

LabView

Die erste komplett grafische
Programmierungsumgebung

Falko Buttler (fbuttler@tzi.de)

Jens Kleinwechter (jenskl@tzi.de)

Gliederung des Referats

- Was versteht man unter LabView?
- Für wen war es gedacht vs. wofür wird es eingesetzt?
- Wie erstellt man ein „Programm“?
- Spracheigenschaften und -besonderheiten
- Vor- und Nachteile gegenüber „herkömmlichen“ Programmiersprachen
- Zusammenfassung

Erste Eckdaten

- **Laboratory Virtual Instrument Engineering Workbench = LabView**
- Hersteller: National Instruments
- <http://www.ni.com/labview/>
- erhältlich seit 1986 für Macintosh Plus mit 1 MB Hauptspeicher
- erste komplett grafische Programmierumgebung
- Version 2.0 1990 mit integriertem Compiler → so schnell, dass Kompilieren kaum gemerkt wurde

Weiterer Lebenslauf

- Vorerst nur für UNIX-Systeme mit 16 oder 32 bit
- Erst ab Windows 3.0 als Wintel-Version und mit Windows 95 erste 32 bit Anwendung
- So zum Industriestandard in der Mess- und Testtechnik und Prozesssteuerung
- Erfolgreichste, *vermarktete* Datenfluss-Programmiersprache

Hauptaufgaben und Umfang

- Visualisierung von Messgeräten
- Simulation von Hardwarefunktionen
- Datenauswertung
- Realisierung von Realzeitsystemen

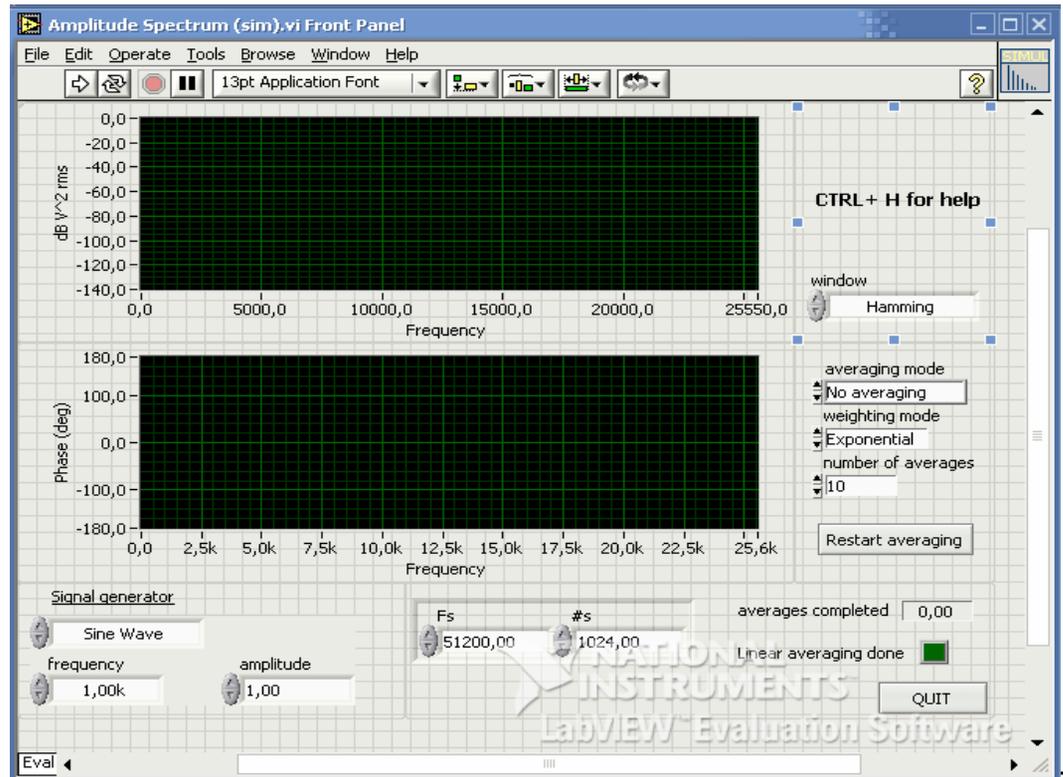
- für viele Plattformen MacOS, Linux, Windows,...
- Vielfältige Funktionsbibliotheken von NI und Fremdherstellern verfügbar
- Schnittstellen für gängige Hardware integriert

Zielgruppe von LabView

- Zielgruppe von NI waren
 - Basic-Programmierer und **nicht** Programmierfremde
 - Ingenieure und Wissenschaftler in der Industrie
- Überwiegend von Nicht-Programmierern für hauseigene Projekte benutzt und positiv aufgenommen

Idee und Struktur

- Konzept ist die Hierarchie von "virtuellen Instrumenten" (VIs)
- Benutzungsschnittstelle **und** Programmlogik aus graphischen Bausteinen
- Connector-Schnittstelle für andere VIs

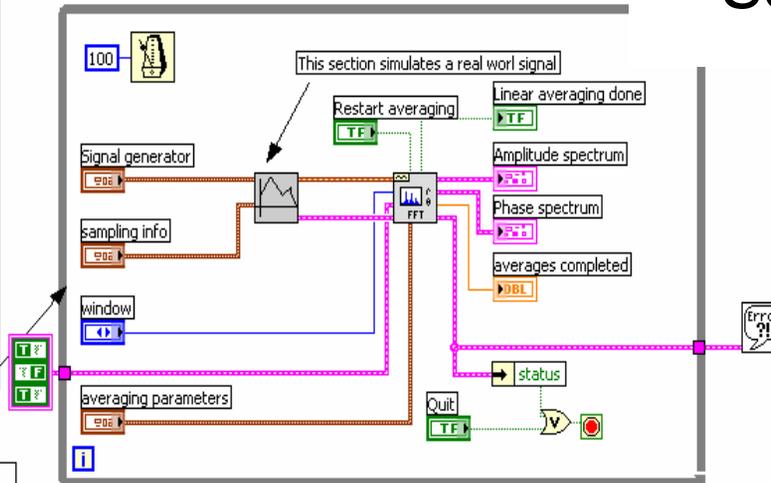
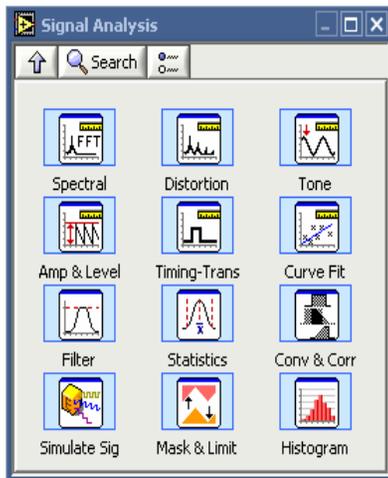


Handhabung

- Frontpanel → eigenes Fenster mit Diagramm
- Diagramm → verbundene Icons
- an Sub-VI können Parameter vererbt werden
- Nutzbar als numerische Kontrolleinheit und über Indikatoren

Programmlogik

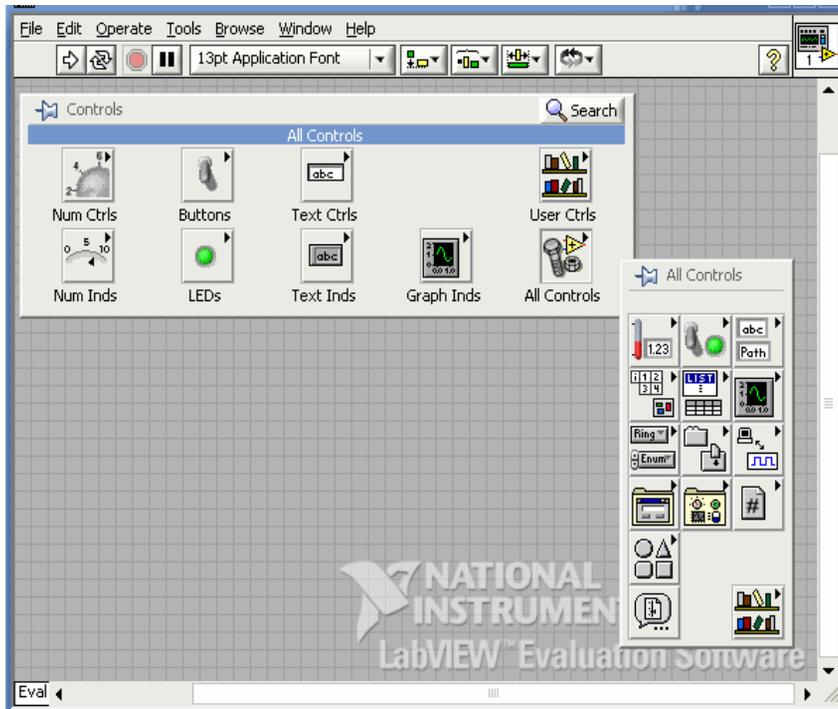
- Graphische Blockschaltbilder
- ähnlich wie elektronische Schaltungen



Default values are
Fs = 51.2 kS/s
of samples = 1024 pts
The record length will be :
 $1024/51200 = 0.02$ seconds = 20 ms
Note : The longer the record length (# of samples)
the better the FFT resolution

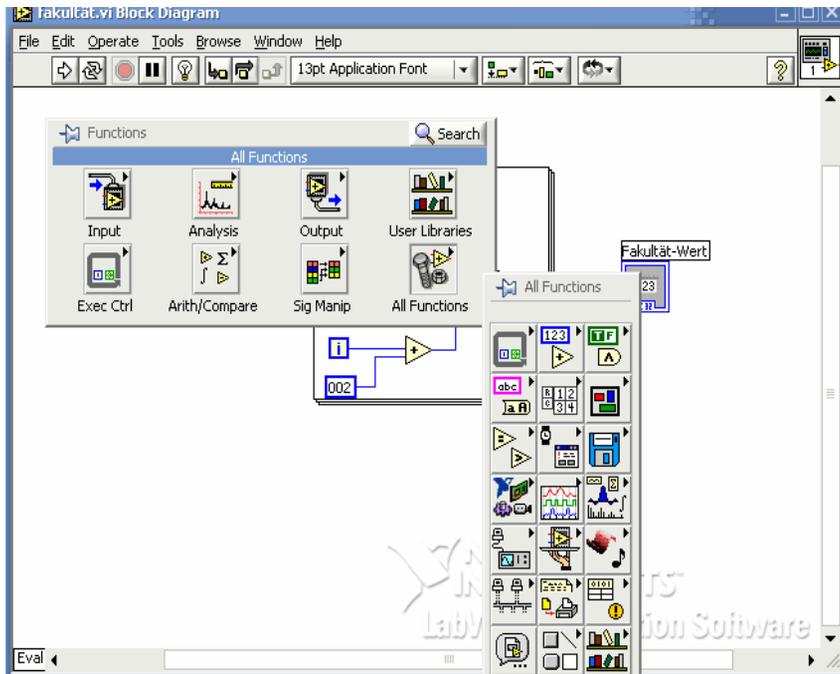
Computes FFT and returns Magnitude
and Phase components
Notes about using averaging :
Linear averaging will stop when the # of records
to average will be reached
You can press the "restart averaging" button

Funktionsvielfalt des Front Panels



- Viele vordefinierte Elemente vorhanden
- Durch benutzerdefinierte Controls erweiterbar
- Regler
- Buttons
- Funktionsgraphen (!)
- LEDs
- Text
- etc.

Funktionsvielfalt des Blockdiagramm



- Strukturen von herkömmlichen Programmiersprachen als Symbole
- Schleifen
- Datenstrukturen
- Variablen
- Mathematische Funktionen
- Inputs und Outputs
- Event Handler
- etc.

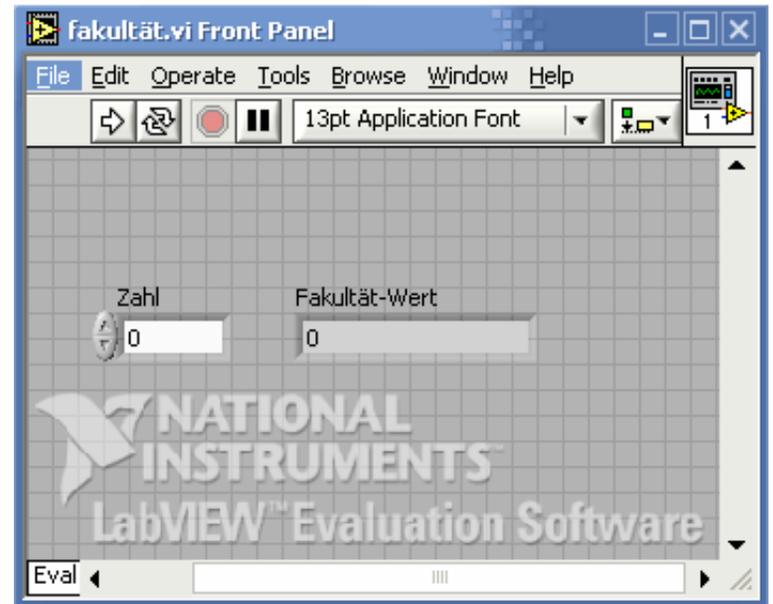
Der Programmierer

- Arbeitet im Diagramm-Fenster
- Nutzt vorgefertigte Icons, Funktionen
- Einfacher Arbeitsfluss
- Eigene Funktionen, Prozeduren sind LabView artfremd
- Eigene, neue Funktionen zu erstellen, setzt Programmierkenntnisse voraus

Der Weg zum Programm

Erster Schritt - Die Oberfläche

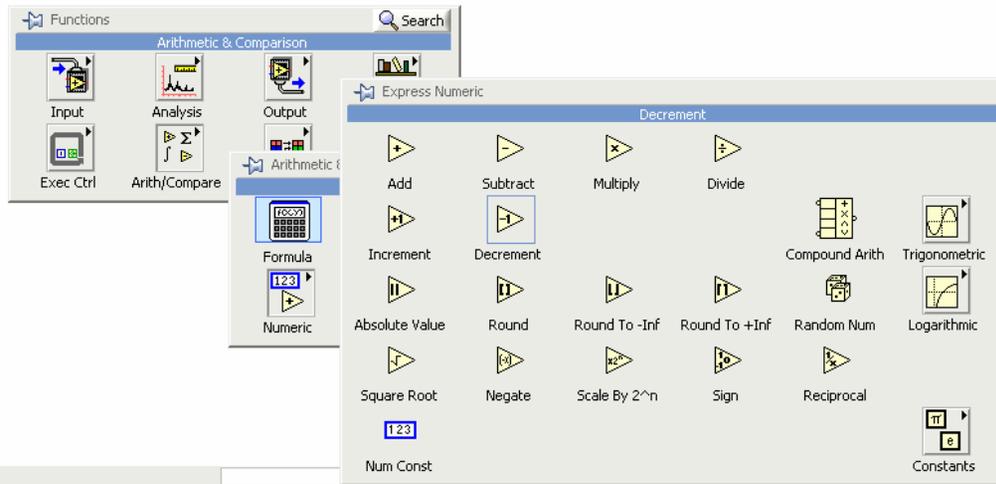
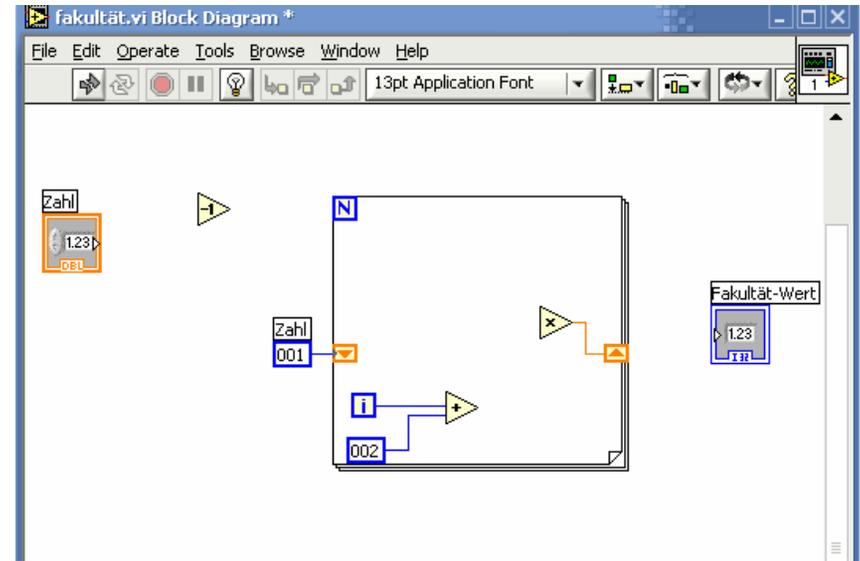
- Graphisch sichtbare Elemente auf dem Frontpanel
 - Organisieren
 - Einstellen
- ➔ keine Funktionalität



Der Weg zum Programm

Zweiter Schritt – Der Algorithmus

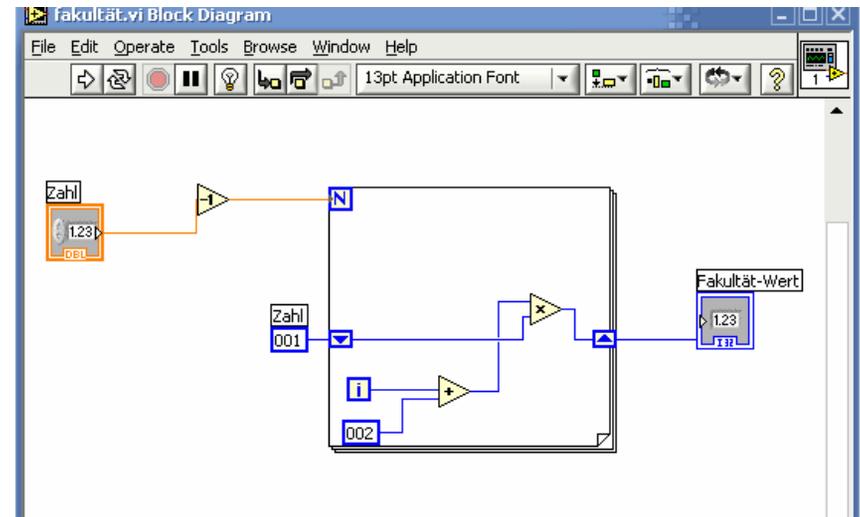
- „Algorithmus“ im Blockdiagramm implementieren
- Aus Control-Leiste auswählen und konfigurieren



Der Weg zum Programm

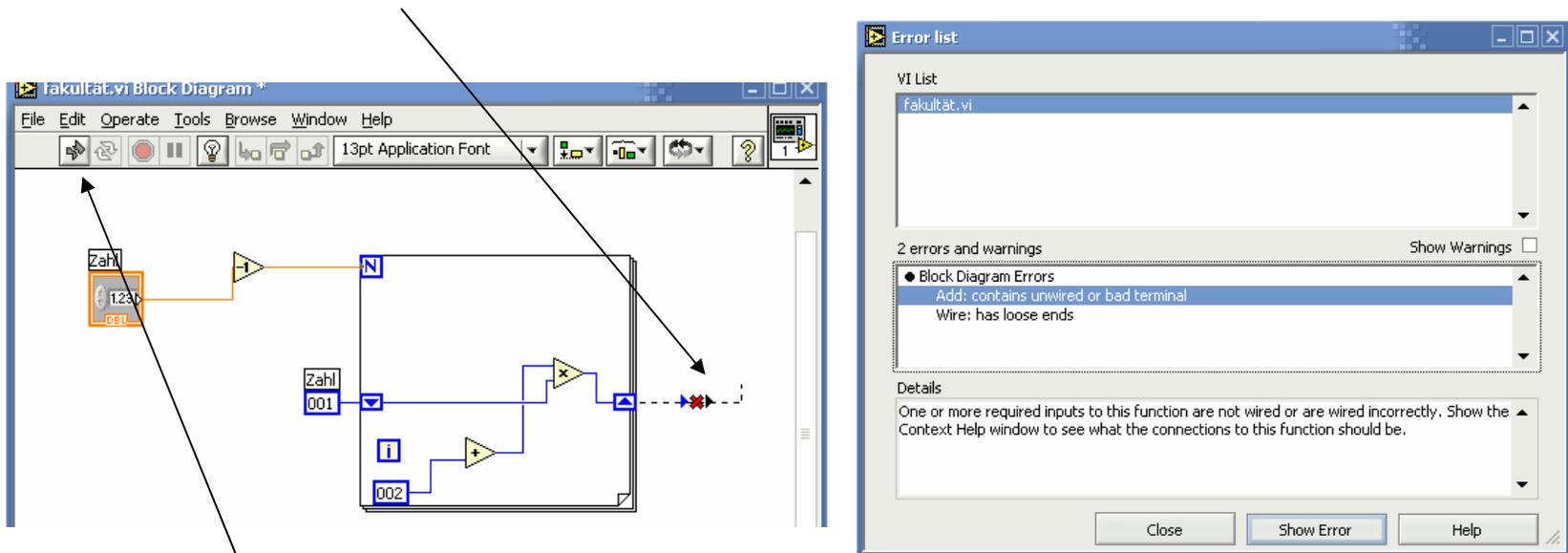
Letzter Schritt – Die Verknüpfung

- Elemente miteinander verknüpfen und schließlich Ein- und Ausgabe an graphische Elemente zuweisen
- grafische Programmiersprache "G"



Fehlersuche und Start

- Fehler in Programm werden durch rote Linien und Kreuze dargestellt inklusive Beschreibung



The image shows two windows from the LabVIEW software. On the left is the 'Block Diagram' window for a VI named 'fakultät.vi'. It displays a block diagram with several components: a numeric control labeled 'Zahl' with the value '123', a numeric indicator labeled 'Zahl' with the value '001', and another numeric indicator labeled 'Zahl' with the value '002'. There are also several mathematical function blocks (addition, multiplication, division) and a loop structure. A red 'X' symbol is visible on a wire, indicating an error. On the right is the 'Error list' window, which displays the following information:

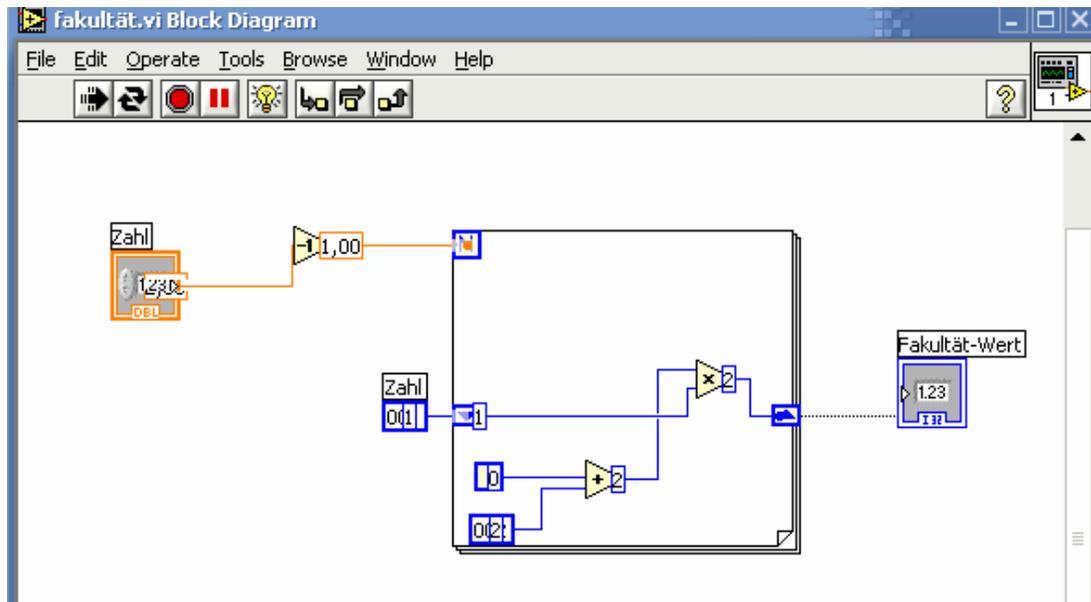
- VI List: fakultät.vi
- 2 errors and warnings
- Block Diagram Errors
 - Add: contains unwired or bad terminal
 - Wire: has loose ends
- Details: One or more required inputs to this function are not wired or are wired incorrectly. Show the Context Help window to see what the connections to this function should be.

Buttons at the bottom of the Error list window include 'Close', 'Show Error', and 'Help'. Arrows from the text above point to the error symbol in the block diagram and the 'Show Error' button in the error list window.

- Ohne Kompilierung startfähig per Button

Debugging

- Debugging einfach durch Animation des Blockdiagramms mit Anzeige aller Variablen → einfache Fehlersuche



Spracheigenschaften I

- Syntax als Blockdiagramm
- Semantik in den Symbolen
- Paradigma der Datenflussdiagramme →
Graphen als formale Grundlage
- Sprache ist
 - universell und implementierbar
 - nicht unbedingt „natürlich“

Spracheigenschaften II

- Keine Polymorphie, Überladung
- Vererbung begrenzt
- Funktionen als eigene Vis
- Nebenläufigkeit explizit möglich
- Rein visuell, keine textuelle Eingabe
- Konsistenz jederzeit erkennbar

Besonderheiten von LabView

- Anpassung unterschiedlicher Datentypen durch automatische Konvertierung
- Einfache Erstellung von parallel laufenden Schleifen mit unterschiedlichen Geschwindigkeiten → Variablen
- automatisches Indizieren und Aufbauen von Matrizen

Vorteile gegenüber textuellen Sprachen

- Darstellung ist Ingenieuren aus Elektrotechnik etc. vertraut
- Einfache Konstruktion einer Benutzerschnittstelle
- Einfaches Debugging, da jedes VI allein testbar
- Geringere Fehleranfälligkeit
- Aufbau komplexer Systeme durch Hierarchiekonzept
- Vereinigung von Entwicklung und Ausführung
- Selbstdokumentierte Programme
- Sehr gute Dokumentation

→ **Zeitaufwand** eines Projekts **geringer**

Nachteile von LabView

- Rekursion und Umsetzung von textuellen Programmen schwierig
- Benutzung ungewöhnlich und „hakelig“
- Nicht kompletter Sprachumfang möglich

Zusammenfassung

- Gut geeignet für
 - Ansteuerung von Instrumenten
 - Programmierneulinge
- Nicht geeignet für
 - Algorithmuslastige Programme
 - „Hacker“

Literatur

- Margaret Burnett; Adele Goldberg; Ted Lewis. „Visual object-oriented programming: concepts and environments“. Manning Publications Co. 1995.
- Urs Lauterburg. „LabVIEW eine grafische Programmiersprache geeignet für den Unterricht“. Physikalisches Institut der Universität Bern. Juli 1998
- National Instruments. „LabView Benutzerhandbuch“. National Instruments. Januar 2002.
- Stefan Schiffer. „Visuelle Programmierung: Grundlagen und Einsatzmöglichkeiten“. Addison-Wesley-Longman. 1997
- Steen Vogelreuter; Oliver A. Braun. „LabVIEW Graphische Programmierung für die Instrumentierung“. Institut für Softwaretechnologie Fakultät für Informatik Universität der Bundeswehr München. März 2000