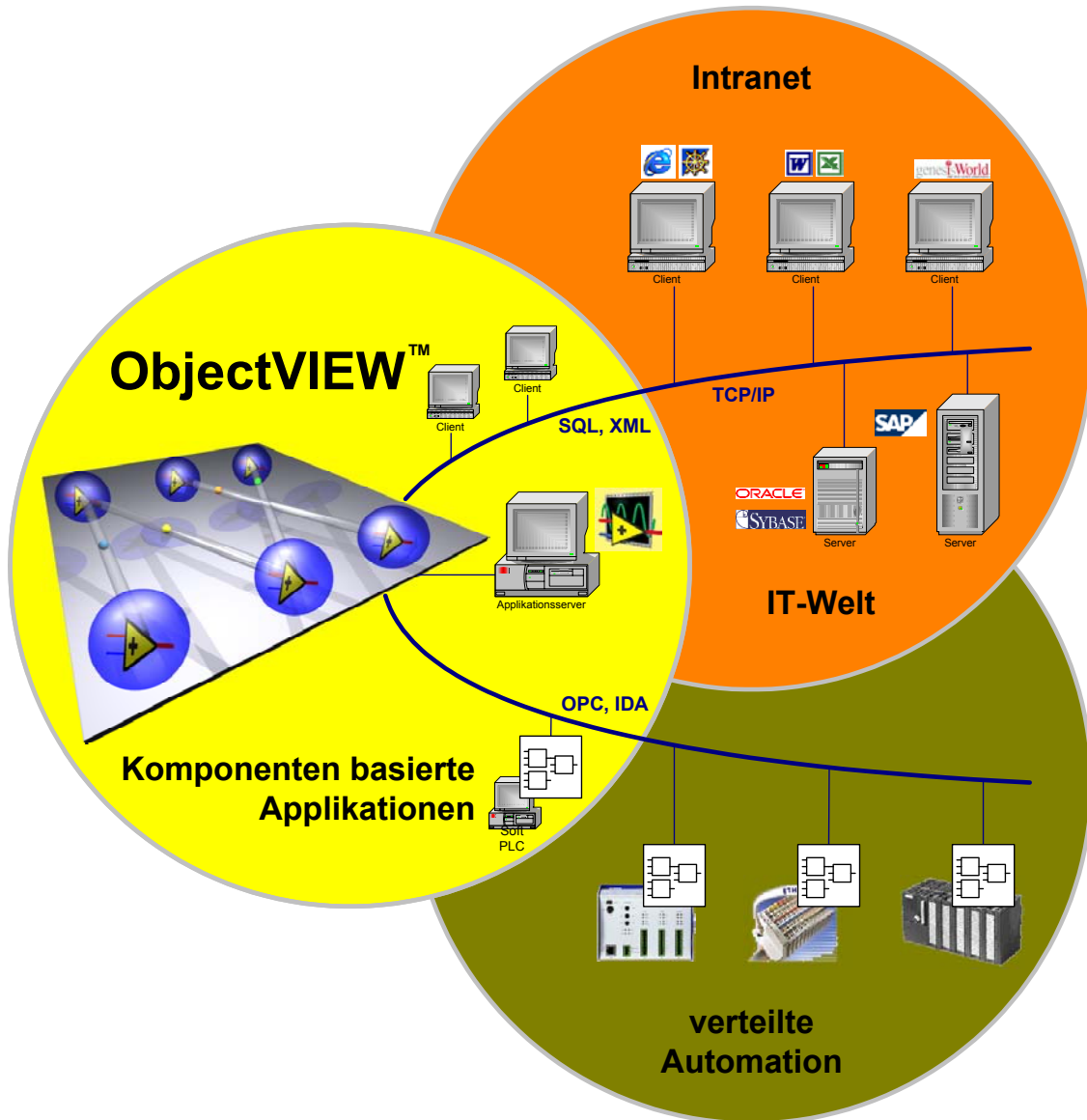


G++, Graphical Object Technology mit ObjectVIEW™



Einführung in Objekt-Netze

Vogel Automatisierungstechnik GmbH, Jenaer Straße 7 D-07778 Dornburg
Tel: 036427 20030, Fax: 036427 20031 info: www.vat.de

Autor: Dipl.-Ing. Jens Vogel (info@vat.de)

Erzeugung von modularen, hierarchisch strukturierten LabVIEW™-Programmen mit ObjectVIEW™

Die Softwareentwicklung mit grafischen verteilten intelligenten Objekten wird ausgehend vom Wasserfallmodell mit Planungs-, Definitions-, Entwurfs- und Implementationsphase einem Beispiel erläutert.

Verteilte Intelligente Objekte

Technische Prozesse die durch Mess- und Automatisierungstechnik zu erfassen, zu steuern und zu regeln sind, sind meist durch hierarchische Teilsysteme strukturierbar. Zum Beispiel Anlage - Teilanlage - Baugruppe - Gerät - Sensor. Die Abbildung dieser Strukturen in der Software ermöglicht eine optimale Bearbeitung der Teilaufgaben bezogen auf zeitliche, räumliche und funktionale Anforderungen. Dabei wird die Bearbeitung jedes Teilprozesses des technischen Systems durch je eine Softwarekomponente vorgenommen. Damit ist eine einfache Skalierung der Systeme möglich. Neue Teilprozesse erfordern nur noch neue Instanzen der jeweiligen Softwarekomponente.

Diese Vorgehensweise erschließt ein gewaltiges Rationalisierungspotential.

Komplexe Systeme können aus vergleichbar einfachen Komponenten zusammengesetzt werden. Diese Komponenten führen die Teilfunktionen selbständig aus und kommunizieren selbständig ereignisgesteuert miteinander. Damit kann die Flexibilität gesteigert und die Gesamtkosten können wesentlich reduziert werden. Die Umsetzung dieser Konzepte für verteilte Systeme führten in der Automatisierungstechnik zu den Standardisierungsbestrebungen „IDA - Interface for distributed Automation“ PROFINet und IEC 1499. In der Informationstechnologie sind gleiche Konzepte mit DCOM, CORBA JAVA/RMI bereits umgesetzt wurden.

Die Graphical Object Technology mit ObjectVIEW™ ist die direkte Umsetzung der Komponenten basierenden Konzepte in die grafische Programmierung mit LabVIEW™. Das ermöglicht die Erzeugung von Programmen bei dem Entwurf und Implementierung gleiche Strukturen aufweisen. Die Komponenten lassen sich wie im Baukasten zusammenschalten. LabVIEW™ wird um ein höheres Abstraktionsniveau die Komponententechnologie erweitert. Komponenten können aus einer Vielzahl von Objekten bestehen. Objektnetze (UML-RT, ROOM) ermöglichen eine anschauliche grafische Beschreibung des Aufbaus der Komponenten. Sie dienen dem Entwurf von verteilten Systemen hoher Komplexität und gleichzeitig zur Dokumentation der daraus entwickelten Software.

Mit ObjectVIEW™ ist es möglich grafische verteilte intelligente Objekte in Objektnetzen zusammenzufassen und deren Verschaltung grafisch im Blockdiagramm zu realisieren.

Dieses Verfahren soll an einer einfachen Aufgabe demonstriert werden.

Das Nanolaserskapel zeigt eine reale Anwendung von ObjectVIEW™ die seit Oktober/2001 in Betrieb ist.

Die Aufgabe

In einer Etage eines Bürogebäudes befinden sich mehrere Räume. Jeder Raum besitzt einen Heizkörper. An ihm befindet sich ein Ventil, das über Pulsweitenmodulation angesteuert wird. Es gibt einen Temperatursensor und einen Fensterkontakt. Mit Hilfe dieser Sensoren und eines Temperatursollwertes soll eine Regelung das Ventil steuern und damit die Temperatur im Raum regeln. Die Regelung lässt sich in verschiedenen Betriebsarten betreiben (Tag, Nacht, usw.). Außerdem besitzt jeder Raum mehrere Lampen, denen jeweils ein Taster zugeordnet ist.

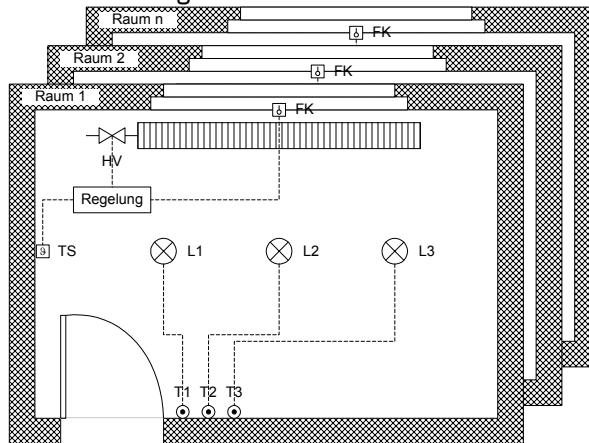


Bild 1: Die Aufgabe

Planungsphase

Die Applikation wird in einer Schichtenstruktur realisiert. Diese besteht aus User Interface, Application Layer und Hardware Layer. Das Objektnetz befindet sich im Application Layer. Die Komponenten der Applikation, die sich im Hardware Layer und im User Interface befinden, werden in diesem Beispiel nicht betrachtet.

Aus der Analyse des realen Systems ergibt sich eine hierarchische Struktur seiner Komponenten. Aus dieser hierarchischen Struktur kann ein hierarchisches Objektnetz entwickelt werden. Der Datenaustausch mit dem Hardware Layer und dem User Interface erfolgt wie innerhalb des Objektnetzes über asynchrone Nachrichten.

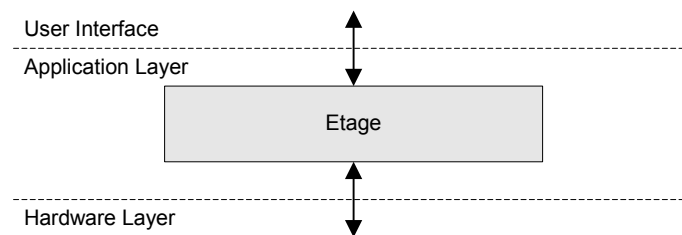


Bild 2: Schichtenstruktur

Die oberste Hierarchieebene bildet die Etage. Die Etage beinhaltet eine Anzahl von Räumen.

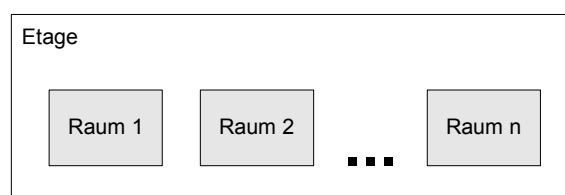


Bild 3: Etage

Durch Betrachtung der Komponenten eines Raumes entsteht die nächste Hierarchieebene. In jedem Raum gibt es eine Heizungsregelung und mehrere Lampen.

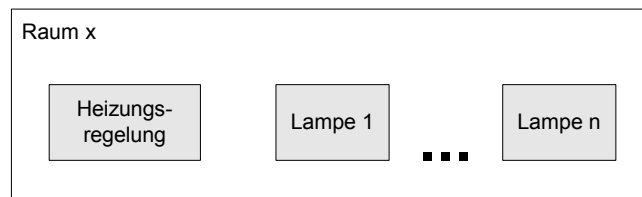


Bild 4: Raum

Die Komponenten der Heizungsregelung gehören zur untersten Hierarchieebene des betrachteten realen Systems. Weil alle realen Objekte auf Software-Objekte abgebildet werden, enthält die Heizungsregelung außer Hardware- auch Software-Komponenten. Die Sollwertvorgabe, der Regler und der Pulsbreitenmodulator sind Software-Komponenten, die für die Funktion des Systems notwendig sind.

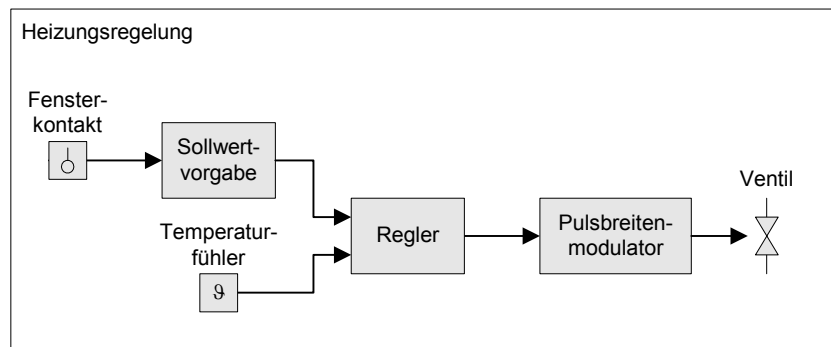


Bild5: Heizungsregelung

Somit ist die folgende Baumstruktur entstanden, in der die Hierarchieebenen deutlich erkennbar sind. Komponenten, die aufgrund ihrer Funktion eine Einheit bilden, sind zu Gruppen zusammengefasst und bilden so Teilsysteme des Gesamtsystems.

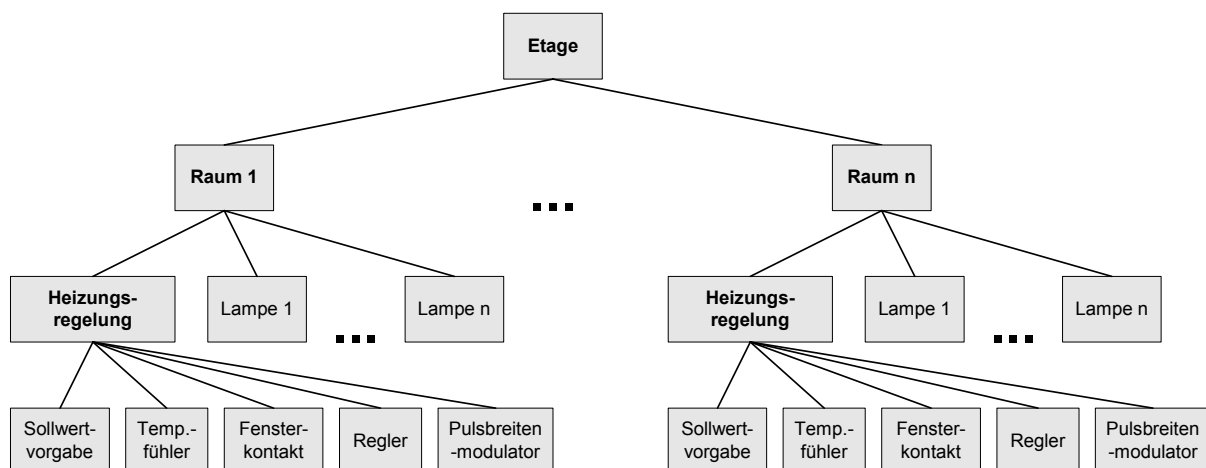


Bild 6: Baumstruktur

An dieser Stelle sei darauf hingewiesen, daß das reale System bewusst in kleine Funktionseinheiten gegliedert wurde, um trotz der Einfachheit des Beispiels eine Struktur mit mehreren Hierarchieebenen zu erhalten.

Entwurf

Der nächste Schritt ist die Transformation der Baumstruktur in ein Objektnetz. Alle Komponenten in der Baumstruktur werden zu Netzobjekten, also zu unabhängigen aktiven Objekten, die untereinander Daten über Verbindungen austauschen. Generell gilt, daß unabhängige aktive Objekte genau dann gebildet werden sollten, wenn die Unabhängigkeit von Funktionseinheiten von ihrer Umgebung gefordert ist. Das entstandene Objektnetz beschreibt genau das reale System, indem es die Blockschemata der Komponenten instanziiert und diese Instanzen hierarchisch anordnet. Die Hierarchieebenen sind mit denen der Baumstruktur identisch. Zur Laufzeit entstehenden Objekte, mit zusammengesetzten Namen z.B. "Floor1.Room1.HR", "Floor1.Room2.HR.WS", usw. .

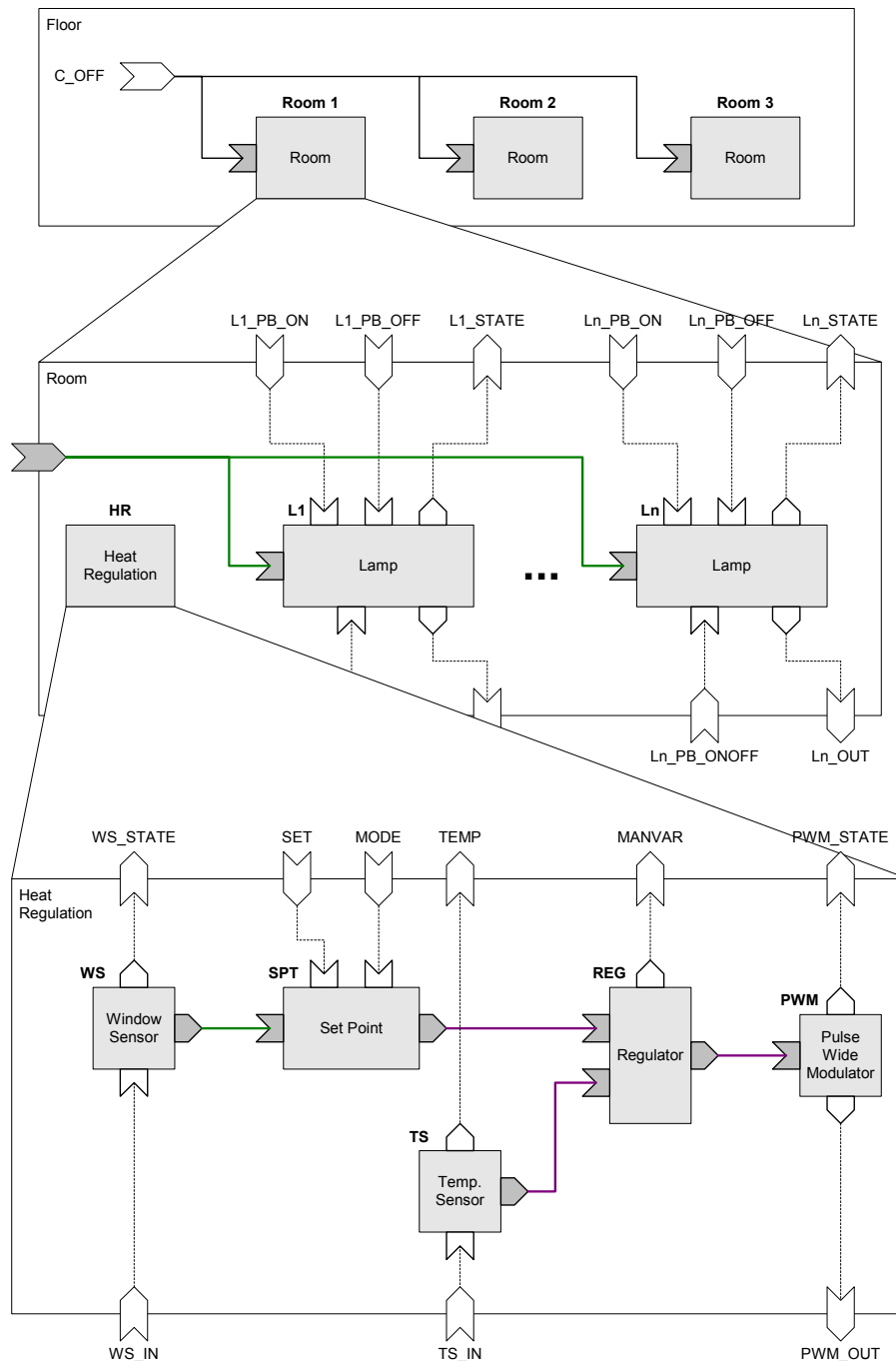


Bild 7: Entwurf

Implementierung

Das sind die drei Objektnetze, die in die Diagramme der hierarchischen Netzobjekte für die Etage, den Raum und die Heizungsregelung gezeichnet werden müssen. Die in den Diagrammen nicht vorhandenen Verbindungen in den Hardware Layer und zum User Interface werden aufgrund des Designs der elementaren Netzobjekte automatisch hergestellt.

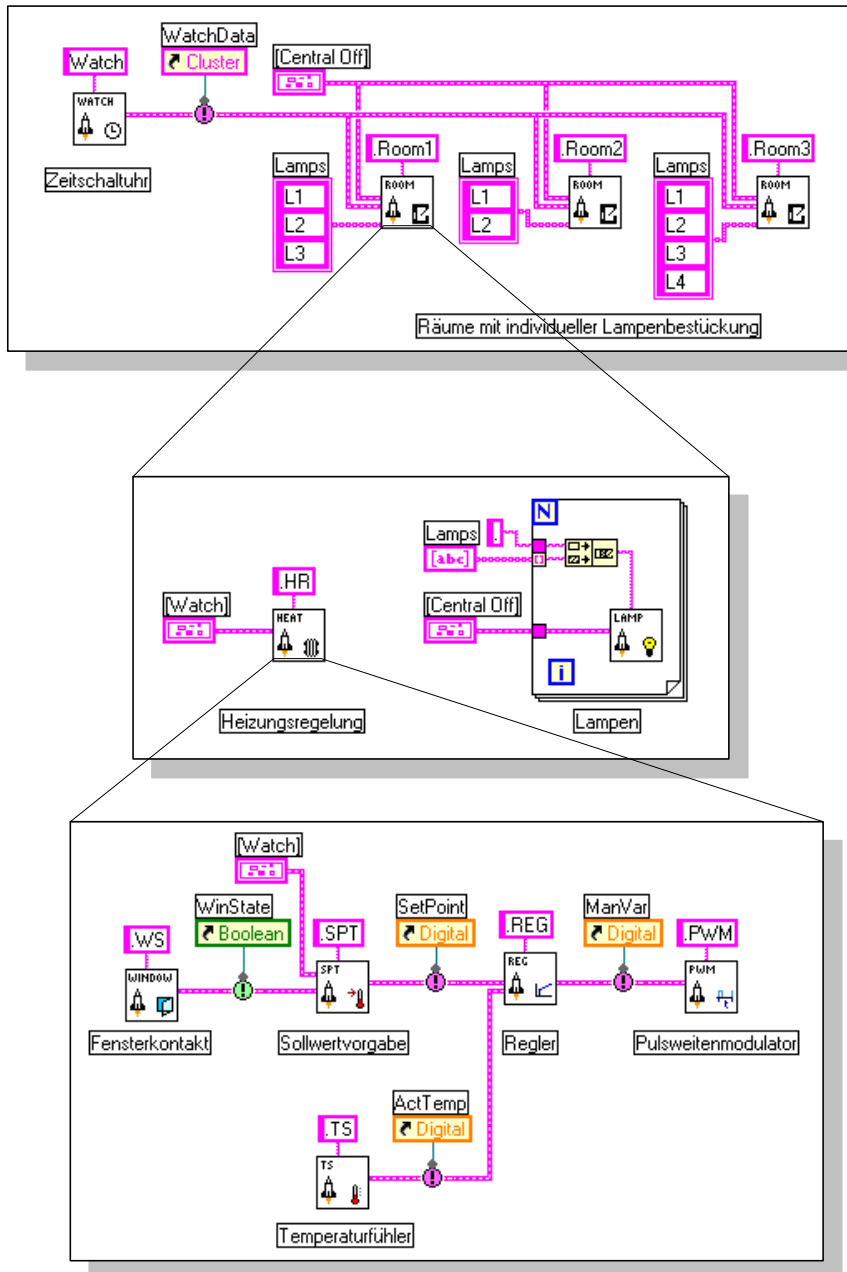


Bild 8: Die Objektnetze – Floor, Room und Heat Regulator

Im Applikation-Layer werden beim Start durch diese Objektnetze 31 unabhängige aktive Objekte dynamisch erzeugt. Diese kommunizieren entsprechend der grafischen Verdrahtung ereignisgesteuert. Die Namen der aktiven Objekte und der Connector-Objekte werden automatisch entsprechend der Baumstruktur gebildet Floor, Floor.Room1, Floor.Room1.HR ... Floor.Room1.L3, Floor.Room1, Die Verarbeitungsfunktion aller Objekte ist vollständig gekapselt und voneinander vollständig unabhängig. Durch die Ereignissteuerung ist es, bei sehr kurzen Antwortzeiten, möglich die Systemauslastung auch bei 31 parallelen Prozessen sehr gering zu halten.

Nanolaserscalpell – Eine reale Anwendung

Hochpräzise Operationen in lebenden Zellen. Umliegendes Gewebe wird nicht geschädigt. Gezielte Schnitte an Chromosomen oder Mitochondrien.

Anwendungsgebiete sind:

- Tumor- und Neurochirurgie
- Gentherapie
- Gendiagnostik
- Entwicklungsbiologie

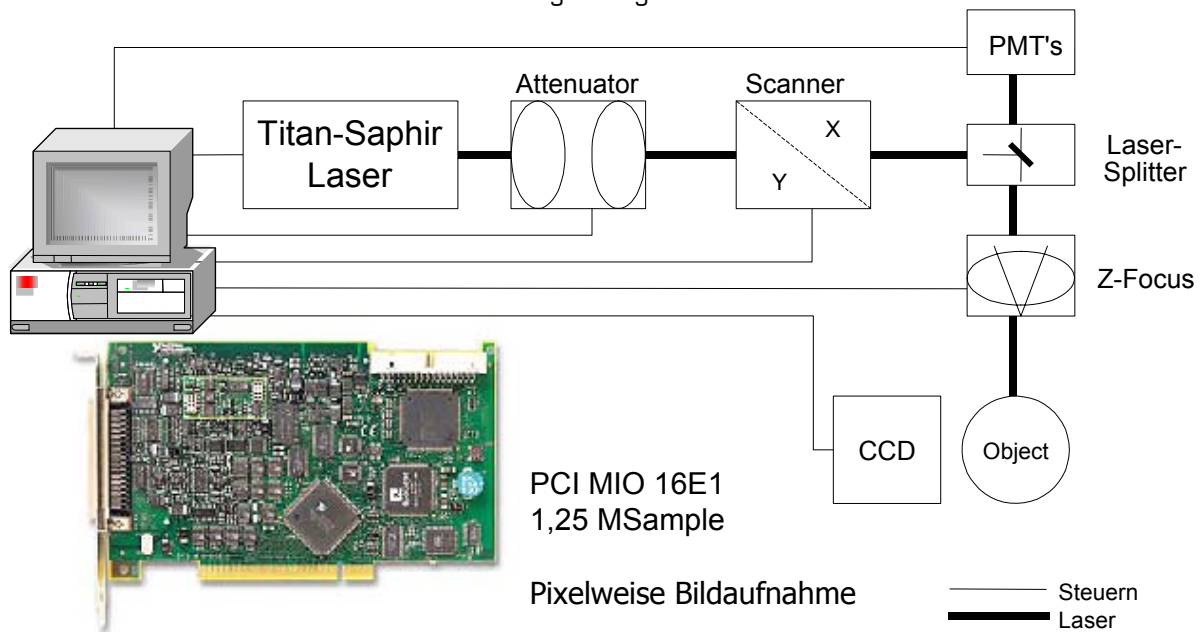


Bild 9: Struktur Nanolaserscalpell

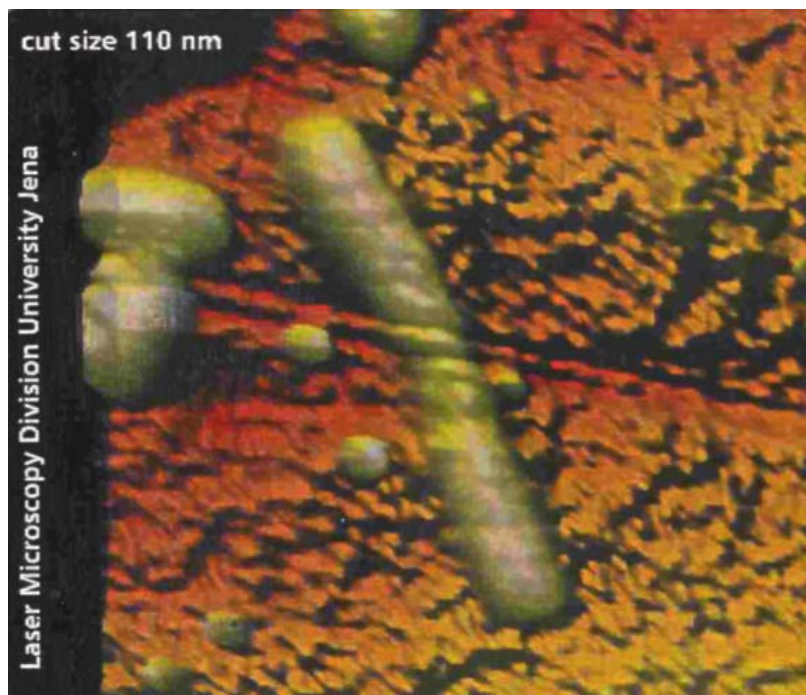


Bild 10: Chromosomen mit 110 nm Schnitt

Die Anwendung von ObjectVIEW™ sparte 50% der Entwicklungszeit.

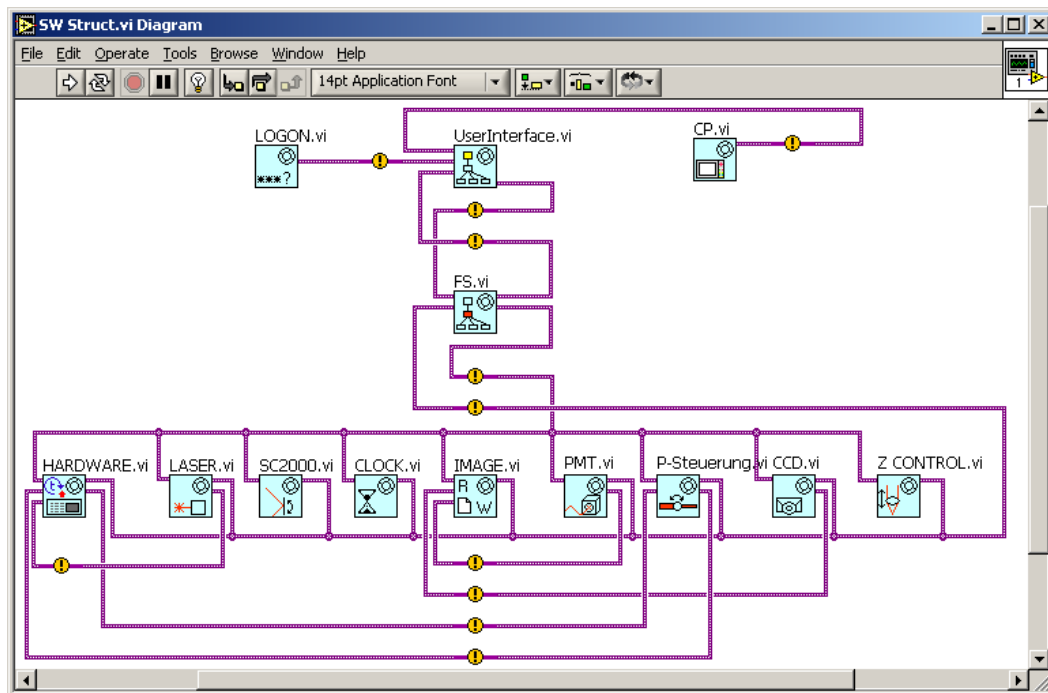


Bild 11: Objekt-Netz Nanolaserskalpel mit aktiven Objekten der Teilkomponenten

Zusammenfassung

Mit ObjectVIEW™ für LabVIEW™ ist es möglich, den ereignisgesteuerten Datenaustausch von aktiven Objekten (Instanzen) untereinander durch grafische Verschaltung zu programmieren. Bei der Erzeugung/Vernichtung eines aktiven Objektes das ein Objekt-Netz enthält werden automatisch alle Unterobjekte erzeugt/vernichtet. Das Object Net Toolset stellt Basisklassen zur Verfügung, von denen der Anwender einfach seine Klassen ableiten und spezialisieren kann. Dies erfolgt mit Hilfe der Werkzeuge des Class Inheritance Toolset von ObjectVIEW™.

ObjectVIEW™ vereinfacht und beschleunigt die Software-Entwicklung von komplexen Anwendungen enorm. Außerdem dokumentiert sich die Anwendung selbst. Änderungen und Erweiterungen der Anwendung sind leicht möglich.

ObjectVIEW™ ermöglicht eine grafische komponentenbasierte Softwareentwicklung, die auf den Entwurfprinzipien Grafischer Datenfluss, Ereignisfluss, und Zustandsfluss basiert. Der Entwurf erfolgt einheitlich in der vollgrafischen Entwicklungsumgebung von LabVIEW™. Grundlage 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 URLs. Damit sind intranetbasierte verteilte intelligente Systeme einfach realisierbar. Die Kombination LabVIEW™ und ObjectVIEW™ ist nicht nur Entwicklungsbasis für einfach skalierbare und sehr gut wartbare Applikationen, die Rapid Prototyping und Rapid Application Development ermöglichen, sondern auch ideal als Basis für die Entwicklung und Umsetzung eigener neuartiger Programmierkonzepte. Eine ausführliche Fassung dieses Beitrages mit Quellenverzeichnis kann unter www.ObjectVIEW.de abgerufen werden.