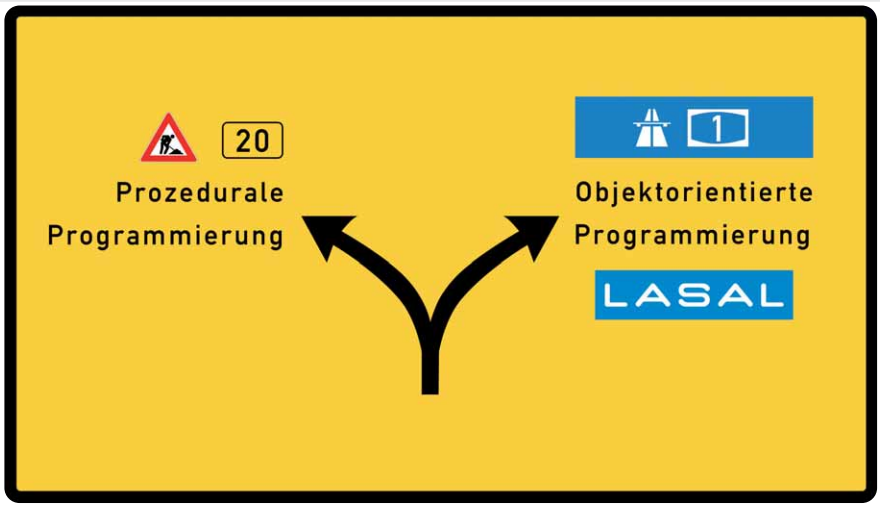

Bild 1: Nach Ansicht der Firma Sigmatek liegt die Zukunft der Programmierung von Steuerungen in der Objektorientierung.

Natürlich objektorientiert

Die Programmierung von Steuerungen ist einer der größten Kostenfaktoren im Maschinenbau. Dies ist einer der Gründe, warum das SPS-Magazin die Tools und Programmiersprachen verschiedenster Steuerungshersteller unter die Lupe nimmt. In diesem Beitrag geht es um den größten Verfechter der objektorientierten Programmierung in der Automatisierungstechnik, den österreichischen Steuerungshersteller Sigmatek. Das Unternehmen setzt seit mehr als zehn Jahren auf das objektorientierte Programmieren und verfügt dementsprechend über einen großen Erfahrungsschatz.

Noch immer ist es so, dass das objektorientierte Programmieren (OOP) im Bereich der Automatisierungstechnik eher ein Schattendasein führt. Es zeichnet sich jedoch ab, dass sich dies ändern wird. Der Automatisierungsexperte Sigmatek setzt bereits seit 1999 auf diese Technologie. Das Unternehmen ist damit nach eigenen Angaben der erste Systemanbieter der Steuerungstechnik am Markt, der OOP eingeführt hatte. Das Engineeringtool bei Sigmatek heißt Lasal und beinhaltet folgende Funktionalitäten:

- Objektorientiertes Programmieren mit grafischer Darstellung und Client-Server-Kommunikation
- Visualisierung
- Motion Control
- Standard- und benutzerdefinierte Bibliotheken
- Programmierung der logischen Verknüpfungen der Safety-Komponenten
- Funktionen für Service und Wartung

Nicht jede dieser Funktionalitäten ist unter einer Oberfläche zu finden. Allerdings gibt es jeweils logische Verbindungen im Hintergrund, die dem Anwender den Eindruck eines durchgängigen Toolsets vermitteln. Lasal unterstützt alle Produkte von Sigmatek wie CPUs, Terminals und Industrie-PCs. Auch ein Wechsel von einer Plattform zur nächsten ist möglich, ohne dass das Programm neu geschrieben werden muss. Es muss lediglich die Onlineverbindung eingestellt werden. Lasal ermöglicht das Erstellen von USB Bootsticks, um Steuerungen ohne die Entwicklungsumgebung updaten zu können.

Die Programmieroberfläche

Die Programmieroberfläche von Lasal setzt sich aus den Elementen Menüleiste, Werkzeug-Symbolleiste, Meldefenster, Eigenschaftenfenster, Projektbaum und der Arbeitsfläche zusammen. Insgesamt kommt die Oberfläche sehr klassisch daher und findet sich in ähnlicher Form bei vielen Windows-Entwicklungstools so wieder. Dies hat den Vorteil, dass sich der Anwender schnell wiederfindet, wenn er bereits mit ähnlichen Tools gearbeitet hat. Bezüglich der Usability wurde darauf geachtet, dass der Anwender möglichst wenig tippen muss. Lasal erstellt den Code für die Deklarationen von Klassen, Variablen, Schnittstellen usw. automatisch im Hintergrund. Dem Anwender bleibt nur mehr die Implementierung der Methoden. Im Projektbaum werden alle Klassen und Dateien, die zum Projekt gehören, angezeigt. Neue Dateien, Globale Typen, Klassen, Libraries und Netzwerke lassen sich dort erstellen, verändern und löschen. Der Nutzer hat folgende Ansicht, wenn er mit Lasal arbeitet:

- Anzeige der veränderten Eigenschaften des aktuellen Elements im Eigenschaftenfenster
- Ausgabe aller Benutzerinformationen (Info-, Warn-, und Fehlermeldungen)
- Netzwerk- und Sourcecodeeditor
- Baumdarstellung aller Projektelemente

Mit dem Macro-Manager lassen sich externe Programme einbinden und ausführen. Mittels Matlab-Simulink Import können C-Files, die über Matlab-Simulink erstellt wurden, eingebunden werden.

Eine automatische Code-Ergänzung für Schlüsselwörter, Konstanten, Variablen, Methoden und Funktionsparameter erleichtert die Eingabe.

Einfacher nicht komplizierter

In der Automatisierungstechnik hat sich der Eindruck verfestigt, dass die objektorientierte Programmierung alles nur noch komplizierter macht. In der Praxis bestätigt sich jedoch genau das Gegenteil. Gerade die Automatisierungstechnik ist ein Paradebeispiel für einfaches, objektorientiertes Design, da reale Komponenten abgebildet werden müssen und sich somit die Klassen nahezu von selbst definieren. Im Wesentlichen ist es die Umstellung auf ein neues Programmierparadigma, welche den Eindruck der Komplexität hervorruft. Zudem ist es bis jetzt nur selten gelungen, das objektorientierte Programmieren verständlich und nachvollziehbar zu erklären. Viel zu häu-

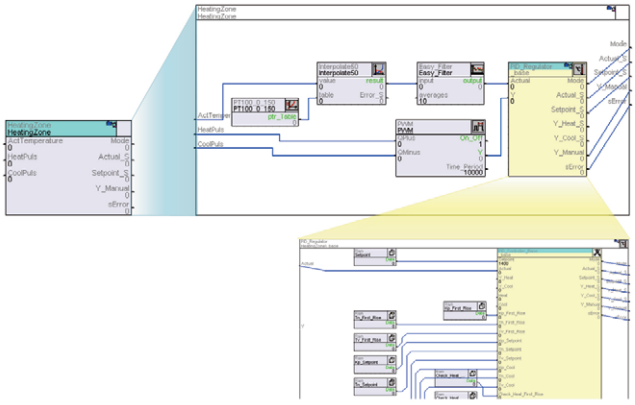

Bild 2: Komplexe Klasse mit eingebetteten Objekten.

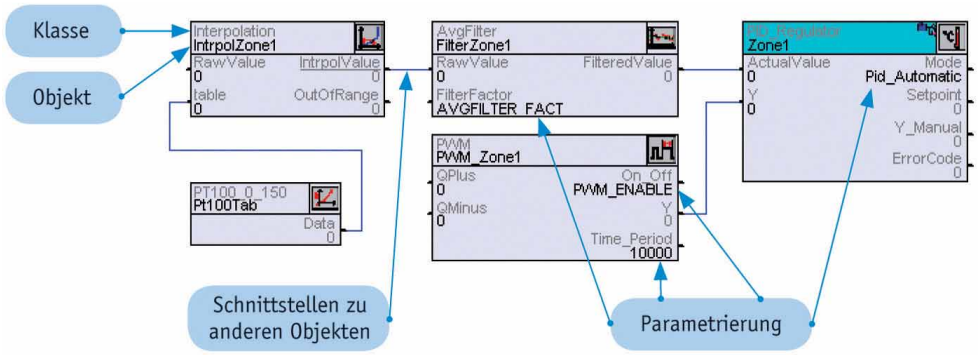

Bild 3: Die grafische Darstellung vereinfacht die objektorientierte Programmierung für den Anwender ganz erheblich.

fig werden Begriffe wie Klassen, Aggregation, Vererbung und Instanzen dazu verwendet, um die eigene Kompetenz zum Ausdruck zu bringen. Hier nimmt Lasal dem Anwender die Scheu, weil in der Entwicklungsumgebung die Klassen über die Oberfläche modelliert werden. Der Entwickler wird mit der komplexeren Syntax der OOP nicht konfrontiert.

Vorteile der objektorientierten Programmierung

In der Praxis bietet OOP eine ganze Reihe von Vorteilen gegenüber der klassischen SPS-Programmierung und der klassischen prozeduralen Hochsprachenprogrammierung. In Bezug auf die Art und Weise, wie bei Sigmatek OOP eingesetzt wird, ergeben sich folgende Vorteile für den Anwender:

- Wiederverwendbarkeit des Quellcodes
- Lesbarkeit des Programmcodes
- Sicherheit
- Erhöhte Qualität des Programmcodes
- Drastische Vereinfachung bei Änderungen über verschiedene Maschinen hinweg
- Vereinfachtes Arbeiten in Teams
- Abbildung von realen Maschinenkomponenten durch Objekte

Objektorientiert und vernetzt

Im Rahmen dieses Beitrages ist es nicht möglich, objektorientierte Programmierung in Gänze zu erklären. Aus diesem Grund werden nur einige Merkmale erläutert, die im Zusammenhang mit der Programmierung mit Lasal wichtig sind. Ein Programm besteht aus Objekten. Also aus selbstständigen Teilen, die für bestimmte Aufgaben zuständig sind. Jedes Objekt enthält Informationen, die eine Aussage über seinen augenblicklichen Zustand machen. Diese Informationen, oder auch Daten, werden Variablen genannt. Die Aufgaben und Aktionen eines Objektes werden durch Funktionen dargestellt, die man bei der OOP als Methode bezeichnet. Eine Klasse ist eine abstrakte Beschreibung eines Anlagenteils mit allen Schnittstellen, Daten und Methoden und bildet so etwas wie die Basis für Objekte. Objekte können als verschiedene Exemplare einer Klasse bezeichnet werden und haben einen Zustand, ein Verhalten und eine Identität (Bild 3). Objekte werden unterschieden nach ihrem internen Aufbau, den nach außen sichtbaren Eigenschaften und ihrer Funktionalität nach außen. Bei Lasal werden die Klassen und Objekte grafisch dargestellt und in Netzwerke eingebunden. Der innere Aufbau wird mit einer der zur Verfügung stehenden Programmiersprachen programmiert. Der Programmfluss entsteht dann durch die Vernetzung unterschiedlichster Objekte auf Basis der nach außen sichtbaren Eigenschaften und Funktionalitäten. Aus mehreren Klassen lässt sich eine komplexe Klasse (Aggregation) erstellen, die eingebettete Objekte enthält (Bild 2).

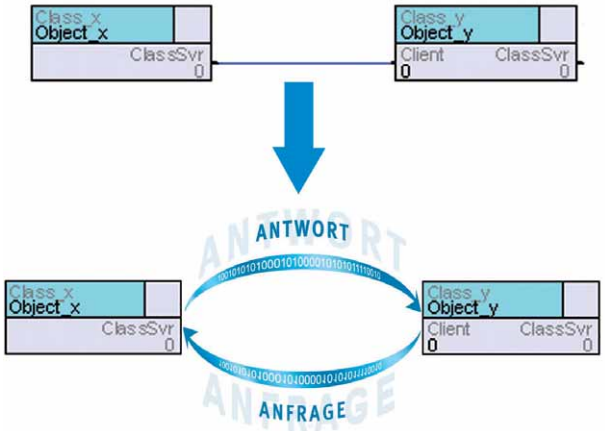

Bild 4: Die Client-Server Technologie funktioniert wie bei einem Netzwerk mit Anfrage und Antwort.

Grafische Programmierung

Die Basis für die Programmierung mit Lasal ist eine grafische Programmierung mit Netzwerken. Bereits erstellte Klassen werden ganz einfach aus dem Projektbaum des Klassen-Fensters (Klassen Bibliothek) per Drag&Drop in ein grafisches Objektnetzwerk fallengelassen, und schon ist ein neues Objekt entstanden.

Zur Verfügung stehende Sprachen

Die Erstellung der Applikation kann in Structured Text, Kontaktplan, Anweisungsliste, jeweils nach IEC 61131-3 Norm, in einer zur Laufzeit interpretierten Ablaufsprache und in C erfolgen. Dabei können die Sprachen innerhalb eines Projekts beliebig gemischt werden.

Debugging

Für die Programmanalyse und Fehlersuche bietet Lasal eine Vielzahl an Debugging-Funktionen:

- Online Diagnose im Netzwerk: Aktueller Status der einzelnen Objekte ersichtlich, Wertänderungen direkt im Netzwerk möglich, Absetzen von ganzen Befehlsketten mit Parametern, erste Diagnose ohne Konfrontation mit Code
- Breakpoints: Standard Breakpoints, Bedingte Breakpoints, Datenbreakpoints
- Durchlaufzähler: Zähler in Statuszeile bei Abarbeitung jener Codezeile, auf der der Cursor steht. Projekt wird dabei nicht angehalten.
- Watch-Window: Anzeige von beliebigen Variablen, gesamten Objekten per Selektion bis hin zu den Prozessorregistern.
- Callstack: Anzeige des Verlaufs der aufgerufenen Funktionen wenn die Steuerung im Breakpoint steht sowie Codeanzeige auf Assemblerebene.
- Trace-Fenster: Im Anwenderprogramm können mittels der Funktion 'Trace' Text und Variablenwerte an die Entwicklungsumgebung geschickt werden.
- Data Analyzer: Echtzeit Datenaufzeichnung, Samplerate bis zu 50µs möglich, Messlinien (horizontal, vertikal) inklusive Messwerte, Starttrigger, Stoptrigger, frei skalierbare Kurven und Zoom, Import-, Exportfunktionen
- Taskmanagement: Anzeige aller zyklischen Tasks (Realtime, Cyclic, Background), Messungen von Tasklaufzeit (aktuell und maximal), Deaktivieren/Aktivieren/Ändern der Zykluszeit von Tasks zur Laufzeit
- TaskTrace: Aufzeichnung der Taskwechsel des Multitaskingsystems in Echtzeit, welcher Task hat den niederprioren Task unterbrochen usw.

Die ist nur ein Teil der für das Debugging zur Verfügung stehenden Möglichkeiten. Daneben stehen noch sogenannte 'Advanced Debug Tools' zur Verfügung.

Bibliotheken

Sigmatek bietet eine Reihe von Standard-Bibliotheken wie Klassen für Hardwarezugriffe, Systemzugriffe, Druckerklassen und Visualisierungsklassen. Daneben stehen auch vorgefertigte Anwendungsklassen wie z.B. PID-Regler, Datenlogger und eine komplette Motion Library zur Verfügung, die dazu beitragen, die Programmierzeiten zu verkürzen. An-

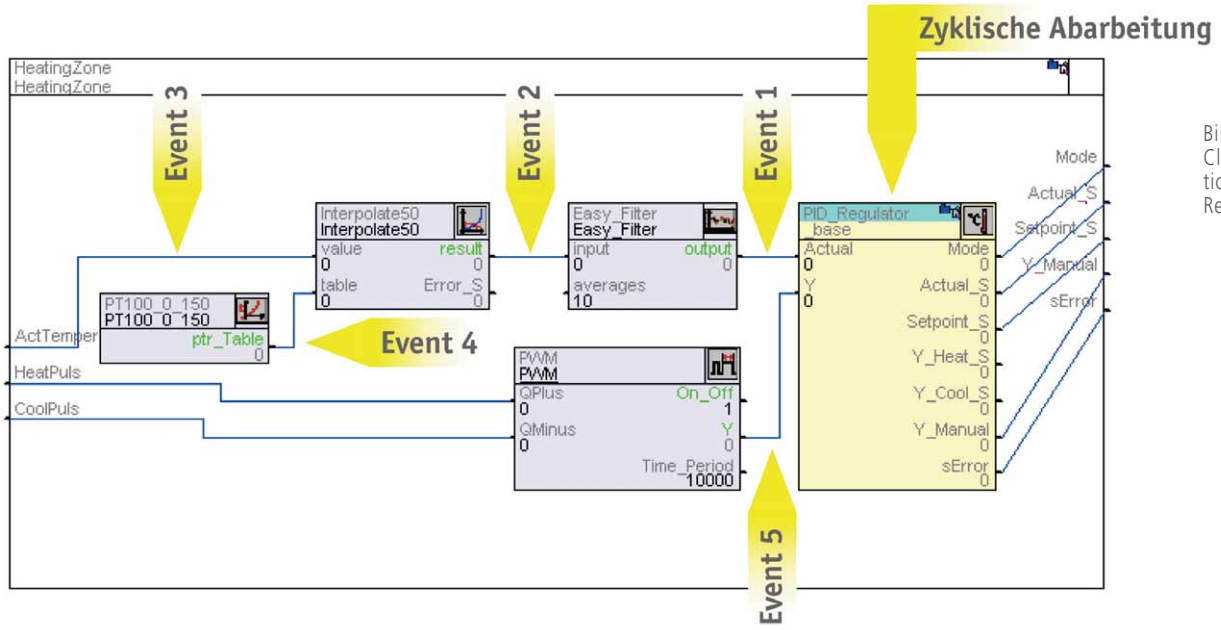

Bild 5: Beispiel für eine Client-Server Kommunikation anhand eines PID-Reglers.

wender können beliebige eigene Bibliotheken erstellen, um Klassen und Netzwerke für eine Wiederverwendung abzulegen.

Client-Server-Kommunikation

Wie in einem Netzwerk funktioniert die Kommunikation zwischen Client und Server bei Lasal über einen Request/Respose-Mechanismus (Frage/Antwort). Der Client fordert einen Dienst an, der vom verbundenen Server zur Verfügung gestellt bzw. bearbeitet wird. Die Kommunikation erfolgt über verschiedene Arten von Kanälen: untypisierte Kanäle zum einfachen Datenaustausch sowie zum Absetzen von Kommandos sowie Objektkanäle, mit denen Methoden der referenzierten Klasse aufgerufen werden können (Bild 4 und 5).

Multitasking

Das Betriebssystem von Sigmatek unterstützt und nutzt sowohl präemptive Tasks (der Betriebssystemkern steuert die Abarbeitung der einzelnen Prozesse und hält jeden Prozess nach einer bestimmten Abarbeitungszeit zu Gunsten anderer Prozesse an) als auch kooperative Tasks (die Rechenleistung wird nacheinander an die Prozesse verteilt). Visualisierung und Steuerungsprogramm können so auf einem Prozessor laufen. Dabei handelt es sich um ein mehrschichtiges Multitasking-System. Das System stellt drei Taskklassen zur Verfügung:

- Echtzeit Task
- Zyklischer Task
- Background Task

Das Betriebssystem behält die Kontrolle über die abzuarbeitenden Aufgaben (präemptives Multitasking), innerhalb einer Taskpriorität wird kooperatives Multitasking angewendet.

Scripting mit Python

Mithilfe von Scripting können in Lasal diverse Abläufe für das Projektmanagement erstellt werden. So können z.B. mithilfe einer Scriptsprache Projekte erstellt und angepasst werden. Dies ist z.B. interessant, wenn es darum geht, ein Basisprogramm auf verschiedene Maschinen gleicher Bauart zu implementieren. Für eine bestimmte Baureihe einer Maschine gibt es ein einziges Basisprojekt. Von diesem können dann automatisch verschiedene Varianten für die kundenspezifischen Maschinen derselben Baureihe abgeleitet werden. Lasal stellt hierfür diverse Befehle der verbreiteten Scriptsprache Python zur Verfügung (Bild 6).

Objektorientierung gehört die Zukunft

Sigmatek kann durchaus als Pionier der objektorientierten Programmierung bezeichnet werden und bietet dazu ein rundes und bewährtes System. Wie im Prinzip bei allen Steuerungsanbietern gibt es noch Optimierungspotenzial im Bereich der Usability und Integration der Tools.

Objektorientiertes Programmieren ist für die Automatisierungstechnik bestens geeignet. Dies zeigt das Beispiel von Sigmatek sehr deutlich. Lasal ist somit für all jene Maschinenbauer interessant, die der zunehmenden Komplexität ihrer Maschinen durch eine leistungsfähige Programmierung entgegen wirken wollen. ■

www.sigmatek-automation.com

Autor: Dipl.-Ing. Martin Buchwitz, Chefredakteur SPS-MAGZIN

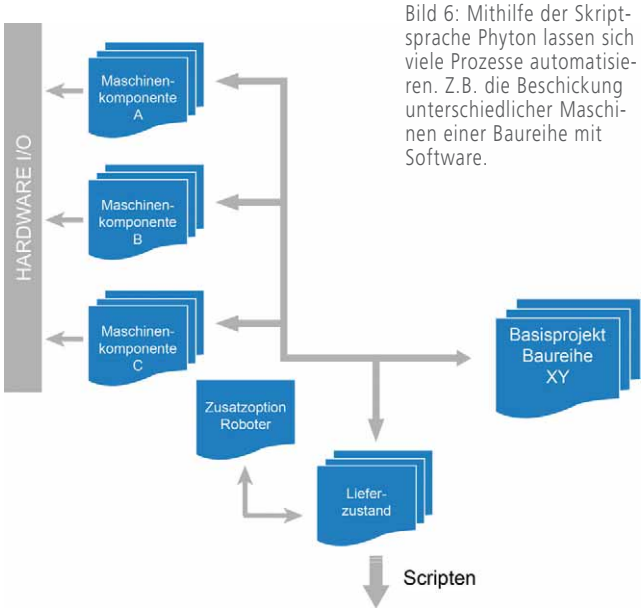

Bild 6: Mithilfe der Skriptsprache Phyton lassen sich viele Prozesse automatisieren. Z.B. die Beschickung unterschiedlicher Maschinen einer Baureihe mit Software.