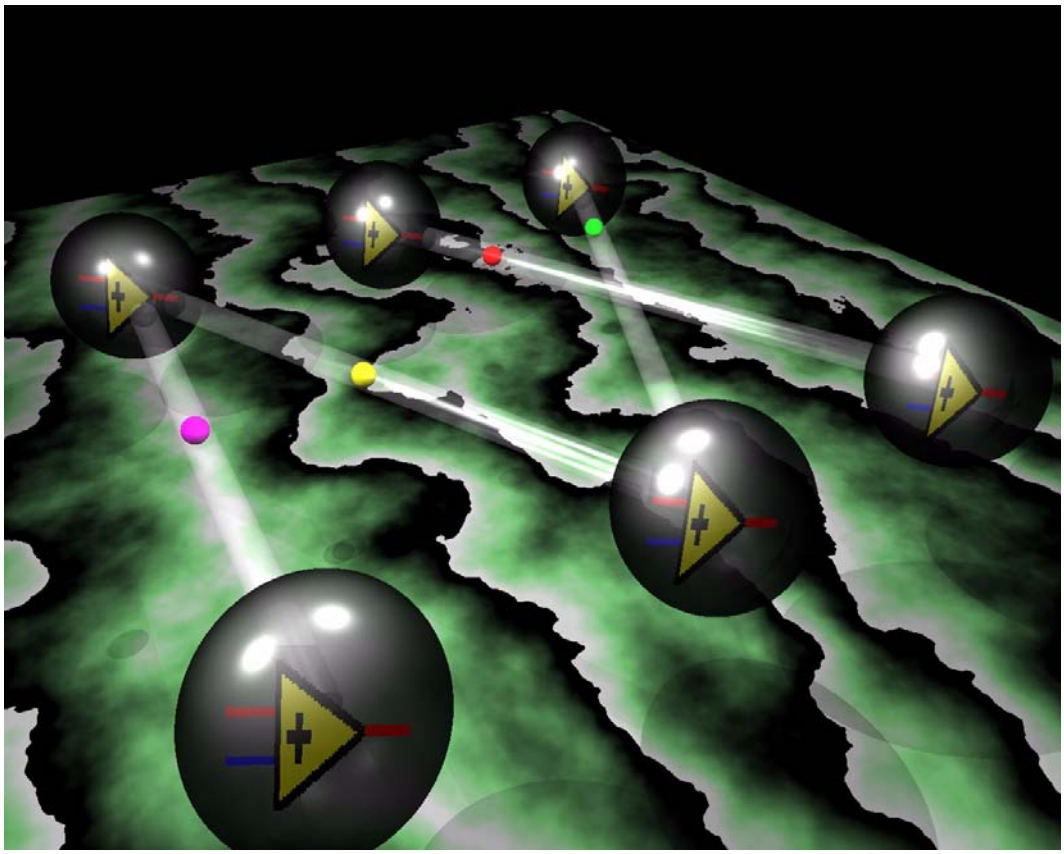
G++ – Objektorientierte und ereignis-getriebene Programmierung mit LabVIEW

Herbert Pichlik
Jens Vogel



Evolution

Ein Bild sagt mehr als tausend Worte. Bei vielen Software- und Systementwicklern scheint dieses Sprichwort der neue Leitsatz für die Lösung von Problemen zu sein. Programmieren mit Bildern ist seit fast zwanzig Jahren Realität.

Um jedoch mit herkömmlichen textbasierten Paradigmen konkurrieren zu können, mussten leistungsfähige Betriebssysteme mit einer grafischen Benutzeroberfläche zur Verfügung stehen.

Der gnadenlose Konkurrenzkampf im Bereich der PC-Technologie beschert dem Endanwender hochperformante und äusserst preisgünstige Plattformen. Der Krieg um die Desktops scheint zu Gunsten von Microsofts Windows Derivaten entschieden zu sein. Alle UNIX-Abkömmlinge (HP-UX, IBM AIX, SINIX, Solaris, Linux, Mac OS X(!), etc.) zusammengenommen erreichen nicht einmal 10% Marktanteil. Ähnlich sieht es auch in der Anwender- und Entwicklersoftware aus. Windows gestützte Systeme supporten mit Abstand die meisten Applikationen.

Erst in der letzten Dekade jedoch konnte Windows (vor allem Windows NT/2000) zu einem modernen und stabilen Betriebssystem reifen. Vorher stellte Windows keine ernsthafte Alternative zu 32bit-Betriebssystemen mit grafischen Benutzeroberflächen wie z.B. MacOS- und somit auch keine Plattform für die Entwicklung grafischer Werkzeuge dar.

Bereits 1983 begannen Jeff Kodosky und sein Team bei National Instruments in Austin Texas die Entwicklung eines vollkommen neuen grafischen Softwareentwicklungssystemes zum schnellen Erstellen, Prüfen und Modifizieren von Messgerätesystemen basierend auf MacOS.

Es entstand ein völlig neues Paradigma, das Problemlösungsansätze in einer neuen Dimension bietet - die Grafische Datenflussprogrammierung in Form sogenannter Virtueller Instrumente.

National Instruments nannte auf Grund seines gedachten Laboreinsatzes das Produkt LabVIEW (**Lab**oratory **VI**rtual **E**ngineering **W**orkbench).

Mittlerweile hat sich LabVIEW eine Vielzahl von neuen Domänen erschlossen. Dass LabVIEW noch nicht in alle IT-Welten eingedrungen ist, liegt unter anderem daran, dass es bis jetzt nicht möglich war, die Vorteile der Grafischen Datenflussprogrammierung mit modernen ereignisgetriebenen und objektorientierten Ansätzen zu verbinden.

Dies hat sich mit der Einführung der Objektorientierten Toolsets für LabVIEW (OTT) der Firma Vogel Automatisierungstechnik grundlegend geändert.

Virtuelle Instrumentierung

LabVIEW ist eine Anwendungsentwicklungsumgebung, die sich von konventionellen, textbasierten Programmiersprachen vor allem in der Art der Kodierung unterscheidet. LabVIEW Programme erstellt man grafisch, und zwar mit Hilfe von Blockdiagrammen und Frontplattenelementen.

In Analogie zu den Frontplatten von Meßgeräten, welche Bedien- und Anzeigeelemente zur Verfügung stellen, und als eigentliche Schnittstelle zwischen Anwender und Gerät dienen, ist das Frontpaneel (*Front Panel)* die Bedienoberfläche eines Programmes, das Anwendern ermöglicht, mit dem Programm in Interaktion zu treten und die Ergebnisse der Anwendungen zu visualisieren.

Die Funktionalität eines Meßgerätes wird durch elektronische/elektrische Komponenten bestimmt, die mit Hilfe von Schaltkreisen miteinander verbunden sind. Das Blockdiagramm (*Block Diagram)* enthält Verbindungen von grafischen Symbolen mit Hilfe von softwarebasierten Datenleitungen in Form von Drähten.

Das *Block Diagram* ist der eigentliche Programmcode - ein grafischer Quellcode, der im Gegensatz zum konventionellen, textorientierten Quellcode als Blockschaltbild im Sinne einer Signalflussdarstellung "gezeichnet" wird. Bestandteile dieses Blockschaltbildes können sowohl einfache mathematische Operatoren wie Addierer, Multiplizierer, etc. sein, als auch komplexe Funktionen wie FFT oder selbstdefinierte SubVIs (Unterprogramme). Mit Hilfe von SubVIs ist eine modulare und hierarchische Abstrahierung ohne Einschränkung in Bezug auf die Anzahl der Hierarchieebenen möglich.

Offenheit, Multiplattformfähigkeit und Parallelität

Die in LabVIEW eingebettete Datenflussmaschine stellt Parallelisierungs- und Multitaskingkonstrukte zur Verfügung. Anders als bei der herkömmlichen textbasierten Programmiersprachen, die von Natur aus sequentiell arbeiten und nur durch aufwendige Programmierung (z.B. mit Hilfe von Semaphoren) parallelisierten Code erzeugen können, ist es in LabVIEW sehr leicht möglich, konkurrierende, einzeln ablaufende und leicht synchronisierbare, Programmteile zu erzeugen. Die Ursache liegt in der Tatsache begründet, daß LabVIEW im Unterschied zu textbasierten Paradigmen, die auf Kontrollflussmechanismen basieren, rein datenflussgetrieben ist. Der datenflussbasierte Ansatz ist eine wesentlich elegantere und natürlichere Art der Formulierung von Problemlösungen. Gerade im Zeitalter der massiv parallelisierenden Strukturen (innerhalb einzelner Prozessoren und durch Einsatz von Multiprozessorsystemen) bei leistungsfähigen Computerarchitekturen spielt der Datenflussansatz seine Stärken aus. Der Anwender hat dadurch unschätzbare Vorteile. Anstatt sich auf irrelevante implementierungsspefische Parallelisierungmechanismen stürzen zu müssen, ist es einem LabVIEW-Programmierer möglich, Probleme zu formulieren und Lösungen zu realisieren, ganz im Sinne des Rapid-Prototyping-Ansatzes.

Das multiplattformfähige LabVIEW supported Multithreading auf allen wichtigen Betriebssystemen, die preemptives Multitasking unterstützen (Windows 2000/98/ME, Linux, Solaris, Concurrent PowerMAX). Preemptives Multitasking hat eine Reihe von gravierenden Vorteilen gegenüber dem kooperativen Pendant. Die Threads können vor Threadende (preemptiv) verlassen werden, um andere Threads weiterbehandeln zu können.

LabVIEW, mit seinem Grafischen Compiler erzeugt sehr schnellen Programmcode, vergleichbar mit C++.

Offenheit wird in Zusammenhang mit Softwareumgebung häufig als Schlagwort gebraucht und hat sich zu einer Art Modewort der Branche entwickelt. Doch beim näheren Betrachten gewährleisten viele, als offen angepriesene, Systeme nicht das, was der Anwender unter Offenheit erwartet. Offenheit kann im praktischen Einsatz einer SW-Entwicklungsumgebung viele Bedeutungen haben, wobei jeder Anwender seine individuellen Schwerpunkte setzen wird. LabVIEW unterstützt alle relevanten Schnittstellen und Kommunikationsmechanismen auf allen relevanten Plattformen. Es gibt Treiber für alle relevanten Mess- und Automatisierungskomponenten.

Nachteile

LabVIEW bietet gegenüber textbasierten Ansätzen eine Vielzahl von strategischen Vorteilen wie die bessere Eignung zu RAD (Rapid Application Development) und RAD (Rapid Application Prototyping). LabVIEW ist ausserdem leichter erlern- und wartbar.

Was LabVIEW bis jetzt nahezu völlig gefehlt hat, sieht man vom Ende der 90er Jahre rudimentär implementierten GOOP-Ansatz (Graphical Object Oriented Programming) ab, ist Grafische Objektorientierung mit Mehrfachvererbung und ein Ereignisflussmodell.

Traditionelle Programmentwicklung in LabVIEW beruht auf dem hierarchisch und modular aufgebauten Datenflussmodell.

Die Entwicklung von LabVIEW-Programmen ist sehr oft getragen von anachronistischen Design-Methodologien wie dem Top-Down Designansatz. Oft ist die grafische Benutzeroberfläche der zentrale Bestandteil des Toplevel-VIs und die SubVIs bilden in der Regel keine strategischen Bestandteile innerhalb der Problemdomäne. Die sogenannte Applikationsschicht (applications layer) ist in herkömmlichen Ansätzen kaum berücksichtigt.

Bis zu einem gewissen Grad sind traditionelle LabVIEW-Programme zwar noch gut skalier- und wartbar. Übersteigen derartige – monolithische - VIs eine gewisse Grössenordnung, sind sehr grosse Anstrengungen zu unternehmen, will man massive Performanceeinbrüche vermeiden. Viele monolithische Programme bestehen aus parallelen State-Machines mit einer Vielzahl von Schleifen und Schieberegistern, die alle parallel abzuarbeiten sind und Ressourcen beanspruchen. Um ein optimales Ressourcenmanagement bei nicht objektorientierten Ansätzen zu bewerkstelligen, ist der breitbandige Einsatz von Softwarekomponenten wie Occurencies, Queues, Notifications, Call by Reference Nodes, Semaphoren und Rendesvousz unumgänglich. Da viele LabVIEW-Programmierer jedoch nur geringe Softwareengineering-Kenntnisse besitzen stellt dies eine bedeutende Hürde bei der Lösung komplexerer Probleme dar.

Monolithische Applikationen haben ausserdem das grosse Problem, dass eine Veränderung des Programmcodes bei Main- oder bei SubVIs leicht zu negativen Seiteneffekten innerhalb der Applikationen führen kann. Der Grund hierfür ist unter anderem darin zu suchen, dass viele Status- und Parametrierinformationen als Globale Variablen implementiert sind. Eine vollständige Applikationskapselung ist damit unmöglich.

Die Wartung sehr grosser monolithischer Anwendungen ist auf Grund der oben beschriebenen Eigenschaften sehr zeit- und damit natürlich auch kostenintensiv.

Da in der Industrie die Mess- und Automatisierungslösungen jedoch sehr lange im Einsatz sind (teilweise über 10 Jahre) ist eine kosteneffiziente Wartung der Applikationen notwendig.

Einen Ausweg aus dieser Krise zeigen die Objektorientierten Toolsets für LabVIEW (OTT) der Firma Vogel Automatisierungstechnik.

Objekttechnologie Toolsets für LabVIEW (OTT)

OT-Toolsets repräsentieren eine Synthese unterschiedlicher Designkonzepte angefangen vom Lab-VIEW-immanenten Datenflussansatz über Objektorientierte Ansätze mit Klassen und Mehrfachvererbung , Ereignis-, Signal- und Botschaftenmanagement, verteilten – inherent parallelen- intelligenten Objekten, Aggregation, Ports, Petrinetzen und Objektnetzen.

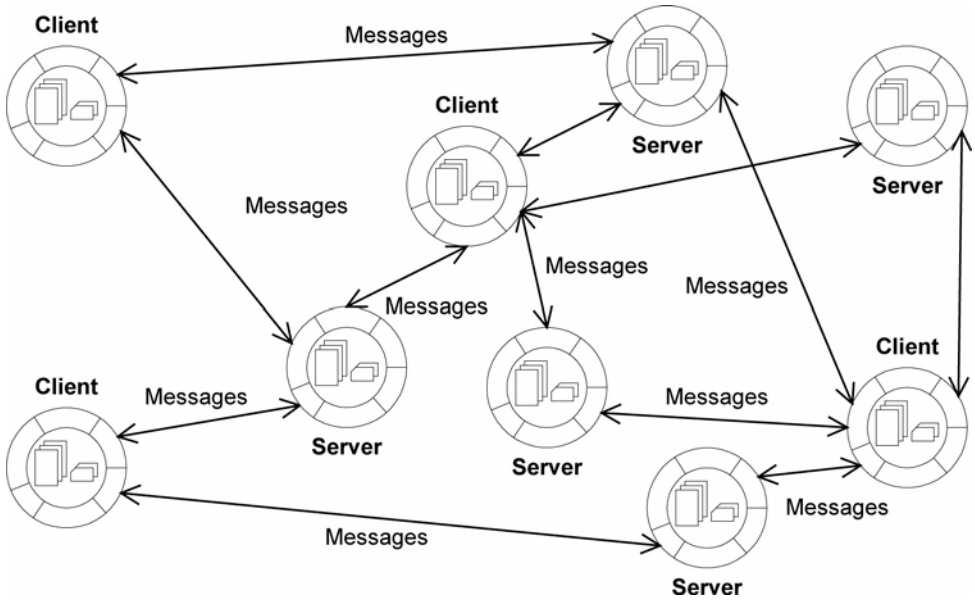


Bild 1: Objects sending messages in a Client Server environment

Die Verwendung dieser Technologien führt zu verteilten-, leicht wiederverwendbaren-, extrem skalier- und wartbaren-, redundanten-, robusten und sehr zuverlässigen intelligenten Objekten, die eine direkte Problembeschreibung und –abstrahierung ermöglichen. Die Objekte sind im LAN und eingeschränkt auch in WAN-Umgebungen (inklusive Internet) verteil- und administrierbar.

Die Basisversion (Object Event Toolset – OET) unterstützt Aktive Objekte und Eventmanagement basierend auf asynchronen Botschaften (mit Hilfe von Queues), Signalisierungen (über Notifikationen) und weit verbreitete Kommunikationstechnologien wie ActiveX und OPC/DataSocket.

Die Professional Version (Object Inheritance Toolset, OIT) stützt sich auf OET Features und bietet darüber hinaus Passive Objekte, Klassenbibliotheken und Klassenbibliotheksmanagement, Mehrfachvererbung, Methoden, Methodenaufrufe sowie Flache Zustandsautomaten. Das Developer Toolset (Object Network Toolset, ONT) ist ein Superset des Professional Toolset. Harel State Machines, Petrinetze (Platz-Transitionsnetze, Kolorierte Petrinetze, Prädikat-Transitionsnetze, Hierarchische Netze, Objektnetze und Ports).

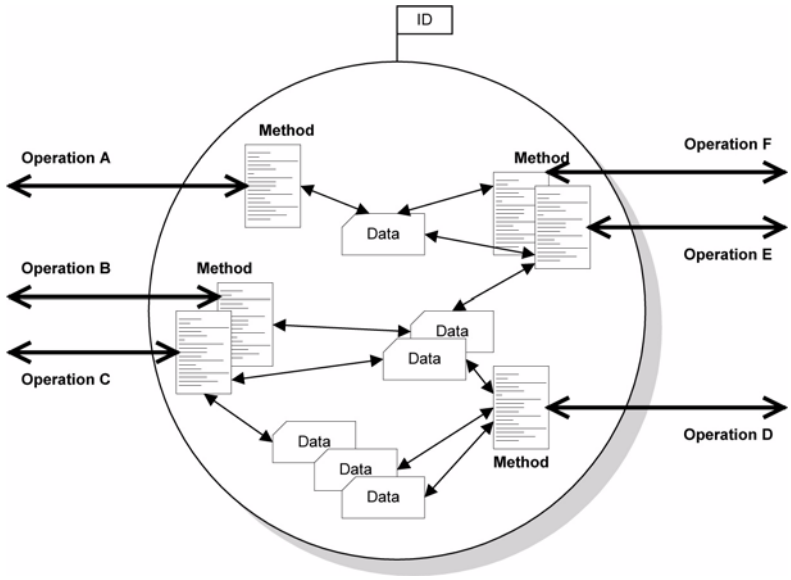


Bild 2: Encapsulation of Objects

Das Universal Toolset (Application Framework Toolset, AFT) nutzt alle Features der untergeordneten Toolsets und bietet unterschiedliche Klassenbibliotheken wie Client Server, Node Communication/node Management, Message-Management, Trends, Error Management, Facility Management, Enterprise Ressource Planning Libraries (ERP) und Manufacturing Execution System Class Libraries (MES).

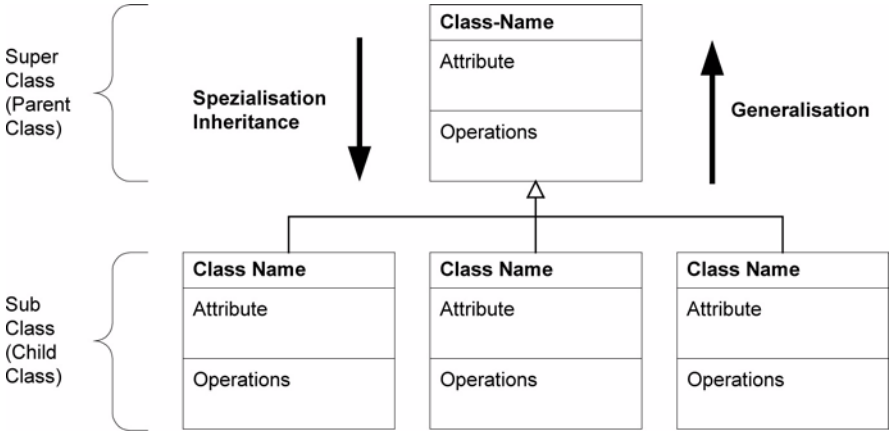


Bild 3: Basic class hierarchy structure

Es ist sehr leicht möglich neue Objekte aus Elternklassen abzuleiten. In der Klassenhierarchietiefe gibt es keine Einschränkungen. Die Abstrahierungsmöglichkeiten der OT-Toolsets gehen weit über die immanenten Möglichkeiten LabVIEWs hinaus.

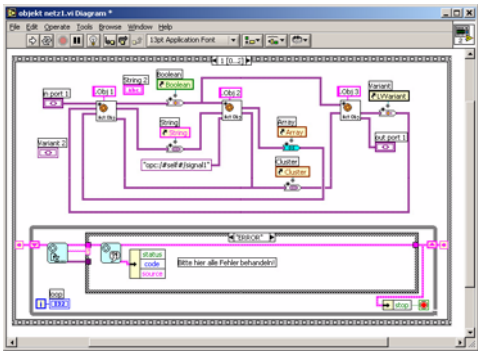


Bild 4: Objekt-netz.tif

Eine herausragende Möglichkeit nebenläufige, nicht deterministische Vorgänge zu abstrahieren besteht in der Verwendung sogenannter Petrinetz. Carl Adam Petri entwickelte bereits in den sechziger Jahren diesen genialen Ansatz der Beschreibung paralleler Abläufe und Prozesse.

Petri-Netze

Ein Petri-Netz besteht aus drei Netzelementen. Das sind einmal die **Plätze** und des weiteren die Übergänge zwischen ihnen. Diese Übergänge bezeichnet man als **Transitionen**. In einem Petri-Netz sind Plätze und Transitionen durch **Kanten** so verkettet, daß sie sich jeweils abwechseln. Zwischen zwei Plätzen befindet sich also immer ein Übergang (Transition). Die Verkettung der Netzelemente erfolgt durch sogenannte gerichtete Kanten. Durch das Petri-Netz bewegt sich eine oder auch mehrere Marken. Ein Platz ist dann aktiv, wenn er mindestens eine Marke besitzt. Die Transitionen bestimmen, ob und wann ein Platz die Marke an seinen Nachfolger abgeben muß. Dafür besitzt jede Transition eine sogenannte Weiterschaltbedingung. Ist diese Bedingung erfüllt, dann bewirkt die Transition die Weitergabe der Marke zwischen den zwei Plätzen, zwischen denen sie sich befindet. Die Weitergabe erfolgt immer gerichtet, entsprechend den Kanten, mit denen Plätze und Transitionen miteinander verkettet sind. Eine Marke kann sich während ihres Flusses durch das Petri-Netz in mehrere Marken aufteilen, die sich später wieder zu einer Marke vereinigen können.

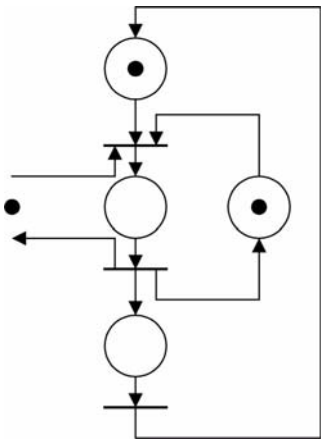


Bild 5: Einfache Petrinetz-Notation (State Transition Net)

Im Gegensatz zu einer Zustandsmaschine kann ein Petri-Netz mehrere aktive Zustände haben. In einem Petri-Netz besteht die Möglichkeit des Parallellaufs von Zuständen sowie der Einrichtung von Unterzuständen. Ein Petri-Netze ist darüber hinaus universeller als ein Programmablaufplan, weil es sich auf einer höheren Abstraktionsebene befindet. Auch ein Programmablaufplan kann nur eine Marke besitzen.

Mit dem Petri-Netz-Konzept ist bei hoher Übersichtlichkeit eine schrittweise Annäherung an die Problemlösung möglich. Damit bietet es sich für den Entwurf von Lösungen mit anfangs nicht genau bestimmbarer Struktur an, wie z.B. komplexe Automatisierungslösungen oder Workflow-Applikationen.

Eine erste Implementierung von 2000 verwendete nur die binäre Markierung. Die OTT Petri-Netze realisieren alle Netztypen ereignisgesteuert. Es sind alle Netztypen möglich, Bedingungs-Ereignissnetze (Plätze mit einer Marke), Stellen-Transtionsnetze (Plätze mehreren Marken) und Prädikat-Transtionsnetze (Plätze tragen beliebige Daten).

Es sind mehrere Petri-Netze erstellbar, diese können hierarchisch strukturiert und synchronisiert werden. Das Werkzeug bietet die Möglichkeit der Instanzierung von Petri-Netzen, d.h. Petri-Netze in aktiven Objekten und Objekten auf den Plätzen von Petrinetzen (Referenznetze).

Das Petri-Netz ist direkt im Blockdiagramm eines VIs erstell- und editierbar. Die Symbolik entspricht der allgemein üblichen Notation für Petri-Netze, wobei die grafischen Möglichkeiten von LabVIEW derzeit noch kleine Kompromisse nötig machen. Für Plätze und Transitionen werden bereitgestellte VI's in das Diagramm eingefügt. Mit diesen Grundelementen können Petri-Netze hoher Komplexität in LabVIEW erstellt werden. Plätze und Transitionen werden durch Verdrahtung miteinander verbunden. Die Verdrahtung symbolisiert die Kanten, über die in einem Petri-Netz die Marken weitergegeben werden. Die Zustände der Plätze können zur weiteren Verarbeitung abgefragt werden. Die Abfrage der Zustände kann aus jedem VI eines Projektes erfolgen. Wie jedes LabVIEW-Diagramm kann das VI mit dem Petri-Netz nach fehlerfreier Erstellung sofort gestartet werden.

Für eine hohe Anschaulichkeit werden die Plätze durch den Anwender Indikatoren bezeichnet. Die Indikatoren zeigen den aktuellen Zustand des Platzes an. Die Ereignisteuerung der Transitionen führt zu einer hohen Abarbeitungsgeschwindigkeit. Alle Plätze werden als Kommunikationsobjekte realisiert . Die Transitionen verschieben die Marken durch Zugriff auf diese Kommunikationsobjekte. Dabei kann jede Transition nur diejenigen Plätze beeinflussen, die als Vorgänger- oder Nachfolgeplätze an sie angeschlossen sind. Über Array-Anschlüsse kann eine große Anzahl von Vorgänger- und Nachfolgeplätzen an jede Transition angeschlossen werden. Durch die Art der Implementation der Weiterschaltbedingung in der Transition ist der konsistente Zugriff auf die Kommunikationsobjekte sichergestellt.

Der Einsatz von Petri-Netzen in LabVIEW-Projekten bietet sich an, wenn eine oder mehrere der folgenden Bedingungen zutreffen:

- Der Objektlebenszyklus graphisch programmiert werden soll
- Ein System gleichzeitig mehrere Zustände annehmen kann
- Zustände in mehrere Unterzustände unterteilt sind
- Die Zustände in einem System von vielen äußeren Bedingungen abhängig sind
- Die Zustände in einem System teilweise voneinander abhängig und teilweise nicht voneinander abhängig sind
- In der Programmstruktur eine nachgeschaltete Datenverarbeitung von der Zustandserkennung eines Prozesses getrennt sein soll

Das Petri-Netz-Konzept bietet Zustandsflussabstrahierungsmöglichkeiten parallel zum Datenfluss. Der Anwender hat damit die Möglichkeit, in einem Projekt beide Programmiermethoden beliebig zu kombinieren. Damit wird die hohe Anschaulichkeit von LabVIEW auch bei zustandsorientierten Anwendungen erreicht.

Das Beispiel der dinnierenden Philosophen verdeutlicht die Einbindung von Petri-Netzen in Objekt-Netze und die Realisierung des Objektlebenszyklus mit Petri-Netzen.

Bild 6: Frontpanel der dinierenden Philosophen

Das Gesamt-Petri-Netz ist aufgeteilt in ein übergeordnetes- und 5 untergeordnete Netze (Hierarchie). Die untergeordneten Netze sind durch aktive Objekte (Objekte mit Prozess) realisiert, die als Instanzen der Klasse "Philosoph" dynamisch erzeugt werden. Das Unternetz "Philosoph" besteht aus zwei Transitionen, zwei Plätzen und einem Netzbeobachte. Die Transitionen und der Netzbeobachter arbeiten parallel und ereignisgesteuert, wobei jede Transition auf Änderungen aller mit ihr durch Kanten verbundenen Plätze ereignisgesteuert reagiert. Ist die Schaltbedingung (alle Vorgänger markiert und alle Nachfolger nicht markiert) gegeben, schaltet die Transition im wechselseitigen Ausschluß mit allen anderen Transitionen. Alle geänderten Plätze sind für weitere Veränderungen gesperrt. Nachdem der Netzbeobachter-Prozess alle Veränderungen registriert hat, erfolgt eine Freigabe der Plätze. Die Schaltgeschwindigkeit des Netzes ist durch die Verzögerung der Freigabe einstellbar.

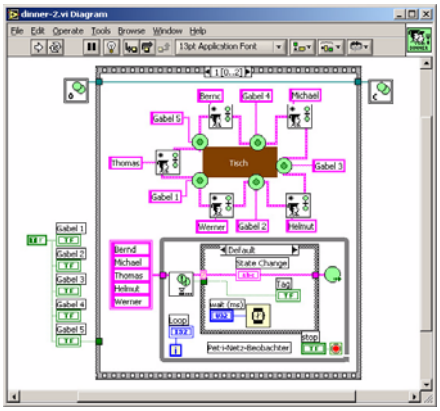


Bild 7: Blockdiagramm der dinierenden Philosophen

Die Plätze "Gabel A" und "Gabel B" jedes Philosophen sind im übergeordneten Netz ("Dinner") definiert. Mit Hilfe von Ports kann das lokale Petri-Netz über diese Plätze verfügen. Das Petri-Objekt-Netz "Dinner" ist durch 16 nebenläufige, ereignisgesteuerte und koordinierte Prozesse realisiert und gibt einen eindrucksvollen Einblick in die Leistungsfähigkeit der Petri-Objekt-Netz-Technolgie.

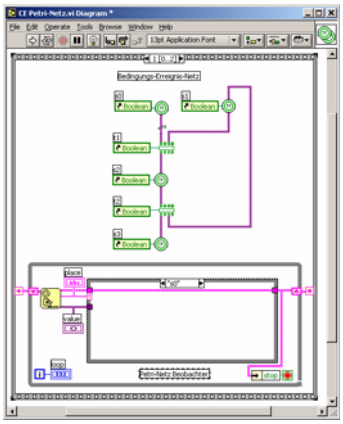


Bild 8: CE-Netz.tif

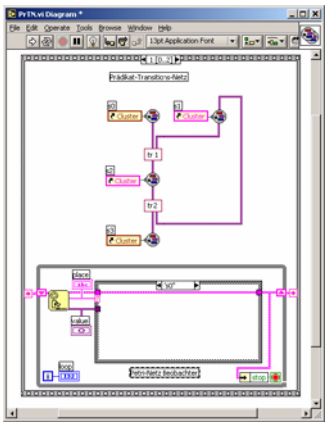


Bild 9: CP-Netz.tif

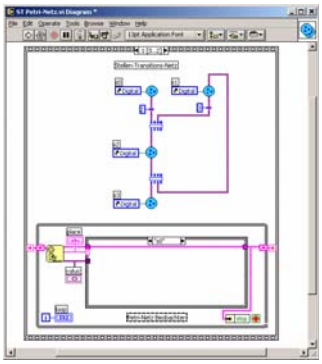


Bild 10: ST-Netz.tif

Anwendungsgebiete

Die Einsatzmöglichkeiten der Kombination LabVIEW und OT-Toolsets sind enorm. Prinzipiell sind alle softwaretechnischen Probleme damit lösbar. In den letzten Monaten sind eine Vielzahl von Applikationen mit Hilfe dieser Entwicklungsplattformen entstanden, die einen Teil der potentiellen Einsatzbandbreite wiederspiegeln. Die Firma Vogel Automatisierungstechnik hat eine komplexe Client-Server Applikation in Form eines Zementwerkautomatisierungssystemes mit einer Vielzahl von I/O-Punkten, Datenbanken Clients.

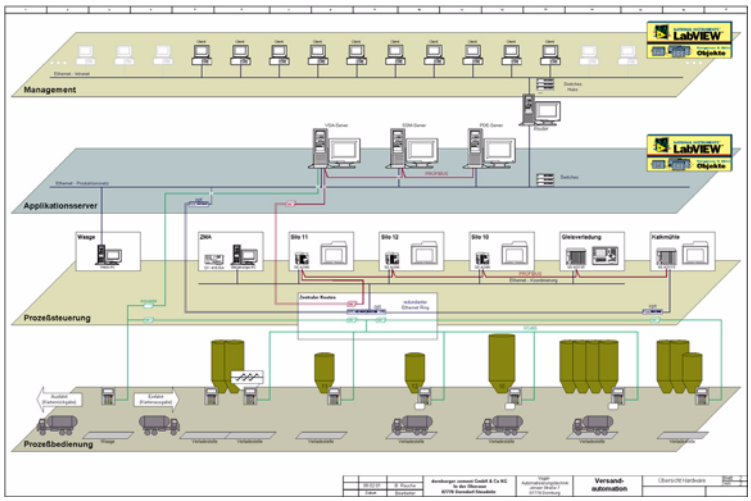


Bild 11: Übersicht Hardware.png

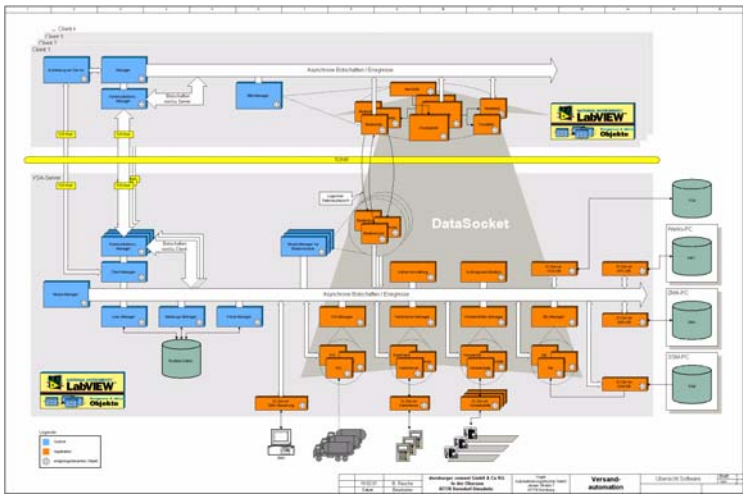


Bild 12: Übersicht Software.png

Die Firma Systec entwickelt zur Zeit ein völlig neues Mess- und Automatisierungskonzept basierend auf netzwerkbasierten Komponenten (PC, PXI, DAQ, SPS, GPIB, Profibus, CAN, etc.) und OT-Tools, genannt Universal Test and Automation System (UTAS).

Die Toolsets können jedoch auch Basis sein für völlig neue Werkzeuge im Bereich der technischen und nichttechnischen Informationstechnologie. MES- (Manufacturing Automation Systems) und ERP-Systeme (Enterprise Ressource Planning) sind neue Toolset-Domänen. Die toolsetbasierte Petrinetztechnologie innerhalb der Grafischen Datenflussprogrammierungsplattform ist ideale Plattform z.B. für die Implementierung hocheffizienter Workflowmanagementkonzepte.

Fazit

Grafische Datenflussprogrammierung unterscheidet sich fundamental von herkömmlichen textorientierten Entwicklungstechniken. Der Idealfall einer grafischen Datenflussprogrammierumgebung ist die Zusammenfassung von Problembeschreibung, Problemlösung und aussagekräftiger, selbsterklärender Dokumentation in einem Arbeitsgang. Das Werkzeug sollte ein, mit herkömmlichen Programmiersprachen vergleichbares Laufzeitverhalten (bezogen auf die Ausführungsgeschwindigkeit; Compiler) haben, eine deklarative- und übersichtliche-, gut strukturierte Darstellung gewährleisten und vor allem Menschen mit unterschiedlichen informationstechnischen Vorkenntnissen einen natürlichen Zugang gewähren.

Objekt- und Eventmanagementtechnologien ermöglichen LabVIEW den Sprung in die Liga der Universalprogrammiersprachen ohne Einschränkungen.

Die Objekttechnologietoolsets für LabVIEW (OTT) der Firma Vogel (www.lvot.de; www.gooptech.com) erweitern LabVIEW um Ojektorientierung mit Mehrfachvererbung, Eventmanagement und unterschiedliche zusätzliche Abstrahierungsmechanismen wie Objekt- und Petrinetze. Alle wichtigen Kommunikationsmechanismen und -protokolle (TCP/IP, ActiveX, OPC, Notifications, Queues, XML, DataSocket) sind unterstützt. Die Toolsets sind ideal zur Entwicklung verteilter intelligenter Systeme (unter anderem gemäss IEC 61499).

Verschiedene Designmethoden sind gleichzeitig verwendbar. Die Formulierung des Problems ist nahezu identisch mit der Codierung und Dokumentation. Anders als bei anderen Entwicklungssystemen sind die Designansätze durchgängig (bei herkömmlichen UML-Ansätzen wird in der Regel nur ein Template erzeugt, das dann als Basis für die weitere Codierung benutzbar ist). Durch diese zusätzlichen Möglichkeiten gibt es keine Einschränkungen mehr in Bezug auf Performance und Skalierbarkeit. Die Erstellung von Klassenbibliotheken mit intelligenten verteilten Objekten ist eine Domäne der Toolsets. OTT-Technologie wird in vier unterschiedlichen Varianten angeboten (OET Object Event Toolset, OIT Object Inheritance Toolset, ONT Object Network Tooset, AFT Application Framework Toolset), wobei die ersten beiden bereits verfügbar sind. LabVIEW-Entwickler sind jetzt in der Lage in bisher völlig neue Terrains der Informationstechnologie vorzustossen. Neben der Automatisierungs- und Messtechnik (mit zehntausenden von I/O-Punkten) werden völlig neue Applikationen in den unterschiedlichsten Bereichen denkbar. Die Kombination von Grafischer Datenflussprogrammierung, Objektorientierung mit Mehrfachvererbung und der zusätzlichen Verwendung von Petrinetzen führt zu einer bisher nicht dagewesenen Freiheit in der Realisierung auch extrem komplexer Systeme. Natürlich profitieren auch kleine Projekte von den Möglichkeiten der Objektorientierung in LabVIEW. Endlich ist es möglich, langfristig wartbare und hochperformante Software (völlig ereignisorientiert) in noch kürzerer Zeit und damit zu noch geringeren Kosten dem Kunden an die Hand zu geben. Auch Entwickler, die bisher LabVIEW aufgrund eingeschränkter Objektorientierung gemieden haben, können jetzt auf auf eine universell einsetzbare Entwicklungsplattform setzen, die ein Optimum an Rapid Prototyping- und Rapid Application Development-Optionen bietet und dazu noch echt multiplattformfähig- und auf verschiedenen Plattformen echtzeitfähig ist.

Referenzen

[1] www.gooptech.com

[2] www.labview-objekt-technologie.de

[3] www.vat.de

[4] www.systec-gmbh.com

[5] www.ni.com

[6] Herbert Pichlik, Jens Vogel, Object Technology Toolsets for LabVIEW - OTT, NI-Week 2001 Proceedings, Austin TX 8/2001

[7] Jens Vogel, Automating Shipping in a Cement Works Using LabVIEW, NI-Week 2001 Proceedings, Austin TX 8/01

[8] Herbert Pichlik, Universal Test- and Automation System using an event driven object oriented LabVIEW approach (UTAS), NI-Week 2001 Proceedings, Austin TX 8/01

[9] OTT User Manual, Vogel/SYSTEC, Dornburg/Nürnberg, 6/01

[10] Rahman Jamal, Herbert Pichlik, „LabVIEW Applications and Solutions“, Prentice Hall 8/98

[11] Herbert Pichlik, „Universal testing of household equipment using LabVIEW“, Best applications of virtual instrumentation, NI-Week 96 Proceedings, Austin TX 8/96

[12] Herbert Pichlik, „Networkcentric test and measurement system“, Best applications of virtual instrumentation, NI-Week 97 Proceedings, Austin TX 8/97

[13] Herbert Pichlik, „Bike Ergometer Tester BET“, Best applications of virtual instrumentation, NI-Week 98 Proceedings, Austin TX 8/98

[14] Herbert Pichlik, „LabVIEW and Transputers“, Proceeding NI User Symposium Munich 11/94