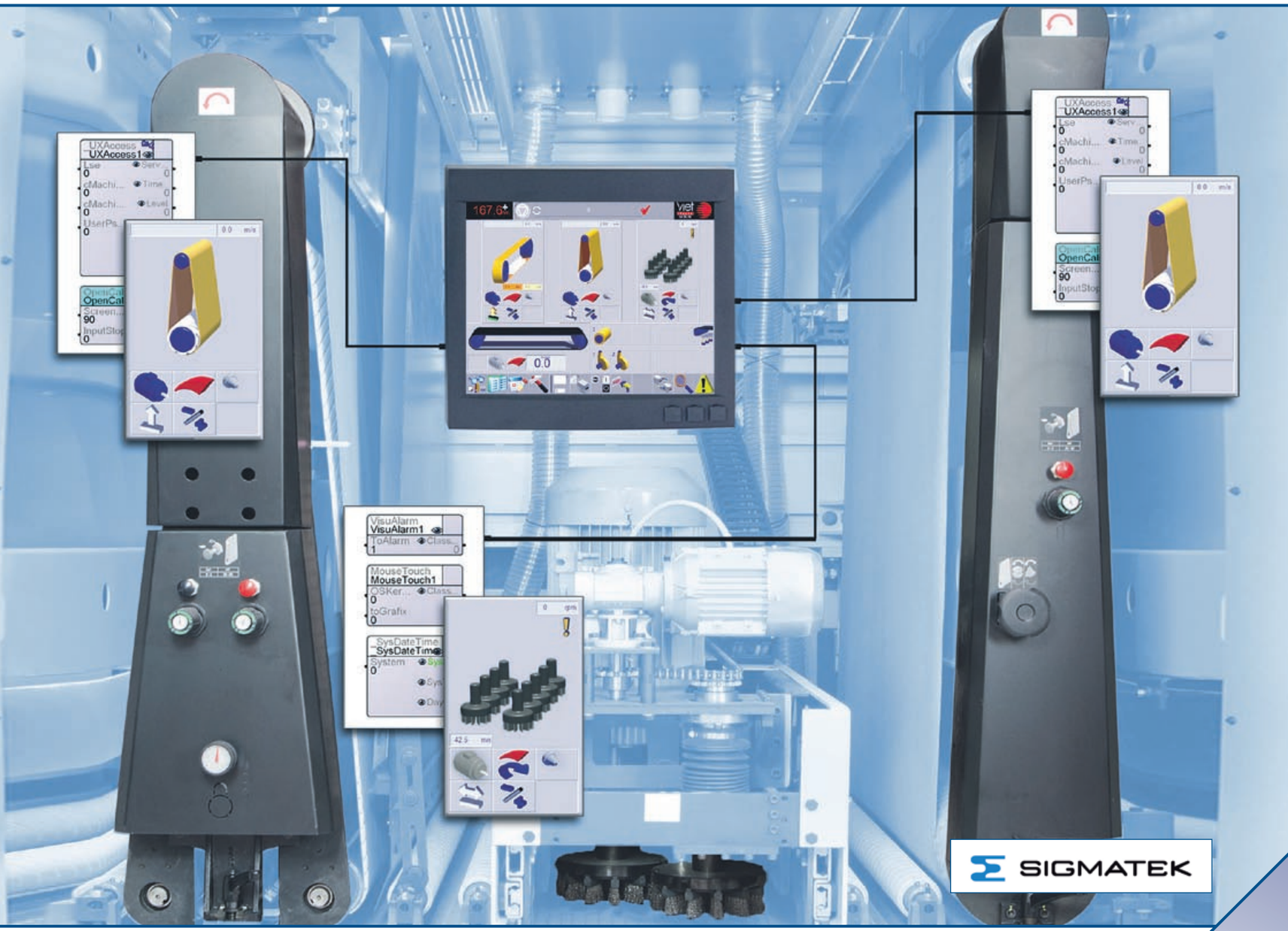
Computer & AUTOMATION 11/2011

11-2011 • 7,50 €
computer-automation.de

# Computer & AUTOMATION

Fachmedium der Automatisierungstechnik




## TITEL: OBJEKTORIENTIERUNG – EIN MUSS!



Ausgabe zur SPS/IPC/Drives

im FOKUS MOBILE AUTOMATION

Bernhard Gangl

# Komplexes einfach umsetzen

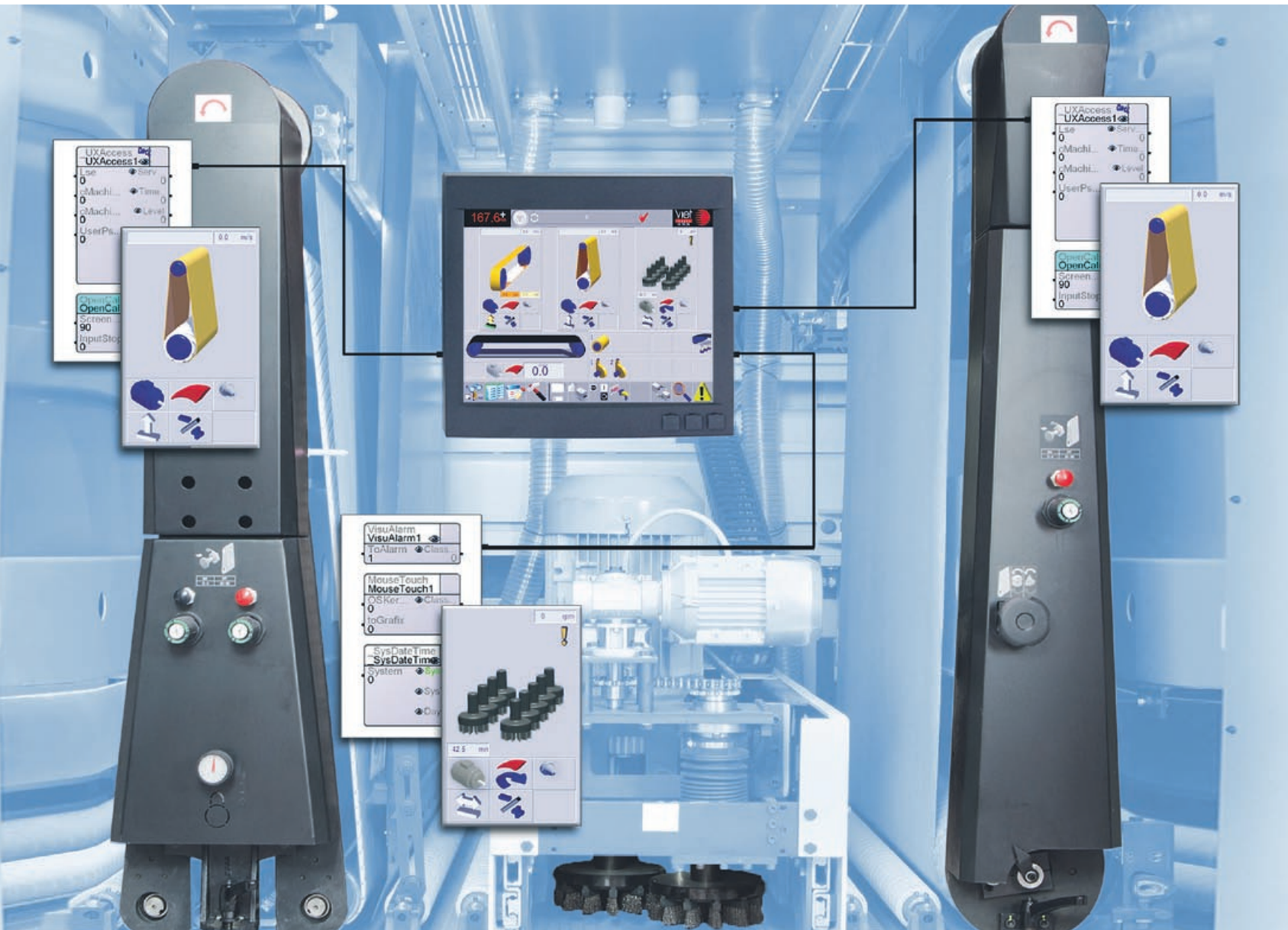
**Viele Anwender haben noch immer eine gewisse Scheu vor der objektorientierten Programmierung – eigentlich unverständlich, denn es gibt für sie kaum ein passenderes Anwendungsgebiet als die Automatisierungstechnik.**

Halle 7 | Stand 370

Effizientes Engineering entwickelt sich zu einem Schlüsselfaktor moderner Maschinen- und Anlagenkonzepte. Gerade hier gibt es – ab einem gewissen Automatisierungsgrad – noch beträchtliche Einsparungspotenziale. Gefragt sind moderne Tools, die dem Anwender helfen, Komplexes einfach und strukturiert umzusetzen. Die Minimierung der Entwicklungszeiten ist dabei nur ein Zielaspekt; eine mindestens ebenso wichtige Forderung lautet, die Software-Wartungskosten erheblich zu reduzieren.

Beispiel Programmierung: Die Verwendung von Hochsprachen wie Structured Text (ST) ist in der SPS-Programmierung seit vielen Jahren Stand der Technik und fest etabliert. Dazu beigetragen haben nicht zuletzt die Berufseinsteiger, die zumeist bereits im Zuge ihres Studiums in Hochsprache programmiert haben. Da die Unterschiede zwischen den verschiedenen Hochsprachen recht marginal sind und sich vor allem durch die Syntax unterscheiden, fällt der Umstieg auf eine andere Hochsprache leicht. Als plattformunabhängige Sprache für diverse Algorithmen hat sich C etabliert. Die Möglichkeit zur Einbindung von C-Sourcen ist in modernen Tools heute somit ein Muss.

Ein Trend jüngeren Datums ist die objektorientierte Programmierung – kurz OOP. Die Idee dabei ist es, Code und Daten in logische Einheiten, in „Objekte“ zusammenzufassen. Software-Entwickler finden in der OOP die Antworten auf die Fragen: Wie mache ich Code wieder verwendbar? Wie mache ich Code lesbar? Wie minimiere ich den Programmieraufwand bei einer Vielzahl von Maschinenoptionen? Für den Einsatz der OOP in der Automatisierungstechnik spricht, dass



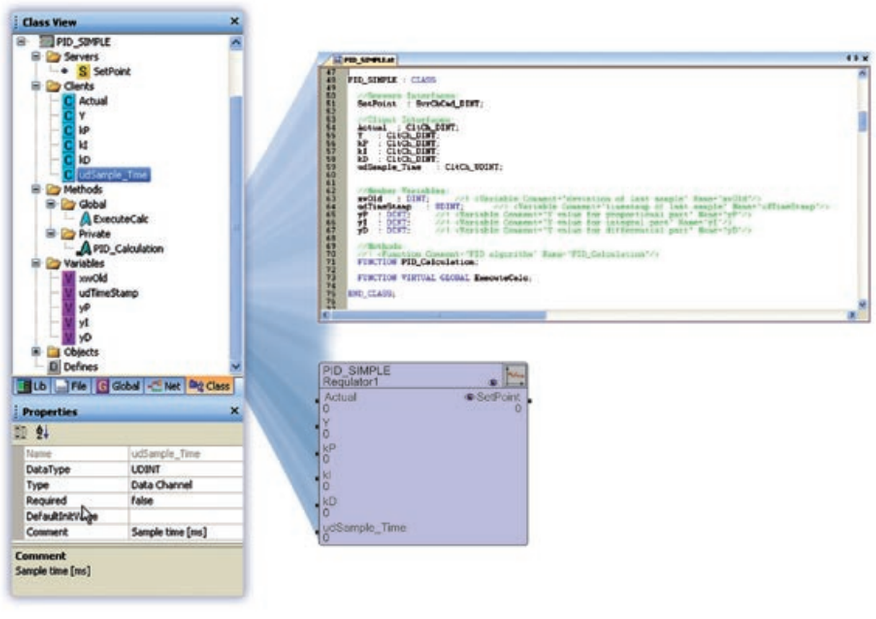
## Die Vorteile der OOP am Beispiel Lasal

Das Nutzen objektorientierter Methoden in der Automatisierungssoftware Lasal bietet dem Anwender eine Menge komfortabler Vorteile:

- Abbildung von realen Maschinenkomponenten durch Softwareobjekte.
- Getestete Funktionsbausteine dank Kapselung – „use and forget".
- Wiederverwendbarkeit der erstellten Funktionsbausteine und des Quellcodes.
- Klare, grafische sichtbare Schnittstellen nach außen. „Unsauberkeiten" wie Zugriffe auf Daten an x Stellen im Projekt sind somit erst gar nicht möglich.
- Vereinfachtes Arbeiten in Teams.
- Durch die grafische Vererbung sind Änderungen und Erweiterungen für bestehende Funktionen mit minimalstem Aufwand durchführbar.
- Durch das grafische Bündeln mehrerer einzelner Funktionsbausteine kann eine komplexe Funktionsabfolge geschaffen werden. Beispiel: Zwei Objekte für ein Zahnrad und ein Objekt für einen Motor ergeben einen neuen Baustein „Getriebemotor".
- Durch die Kapselung können die Bausteine einfach in Libraries aufbewahrt werden.
- Abgeschlossene Bausteine schaffen die Möglichkeit, Projekte vollkommen ferngesteuert über Scripts (Python) zu erstellen oder abzuändern.
- Lesbarkeit und Qualität des Programmcodes steigen.

hier genau wie in der OOP in Komponenten (Objekten) gedacht wird – also zum Beispiel in Motoren, Getrieben und daraus folgenden Antriebssträngen. Somit ist das objektorientierte Design der Software – der wahrscheinlich wichtigste Aspekt bei diesem Programmierkonzept – ein Leichtes, da es gilt, reale mechanische Komponenten in modularen Softwarekomponenten abzubilden und sich die Klassen somit nahezu von selbst definieren.

Eine Klasse ist der „Bauplan" für ein Objekt und definiert den Programmcode und die Datenelemente. Jede Klasse übernimmt eine bestimmte Aufgabe wie beispielsweise die Ansteuerung eines Antriebs oder eines Ventils. Der eigentliche Programmcode eines Objektes wird in den gebräuchlichen Sprachen der IEC 61131-3 (Strukturierter Text, Kontaktplan oder C) implementiert. Dies ist ein wesentlicher Akzeptanzfaktor, da so die Methoden der objektorientierten Programmierung als durchgängige Erweiterung der vertrauten und bewährten Sprachen zur Verfügung stehen.




**Über den „Klassentree" von Lasal Class wird die Klasse mit ihren Eigenschaften, Methoden und Schnittstellen definiert, die dazu notwendige Deklaration und grafische Darstellung erledigt das Werkzeug automatisch.**

Die Vorteile der objektorientierten Programmierung sprechen für sich – warum aber scheuen sich trotzdem noch viele Anwender in der Automatisierungstechnik, diese Methoden zur Entwicklung ihrer Maschinen und Anlagenkonzepte einzusetzen? Es die Umstellung auf ein neues Programmierparadigma, die den Eindruck der Komplexität vermittelt. Zudem wird vielfach mit den Begriffen Objekt, Klasse, Instanz, Vererbung und Aggregation „jongliert", um Kompetenz zu vermitteln, anstatt dem Anwender die Scheu zu nehmen und ihm praktisch die Vorteile aufzuzeigen.

Vor diesem Hintergrund stand daher bei der Entwicklung des objektorientierten Engineering-Tools „Lasal" von Sigmatek von Anfang an das Ziel im Vordergrund, dem Anwender alle Vorteile der OOP zur Verfügung zu stellen, ohne dass er in der praktischen Anwendung mit der komplexeren Syntax der Objektorientierung in Berührung kommt. Das heißt: Sämtliche Aktionen für die Erstellung oder Änderung einer Klasse können vom Programmierer über die Bedienoberfläche erledigt werden, alle notwendigen Deklarationen führt Lasal selbstständig im Hintergrund aus. Der Anwender kann sich daher auf die Implementierung der Methoden (Funktionen) konzentrieren.

Die grafische Darstellung trägt ebenfalls zur Vereinfachung bei. Beim grafischen Ansatz werden die von Klassen erzeugten Objekte (Maschinenmodule) in so genannten Netzwerken dargestellt. Mit anderen Worten: Der Entwickler sieht auf den ersten Blick die Eigenschaften eines Maschinenteils sowie die Kommunikation mit anderen Objekten, sprich Maschinenteilen. Das Ändern einer Klasse über die Bedienoberfläche wirkt sich sofort auf die grafische Darstellung aus. Kurzum: Die Maschine wird in der Software grafisch nachgebildet.

Da sich die verschiedenen Maschinenteile und deren Interaktion miteinander in der Software widerspiegeln, gestaltet sich auch die Kommunikation zwischen dem Konstrukteur beziehungsweise dem Elektrotechniker und dem Software-Entwickler viel einfacher. Einmal erstellte und getestete Softwaremodule (Objekte) lassen sich in Bibliotheken ablegen. Die Objekte sind so-

(Bilder: Sigmatek)

mit in unterschiedlichen Projekten oder Systemteilen wieder verwendbar und können zu hochkomplexen Programmstrukturen zusammengefügt werden. Software wird somit „nachhaltig“, was heute ein gewichtiges Argument ist. Denn wenn Code über längere Zeit verwendet wird – was ja wünschenswert ist – entsteht bei bisherigen Programmiermethoden oft ein undurchschaubares Softwaregebilde. Durch die Strukturiertheit der OOP ist es hingegen möglich, bestehende Klassen zu erweitern beziehungsweise zu verändern (Ableitung) – und zwar ohne dazu in die ursprüngliche Klasse eingreifen zu müssen. Die Änderungen sind somit immer nachvollziehbar und einfach zu erkennen. Dies spart Zeit und schont die Nerven.

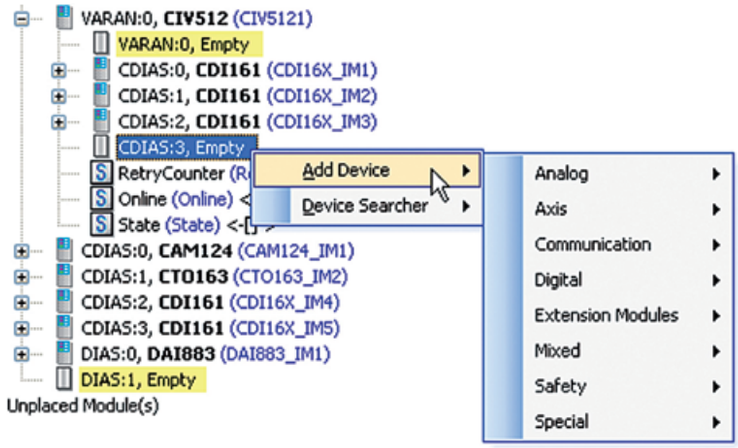
## Komfort und Usability im Fokus

Sigmatek hat bereits vor rund zehn Jahren damit begonnen, objektorientierte Konzepte aus der IT-Welt in die Automatisierungstechnik zu übertragen beziehungsweise dafür zu optimieren. Das dahinter stehende Tool Lasal ermöglicht es dem Maschinenbauer, neben der eigentlichen Steuerungsprogrammierung auch Visualisierungs-, Motion-Control-, Safety- sowie Service- und Fernwartungs-Aufgaben effizient zu realisieren. Bei der aktuellen, zweiten Lasal-Generation liegt das Hauptaugenmerk nun voll und ganz auf den Komfortfunktionen sowie der weiteren Verbesserung der Usability – und das trotz oder besser gesagt durch die Objektorientierung!

So wird beispielsweise über häufig benötigte Funktionalitäten eine weitere Bedienoberfläche gesetzt, um Einzelschritte zu bündeln, damit der Anwender noch schneller und komfortabler zum Ziel kommt. Beispiele dafür sind der Hardware-Editor, fertige in der Bibliothek hinterlegte Komponenten (zum Beispiel eine komplette Betriebsdatenerfassung oder Motion-Technologie-Module) oder auch die automatische Softwaregenerierung.

Hohen Komfort bietet das Engineering-Tool beispielsweise außerdem bei der Konfiguration der Hardware. Über den integrierten Hardware-Editor lassen sich alle Module projektieren, parametrieren und diagnostizieren. Aktionen wie das Erzeugen einer Instanz (Objekt) für ein Hardwaremodul mit dem dazugehörigen Softwarebaustein (Klasse), führt er automatisch im Hintergrund aus. Dies entlastet den Anwender, da er selbst nichts programmieren muss. In der ersten Version musste sich der Anwender um die Erzeugung der Objekte noch manuell kümmern. Die Parametrierung von Modulen sowie einzelner I/Os erfolgt in gewohnter Weise über den so genannten Property-Browser.

Zusätzlich bietet der Hardware-Editor Unterstützung bei der Verknüpfung der I/Os mit den Anwenderobjekten. Mit dem I/O-Connection-Manager lassen sich Eingänge, Ausgänge und Schnittstellen über eine übersichtliche Oberfläche (Auswahl in Listenform) einfach




**Mittels des Hardware-Editors lassen sich die benötigten Hardwaremodule einfach zusammenstellen: über gruppierte Listen, per Suchdialog oder – bei einer bereits bestehenden Anlage – per automatischer Ermittlung über die Online-Verbindung.**

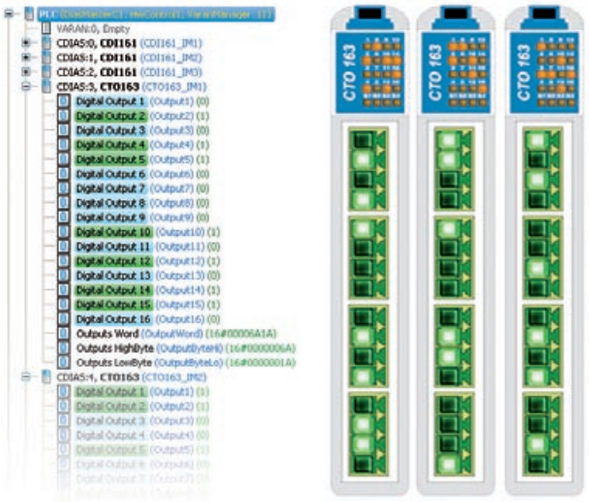
verdrahten. Die Möglichkeit für Import und Export dieser I/O-Verbindungsliste gibt dem Anwender die notwendige Flexibilität bei der Projektierung unterschiedlicher Maschinenvarianten.

Ein Klasse hat zunächst bestimmte Aufgaben in der Steuerung zu übernehmen. In Lasal Class bietet eine Klasse zusätzliche Features wie eine direkt integrierte Revisions-Historie und die entsprechende Dokumentation der einzelnen Klassenelemente. Zudem können Hilfedateien angehängt werden, die wichtige Informationen zum „warum“, „wieso“, „wo“ und „wie“ enthalten. Dies erweist sich bei der Wiederverwendung einer Klasse aus seiner Library als sehr hilfreich. Durch einen Import kann der Anwender per F1-Taste das hinterlegte Dokument – zum Beispiel ein Word- oder ein PDF-Dokument – direkt öffnen.

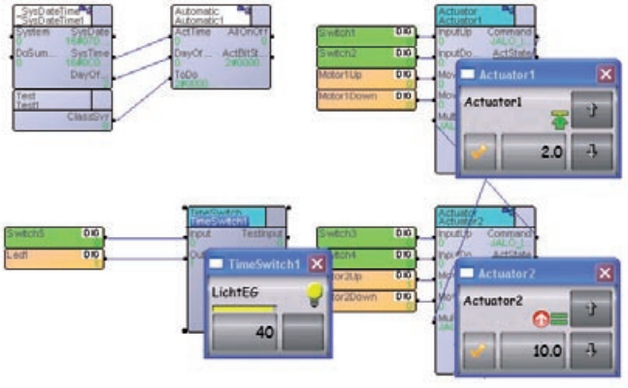
## Objektorientierte Vorteile auch in puncto Visualisierung

Die Objektorientierung eröffnet nicht nur im Bereich der Ablaufprogrammierung von Maschinen neue Möglichkeiten, sondern auch hinsichtlich der Visualisierung. Mit „Lasal Screen“ etwa lassen sich grafische Objekte definieren. Dabei verwendet der Anwender Datenpunkte einer Klasse aus dem Steuerungsprojekt, um das visuelle Erscheinungsbild dieser Klasse zu definieren. Das heißt: Genau so wie im Steuerungsprojekt eine Klasse der Bauplan für die Objekte ist, kann auf der Visualisierungsseite ein Bauplan für die grafische Anzeige der Objekte einer Klasse erstellt werden. Auf der Bildschirmseite muss anschließend nur noch für jedes Visualisierungsobjekt der entsprechende Objektname aus dem Steuerungsprojekt zugeordnet werden. Die Erstellung einer neuen Visualisierung erfolgt demnach mittels Parametrierung, eine aufwendige Programmierung entfällt.

Damit der Anwender die grafischen Objekte nicht nur in der Visualisierung einsetzen kann, hat Sigmatek den „VisualObject-View“ (VOV) entwickelt. Damit ist die Visualisierung einzelner Klassen direkt in der Programmierumgebung Lasal Class nutzbar, um beispielsweise einzelne Anlagenkomponenten offline zu parametrieren oder per Onlineverbindung visualisieren und testen zu können. Diese grafischen Visualisierungsobjekte lassen sich auch direkt an eine Klasse anhängen. Bei Bedarf können die Vi-




**Dieses Beispiel zeigt, wie drei im Hardware-Editor konfigurierte Ausgangsmodule in der Visualisierung dargestellt werden können. Dabei wurde das Aussehen einmal definiert, anschließend jedoch dreimal platziert und auf das entsprechende Hardware-Objekt zugewiesen.**




**Im VisualObjektView (VOV) lassen sich Visualisierungsobjekte auch in Lasal Class einblenden. Softwareteile können so direkt in der Entwicklungsumgebung visualisiert und getestet werden.**

sualisierungsobjekte dann schnell mit einer Klasse verknüpft werden. In diesem Fall ist das Erscheinungsbild in der Visualisierung fest mit dem Funktionsbaustein verbunden.

## Flexible Updates per Kapselung

Lasal arbeitet mit gekapselten Objekten, die über Schnittstellen mit der „Außenwelt“ kommunizieren. Wenn diese Schnittstellen klar definiert sind, sind Objekte später mit Leichtigkeit gegen andere austauschbar. Software wird so wartungsfreundlich und leicht testbar – und das noch nach Jahren.

Mit der Kapselung bei der OOP ergeben sich somit neue und höchst flexible Möglichkeiten bei Programm-Updates oder Fehlerkorrekturen in der Software. Ein Beispiel: Angenommen ein Softwarebaustein hat die Aufgabe, eine Temperatur zu regeln. Es sind tausende Maschinen in unzähligen Maschinenvarianten am Markt und plötzlich stellt man fest, dass der Temperaturregler einen schwerwiegenden Fehler enthält. Klassischerweise müsste der Hersteller nun für jede Maschine den spezifischen Softwarestand korrigieren und an der Maschine updaten. Lasal Class erstellt in diesem Fall auf Knopfdruck einen USB-Stick mit der korrigierten Klasse, in unserem Beispiel mit dem korrekten Reglerbaustein. Dieser kann nun überall per Stick oder Online-Fernwartung eingespielt werden, ohne dafür die tatsächlichen Programme auf den Maschinen kennen zu müssen. *gh*

**Bernhard Gangl**

ist Abteilungsleiter Software bei Sigmatek.