

Integration speicherprogrammierbarer Steuerungen in verteilte, objektorientierte Automatisierungsanwendungen

R. Kröger, Th. Gräff, M. Gröger, M. Núñez

Fachhochschule Wiesbaden, Fachbereich Informatik
Kurt-Schumacher-Ring 18, D-65197 Wiesbaden
email: kroeger@informatik.fh-wiesbaden.de

M. Wack

Klöckner-Moeller GmbH, Bereich Automatisierungsgeräte
Hein-Moeller-Straße 7-11, D-53115 Bonn

Zusammenfassung

This paper presents an approach for integrating a PLC-controlled technical subprocess into a general object-oriented distributed automation application. The approach is based on OMG CORBA principles and has been implemented successfully in a CAN-bus based environment using a Klöckner-Moeller PLC by the DIRECT project of the Wiesbaden Polytechnic. Furthermore, a new method for programming networked PLC applications is proposed. This method uses the object-based approach as well. The application programmer has not to be aware of the distributed environment at design time. He develops his distributed application as a local IEC 1131-3 program without any need for explicit message-based I/O. Based on a decision for assigning the function block instances to the different PLC nodes, the necessary communication code is generated automatically. This method is currently being implemented in a collaboration between Klöckner-Moeller (Bonn) and Wiesbaden Polytechnic.

1. Einleitung

Die Integration von speicherprogrammierbaren Steuerungen (SPSen) in verteilte Netzwerke gewinnt in zunehmendem Maße für Industrieunternehmen an Bedeutung. Eine konsequente Dezentralisierung von Automatisierungsstrukturen entspricht dem Trend, Intelligenz dort haben zu wollen, wo diese benötigt wird. Hieraus ergeben sich neue zusätzliche Anforderungen an die Vernetzung, die Integration in übergeordnete Kommunikationsstrukturen sowie an die Kommunikation von SPS-Systemen untereinander. Zum einen werden SPSen i.d.R. über Feldbusse vernetzt. Diese sind von den Hardwarestrukturen und dem Durchsatz beschränkt und eher für kleine Datenmengen der E/A-Ebene geeignet. Zum anderen ist bisher nur eine lose Kopplung

mit externen Systemen (z.B. Leitsysteme) üblich. Es ergibt sich nun die Aufgabe, das Paradigma von SPSen so zu erweitern, daß diese neuen Kommunikationsformen einfach und transparent unterstützt werden.

Die kostengünstige Entwicklung verlässlicher Anwendungs-Software stellt in diesem Zusammenhang aber noch eine große Herausforderung dar. Die herkömmlichen Methoden für Entwurf und Erstellung von Echtzeitsystemen sind diesen Anforderungen nicht mehr gewachsen. Die Programmierung von SPSen hat zwar mit Einführung der neuen Norm IEC 1131-3 deutliche Fortschritte in Hinblick auf die Modularisierung und Portierung von Steuerungsprogrammen gebracht, allerdings wird wenig und ausschließlich auf Nachrichtenaustausch ausgerichtete Unterstützung bei der Entwicklung dezentraler SPS-Anwendungen geleistet /5/.

Objektorientierte Methoden für Analyse und Entwurf sind in den vergangenen Jahren in zahlreichen Varianten vorgestellt worden (z.B. /2/, /8/). Sie versprechen, Entwickler zu leiten, die Effizienz ihrer Arbeit zu steigern und die Entwicklungskosten durch bessere Wartbarkeit und Wiederverwendbarkeit von Bausteinen längerfristig zu reduzieren. Eine Unterstützung für die spezifischen Probleme von Automatisierungsanwendungen wird aber erst in Ansätzen geboten /9/. Für den Bereich der Büroanwendungen hat die Object Management Group (OMG), ein Konsortium führender Unternehmen der Informationstechnologie, mit der Object Management Architecture (OMA) ein weit beachtetes Referenzmodell für objektorientierte Verarbeitung in verteilten, heterogenen, offenen Umgebungen definiert /6/, das die Basis für eine Reihe von Spezifikationen bildet. Ein Object Request Broker (ORB) ist dabei eine zentrale Komponente, die es Objekten erlaubt, in der verteilten Umgebung transparent Aufrufe an andere Objekte zu tätigen bzw. von diesen entgegenzunehmen. Die Common Object Request Broker Architecture (CORBA) Spezifikation identifiziert die Komponenten eines ORBs und spezifiziert deren Programmierschnittstellen /7/.

Das DIRECT-Projekt (DIstributed REaltime ConTrol) an der Fachhochschule Wiesbaden versucht, die Prinzipien der objektorientierten verteilten Programmierung entsprechend dem OMA/CORBA-Referenzmodell der OMG für die Entwicklung verteilter Anwendungen in der Automatisierungstechnik zu nutzen /3/, /4/. In einer Zusammenarbeit zwischen der Fachhochschule Wiesbaden und der Klöckner-Moeller GmbH, Bonn, werden diese Ansätze derzeit auf die Integration von SPS-gesteuerten Teilsystemen in komplexe objektorientierte Automatisierungsanwendungen sowie auf die Programmierung vernetzter SPS-Systeme übertragen.

Im folgenden Kapitel wird ein Überblick über die dem Projekt DIRECT zugrundeliegende Architektur eines komplexen, objektorientierten Automatisierungssystems und die angewendete Vorgehensweise zur Anwendungsentwicklung gegeben. Danach wird die erfolgreich abgeschlossene Integration eines SPS-gesteuerten Teilsystems vorgestellt. Anschließend wird eine neue, auf objektorientierten Prinzipien basierende Methode zur Programmierung vernetzter SPS-Systeme beschrieben, die im Rahmen der Zusammenarbeit entwickelt wurde und die derzeit realisiert wird.

2. Die DIRECT-Architektur

Der Entwicklung einer Automatisierungsanwendung im Rahmen von DIRECT liegt das CORBA-Modell zugrunde [3]. Eine Anwendung wird durch eine Menge von kooperierenden Objekten modelliert und implementiert, die auf den verschiedenen Stellen des verteilten Systems residieren. Die Spezifikation von Objektschnittstellen geschieht in der C++-ähnlichen CORBA Interface Definition Language (IDL). Vererbung von Schnittstelleneigenschaften ist in CORBA IDL möglich. Objektimplementierungen erfolgen in C++ und C. Aus der Sicht des Anwendungsentwicklers erfolgt ein Aufruf einer Operation eines entfernten oder lokalen Objekts gleichermaßen durch einen üblichen Prozeduraufruf. Die Verteiltheit des Systems ist für den Anwendungsentwickler damit weitgehend transparent und spielt primär bei der Konfigurierung und Initialisierung des Systems eine wesentliche Rolle.

Schnittstellenbeschreibungen abstrahieren zunächst von der Implementierung der angebotenen Funktionalität. So sind z.B. für den Nutzer einer Laborgeräteschnittstelle die sehr vielfältigen unterlagerten physikalischen Schnittstellen der Laborgeräte nicht mehr von Interesse. Mittels Vererbung von Schnittstellen können Familien von Geräteklassen dargestellt werden. Spezielle Eigenschaften von Geräten bestimmter Hersteller lassen sich z.B. in abgeleiteten Klassen isolieren, während die gemeinsame Funktionalität der Geräteklasse in einer allgemeinen Schnittstelle zusammengefaßt wird. Vererbung erlaubt auch die Wiederverwendbarkeit von Implementierungsklassen.

Es wird angenommen, daß alle Komponenten eines komplexen Automatisierungssystems durch ein zweistufiges Kommunikationsnetzwerk verbunden sind (vgl. Abb. 1). Ein LAN (z.B. Ethernet) verbindet Server-Rechner und Bedienplätze und bietet die Möglichkeit zur Integration des Systems in höhere Schichten der CIM-

Architektur. Alle Geräte sowie komplexe SPS-gesteuerte Subsysteme werden über μ Controller-basierte Frontends an echtzeitfähige Feldbus-Segmente angebunden, welche selbst über Gateways mit dem LAN verbunden sind. Als Feldbus wird derzeit in DIRECT der CAN-Bus (Controller Area Network) eingesetzt, ein Multicast-fähiges Feldbussystem, das ursprünglich für den Automobilbereich entwickelt wurde. Alle Knoten werden derzeit unter Verwendung des PXROS Echtzeit-Kerns (HighTec, Saarbrücken) betrieben. Insgesamt liegt damit eine verteilte Systemarchitektur vor.

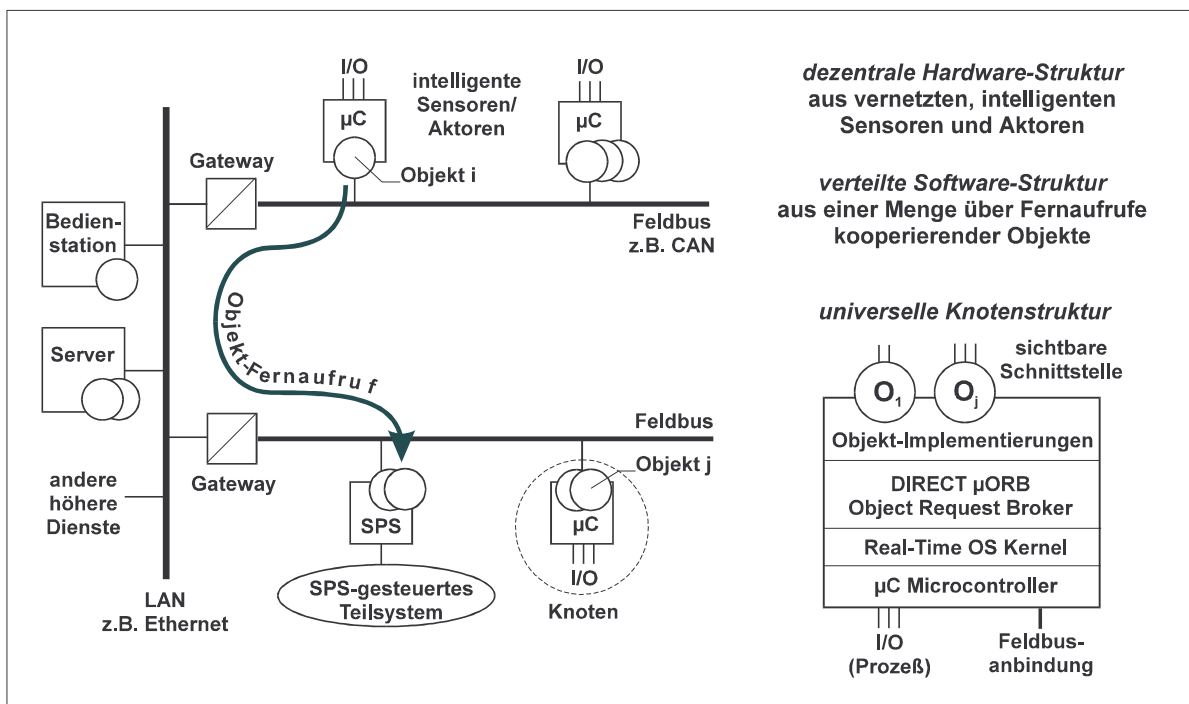


Abb. 1: Die DIRECT-Architektur

Während für die LAN-basierte Ebene bereits mehrere CORBA-konforme Produkte verschiedener Hersteller existieren, sind objektorientierte Ansätze für die Echtzeit-relevante Feldbus-Ebene weitgehend neu. Im Rahmen von DIRECT wurde ein zur Erzielung von Echtzeiteigenschaften vereinfachter, nach CORBA-Prinzipien arbeitender Object Request Broker entworfen und als Middleware-Komponente auf einem Echtzeit-Betriebssystemkern implementiert. Da der Object Request Broker auch auf kleinen μ Controller-basierten Systemen lauffähig ist, wird er als μ ORB bezeichnet. Er ermöglicht es, daß alle relevanten Komponenten einer Automatisierungsanwendung als intelligente Objekte erscheinen. Objekte leben dabei, ev. zu mehreren, in Tasks des unterlagerten Echtzeitkerns. Die Kooperation zwischen den Objekten geschieht durch den Aufruf von Operationen, die durch den μ ORB im Netzwerk vermittelt werden. Ein

Aufrufer einer Operation wird im folgenden auch als Client bezeichnet, eine dienstbringende Instanz auch als Server. Zur Durchführung von Objektfernaufrufen enthält der μ ORB ein sogenanntes Remote Procedure Call (RPC)-Protokoll. Details der Implementierung sind in /1/ und /3/ beschrieben.

3. SPS-gesteuerte Teilanwendungen

Ein wichtiger Problembereich bei der Entwicklung komplexer Automatisierungsanwendungen ist die Integration von SPS-gesteuerten Teilsystemen. In diesem Kontext prallen häufig die unterschiedlichen, durch die verschiedenen Verarbeitungsmodelle geprägten Sichtweisen der Beteiligten aufeinander. In DIRECT wird versucht, eine gemeinsame Basis durch CORBA-basierte "Wrapper" zu finden. Ein Wrapper verpackt eine zyklisch ablaufende SPS-Anwendung und verleiht ihr eine objektorientierte Hülle, die das SPS-Subsystem nach außen, etwa zum Zwecke der Parametrisierung oder Modusänderung, als eine Menge von Objekten erscheinen läßt. Die Integrierbarkeit der Teilanwendung in die Sichtweise der übergeordneten objektorientierten Anwendung ist dabei vollständig gegeben. Die durch die SPS realisierten Objekte können sowohl eine Client-Rolle wie auch eine Server-Rolle einnehmen.

Die Integration einer Klöckner-Moeller SPS PS 416 in das CAN-Bus-basierte Experimentalsystem wurde erfolgreich abgeschlossen. Die Vorgehensweise ist in Abb. 2 dargestellt. Die SPS wird um eine zusätzliche Prozessor-Karte erweitert. Auf ihr läuft der μ ORB wie auf den anderen CAN-Knoten im System. Auf ihm residieren Objekte, die aus der Sicht entfernter Klienten als dienstbringende Objekte erscheinen, in Wirklichkeit aber nur sogenannte Proxy-Objekte sind, die einen eingehenden Aufruf über den Rückwandbus der SPS an zugehörige, die Objekte tatsächlich realisierende Funktionsbaustein-Instanzen in der eigentlichen SPS-CPU weitergeben. Die Proxy-Objekte bilden zusammen mit diesen Instanzen den o.a. Wrapper.

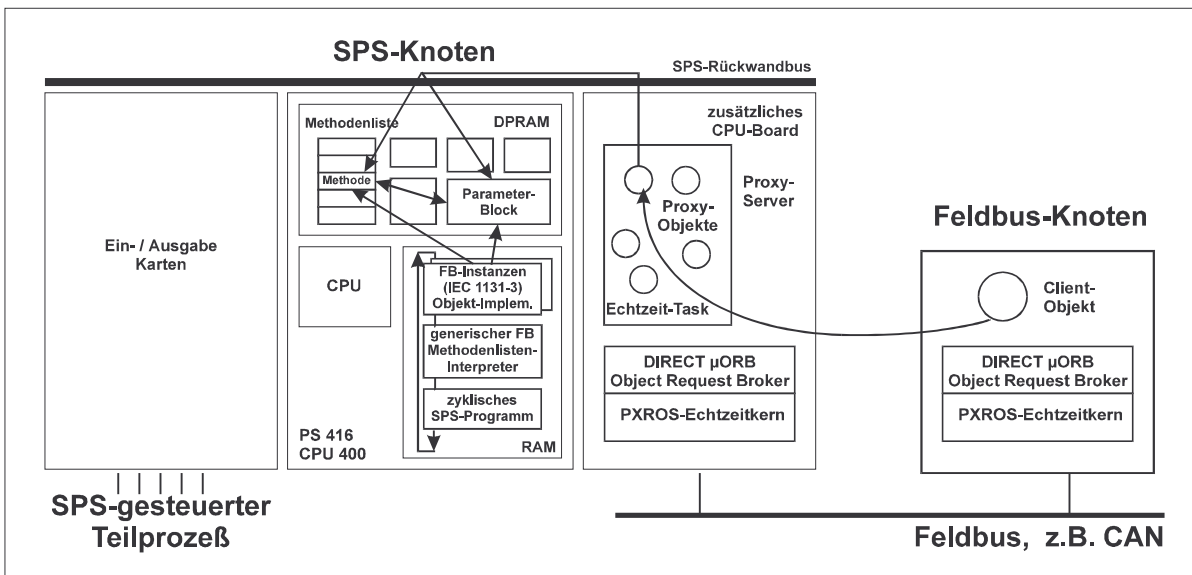


Abb. 2: SPS-Integration

Genauer wird die Kommunikation der Zusatz-Karte mit der SPS-CPU über ein spezielles dual-ported RAM (DPRAM) abgewickelt, das für beide Teilnehmer zugreifbar ist. Es enthält eine sogenannte Objektliste mit einem Eintrag für jedes sichtbare Objekt. Jeder Eintrag enthält ein sogenanntes Methoden-Feld, das die auszuführende Operation identifiziert. Weiter existiert auf SPS-Seite ein generisches Funktionsmodul, das für die Bearbeitung aller rechenwilligen Methodenaufrufe im Rahmen der zyklischen Verarbeitung der SPS zuständig ist.

Im Falle eines zur Laufzeit bei einem Proxy-Objekt eingehenden Aufrufs einer durchzuführenden Operation schreibt das Proxy-Objekt zuerst alle Parameter in den hierzu vorgesehenen Speicherbereich im DPRAM. Anschließend wird die Verarbeitung durch das Schreiben der aufzurufenden Methode in die Objektliste freigegeben. Im Rahmen der zyklischen Verarbeitung der SPS wird der generische Funktionsbaustein abgearbeitet, der die Objektliste auf anstehende Aufrufe durcharbeitet und im Bedarfsfall die den Objekten zugehörigen Funktionsbausteine aktiviert.

4. Objektorientierte Programmierung vernetzter SPS-Systeme

4.1 Überblick

Der Entwurf einer dezentralen objektorientierten Automatisierungsanwendung erfolgt in drei Stufen, die durch die DIRECT Struktur geprägt sind. Zunächst wird eine Applikation als eine Menge von interagierenden Objekten modelliert. Die Objekte können dabei einen direkten Bezug zur Prozeß-Peripherie haben, wie z.B. Ventilinseln, Motoren oder Achsen, oder aber auch abstrakte Softwarekomponenten der Anwendung

mit bestimmten Eigenschaften repräsentieren (z.B. Temperaturzonenregelung). Das Ergebnis der Modellierung ist ein Geflecht voneinander abhängiger Objekte. Jeder Objekttyp wird anschließend durch einen Funktionsbaustein, jedes konkrete Objekt durch eine Instanz eines Funktionsbausteins im Sinne der IEC 1131-3 repräsentiert. Den Beziehungen von Objekten untereinander entsprechen Aufrufe zwischen den zugeordneten Funktionsbausteinen. Bei der Programmierung der Objekte muß auf die spätere räumliche Verteilung noch keine Rücksicht genommen werden.

Durch eine Stationierungsentscheidung wird in der zweiten Stufe eine Zerlegung der Gesamtanwendung und eine Zuordnung der entsprechenden Funktionsbaustein-Instanzen auf die einzelnen Automatisierungsgeräte vorgenommen. Jede Beziehung zwischen verteilten Objekten erfordert vom Laufzeitsystem entsprechende Unterstützung. Kommunikationsvorgänge die durch die Zerlegung notwendig gewordenen sind, werden vom System automatisch erbracht. Existieren zwischen zwei Objekten dagegen nur lokale Beziehungen, d.h. liegen beide Objekte auf demselben Automatisierungsknoten, sind weitere Aktionen nicht notwendig.

Durch eine dritte Stufe werden die IEC 1131-Teilanwendungen in entsprechende Automatisierungsgeräte geladen und dort zur Ausführung gebracht. Der dazu notwendige synchronisierte Austausch von Daten wird durch den in das SPS-Betriebssystem integrierten µORB übernommen. Im folgenden wird die Vorgehensweise detaillierter dargestellt.

4.2 Realisierung des IEC-Modells bei Klöckner-Moeller Steuerungen

Die Realisierung der IEC 1131-3 (Anweisungsliste AWL) für Klöckner-Moeller Steuerungen sieht vor, beliebig viele, nur durch den Speicherplatz beschränkte, typisierte Einheiten, die Instanzen genannt werden, aus einer abstrakten Spezifikation, der Funktionsbaustein (FB)-Typdefinition, zu erzeugen. Dieser Vorgang wird Instanziierung genannt. Eine Instanz kann damit als Inkarnation eines abstrakten Datentyps, bestehend aus einer Menge von Daten und aus einer Anzahl von erlaubten Operationen auf diesen Daten angesehen werden. Die funktionale Schnittstelle des Funktionsbausteins wird durch die Menge der Operationen eines Typs definiert.

Klöckner-Moeller verwendet die Möglichkeit des Code-Sharing. Dieser Ansatz zeichnet sich dadurch aus, daß alle Instanzen einer FB-Typdefinition den gleichen Code verwenden. Strukturell teilt sich eine Instanz auf in einen Ein-/Ausgabebereich, den ein Aufrufer kennt und modifizieren kann, und einen lokalen, nur innerhalb der Operationen sichtbaren Datenbereich. Durch die Parametrierung des Eingabebereichs einer Instanz definiert ein Aufrufer die auszuführende Operation inklusive der

Übergabeparameter. Ein Objekt einer verteilten, objektorientierten Anwendung wird damit durch genau eine Instanz repräsentiert. Durch die SPS typische Programmverarbeitung wird der Code eines Funktionsbausteins ein- oder mehrfach im Programm mit entsprechenden Instanzen aufgerufen und abgearbeitet.

4.3 Implementierungskonzept

Jedem entfernten Objekt ist auf dem eigenen Knoten ein lokales Stub-Objekt zugeordnet, welches auch mit Stellvertreter-Objekt bezeichnet wird und sich durch eine zum eigentlichen Objekt identische Schnittstelle auszeichnet. Ein potentieller Benutzer kann daher nicht zwischen einem lokalen und einem entfernten Objekt unterscheiden. Die Aufgabe eines Stub-Objektes ist es, einen Methodenaufruf entgegenzunehmen, dabei die Argumente geeignet in eine Nachrichten-orientierte Form zu transformieren und diese über das unterliegende Netzwerk an den Knoten zu versenden, der das gewünschte Zielobjekt beherbergt. Dort werden die Nachricht entgegengenommen, die Argumente ausgepackt und ein lokaler Methodenaufruf zu dem gewünschten Zielobjekt durchgeführt. Mittels der Parameter und den aktuellen Attributen kann nun die eigentliche programmierte Aktion ausgeführt werden. Die dabei entstehenden zugehörigen Ergebnisse werden auf dem umgekehrten Weg zurück zum Aufrufer transportiert.

Das auf einem entfernten Knoten lokalisierte Zielobjekt enthält den unveränderten Code des ursprünglich definierten Funktionsbausteins. Der Stub-Code eines Stellvertreter-Objektes kann aufgrund der Homogenität der eingesetzten SPSen, in Vereinfachung zu allgemeinen RPC-Systemen, generisch realisiert werden. Alle zur Übertragung in Frage kommenden Instanzen werden vom System als eine Byte-Folge betrachtet. Nur die physikalische Adresse und die Länge einer Instanz sind von gesondertem Interesse. Beides ist dem Stub-Code bekannt bzw. kann vom Laufzeitsystem erfragt werden. Die innere Struktur spielt aufgrund der angenommenen Homogenität keine Rolle.

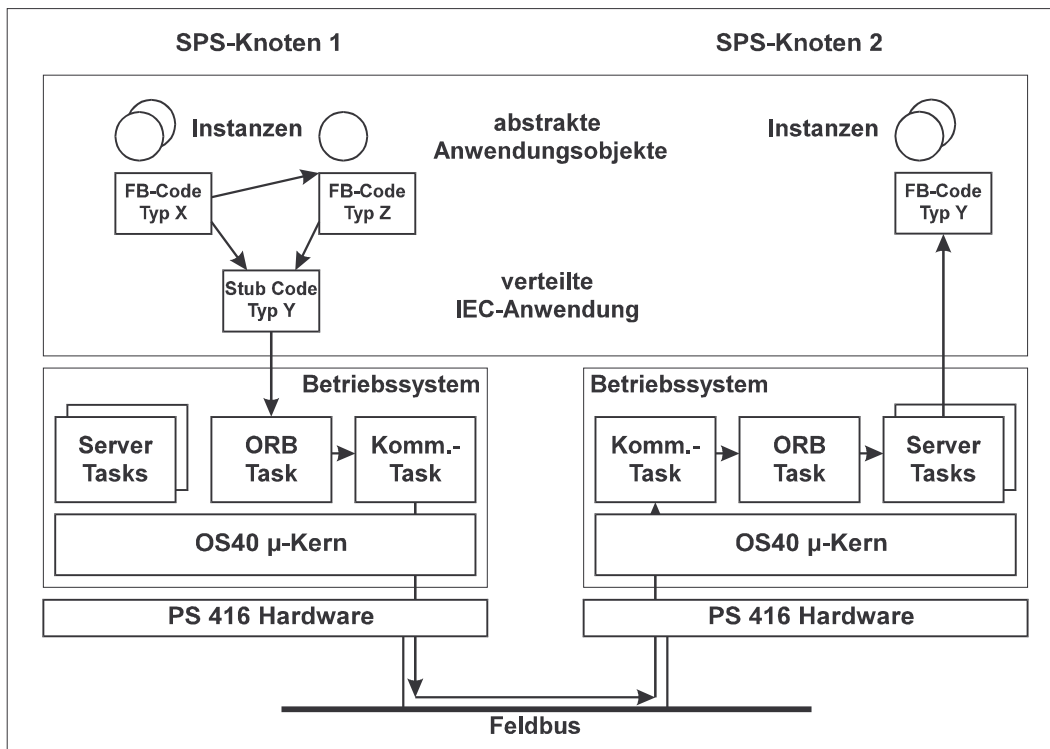


Abb. 3: Implementierungskonzept für die Klöckner-Moeller SPS PS 416

Für das Klöckner-Moeller SPS-Betriebssystem OS40 ist die Erweiterung um eine Variante des DIRECT µORB vorgesehen. Wie oben beschrieben werden hierdurch die Aufrufe aller Client-Stubs entgegengenommen und unter Nutzung des unterlagerten Feldbusses auf entfernte SPS-Knoten vermittelt. Der Gesamtzusammenhang einer solchen, für den Anwendungsprogrammierer transparenten Objektverteilung ist in Abb. 3 dargestellt.

5. Zusammenfassung

In diesem Beitrag wurde ein Konzept zur Integration speicherprogrammierbarer Steuerungen in verteilte, objektorientierte Automatisierungsanwendungen beschrieben. Ausgehend von der heute nur ungenügend unterstützten Anbindung von SPS-Systemen in offene Kommunikationsstrukturen basiert die vorgestellte Vorgehensweise auf einem verteilten, objektorientierten Programmier-Paradigma entsprechend dem OMA/CORBA-Referenzmodell der OMG. Die im Rahmen einer Kooperation zwischen dem DIRECT-Projekt der Fachhochschule Wiesbaden und der Klöckner-Moeller GmbH, Bonn, abgeschlossene Integration eines SPS-gesteuerten Teilsystems wurde erläutert. Ferner wurde eine neue dreistufige Methode zur objektorientierten Programmierung vernetzter

SPS-Systeme beschrieben, deren Realisierung in der weiteren Zusammenarbeit verfolgt wird.

Literatur

- /1/ Bauer, A.; Remždios, O.: "Objektorientierte Verarbeitung in verteilten Automatisierungssystemen", Diplomarbeit, Fachhochschule Wiesbaden, 1995
- /2/ Coad, P.; Yourdan, E.: "Object-Oriented Analysis" (Second Ed.) and "Object-Oriented Design", Prentice-Hall, 1991
- /3/ Kröger, R.; Bauer, A.; Remždios, O.; Thoss, M.: "Objektorientierte Programmierung verteilter Echtzeitanwendungen", Echtzeit'95, Karlsruhe, 20.-22. Juni 1995, Network GmbH, 1995
- /4/ Kroeger, R.: "Using Object-Oriented Standards for Developing Realtime Control Applications", Proc. 3rd IFAC/IFIP Workshop on Algorithms and Architectures for Real-Time Control (AARTC'95), Ostend, May 31 - June 2, 1995
- /5/ Neumann, P.; Grötsch, E.E.; Lubkoll, C.; Simon, R.: "SPS-Standard: IEC 1131-Programmierung in verteilten Automatisierungssystemen", Oldenbourg-Verlag, 1995
- /6/ OMG: "Object Management Architecture Guide", Object Management Group, Framingham, MA, USA, 1992
- /7/ OMG: "The Common Object Request Broker, Architecture and Specification, Rev.1.2", Object Management Group, Framingham, MA, USA, 1993
- /8/ Rumbaugh, J. et al.: "Object-Oriented Modeling and Design", Prentice-Hall, 1991
- /9/ Selic, B.; Gullekson, G.; Ward P. T.: "Real-Time Object-Oriented Modeling", J. Wiley & Sons, New York, 1994