - Netze

Für alle nichttrivialen Objektlebenszyklen kann dieser durch endliche Automaten beschrieben werden. Es gibt Klassen für flache und Harel-Automaten .

Petri-Netze erlauben die Abbildung von dynamischen Systemen mit nebenläufigen und nicht-deterministischen Vorgängen. Eine direkte graphische Eingabe der Netze ist realisiert.

Als Netztypen sind

- Bedingungs/Ereignis-Netze (Marken Boolean),
- Stellen/Transitions-Netze (mit mehreren Marken) und
- Prädikat/Transitions-Netze (mit individuellen, gefärbten Marken)

möglich.

Unterschiedliche Netze (Typen) können gekoppelt und hierarchisch strukturiert werden. Es sind Petri-Netze in Objekten und Petri-Netze mit Objekten möglich.

[ZaHe 98], [NüFeBö 99], [HoVe]

Klassenbibliotheken

Alle Funktionen sind als Klassen realisiert (nicht bei der Basic-Version). Durch Templates ist der Anwender in der Lage, sich von allen Systemklassen eigene Anwenderklassen abzuleiten und die Funktion zu erweitern.

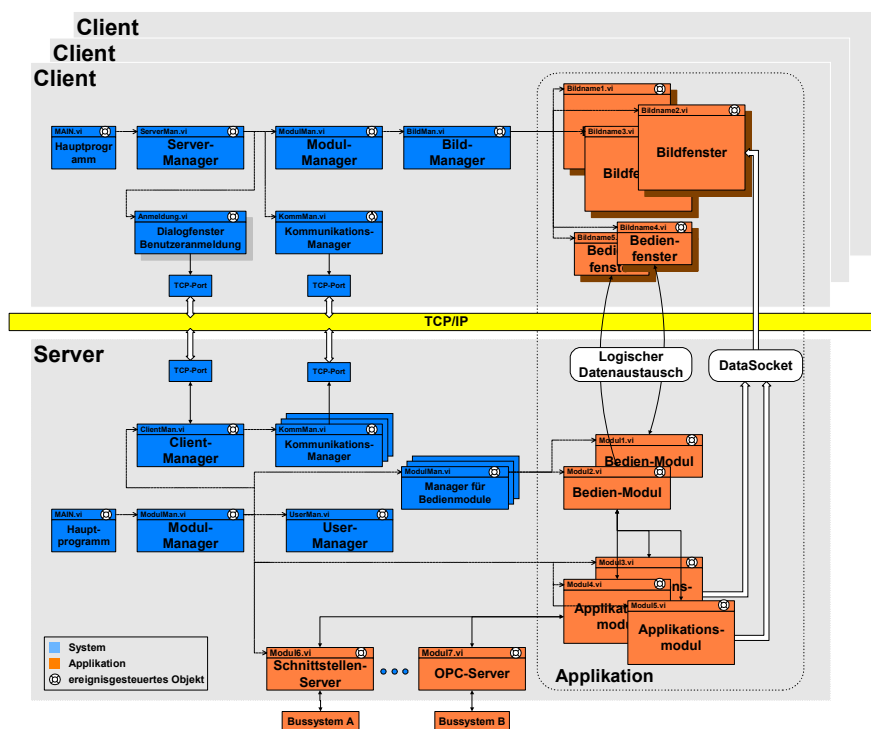
Auch von diesen Anwenderklassen können nach Erstellung von Templates wieder neue Klassen abgeleitet werden. Die Mehrfachvererbung, eine Unterklasse hat mehrere Oberklassen, wird unterstützt.

Framework für Client Server Systeme

Dezentrale Systeme erfordern anwendungsunabhängige Grundbausteine. Diese Klassen zur Realisierung von komplexen Client-Server-Applikationen werden in Form eines Frameworks zur Verfügung gestellt. Das Framework ist modular aufgebaut und enthält Benutzer- und Zugriffsverwaltung, Kommunikationsbausteine, Templates für Trend und Meldungsverwaltung, Bild- und Menü-Manager sowie Templates für anwendungsspezifische Objekte.

Die Wirkungsweise aller Module kann durch eine Musterapplikation Gebäudemanagement einfach nachvollzogen werden.

Das Universal-Toolset ermöglicht es dem Anwender, hochwertige Client-Server-Visualisierungssysteme auf Basis der Datenflussprogrammierung und Objekt-Technologien zu erstellen



Literaturverzeichnis

(Fachbücher)

[JaHa 01] Jamal, Rahman; Hagestedt, Andre: **LabVIEW Das Grundlagenbuch**, Addison-Wesley Pearson Education Deutschland GmbH, München, 3. Auflage, ISBN 3-8273-1714-2, 2001

[Balz 00] Balzert, Helmut: **Lehrbuch der Software- Technik**, Software-Entwicklung, Spektrum Akademischer Verlag Heidelber, Berlin, 2. Auflage, ISBN 3-8274-0480-0, 2000

[Balz 00] Balzert, Heide: **Objektorientierung in 7 Tagen**, vom UML-Modell zur fertigen Web- Anwendung, Spektrum Akademischer Verlag Heidelberg, Berlin, ISBN 3-8274-0599-8, 2000

[Baum 96] Baumgarten, Bernd: **Petri- Netze Grundlagen und Anwendungen 2. Auflage**, Spektrum Akademischer Verlag Heidelberg, ISBN 3-8274-0175-5, 1996

[Flan 98] Flanagan, David: **Java in a Nutshell** (deutsche Ausgabe für Java 1.1, O'Reilly Verlag Köln, ISBN 3-89721-100-9, 1998

[Furr 99] Furrer, Frank J.: **Objekt- Technologie** (Band 1), SYSLOGIC-Press (Schweiz), ISBN 3-9520919-1-X, 1999 <http://www.objecttechnologysource.com>

[HeGuDy 95] Hering, Ekbert, Gutekunst; Jürgen; Dyllong, Ulrich: **Informatik für Ingenieure**, VDI-Verlag, ISBN 3-18-400944-0, 1995

[Hüse 94] Hüsener, Thomas: **Entwurf komplexer Echtzeitsysteme**, BI- Wiss.- Verlag Mannheim, Leipzig, Wien, Zürich, 1994

[Hüse 95] Hüsener, Thomas: **Objektorientierter Entwurf von nebenläufigen , verteilten und echtzeitfähigen Softwaresystemen**, Spektrum Akademischer Verlag Heidelberg, ISBN 3860257080, 1995

[Jens 97] Jensen, Kurt: **Coloured Petri Nets Basic Concepts, Analysis Methods and Practical use** (Band 1..3), Springer Verlag Berlin- Heidelberg- New York, ISBN 3-540-60943-1..3, 1997

Literaturverzeichnis

(Fachzeitschriften, Internet)

[HoVe] Holvoet, Tom; Verbeaten, Pierre: **PN- TOX: a Paradigm and Development Environment for Object Concurrency Specifications**, Dept. of Computer Science, K.U.Leuven, Belgien <http://www.cs.kuleuven.ac.be/~tom/publications.html>

[NüFe 97] Nützel, Jürgen; Fengler, Wolfgang: **Objektorientiertes Entwurfsmodell für Steuerungssysteme auf Basis der Petri-Netz- Theorie**, TU- Ilmenau, www.theoinf.tu-ilmenau.de/ossi-project , 1997

[NüFeBö 99] Nützel, Jürgen; Fengler, Wolfgang; Böhme, Thomas: **Objektorientierter Entwurf verteilter eingebetteter Echtzeitsysteme auf Basis höherer Peti- Netze**, Promotion TU-Ilmenau, , <http://www.theoinf.tu-ilmenau.de/~nuetzel>

[ZaHe 98] Zapf, Michael; Heinzl, Armin: **Ansätze zur Integration von objektorientierten Konzepten und Petri- Netzen**, Universität Bayreuth, www.wi.oec.uni-bayreuth.de, <http://wi.oec.uni-bayreuth.de/mitarbeiter/zapf/> , 1998

[ZeHaSa 97] Zeigler, Bernard P.; Hall, Steve B.; Sarjoughian, Hessam S.: **DEVJAVA User's Guide**, The University of Arizona, www.acims.arizona.edu/DEVS_HLA/devs_hla.html, 1997

[ZeHaSa 97] Zeigler, Bernard P.; Hall, Steve B.; Sarjoughian, Hessam S.: **DEVs-Java Reference Guide**, The University of Arizona, http://www.acims.arizona.edu/DEVs_HLA/devs_hla.html, 1997



LabVIEW ist ein eingetragenes Warenzeichen der National Instruments Corporation. ni.com/labview

Graphical Object Technology

ObjectVIEW™ für LabVIEW™

Autor: Dipl.-Ing. (FH) Jens Vogel

Geschäftsführer Vogel Automatisierungstechnik GmbH

© Copyright 04/2001 Vogel Automatisierungstechnik GmbH. Alle Rechte vorbehalten. Produkt- und Firmennamen sind eingetragene Warenzeichen ihrer Hersteller.



Jenaer Straße 7
D- 07778 Dornburg

Phone:
+49 (0)36427-200 30

Fax:
+49 (0)36427-200 31

Email : info@vat.de
<http://www.vat.de>