



Commande machine

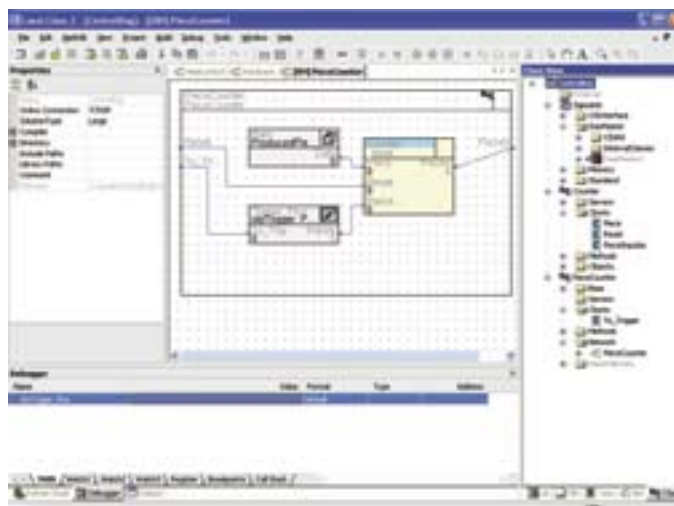
Utilisez la programmation orientée objet avec la norme IEC 61131

La programmation orientée objet est tout sauf neuve. La force de l'orientation objet (OO), par rapport à la programmation traditionnelle, réside dans la structure plus claire qu'elle propose. Pour la construction de machines, la combinaison de l'IEC 61131 avec la programmation orientée objet (IEC 31113-3) peut faciliter la vie du développeur. Et plus important encore: le délai de développement d'une commande machine – et surtout des variantes ultérieures – sera sensiblement réduit.

La commande machine remplace de plus en plus le PLC traditionnel. Cela n'a rien de surprenant car le programme cyclique tout ou rien qui tourne en continu dans le PLC n'offre plus une structure suffisante. Les commandes actuelles sont plus complexes: outre les tâches logiques, le PLC doit aussi gérer une série d'autres tâches de commande comme le contrôle de mouvement, la HMI, les opérations de calcul, l'enregistrement de données, les alarmes et la communication. La norme IEC 61131 a déjà apporté de grandes améliorations mais nous en sommes – malgré tout – toujours à une situation où 'plus mon programme est long, plus la gestion de toutes ces lignes est longue'.

Modularité

Et c'est là qu'intervient l'OO. Dans un environnement OO, chaque tâche distincte (objet) est dotée d'une fonction propre et d'un code propre (méthode), de sa base de données et de son temps de cycle. De ce fait, la maintenance d'un programme écrit en OO est fortement simplifiée. L'OO met la modularité à l'avant-plan. Néanmoins, sa force réside aussi dans la simplification de la programmation par rapport aux langages de pro-



Les données entre les classes sont visualisables en ligne. Le compteur de pièces se compose ici de trois classes : le Counter, le ProducedPie et le trigger (entrée de comptage).

grammation à orientation plus traditionnelle. Un objet ne doit être programmé qu'une seule fois, ensuite il est en permanence réutilisable. Cette facilité a son importance: un objet utilisé une fois peut être testé comme unité indépendante. En d'autres termes: l'objet peut être isolé du reste du programme.

Innovateur

Le grand avantage de l'OO est évident: le gain de temps. La technologie client/serveur en combinaison avec la programmation orientée objet et une présentation graphique, permet une réalisation plus rapide et plus simple des commandes machine. La méthode de réflexion et la manière de présentation jettent un pont entre le constructeur et le développeur de logiciels. Avant de démarrer l'écriture de la première ligne de code, il faut d'abord déterminer les différentes classes (fonctions) de la machine. Chaque classe est décrite et les relations réciproques entre les classes sont nommées (canaux client-serveur). Vient ensuite la description de la tâche d'une classe, c.-à-d. les fonctions (méthodes) dont elle doit s'acquitter. Pensez par exemple à

l'initialisation, au déclenchement d'alarmes, à la lecture et l'écriture de variables, aux tâches cycliques (PID, forage, déplacement, activation-désactivation de vannes...), aux tâches temps réel (positionnement asservi...), aux tâches en arrière-plan (HMI, archivage...). Une 'classe' dissimulera ses données aux autres classes. Les autres classes ne peuvent s'approcher directement de ces données, seulement via des canaux client-serveur publics. Il existe une séparation claire entre l'interface et l'implémentation. Voilà l'essence de la programmation orientée objet. Chaque classe peut désormais être construite par un autre programmeur et testé séparément. Une classe qui a été approuvée peut être reprise sans problème dans le projet et, naturellement, dans la bibliothèque.

IEC 31113-3: y compris l'OO

La norme IEC 61131-3 va un pas plus loin car elle est étendue avec la programmation orientée objet. Un programmeur peut se concentrer sur le fonctionnement de sa spécialité. Il se charge de l'interface avec l'environnement, comme discuté auparavant. Dès

qu'il a obtenu son information, il peut la traiter et mettre de nouvelles informations à disposition. En d'autres termes: dès qu'il reçoit des photocellules l'information qu'une pièce de travail est en place, le traitement peut démarrer. Lorsque cette opération est prête, un signal part vers l'action suivante, par exemple le transporteur qui initie son mouvement. Le transporteur ou convoyeur à rouleaux a sa propre classe. Celle-ci est décrite de manière à reconnaître automatiquement si la bande se trouve en début de ligne, entre deux stations ou à la fin. Le remplissage ou le vidage est automatiquement surveillé. Cette classe universelle est maintenue par un seul programmeur qui traite les éventuelles nouvelles demandes de collègues. Ses collègues disposent toujours de la dernière version. La reconnaissance est fortement accrue, la fiabilité augmente car les classes peuvent être testées extensivement et le délai de développement des nouvelles applications est sensiblement raccourci. La mise en service est réduite à son minimum.

Débogage

La représentation graphique des classes montre clairement 'en ligne' la cohésion entre les objets. Le projet global, la fonctionnalité, le trafic de données et les interfaces sont disponibles dans un seul aperçu. Les liaisons complexes se reconnaissent, se contrôlent et se modifient dès lors aisément. Le constructeur reconnaît sa machine et le programmeur reconnaît les classes. Les informations escomptées et les informations manquantes sont clairement identifiées. Les classes sont testées séparément, les données internes n'importent donc plus pour l'aperçu. En ramenant l'information à l'essentiel, l'aperçu est agrandi, la cause d'un problème sera plus vite localisée et le programmeur de la classe 'problématique' n'a plus qu'à vérifier son code. <<



Comment fonctionne la norme IEC 61131-3 orientée objet dans la pratique?

Comment fonctionne la combinaison de l'OO et de la norme IEC 61131 (ou IEC 61131-3) dans la pratique? Nous avons examiné LASAL Class, un des premiers environnements de programmation orientés objet, spécialement conçu pour les constructeurs de machines/utilisateurs de PLC.

Le progiciel permet d'écrire un code programme en Texte structuré selon la norme IEC-61131, en liste d'instructions (AWL/LIST), en schéma à contact (LD/CONT), en C++ et offre un interpréteur pour les commandes séquentielles. Il a recours à la technologie client-serveur, à la présentation graphique et à l'orientation objet.

Plusieurs personnes peuvent travailler simultanément en toute simplicité à une application dans différents langages ('multi development possibility'). Le programmeur connecte les objets de manière graphique. Il place par exemple un module d'E/S (un objet) à l'écran. Ensuite, il tire une ligne d'un canal serveur vers le canal client souhaité d'un régulateur PID par exemple (un objet). La méthode sous-jacente 'code logiciel' les relie ensemble par logiciel. Ce que fait la commande avec les signaux d'E/S est déterminé dans l'objet qui va de pair avec le PLC. Cela peut être une fonction simple mais cela peut être aussi une fonction très complexe. Un actionneur peut être piloté directement. Cependant, une fonction de temporisation ou de comptage ou encore, une

fonction de diagnostic est également envisageable. Même l'asservissement d'un axe complet, une unité d'enlèvement d'une machine de pulvérisation, un distributeur d'une machine d'emballage... peuvent constituer un objet. Il est ainsi tout aussi facile de relier un axe asservi que de raccorder un capteur à une entrée. Pas besoin de connaissances spécifiques des classes pour les utiliser. Les classes, y compris leurs méthodes (fonctions) sont définies dans une bibliothèque et sont aisément réutilisables par simple glissement et remplissage éventuel des paramètres adéquats. Outre LASAL Class, il y a aussi LASAL Screen Editor et Lasal Text. LSE permet de développer des objets graphiques qui ont un lien direct avec leurs objets de commande. LSE offre à l'utilisateur une interface HMI directement intégrée dans son environnement de programmation. Les processeurs actuels sont tellement puissants qu'une seule CPU peut se charger - moyennant une programmation OO - tant de la tâche de commande que de la tâche HMI.

Exemples d'applications

Depuis les premières explorations prudentes, diverses applications ont été réalisées avec la méthode OO. Les collègues suscitent l'enthousiasme d'autres collègues et les intégrateurs de système apprécient les avantages de la réutilisation, les possibilités multi-développeur et l'indépendance de la plate-forme.

Une foreuse relativement simple qui doit déplacer un bout de métal et forer des trous avec un point de réglage. Une trancheuse pour le

fromage/la charcuterie/le pain qui doit être réglée en fonction de l'épaisseur de coupe, du poids, de la présentation. Une scelleuse d'emballage vertical plein pour l'emballage de chips, bonbons, légumes... laissant le choix entre un entraînement servo, à courant triphasé ou hydraulique. Un autoclave, des robots multi-axe. Un élevage de vaches laitières nécessitant plusieurs régulations de vannes identiques, une station météo et une commande d'éclairage. Une trieuse permettant au client de déterminer lui-même le nombre d'unités de sortie, chaque unité se matérialisant par le simple rajout d'une classe à partir de la bibliothèque, en fonction de la quantité souhaitée. Ou encore une bobineuse avec une cisaille à la volée, une balance industrielle, une planteuse, une machine de traitement de pâte, des épilateurs d'oignons... Même des machines assez simples profitent de la méthode OO.

Remarque finale

La disponibilité d'outils de programmation OO ne fera qu'augmenter dans les prochaines années. Les étudiants disposent en effet de ce bagage en fin d'études et sont habitués à réfléchir dans ces structures. Pour les programmeurs de PLC actuels, l'appropriation de cette technique constituera un nouveau défi. Pas de doute qu'ils y parviendront. En effet, le pas du registre à décalage à la matrice et au struct a également été réussi le siècle dernier.<<