(12) NACH DEM VERTRAG ÜBER DIE INTERNATIONALE ZUSAMMENARBEIT AUF DEM GEBIET DES PATENTWESENS (PCT) VERÖFFENTLICHTE INTERNATIONALE ANMELDUNG

(19) Weltorganisation für geistiges Eigentum
Internationales Büro

(43) Internationales Veröffentlichungsdatum
19. Juni 2003 (19.06.2003)

PCT

(10) Internationale Veröffentlichungsnummer
WO 03/050716 A2

(51) Internationale Patentklassifikation[7]: G06F 17/30

(21) Internationales Aktenzeichen: PCT/DE02/04375

(22) Internationales Anmeldedatum:
28. November 2002 (28.11.2002)

(25) Einreichungssprache: Deutsch

(26) Veröffentlichungssprache: Deutsch

(30) Angaben zur Priorität:
101 61 111.0 12. Dezember 2001 (12.12.2001) DE

(71) Anmelder *(für alle Bestimmungsstaaten mit Ausnahme von US)*: **SIEMENS AKTIENGESELLSCHAFT** [DE/DE]; Wittelsbacherplatz 2, 80333 München (DE).

(72) Erfinder; und

(75) Erfinder/Anmelder *(nur für US)*: **LANGKAFEL, Dirk** [DE/DE]; Bergstr. 15a, 91090 Effeltrich (DE). **THURNER, Elmar** [AT/DE]; Gneisenaustr. 12, 90491 Nürnberg (DE).

(74) Gemeinsamer Vertreter: **SIEMENS AKTIENGESELLSCHAFT**; Postfach 22 16 34, 80506 München (DE).

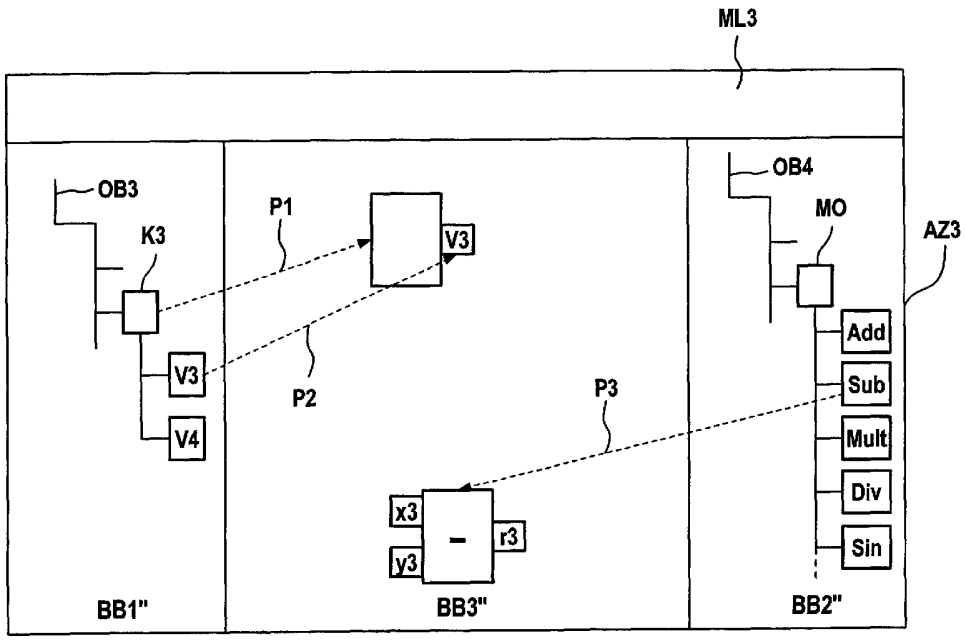
(81) Bestimmungsstaat *(national)*: US.

(84) Bestimmungsstaaten *(regional)*: europäisches Patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, SK, TR).

[Fortsetzung auf der nächsten Seite]

(54) Title: SYSTEM AND METHOD FOR PROJECTING TRANSFORMATIONS OF OBJECT TREES

(54) Bezeichnung: SYSTEM UND VERFAHREN ZUR PROJEKTIERUNG VON TRANSFORMATIONEN VON OBJEKTBÄUMEN



(57) Abstract: A system and method for projecting the transformation of object trees (OB1 - OB4), especially in MES systems, wherein the source object tree (QB) and target object tree (ZB) are represented in a common meta-object model of a software system, especially a framework (IF). The source object tree (QB) is transformed into the target object tree (ZB) by adjustment. Transformation occurs directly in the object of the meta-object model (component), thereby enabling communication between connected applications by means of pure data exchange.

[Fortsetzung auf der nächsten Seite]

WO 03/050716 A2

Erklärungen gemäß Regel 4.17:

— *hinsichtlich der Berechtigung des Anmelders, ein Patent zu beantragen und zu erhalten (Regel 4.17 Ziffer ii) für die folgenden Bestimmungsstaaten europäisches Patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, SK, TR)*

— *Erfindererklärung (Regel 4.17 Ziffer iv) nur für US*

Veröffentlicht:

— *ohne internationalen Recherchenbericht und erneut zu veröffentlichen nach Erhalt des Berichts*

Zur Erklärung der Zweibuchstaben-Codes und der anderen Abkürzungen wird auf die Erklärungen ("Guidance Notes on Codes and Abbreviations") am Anfang jeder regulären Ausgabe der PCT-Gazette verwiesen.

(57) Zusammenfassung: System und Verfahren zur Projektierung der Transformation von Objektbäumen (OB1 - OB4), insbesondere in MES-Systemen, wobei die Quell- (QB) und Zielobjektbäume (ZB) in einem gemeinsamen Metaobjektmodell eines Softwaresystems, insbesondere eines Frameworks (IF), repräsentiert sind. Der Quellobjektbaum (QB) wird durch Regeln in den Zielobjektbaum (ZB) transformiert. Die Transformation findet direkt auf den Objekten des Metaob-jektmodells (Component) statt. Dadurch wird eine Kommunikati-on über einen reinen Datenaustausch zwischen angekoppelten Applikationen ermöglicht.

Beschreibung

System und Verfahren zur Projektierung von Transformationen von Objektbäumen

Die Erfindung betrifft ein System und ein Verfahren zur Projektierung von Transformationen von Objektbäumen, wobei die Quell- und Zielobjektbäume in einem gemeinsamen Metaobjektmodell eines Softwaresystems repräsentiert sind und wobei das Softwaresystem auf mindestens einer Rechnereinheit gespeichert ist.

Ferner betrifft die Erfindung ein entsprechendes Computerprogramm, ein Computerprogrammprodukt sowie eine Datenverarbeitungseinrichtung.

Wenn in einem Softwaresystem heterogene Applikationen, z.B. MES-Applikationen miteinander verbunden werden, dann sind die Objekte bzw. Objektbäume der jeweiligen Applikationen, im Metamodell des Softwaresystems meistens in unterschiedlicher Weise repräsentiert, auch wenn die Objekte bzw. Objektbäume semantisch übereinstimmen. Durch diese unterschiedliche Repräsentation wird die Kommunikation zwischen den Applikationen erschwert.

Aus "Software für die Automatisierung - Transparenz über die Abläufe schaffen", Artikel von Dirk Kozian in Elektronik für die Automatisierung 11, 17.11.1999 ist bekannt, für die Automatisierung von Produktions- bzw. Fertigungsabläufen so genannte Manufacturing Execution Systems (MES) einzusetzen. Diese Systeme integrieren die Automatisierungsebene (Controls) mit den ERP-Systemen (ERP: Enterprise Resource Planning) der Unternehmensleitebene. Manufacturing Execution Systems sind Systeme, die z.B. Informationen zur Optimierung von Produktionsabläufen bereitstellen. Zum einen müssen die Manufacturing Execution Systems die groben Planungsdaten der ERP-Systeme um anlagenspezifische und aktuelle Feinplanungs-

daten ergänzen und diese entsprechend an die unterlagerte Automatisierungsebene weiterleiten, zum anderen haben sie die Aufgabe, aus der Automatisierungsebene produktionsrelevante Informationen zu übernehmen, diese aufzubereiten und an die Unternehmensleitebene weiterzumelden. MES-Systeme erfüllen somit die Aufgabe einer vertikalen Integration zwischen der Unternehmensleitebene und der Automatisierungsebene. Typische Einzelaufgaben von MES-Systemen sind Enterprise Asset Management, Maintenance Management, Information Management, Scheduling, Dispatching und Trace & Track. Diese Aufgaben werden jeweils von MES-Komponenten bzw. MES-Applikationen ausgeführt.

Aufgrund der software- und datentechnischen Heterogenität der MES-Applikationen lassen sich diese aber nur sehr schwer in ein gemeinsames System bzw. Projekt integrieren und der Datenaustausch zwischen diesen Applikationen ist nur mit Aufwand möglich.

Aus "Massive Wiederverwendung: Konzepte, Techniken und Organisation", Artikel von Ulrich Lindner in OBJEKTspektrum 1/96, S. 10 - 17, ist es bekannt, Softwarekomponenten durch so genannte Adapter oder durch Wrapping (Verpacken) in ein Softwaresystem zu integrieren. Ziel ist es dabei die Wiederverwendbarkeit von Softwarekomponenten zu erhöhen.

Aus "XML - Schlüsseltechnologie für Softwarearchitekturen", Artikel von Alexander Jung in OBJEKTspektrum 1/2001, S. 71 - 74, ist es bekannt, XML (eXtensible Markup Language) für den Datenaustausch zwischen Fremdsystemen einzusetzen und dabei Transformationen vorzunehmen. Im Rahmen der XML-Familie ist ein Standardvorgehen zur Transformation von XML-Dokumenten definiert: XSL Transformations (XSL steht für Extensible Stylesheet Language). Mit XSL Transformations lassen sich auch Baumtransformationen beschreiben, aber nur, wenn die Objekte der Bäume im XML-Format vorliegen und das jeweilige XML-Format einem Anwender bekannt ist. Wenn ein Anwender eine

Transformation eines Objektbaumes durchführen will, benötigt er die zugehörige Repräsentation der Objekte im XML-Format und er muss auf der XML-Ebene die Transformation definieren. Das bedeutet Aufwand, denn der Anwender muss die entsprechenden XML-Formate der Objekte erst besorgen.

Aus "Konfigurieren statt Programmieren - Die Empfehlungen des ZVEI-Arbeitskreises Systemaspekte, 1. Teil", Artikel in Elektronik 8/1994, S. 112 - 117, ist es bekannt, die Software Automatisierungssysteme zu Konfigurieren bzw. zu Projektieren. Es sind aber keine konfigurierbaren Systeme für branchenübergreifende Problemstellungen vorhanden.

Aufgabe der zugrundeliegenden Erfindung ist es ein System und ein Verfahren zur Projektierung von Transformationen von Objektbäumen zur Verfügung zu stellen, dass es ermöglicht, wobei die Definition der Transformationen für den Anwender komfortabel und einfach ist.

Gemäß der Erfindung wird die gestellte Aufgabe für ein System zur Projektierung von Transformationen von Objektbäumen durch die Merkmale von Anspruch 1 gelöst. Die Transformation der Objekte, bzw. der Objektbäume findet dabei in der Domänenwelt des Anwenders statt, d.h. der Anwender kann vom zugrunde liegenden Format der Objekte abstrahieren. Ein weiterer Vorteil liegt darin, durch die Transformation Objektstrukturen in eine Form gebracht werden, die für Empfänger (z.B. andere Applikationen) dieser Objektstrukturen jeweils verständlich sind. Es liegt nämlich oft der Fall vor, dass in einem gemeinsamen Metaobjektmodell, semantisch gleiche Objekte (z.B. ein Produktionsauftrag) unterschiedlich repräsentiert sind. Durch die Transformationen werden für Objekte mit gleicher Semantik auch einheitliche Repräsentationen geschaffen. Dadurch wird eine Kommunikation zwischen unterschiedlichen Applikationen über einen reinen Datenaustausch ermöglicht. Ein Anwender kann sehr komfortabel die Projektierung der Trans-

formationen vornehmen durch grafisches Verschalten (Verbinden) der Symbole. Durch das Verschalten werden auch weitere Regeln erzeugt, die weiterverarbeitet bzw. wiederverwendet werden können. Durch die verwendete grafische Oberfläche sind die Transformationen und die verwendeten Regeln für einen Anwender transparent.

Eine erste vorteilhafte Ausgestaltung der vorliegenden Erfindung für ein System liegt darin, dass die Transformation in der für einen Anwender zugrundeliegenden Domäne durchführbar ist. Erfolgt die Transformation direkt auf der Domänenebene des Anwenders, kann der Anwender von den internen Formaten der Objekte bzw. Objektbäume abstrahieren. Die Effektivität eines Anwenders wird dadurch erhöht, da er sich in seiner bekannten Welt (bekannte Nomenklatur, bekannte Objekte etc.) bei der Erstellung von Lösungen aufhält. Ein Anwender muss somit die zu transformierenden Objekte nicht auf der Ebene ihres internen Formats transformieren und danach wieder in die Domänenwelt zurückübersetzen.

Eine weitere vorteilhafte Ausgestaltung der vorliegenden Erfindung für ein System liegt darin, dass die Regeln als Objekte des Softwaresystems repräsentiert sind. Dadurch erscheinen die Objekte einem Anwender als Teil des Softwaresystems. Ein Anwender kann die Regeln auch wie Objekte handhaben, z.B. grafisch verknüpfen.

Eine weitere vorteilhafte Ausgestaltung der vorliegenden Erfindung für ein System liegt darin, dass das Positionieren und Verschalten der Knoten und/oder der Operatoren und/oder der Regeln via drag&drop durch Eingabehilfsmittel erfolgt. Dadurch wird ein Anwender in seiner Arbeitsweise unterstützt und seine Effektivität erhöht.

Eine weitere vorteilhafte Ausgestaltung der vorliegenden Erfindung für ein System liegt darin, dass von einem Anwender weitere Operatoren und/oder Regeln zur Transformation defi-

nierbar sind. Dadurch wird die Flexibilität für einen Anwender erhöht und ein Anwender kann sich Regeln definieren, die für die zugrundeliegenden Strukturen und Problemstellungen angemessen sind.

Eine weitere vorteilhafte Ausgestaltung der vorliegenden Erfindung für ein System liegt darin, dass das Softwaresystem durch mindestens ein Rahmenprogramm realisiert ist. Dadurch liegen die Vorteile eines Rahmenprogramms (Framework) vor. Ein Framework definiert die Architektur einer Familie von Softwaresystemen, stellt die grundlegenden Bausteine zur Erstellung dieser Systeme zur Verfügung und legt deren Zusammenspiel fest. Durch Frameworks werden Entwicklungszeit und Entwicklungskosten eingespart.

Eine weitere vorteilhafte Ausgestaltung der vorliegenden Erfindung für ein System liegt darin, dass Quell- und Zielobjektbaum durch die Objektmodelle von angekoppelten Adaptern vorgegeben sind. Adapter sind eine Abstraktionsstufe höher als Wrapper. Sie bieten eine einheitliche Sicht auf angekoppelte Applikationen. Ein Adapter bietet Funktionalität, um die anzukoppelnde Komponente zu starten, zu bedienen etc. Ein Adapter entspricht in der Sprache der Design Patterns einer "Facade". Ein Wrapper dagegen bildet das API (Application Programmable Interface) einer Fremdkomponente (z.B. eine MES-Applikation eines Drittanbieters) in das Objektmodell eines Softwaresystems ab. So wird z.B. aus einer Methode des API der Fremdkomponente eine Methode des Softwaresystems bzw. aus einem Integer-Datentyp des API der Fremdkomponente wird ein Integer-Datentyp des Softwaresystems, usw. Durch Adapter werden die Objekte der zu integrierenden Applikationen in das Metaobjektmodell des Softwaresystems abgebildet und im Metaobjektmodell als Quell- bzw. Zielobjektbaum repräsentiert. Wenn nun die Adapter die Struktur dieser Bäume vorgeben, kann eine nachfolgende Transformation von Quell- auf Zielobjektbaum sehr leicht erfolgen, evtl. sogar automatisiert.

Gemäß der Erfindung wird die gestellte Aufgabe für ein Verfahren zur Projektierung von Transformationen von Objektbäumen durch die Merkmale von Anspruch 8 gelöst. Die Transformation der Objekte, bzw. der Objektbäume findet dabei in der Domänenwelt des Anwenders statt, d.h. der Anwender kann vom zugrunde liegenden Format der Objekte abstrahieren. Ein weiterer Vorteil liegt darin, durch die Transformation Objektstrukturen in eine Form gebracht werden, die für Empfänger (z.B. andere Applikationen) dieser Objektstrukturen jeweils verständlich sind. Es liegt nämlich oft der Fall vor, dass in einem gemeinsamen Metaobjektmodell, semantisch gleiche Objekte (z.B. ein Produktionsauftrag) unterschiedlich repräsentiert sind. Durch die Transformationen werden für Objekte mit gleicher Semantik auch einheitliche Repräsentationen geschaffen. Dadurch wird eine Kommunikation zwischen unterschiedlichen Applikationen über einen reinen Datenaustausch ermöglicht. Ein Anwender kann sehr komfortabel die Projektierung der Transformationen vornehmen durch grafisches Verschalten (Verbinden) der Symbole. Durch das Verschalten werden auch weitere Regeln erzeugt, die weiterverarbeitet bzw. wiederverwendet werden können. Durch die verwendete grafische Oberfläche sind die Transformationen und die verwendeten Regeln für einen Anwender transparent.

Eine erste vorteilhafte Ausgestaltung der vorliegenden Erfindung für ein Verfahren liegt darin, dass die Transformation in der für ein Anwender zugrundeliegenden Domäne durchgeführt wird. Erfolgt die Transformation direkt auf der Domänenebene des Anwenders, kann der Anwender von den internen Formaten der Objekte bzw. Objektbäume abstrahieren. Die Effektivität eines Anwenders wird dadurch erhöht, da er sich in seiner bekannten Welt (bekannte Nomenklatur, bekannte Objekte etc.) bei der Erstellung von Lösungen aufhält. Ein Anwender muss somit die zu transformierenden Objekte nicht auf der Ebene ihres internen Formats transformieren und danach wieder in die Domänenwelt zurückübersetzen.

Eine weitere vorteilhafte Ausgestaltung der vorliegenden Erfindung für ein Verfahren liegt darin, dass die Regeln als Objekte des Softwaresystems repräsentiert werden. Dadurch erscheinen die Objekte einem Anwender als Teil des Softwaresystems. Ein Anwender kann die Regeln auch wie Objekte handhaben, z.B. grafisch verknüpfen.

Eine weitere vorteilhafte Ausgestaltung der vorliegenden Erfindung für ein Verfahren liegt darin, dass das Positionieren und Verschalten der Knoten und/oder der Operatoren und/oder der Regeln via drag&drop durch Eingabehilfsmittel durchgeführt wird. Dadurch wird ein Anwender in seiner Arbeitsweise unterstützt und seine Effektivität erhöht.

Eine weitere vorteilhafte Ausgestaltung der vorliegenden Erfindung für ein Verfahren liegt darin, dass von einem Anwender weitere Operatoren und/oder Regeln zur Transformation definiert werden. Dadurch wird die Flexibilität für einen Anwender erhöht und ein Anwender kann sich Regeln definieren, die für die zugrundeliegenden Strukturen und Problemstellungen angemessen sind.

Eine weitere vorteilhafte Ausgestaltung der vorliegenden Erfindung für ein Verfahren liegt darin, dass das Softwaresystem durch mindestens ein Rahmenprogramm realisiert wird. Dadurch liegen die Vorteile eines Rahmenprogramms (Framework) vor. Ein Framework definiert die Architektur einer Familie von Softwaresystemen, stellt die grundlegenden Bausteine zur Erstellung dieser Systeme zur Verfügung und legt deren Zusammenspiel fest. Durch Frameworks werden Entwicklungszeit und Entwicklungskosten eingespart.

Eine weitere vorteilhafte Ausgestaltung der vorliegenden Erfindung für ein Verfahren liegt darin, dass Quell- und Zielobjektbaum durch die Objektmodelle von angekoppelten Adaptern vorgegeben werden. Adapter sind eine Abstraktionsstufe höher als Wrapper. Sie bieten eine einheitliche Sicht auf angekop-

pelte Applikationen. Ein Adapter bietet Funktionalität, um die anzukoppelnde Komponente zu starten, zu bedienen etc. Ein Adapter entspricht in der Sprache der Design Patterns einer "Facade". Ein Wrapper dagegen bildet das API (Application Programmable Interface) einer Fremdkomponente (z.B. eine MES-Applikation eines Drittanbieters) in das Objektmodell eines Softwaresystems ab. So wird z.B. aus einer Methode des API der Fremdkomponente eine Methode des Softwaresystems bzw. aus einem Integer-Datentyp des API der Fremdkomponente wird ein Integer-Datentyp des Softwaresystems, usw. Durch Adapter werden die Objekte der zu integrierenden Applikationen in das Metaobjektmodell des Softwaresystems abgebildet und im Metaobjektmodell als Quell- bzw. Zielobjektbaum repräsentiert. Wenn nun die Adapter die Struktur dieser Bäume vorgeben, kann eine nachfolgende Transformation von Quell- auf Zielobjektbaum sehr leicht erfolgen, evtl. sogar automatisiert.

Eine weitere vorteilhafte Ausgestaltung der vorliegenden Erfindung liegt darin, dass das erfindungsgemäße Verfahren durch ein Computerprogramm implementiert ist. Dadurch können eventuelle Modifizierungen bzw. Anpassungen leicht durchgeführt werden.

Eine weitere vorteilhafte Ausgestaltung der vorliegenden Erfindung liegt darin, dass das Computerprogramm für das erfindungsgemäße Verfahren auf einem Datenträger gespeichert ist. Dadurch ist das Verfahren bezüglich der Logistik und Verteilung leicht handhabbar.

Eine weitere vorteilhafte Ausgestaltung der vorliegenden Erfindung liegt darin, dass das Computerprogramm für das erfindungsgemäße Verfahren auf einer Datenverarbeitungseinrichtung installiert ist. Dadurch wird die Performance erhöht.

Weitere Vorteile und Details der Erfindung ergeben sich anhand der nun folgenden Beschreibung vorteilhafter Ausführungsbeispiele und in Verbindung mit den Figuren. Soweit in

unterschiedlichen Figuren Elemente mit gleichen Funktionalitäten beschrieben sind, sind diese mit gleichen Bezugszeichen gekennzeichnet. Es zeigen:

FIG 1 in einer Übersichtsdarstellung die "Unternehmenspyramide" mit drei Steuerungsebenen,

FIG 2 ein beispielhaftes Übersichtsbild mit Software- und Hardwareeinheiten für MES-Lösungen,

FIG 3 die zentrale Stellung des die Softwareapplikationen koppelnden Rahmenprogramms,

FIG 4 ein Beispiel für die Transformation von Objekten,

FIG 5 ein Beispiel für die Transformation von Objekten mit Hilfe von mathematischen Objekten,

FIG 6 ein Beispiel für die Projektierung einer Transformation,

FIG 7 den prinzipiellen Aufbau eines Adapters und

FIG 8 eine "Component" als Metaobjektmodell in UML-Notation.

Die Darstellung gemäß FIG 1 zeigt in einer Übersichtsdarstellung die drei Steuerungsebenen, wie sie üblicherweise in einem produzierenden bzw. fertigenden Unternehmen zu finden sind. Durch die Pyramidenform wird ausgedrückt, dass nach Oben hin eine Verdichtung der Informationen stattfindet. Die oberste Ebene ist die ERP-Ebene (Enterprise Resource Planning. Auf dieser Unternehmensleitebene werden üblicherweise die betriebswirtschaftlichen und vertrieblichen Aufgaben in einem Unternehmen durchgeführt (z.B. Finanzwesen, Vertriebswesen, Personalwesen, Berichterstattung). Aber auch produktionsanlagenübergreifende logistische Aufgaben (z.B. Auftrags-

und Materialverwaltung) werden auf dieser Ebene durchgeführt. Das System SAP R/3 ist ein ERP-System, das auf der Unternehmensleitebene sehr häufig verwendet wird.

Die unterste Ebene der Pyramide ist die Automatisierungs-Ebene (Controls). Auf dieser Ebene kommen üblicherweise speicherprogrammierbare Steuerungen (SPS) in Verbindung mit Visualisierungs- und Prozessleitsystemen (PLS) zum Einsatz. Die Antriebe, Aktoren und Sensoren der Produktions- und/oder Fertigungseinrichtungen stehen direkt mit den Systemen dieser Ebene in Verbindung.

Das Verbindungsglied zwischen der ERP-Ebene und der Automatisierungs-Ebene wird durch die MES-Ebene gebildet. Die Applikationen der MES-Ebene sorgen somit für eine vertikale Integration zwischen der ERP-Ebene und der Automatisierungs-Ebene. Die MES-Applikationen müssen einerseits die Grobplanungen der ERP-Systeme um produktionsanlagenspezifische Feinplanungen ergänzen und an die Systeme der Automatisierungs-Ebene weiterleiten, andererseits ist es Aufgabe der MES-Applikationen produktionsrelevante Daten der Automatisierungs-Ebene aufzunehmen, aufzubereiten und an die ERP-Ebene (Unternehmensleitebene) weiterzuleiten.

Typische MES-Applikationen sind u.a. Quality Management (QM), Maintenance Management (MM), Performance Analysis (PA), Process Management, Labor Management, Asset Management. Durch jeweils drei Punkte wird in FIG 1 ausgedrückt, dass sich auf einer Ebene weitere Elemente (Applikationen, Systeme etc.) befinden können.

MES-Systeme bzw. ERP-Systeme enthalten in der Regel ein so genanntes Laufzeitsystem zur zeitlichen Ablaufsteuerung der beteiligten Komponenten (Teilkomponenten, Module, Tasks, Prozesse des Betriebssystems etc.), sowie ein so genanntes Engineeringsystem zum Erstellen und Editieren von Programmen, welche zur Ausführung im Laufzeitsystem vorgesehen sind.

Die Darstellung gemäß FIG 2 zeigt ein beispielhaftes Übersichtsbild mit Software- und Hardwareeinheiten für MES-Lösungen. Die einzelnen MES-Applikationen A1 bis A3 sind über Adapter AD1 bis AD3 mit einem Rahmenprogramm (Framework) IF verbunden. Über einen bidirektionalen Informationspfad I1 ist ein Benutzerarbeitsplatz PIW1 mit dem Rahmenprogramm IF gekoppelt und kann somit die daran hängenden bzw. integrierten MES-Applikationen verwalten und überwachen. Der Benutzerarbeitsplatz PIW1 besteht üblicherweise aus einer Anzeigevorrichtung (Monitor, Display, etc.), einer Datenverarbeitungsanlage (z.B. PC) mit Prozessor und Speichereinrichtungen sowie Eingabeeinheiten (Keyboard, Maus, etc.). Die MES-Applikationen A1 bis A3 sowie das Rahmenprogramm IF können auf eigenen Datenverarbeitungseinheiten bzw. Prozessoren ablaufen, es ist aber auch möglich, dass sie auf der Datenverarbeitungseinheit des PIW1 ablaufen.

Über Adapter AD1 bis AD3 sind die jeweiligen MES-Applikationen A1 bis A3 mit dem Rahmenprogramm IF verbunden. Die Adapter sind somit die Koppelbausteine zwischen dem Rahmenprogramm IF und den Applikationen. Über die Adapter können somit auch an sich heterogene Applikationen miteinander verbunden werden, und durch die Integration mit dem Rahmenprogramm IF ist es möglich, zwischen den Applikationen zu kommunizieren und Datenaustausch zu betreiben. Die Adapter sind Softwaremodule, die Verbindungen zu verschiedenen Anwendungen bzw. Applikationen herstellen. In typischen Integrationsszenarien sind dies Integrationen zu Systemen aus der MES-, ERP-, SCADA- oder Controls-Welt. Ein Adapter bietet Funktionalität, um eine anzukoppelnde Komponente zu starten, zu bedienen, etc. Ein Adapter erlaubt den Zugriff auf Daten und Funktionen der anzukoppelnden Anwendung bzw. Applikation, stellt bestimmte Runtimedaten zur Verfügung und erlaubt das Laden von Engineeringinformationen aus der anzukoppelnden Anwendung bzw. Applikation. Adapter können sich hinsichtlich ihres Aufbaus und Umfangs unterscheiden. So können Adapter z.B. fest programmiert sein oder sie können konfiguriert bzw.

modelliert werden. Auch bezüglich der Möglichkeiten des Zugriffs auf die anzukoppelnde Applikation können sie sich unterscheiden, so können z.B. Adapter nur einen datentechnischen Zugang gestatten, es ist aber auch möglich, dass Adapter einen Zugang auf höherwertige Geschäftsabläufe zulassen. Beim Hochfahren werden die Adapter mit den hinterlegten Modellen und Zustandsinformationen geladen. Zur Laufzeit wird dann überprüft, ob und wie die unterschiedlichen integrierten Applikationen bzw. Anwendungen zusammenpassen. Über eine Visualisierungs- bzw. Monitoringkomponente ist es möglich, den Status eines Adapters abzufragen und am Benutzerarbeitsplatz PIW1 darzustellen (auch graphisch). Durch Adapter erhält das System und auch der Anwender eine standardisierte und einheitliche Sicht auf Applikationen (je nachdem, welche Abstraktionsebene bei den Adaptern vorhanden ist).

Eine weitere Möglichkeit Softwarekomponenten zu integrieren, ist es, Wrapper einzusetzen. Ein Wrapper bildet das API (Application Programmable Interface) einer Fremdkomponente (z.B. eine MES-Applikation) in das Objektmodell des Rahmenprogrammes ab. So wird z.B. aus einer Methode des API der Fremdkomponente eine Methode des Rahmenprogramms bzw. aus einem Integer-Datentyp des API der Fremdkomponente wird ein Integer-Datentyp des Rahmenprogramms.

Neben MES-Applikationen können auch Applikationen aus der Unternehmensleitebene (Enterprise Resource Planning-Ebene) und/aus der Automatisierungsebene (Controls-Ebene) über das Rahmenprogramm IF integriert werden und über den Arbeitsplatz PIW1 (das Acronym PIW steht für Personalized Industrial Workplace) überwacht bzw. verwaltet werden. Das Rahmenprogramm IF bildet somit eine Integrationsplattform für den gesamten industriellen Bereich. Unterschiedliche Applikationen aus der Unternehmensleitebene, der MES-Ebene und der Automatisierungsebene lassen sich durch das Rahmenprogramm IF einfach und wirtschaftlich mit Hilfe von Adaptern und/oder Wrappern integrieren. Das Rahmenprogramm IF ist somit als Middle-

ware-Plattform und als Manufacturing Application Integration-Werkzeug anzusehen. Über den Arbeitsplatz PIW1 kann ein Benutzer (z.B. der Anlagenfahrer) die jeweiligen Zustände der zu überwachenden Applikationen sehen, und er kann auch auf Daten und auf Methoden der Applikationen zugreifen, und weiterhin kann er durch diesen Zugriff Applikationen miteinander in Verbindung setzen.

Das Rahmenprogramm IF ermöglicht es somit, zum einen eine vertikale Integration von Applikationen aus unterschiedlichen Unternehmensebenen zu erreichen und zum anderen ermöglicht das Rahmenprogramm IF eine horizontale Integration von Applikationen der MES-Ebene.

Der Arbeitsplatz PIW1 stellt für einen Benutzer an der Frontendseite von MES-Applikationen oder anderen Applikationen aus dem Unternehmen ein "One Window to the World" dar. Das heißt, der Arbeitsplatz ermöglicht, über eine gemeinsame einheitliche Oberfläche einen integrativen Zugang auf unterschiedliche, auch heterogene Anwendungen im Unternehmen. Der Benutzer des Arbeitsplatzes PIW1 kann somit von diesem einen Arbeitsplatz aus alle integrierten MES- oder anderen Anwendungen überwachen und verwalten. Dieser Arbeitsplatz kann über das Internet, das Intranet, LAN (Local Area Network) oder andere denkbare Verbindungen mit den Applikationen verbunden sein. Es ist auch möglich, diesen Arbeitsplatz als mobile Station, z.B. als mobiles Endgerät (PDA, Handy) auszugestalten. Diese Mobilität würde für einen Benutzer noch weitere Vorteile bringen.

Die Darstellung gemäß FIG 3 zeigt die zentrale Stellung des die Softwareapplikationen koppelnden Rahmenprogramms. Um das erfindungsgemäße System oder Verfahren zu realisieren, bietet es sich an, eine Client-Server-Architektur zu wählen. Das Rahmenprogramm (IF; FIG 2) kann dabei auf einem einzigen Server oder auf mehreren beliebigen Servern, die sich in einer IT-Landschaft verteilen können, realisiert sein. In FIG 3 ist

dargestellt, dass sich das Rahmenprogramm (IF; FIG 2) auf einem Server IFS (Industrial Framework Server) befindet. An diesem zentralen Server IFS hängen durch die bidirektionalen Informationspfade I2 - I8 verbunden, die Clients. Zu den Clients zählen zum einen die Applikationen aus der ERP-, der MES- und der Automatisierungsebene. In FIG 3 sind diese Applikationen am unteren Bildrand dargestellt. Über die Adapter AD4 - AD6 sind diese Applikationen mit dem Rahmenprogramm (IF; FIG 2) und somit mit dem Server IFS verbunden. Die Verbindung der Adapter AD4 - AD6 mit den Applikationen erfolgt über API-Schnittstellen API1 - API3 (API steht für Application Programmable Interface). Ein Application Programmable Interface stellt eine Schnittstelle mit einer Menge von Kommandos dar. APIs werden auch verwendet bei der Umsetzung von Parameterlisten von einem Format in ein anderes Format und bei der Interpretation der Argumente in eine oder beide Richtungen. Die APIs sind sozusagen der Klebstoff zwischen den Applikationen und den Adaptern. Die Verbindung zwischen den Adaptern AD4 - AD6 mit dem Rahmenprogramm (IF; FIG 2) (in FIG 3 dargestellt durch die bidirektionalen Informationspfade I3 - I5) geschieht über geeignete Datenformate (z.B. XML), geeignete Protokolle (XOP, OPC, etc.) und geeignete Transportmechanismen (z.B. DCOM oder MSMQ). Auch HTTP (Hyper Text Transfer Protocol) kann hierbei verwendet werden. Auch das auf XML eXtensible Markup Language) beruhende Protokoll SOAP (Simple Object Access Protocol) kann für die Integration der Adapter AD4 - AD6 an das Rahmenprogramm (IF; FIG 2) bzw. den zugehörenden Server IFS verwendet werden. Clients bzw. Applikationen, die ActiveX-Dokumente bzw. -Aufrufe unterstützen, lassen sich besonders vorteilhaft in das Rahmenprogramm (IF; FIG 2), bzw. den Server IFS integrieren. Die Anbindung der Applikationen an das Rahmenprogramm kann neben Adaptern auch durch Wrapper oder anderen Integrationsmechanismen erfolgen.

Als weiterer Client kann mit dem Server IFS das Repository IFR (Industrial Framework Repository) verbunden sein. In FIG 3 ist die Verbindung durch den bidirektionalen Informati-

onspfad I2 dargestellt. Das Repository IFR wird verwendet, um Daten sicher und persistent zu halten. Über Methodenaufrufe kann auf diese Daten zugegriffen werden. Im Repository sind u.a. Objekte, Methoden und Laufzeitdaten gespeichert.

Auf der oberen Bildhälfte sind weitere Clients des Servers IFS dargestellt. Der Personalized Industrial Workplace PIW2 und eine eventuell vorhandene Engineeringumgebung EU sind Clients des Servers IFS. Der Personalized Industrial Workplace PIW2 ist durch den bidirektionalen Informationspfad I6 mit dem Rahmenprogramm (IF; FIG 2) bzw. mit dem Server verbunden, die Engineeringumgebung EU entsprechend durch den bidirektionalen Informationspfad I7. Durch die drei Punkte ist dargestellt, dass weitere Clients am Server IFS hängen können. In FIG 3 ist angedeutet, dass außerdem ein weiterer Client C, verbunden durch den Informationspfad I8, am Server IFS hängt.

Die Verbindung der Clients IFR, PIW2, EU, C geschieht entsprechend über APIs bzw. über gängige Datenformate (z.B. XML), gängige Protokolle (XOP, OPC, etc.) und gängige Transportmechanismen (z.B. DCOM, HTTP oder MSMQ).

Die eingesetzten Adapter AD4 - AD6 ermöglichen den Zugang zu Daten und auch zu Methoden der einzelnen Applikationen, die sie mit dem Rahmenprogramm (IF; FIG 2) verbinden. Diese Adapter sind sehr flexibel und nicht auf einzelne spezielle Protokolle oder spezielle Transportmechanismen festgelegt. Wenn die Adapter in einer Laufzeitumgebung eingesetzt werden, dann sind sie so konfiguriert, dass sichergestellt ist, dass bestimmte benötigte Daten aus einer Applikation zum richtigen Zeitpunkt in der Serverumgebung vorhanden sind. Dies kann, wie schon gesagt, über unterschiedliche Protokolle und Transportmechanismen erfolgen. In einer Laufzeitumgebung können sich mehrere Adapter, die auch kleine Servereigenschaften (wie beispielsweise das Ausführung von Workflows, die Bereitstellung verschiedener Kommunikationsmöglichkeiten, ...) besitzen können, befinden. Diese Adapter können auf dem jewei-

ligen Applikationsrechner laufen. Sie müssen aber nicht nur auf einer Maschine laufen, sie können auch verteilt sein.

Softwareapplikationen, insbesondere MES-Applikationen (Manufacturing Execution Systems), liegen oft in einer heterogenen Form vor, z.B. können sie unterschiedliche Datenformate oder Objektmodelle besitzen oder sie sind in unterschiedlichen Programmiersprachen realisiert. Das erfindungsgemäße System bzw. Verfahren ermöglicht es, solche heterogenen Applikationen mit Hilfe eines Rahmenprogramms zu integrieren. Die Kommunikation zwischen diesen Applikationen erfolgt auf der Basis von Kommunikationsmitteln wie z.B. HTTP, DCOM, MSMQ, etc. Die Kommunikation, d.h. der Datenaustausch zwischen den Applikationen ist aber unabhängig von diesen Kommunikationsmitteln, d.h. die Applikationen können von den Applikationsmitteln abstrahieren.

In der Darstellung gemäß FIG 4 ist ein Beispiel für die Transformation von Objekten bzw. Objektbäumen dargestellt. In einem gemeinsamen Meta-Objektmodell können semantisch gleiche Objekte durchaus unterschiedlich repräsentiert sein. Um eine Kommunikation über den reinen Datenaustausch zu ermöglichen, ist es notwendig, dass die auszutauschenden Objektstrukturen in eine Form gebracht werden, die für den jeweiligen Empfänger verständlich ist. Die Darstellung gemäß FIG 4 zeigt schematisch die Oberfläche einer Projektierungsumgebung zur Transformation von Objekten bzw. Objektbäumen mit einer Anzeigevorrichtung AZ 1. Die in den Bildschirmbereichen BB1 bzw. BB2 dargestellten Objektbäume stellen semantisch gleiche Objekte (z.B. einen Produktionsauftrag) dar, die aber unterschiedlich repräsentiert sind. Der Quellbaum QB besteht aus einer Wurzel 1 und einem darunter liegenden Element 2, das wiederum als Unterelemente S1 und S2 besitzt. Der Zielbaum ZB besitzt als Wurzel 1' und als darunter hängende Elemente 2' und 2''. Das Element 2' ist mit dem Element S1 verbunden, das Element 2'' mit dem Element S2. Die Baumrepräsentationen QB bzw. ZB gehören zu Applikationen (z.B. MES-Applikationen),

die über Adapter das Meta-Modell des zugrunde liegenden Softwaresystems (z.B. Frameworks) abgebildet wurden. Da die beiden Baumstrukturen, obwohl sie semantisch gleiche Objekte repräsentieren, in einer unterschiedlichen Form im Meta-Objektmodell repräsentiert sind, ist eine Kommunikation zwischen den Applikationen, zu denen sie gehören, in dieser Form noch nicht effektiv möglich. Eine effektive Kommunikation wird dadurch ermöglicht, dass ein Objektbaum auf den anderen abgebildet wird. In der Darstellung gemäß FIG 4 wird der Quellbaum QB in den Baum ZB transformiert. Wie diese Transformation erfolgt, ist im Bildschirmbereich BB3 dargestellt. Die Elemente S1 und S2 stellen dabei Objekte (z.B. Variablen) dar, die in den beiden Bäumen QB und ZB die gleiche Bedeutung haben. Im Bildschirmbereich BB3 ist dargestellt, wie eine Transformation erfolgen kann, die ausdrückt, dass die Objekte S1 und S2 des Quellbaumes QB und des Zielbaumes ZB semantisch zusammengehören. Die Anzeigevorrichtung AZ1 kann als weiteren Bildschirmbereich eine Menüleiste ML1 enthalten, auf der Funktionen (z.B. durch Buttons) dargestellt sind, die ein Anwender für die Transformation verwenden kann. Als Anzeigevorrichtung dient üblicherweise ein Monitor oder ein Display. Über Eingabehilfsmittel, z.B. Maus oder Keyboard, kann ein Anwender die Elemente der Anzeigevorrichtung bedienen und aktivieren. Die Anzeigevorrichtung AZ1 kann auch weitere Bildschirmbereiche beinhalten. So ist es z.B. auch vorstellbar, dass der Bildschirmbereich BB3 weiter unterteilt wird, es können aber auch weitere Menüleisten ML1 vorhanden sein. Einmal vom Anwender definierte Transformationen können abgespeichert werden und für spätere Anwendungen wieder verwendet werden. Auch kann ein Anwender Regeln definieren, um diese Transformationen durchzuführen. Auch diese Regeln kann ein Anwender abspeichern und für spätere Transformationen wieder verwenden. Es ist auch denkbar, dass eine Regel Teil des Meta-Objektmodells des zugrunde liegenden Softwaresystems wird und dann als Objekt einem Anwender zur Verfügung steht. Die erwähnten Regelobjekte entstehen automatisch bei der Definition einer Transformation. Zur Definition von Transformatio-

nen können mathematische Funktionen (Addition, Subtraktion, Sinus, Kosinus, etc.), aber auch Timer oder andere Adapter (z.B. Excel) verwendet werden. Über Drag&Drop-Mechanismen können sehr leicht die Transformationen und die Regeln erstellt werden. Durch die oben beschriebene Transformation werden zwei semantische Darstellungen (z.B. Objektbäume), die durch die Adaption in das Meta-Objektmodell gebracht worden sind, ineinander übergeführt. Dadurch ist es sehr leicht, dass die adaptierten Applikationen (z.B. unterschiedliche MES-Applikationen) miteinander kommunizieren können. Bei der beschriebenen Transformation werden die Objekte bzw. die Objektbäume direkt aufeinander abgebildet und nicht die Repräsentationen dieser Objekte in irgendwelchen Formatstrukturen. Die Transformation kann somit in der Domänenwelt des Anwenders erfolgen, und ein Anwender kann von der internen Repräsentation der Objekte, d.h. von den internen Formaten, abstrahieren. Die hier beschriebene Projektierungs- bzw. Engineeringumgebung kann z.B. als Client in einem Framework oder im Softwaresystem realisiert sein. Es ist aber auch möglich, dass diese Umgebung auf einem stand-alone-PC realisiert wird.

Die Darstellung gemäß FIG 5 zeigt ein Beispiel für die Transformation von Objekten bzw. Objektbäumen mit Hilfe von mathematischen Objekten. Die Bildschirmbereiche BB1' und BB2' enthalten Objektbäume OB1 bzw. OB2, die ineinander übergeführt werden sollen. OB1 enthält in seiner Struktur als Element die Komponente K1, OB2 enthält in seiner Struktur als Element die Komponente K2. Über einen Drag&Drop-Mechanismus werden die Komponenten K1 bzw. K2 auf die Arbeitsfläche BB3' geschoben, und auf der Arbeitsfläche BB3' entspricht die Komponente K1' der Komponente K1 vom Objektbaum OB1 und die Komponente K2' entspricht der Komponente K2 des Objektbaumes OB2. Zwischen diesen Komponenten K1' und K2' soll eine Transformation hergestellt werden. K1' enthält die Variablen V1 und V2, K2' enthält die Variablen S und D. In der Menüleiste ML2 sind Operatoren, z.B. Addition, Subtraktion, Division, Sinus dargestellt, die für die Definition einer Transformation verwen-

det werden können. Es können auch weitere mathematische Operatoren zur Verfügung gestellt werden, um eine Transformation zu definieren, wie z.B. Multiplikation, Tangens, Arcus Tangens, Kosinus, Exponentialfunktion, Logarithmus, Wurzelfunktion, Signum, Absolutfunktion, usw. Des Weiteren können auch Timer für die Definition von Transformationen verwendet werden. Timer liefern z.B. die aktuelle Zeit oder auch definierbare Zeittakte, die z.B. mit Hilfe der erwähnten mathematischen Operationen manipuliert werden können. So ist es z.B. denkbar, Werte, die aus einem Timer kommen, mit anderen Werten aufzuaddieren und erst dann weiterzuleiten. Ein Anwender kann aber auch eigene Operatoren bzw. Operationen implementieren mittels Skript oder Implementierung von COM-Objekten. Im Bildschirmbereich BB3' ist dargestellt, dass die zwei Variablen V1 und V2 der Komponente K1 bzw. K1' aufaddiert werden. Das Resultat dieser Addition (dargestellt durch die Variable R1) wird mit der Variablen S der Komponente K2' bzw. K2 verbunden. Bei der Erstellung einer solchen Transformation entsteht ein Regelobjekt, das auch wieder für spätere Transformationen verwendet werden kann. Diese Regelobjekte können z.B. in der Kommunikationsstruktur eines Adapters (siehe FIG 7) abgelegt und verwendet werden. Durch die drag&drop-mäßige Verbindung von Objektbäumen untereinander bzw. Objekten von Objektbäumen, indem sie auf die Arbeitsfläche BB3' geschoben werden, werden automatisch die Verbindungen für die Transformation erzeugt. D.h. im Hintergrund wird für die zu verbindenden Objekte diese Verbindung angelegt.

Durch die Möglichkeiten der graphischen Projektierung mit Drag&Drop-Mechanismus muss ein Anwender einzelne Verbindungen nicht mehr programmieren, sondern er kann sie graphisch projektieren. Die Effektivität eines Anwenders steigt dadurch enorm. Nach einer Transformation liegen zwei Objektbäume, die miteinander verbunden werden sollen, in einer gemeinsamen Repräsentation vor. Dadurch ist es sehr leicht möglich, eine Kommunikation, die auf einem reinen Datenaustausch basiert, zwischen diesen Objekten, d.h. zwischen den dazugehörenden

Applikationen, einzurichten. Die Performance einer solchen Kommunikationsverbindung ist sehr hoch.

Die Darstellung gemäß FIG 6 zeigt ein Beispiel für die Projektierung einer Transformation. In FIG 6 ist eine Anzeigevorrichtung AZ3 (z.B. Display, Monitor) dargestellt. Mit den Bildschirmbereichen BB1", BB3", BB2" und dem Bildschirmbereich ML3. Im Bildschirmbereich BB1" ist der Objektbaum OB3 mit der Komponente K3 dargestellt. Die Komponente K3 enthält die Variablen v3 und v4. Mit den Pfeilen P1 und P2 wird schematisch dargestellt, dass die Komponente K3 bzw. die dazugehörige Variable v3 auf den Bildschirmbereich BB3" via Drag&Drop gezogen werden. Der Bildschirmbereich BB3" stellt somit eine Arbeitsoberfläche in der Projektierungsumgebung dar. Der Bildschirmbereiche BB2" enthält den Objektbaum OB4. Dieser Objektbaum OB4 enthält einen Knoten MO mit mathematischen Objekten. Diese mathematischen Objekte können ebenfalls auf die Arbeitsfläche BB3" geschoben werden. Dies wird schematisch durch den Pfeil P3 dargestellt. Durch den Pfeil P3 wird das mathematische Objekt "Subtraktion" (Sub) ausgewählt und auf die Arbeitsfläche geschoben. Dieses mathematische Objekt "Subtraktion" enthält die Eingangsvariablen x3 und y3 sowie die Ausgangsvariable r3. Die Variable r3 stellt das Resultat der Differenzbildung dar. In der Darstellung gemäß FIG 6 ist dargestellt, dass der Knoten mit den mathematischen Objekten MO neben der Subtraktion auch die Addition, die Multiplikation, die Division, den Sinus enthalten kann. Es ist aber auch möglich, dass dieser Knoten weitere mathematische Objekte wie Arcus, Arcustangens oder vom Anwender selbst definierte mathematische Funktionen enthält. Der Anwender kann auf der Arbeitsfläche BB3'' die Objekte miteinander verknüpfen und dabei Regeln für die Transformation der Objektbäume herstellen. Es ist auch denkbar, dass die mathematischen Objekte zur Erstellung der Regeln in der Menüleiste ML3 hinterlegt sind. Von dieser Menüleiste ML3 können diese mathematischen Objekte dann auch via Drag&Drop oder über Mausklick für die Arbeitsfläche BB3' abgewählt werden. Die Menüleiste ML3

kann auch weitere Funktionen beinhalten, die auch vom Anwender definiert sein können. Die Bildschirmbereiche, wie sie in der Anzeigevorrichtung AZ3 dargestellt sind, können auch beliebig anders angeordnet werden. Es ist auch denkbar, dass weitere Bildschirmbereiche angezeigt werden bzw. dass weniger Bildschirmbereiche angezeigt werden. Die Eingaben für die Projektierumgebung können auch über andere Eingabehilfsmittel, z.B. über Tastatur erfolgen.

Darstellung gemäß FIG 7 zeigt den prinzipiellen Aufbau eines Adapters. Jeder Adapter ist eine spezielle Component mit der besonderen Eigenschaft, dass die Component eines Adapters jeweils in einem eigenen Prozess läuft. Jeder Adapter bringt schon eine bestimmte Default-Struktur mit, die wieder als Baumstruktur von Objekten des Rahmenprogramms, d.h. Components und Variablen, dargestellt ist, und die bestimmte Stellen zur Verfügung stellt, an denen der Adapter bestimmte Informationen nach außen legen kann. Beispiele dafür sind Statusinformationen über den Zustand des Adapters (ist der Adapter mit seiner Anwendung verbunden, läuft die Anwendung, Versionsinformationen, usw.). Weiterhin kann ein Adapter Informationen über das externe System, d.h. die externe Applikation, nach außen geben. Durch den "Object Space" kann ein Adapter Strukturen nach außen legen, auf die andere Adapter bzw. andere Applikationen zugreifen können. Auch kann ein Adapter Informationen bezüglich einer Kommunikationsinfrastruktur nach außen geben. Unter Kommunikationsinfrastruktur versteht man Objekte, um Nachrichten zu senden, zu empfangen, Nachrichten zu transformieren. Aber auch Mechanismen, um ein Routing vorzunehmen und Mechanismen, um den Datenaustausch des Adapters zu protokollieren. Weiterhin gehören zur Kommunikationsinfrastruktur eines Adapters so genannte Inports und Outports, die physikalisch die Nachrichten empfangen oder versenden. Ein Adapter ist als eigener Prozess des Betriebssystems vorhanden, d.h. wenn ein Adapter abstürzt, dann hat das keine Auswirkungen auf andere Adapter. Ein Adap-

ter ist somit eine eigene geschlossene Anwendung, die alleine ausführbar ist

Adapter und Wrapper sind Mechanismen für die Integration von Softwarekomponenten in ein Softwaresystem. Ein Wrapper bildet das API (Application Programmable Interface) einer Fremdkomponente bzw. einer zu integrierenden Applikation in das Objektmodell des Softwaresystems, hier Rahmenprogramms (IF; FIG 2) ab. So wird z.B. aus einer Methode des API der zu integrierenden Applikation eine Methode des Rahmenprogramms bzw. aus einem Integer-Datentyp des API der zu integrierenden Applikation wird ein Integer-Datentyp des Rahmenprogramms, usw. Ein Wrapper bringt somit das API der zu integrierenden Applikation in die Sprache, in die Nomenklatur und in die Objektwelt des Rahmenprogramms.

Ein Adapter dagegen ist eine Abstraktionsstufe höher als ein Wrapper. Adapter stellen eine standardisierte Sicht auf zu integrierende Applikationen dar, und das Rahmenprogramm, in das die zu integrierende Applikation eingehängt wird, kann von dieser Applikation abstrahieren. Ein Adapter stellt Funktionalität zur Verfügung, um das zu integrierende System, d.h. die zu integrierende Applikation, zu starten, zu bedienen und zu handeln. Mit Hilfe der Adapter kann z.B. ein Anwender eine SAP-Applikation benutzen, ohne ein SAP-Experte zu sein. Durch die IDOC-Adapter werden SAP-Applikationen in das Objektmodell des Rahmenprogramms abgebildet und werden dann als Objekte des Rahmenprogramms (Component IDOC) verwendet.

Durch das erfindungsgemäße System und Verfahren können zwei Applikationen (z.B. MES-Applikationen) peer-to-peer-mäßig zusammengebracht werden, ohne dass eine solche Verbindung jeweils peer-to-peer-mäßig programmiert werden muss. Diese Verbindungen werden erfindungsgemäß projektiert durch das Etablieren einer Connection.

Die bei der Transformation verwendeten Regeln können Teil eines Adapters sein bzw. die Regeln können als Information auf einem Adapter hinterlegt sein und ins System eingebracht werden.

Es gibt z.B. Adapter für Spreatsheets (z.B. Excel) oder andere "Commercials off the shelf"-Programme.

Darstellung gemäß FIG 8 zeigt die Objektstruktur einer "Component" als Metaobjektmodell des Rahmenprogramms (IF; FIG 2) in UML (Unified Modelling Language). Beim erfindungsgemäßen System und Verfahren werden die zu integrierenden Softwareapplikationen und die zugrunde liegenden Kommunikationsmittel (z.B. HTTP, MSMQ, etc.) auf ein Objektmodell des Rahmenprogramms (IF; FIG 2) abgebildet. Dadurch besitzen an sich heterogene Applikationen ein gemeinsames Objektmodell. Dadurch werden alle Methoden, Datenstrukturen, Objekte usw. der zu integrierenden Fremdapplikationen in einem gemeinsamen Objektmodell dargestellt. Dieses Objektmodell ist sehr generisch und stellt ein Metaobjektmodell dar. Der Kern dieses Objektmodells besteht aus einem Objekttyp namens "Component". Eine Component kann wiederum andere Components aggregieren und/oder spezielle Typen von Components, so genannte Variablen, denen im Betrieb bestimmte Werte zugewiesen sind, enthalten. Components und Variablen können jeweils auch zusätzliche Attribute besitzen.

Dieses Objektmodell bildet die Basis der Adaptierung von MES-Applikationen oder anderen Applikationen. Das bedeutet, dass die Strukturen der Applikationen in Strukturen dieses Objektmodells übersetzt bzw. abgebildet werden. Alle zu integrierenden Applikationen werden innerhalb des Rahmenprogramms (IF, FIG 2) in der Darstellung dieses Objektmodells repräsentiert. Auch die zugrunde liegenden Kommunikationsmittel werden auf dieses generische Objektmodell abgebildet. Im Rahmenprogramm (IF; FIG 2) sind nun alle Applikationen und alle zugrunde liegenden Kommunikationsmittel in einem einheitli-

chen und homogenen Objektmodell repräsentiert. Dadurch können sehr einfach und leicht Kommunikationsbeziehungen zwischen den Applikationen eingerichtet werden.

In der Darstellung gemäß FIG 8 ist die generische Komponente "Component", die den Kern des Objektmodells darstellt, in UML-Notation aufgezeigt.

Die rechteckigen Kästchen stellen Teile des Objektmodells dar. Durch eine Rautenbeziehung wird eine Aggregation (ist Teil von-Beziehung) dargestellt, durch eine Dreiecksbeziehung wird die Vererbung (ist ein-Beziehung) dargestellt. In der Darstellung gemäß FIG 8 wird somit gezeigt, dass eine Component aus mehreren Components bestehen kann, weiterhin kann eine Component Teil einer anderen Component sein. Durch die Rautenbeziehung ist eine Component mit Attributen und Variablen verbunden. Das heißt, eine Component kann Attribute und Variable beinhalten. Attribute sind beschreibende Daten. Alle Objekte einer Klasse besitzen dieselben Attribute, jedoch im Allgemeinen unterschiedliche Attributwerte. Ein Attributwert ist ein einem Attribut zugeordneter Wert aus seinem Wertebereich. Eine Variable kann Werte von bestimmten Datentypen (z.B. Integer, Real etc.) annehmen. Wie durch die Rautenbeziehung dargestellt, kann eine Component mehrere Variablen enthalten. Eine Component kann aber auch eine Oberklasse einer Variable sein, wie durch die Dreiecksbeziehung dargestellt. Das heißt, eine Variable kann eine abgeleitete Component sein. Die Rauten- und die Dreiecksbeziehungen können auch Kardinalitäten beinhalten.

Durch Abbildungsmechanismen wie z.B. Adapter oder Wrapper werden die zu integrierenden Softwareapplikationen und die zugrunde liegenden Kommunikationsmittel auf diese generische Objektstruktur, d.h. "Component"-Struktur, des Rahmenprogramms (IF; FIG 2) abgebildet.

System und Verfahren zur Projektierung der Transformation von Objektbäumen, insbesondere in MES-Systemen, wobei die Quell- und Zielobjektbäume in einem gemeinsamen Metaobjektmodell eines Softwaresystems, insbesondere eines Frameworks repräsentiert sind. Der Quellobjektbaum wird durch Regeln in den Zielobjektbaum transformiert. Die Transformation findet direkt auf den Objekten des Metaobjektmodells (Component) statt. Dadurch wird eine Kommunikation über einen reinen Datenaustausch zwischen angekoppelten Applikationen ermöglicht.

Das oben beschriebene erfindungsgemäße System bzw. Verfahren lässt sich als Computerprogramm in dafür bekannten Sprachen implementieren. Ein derartig implementiertes Computerprogramm kann in ebenfalls bekannter Weise über elektronische Datenwege, aber auch auf Datenträgern abgespeichert und transportiert werden.

Patentansprüche

1. System zur Projektierung von Transformationen von Objektbäumen (QB, OB1 - OB4)), wobei die Quell- (QB) und Zielobjektbäume (ZB) in einem gemeinsamen Metaobjektmodell eines Softwaresystems repräsentiert sind und wobei das Softwaresystem auf mindestens einer Rechnereinheit gespeichert ist,
d a d u r c h g e k e n n z e i c h n e t, dass
a) in einem ersten Bildschirmbereich (BB1, BB1', BB1'') auf einer Anzeigevorrichtung (AZ1 - AZ3) ein Quellobjektbaum (QB) dargestellt ist,
b) in einem zweiten Bildschirmbereich (BB2, BB2') der Anzeigevorrichtung (AZ1 - AZ3) ein Zielobjektbaum (ZB) dargestellt ist,
c) in einem dritten Bildschirmbereich (BB3, BB3', BB3'') der Anzeigevorrichtung (AZ1 - AZ3) eine Arbeitsfläche (BB3, BB3', BB3'') dargestellt ist, zum Positionieren der Knoten der Objektbäume (QB, ZB, OB1 - OB4),
d) in einem vierten Bildschirmbereich (ML1 - ML3, BB2'') der Anzeigevorrichtung (AZ1 - AZ3) Operatoren und/oder Regeln dargestellt sind, die optional auf der Arbeitsfläche (BB3, BB3', BB3'') positionierbar sind und
e) auf der Arbeitsfläche (BB3, BB3', BB3'') eine Transformation projektierbar ist, durch Verschalten der auf der Arbeitsfläche (BB3, BB3', BB3'') befindlichen Symbole.

2. System zur Projektierung nach Anspruch 1,
d a d u r c h g e k e n n z e i c h n e t, dass die Transformation in der für einen Anwender zugrundeliegenden Domäne durchführbar ist.

3. System zur Projektierung nach Anspruch 1 oder 2,
d a d u r c h g e k e n n z e i c h n e t, dass die Regeln als Objekte des Softwaresystems repräsentiert sind.

4. System zur Projektierung nach Anspruch 1, 2 oder 3,

d a d u r c h g e k e n n z e i c h n e t, dass das Positionieren und Verschalten der Knoten und/oder der Operatoren und/oder der Regeln via drag&drop durch Eingabehilfsmittel erfolgt.

5. System zur Projektierung nach einem der Ansprüche 1 bis 4, d a d u r c h g e k e n n z e i c h n e t, dass von einem Anwender weitere Operatoren und/oder Regeln zur Transformation definierbar sind.

6. System zur Projektierung nach einem der Ansprüche 1 bis 7, d a d u r c h g e k e n n z e i c h n e t, dass das Softwaresystem durch mindestens ein Rahmenprogramm (IF) realisiert ist.

7. System zur Projektierung nach einem der Ansprüche 1 bis 6, d a d u r c h g e k e n n z e i c h n e t, dass Quell- (QB) und Zielobjektbaum (ZB) durch die Objektmodelle von angekoppelten Adaptern (AD1 - AD6) vorgegeben sind.

8. Verfahren zur Projektierung von Transformationen von Objektbäumen (QB, OB1 - OB4), wobei die Quell- (QB) und Zielobjektbäume (ZB) in einem gemeinsamen Metaobjektmodell eines Softwaresystems repräsentiert werden und wobei das Softwaresystem auf mindestens einer Rechnereinheit gespeichert wird, d a d u r c h g e k e n n z e i c h n e t, dass

a) in einem ersten Bildschirmbereich (BB1, BB1', BB1'') auf einer Anzeigevorrichtung (AZ1 - AZ3) ein Quellobjektbaum (QB) dargestellt wird,
b) in einem zweiten Bildschirmbereich (BB2, BB2') der Anzeigevorrichtung (AZ1 - AZ3) ein Zielobjektbaum (ZB) dargestellt wird,
c) in einem dritten Bildschirmbereich (BB3, BB3', BB3'') der Anzeigevorrichtung (AZ1 - AZ3) eine Arbeitsfläche (BB3, BB3', BB3'') dargestellt wird, auf der Knoten der Objektbäume positioniert werden,

d) in einem vierten Bildschirmbereich (ML1 - ML3, BB2'')der Anzeigevorrichtung (AZ1 - AZ3) Operatoren und/oder Regeln dargestellt werden, die optional auf der Arbeitsfläche positioniert werden und
e) auf der Arbeitsfläche eine Transformation projektiert wird durch Verschalten der auf der Arbeitsfläche befindlichen Symbole.

9. Verfahren zur Projektierung nach Anspruch 8,
d a d u r c h g e k e n n z e i c h n e t, dass die Transformation in der für ein Anwender zugrundeliegenden Domäne durchgeführt wird.

10. Verfahren zur Projektierung nach Anspruch 8 oder 9,
d a d u r c h g e k e n n z e i c h n e t, dass die Regeln als Objekte des Softwaresystems repräsentiert werden.

11. Verfahren zur Projektierung nach Anspruch 8, 9 oder 10,
d a d u r c h g e k e n n z e i c h n e t, dass das Positionieren und Verschalten der Knoten und/oder der Operatoren und/oder der Regeln via drag&drop durch Eingabehilfsmittel durchgeführt wird.

12. Verfahren zur Projektierung nach einem der Ansprüche 8 bis 11,
d a d u r c h g e k e n n z e i c h n e t, dass von einem Anwender weitere Operatoren und/oder Regeln zur Transformation definiert werden.

13. Verfahren zur Projektierung nach einem der Ansprüche 8 bis 12,
d a d u r c h g e k e n n z e i c h n e t, dass das Softwaresystem durch mindestens ein Rahmenprogramm (IF) realisiert wird.

14. Verfahren zur Projektierung nach einem der Ansprüche 8 bis 13,

dadurch gekennzeichnet, dass Quell- (QB) und Zielobjektbaum (ZB) durch die Objektmodelle von angekoppelten Adaptern (AD1 - AD6) vorgegeben werden.

15. Computerprogramm, das ein Verfahren nach einem der Ansprüche 8 bis 14 implementiert.

16. Datenträger, auf dem ein Computerprogramm nach Anspruch 15 gespeichert ist.

17. Datenverarbeitungseinrichtung (PIW1, PIW2, IFS) auf der ein Computerprogramm nach Anspruch 15 installiert ist.

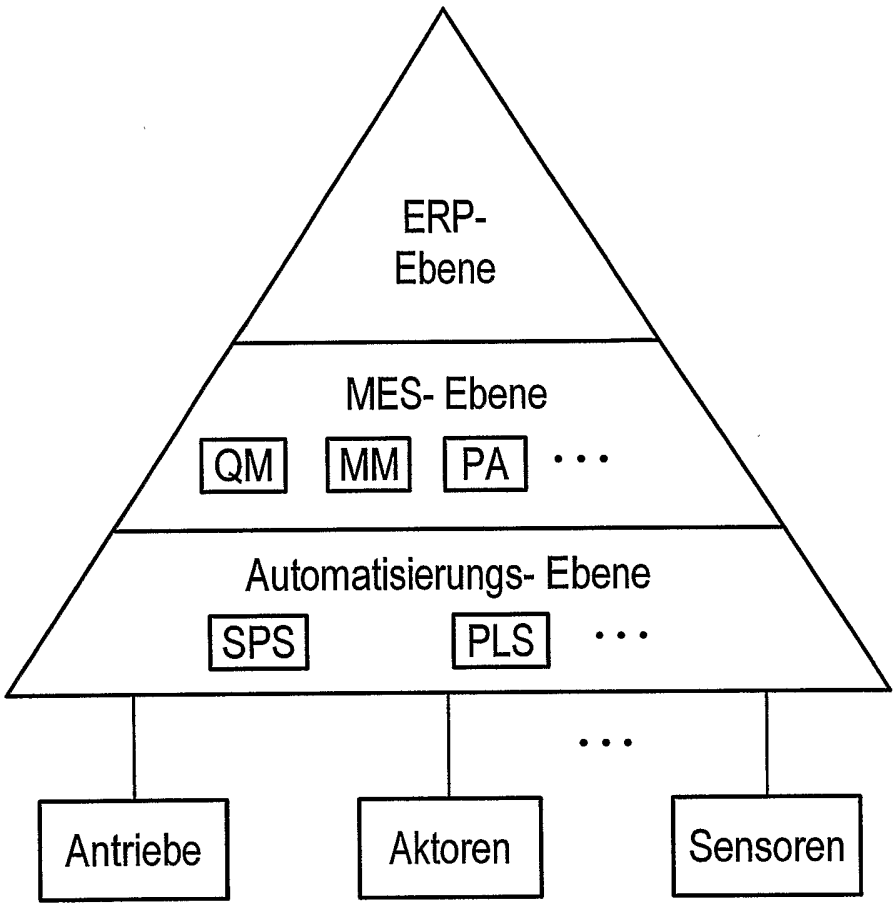

ERP-
Ebene
MES- Ebene
QM
MM
PA
Automatisierungs- Ebene
SPS
PLS
Antriebe
Aktoren
Sensoren

FIG 1

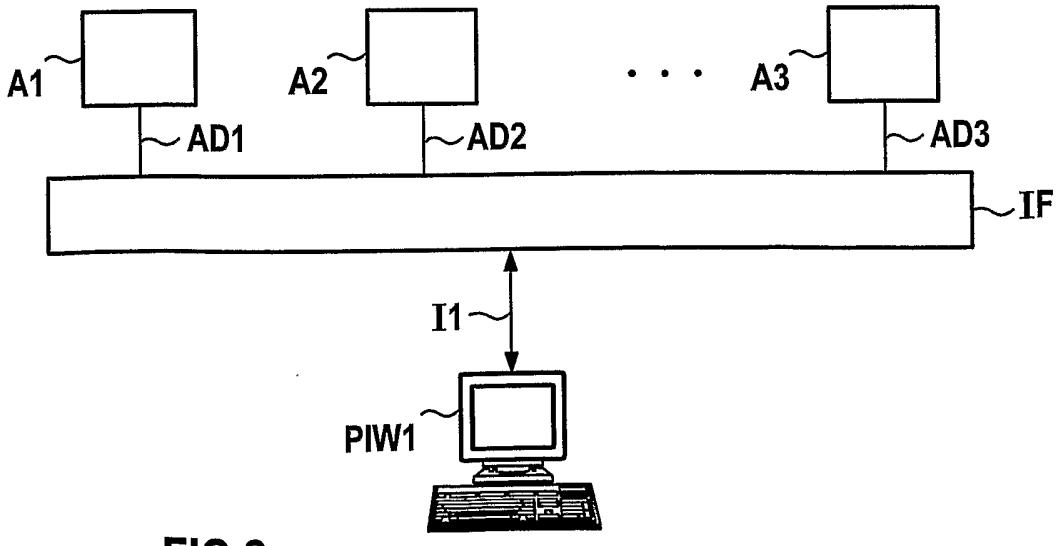

A1
A2
A3
AD1
AD2
AD3
IF
I1
PIW1

FIG 2

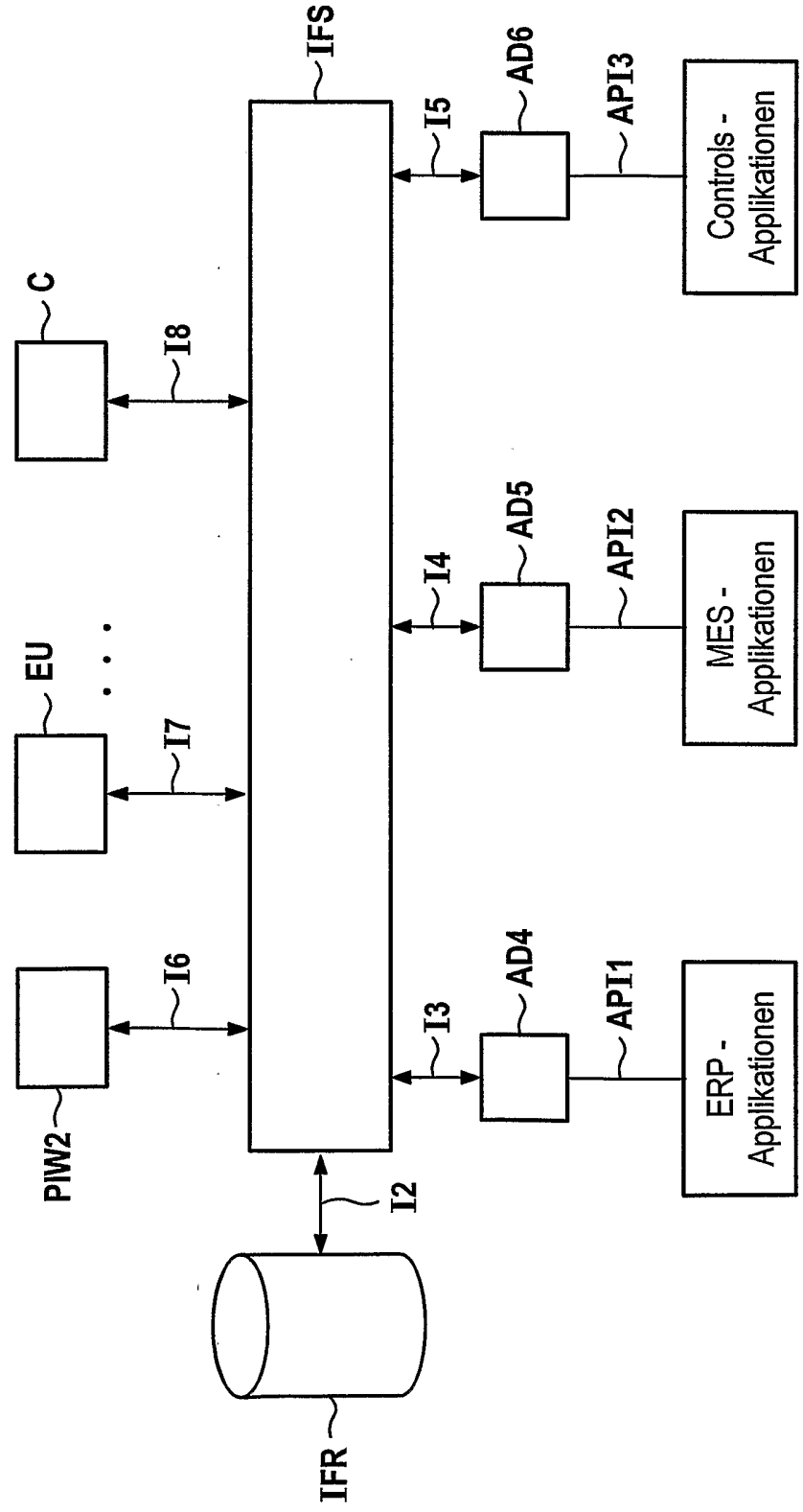

IFS
I5
AD6
API3
Controls - Applikationen
C
I8
EU
I7
I4
AD5
API2
MES - Applikationen
PIW2
I6
I3
AD4
API1
ERP - Applikationen
I2
IFR

FIG 3

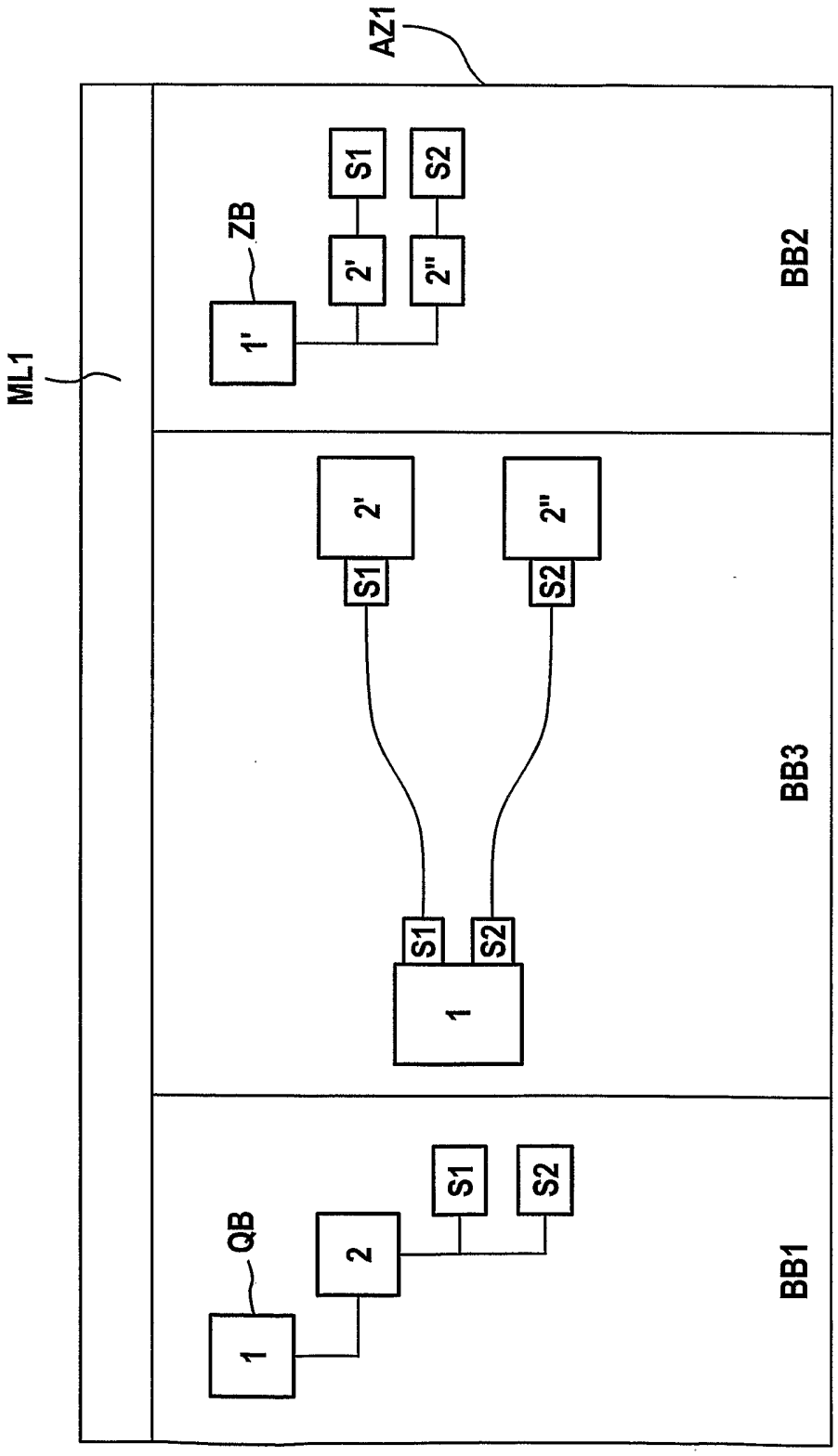


FIG 4

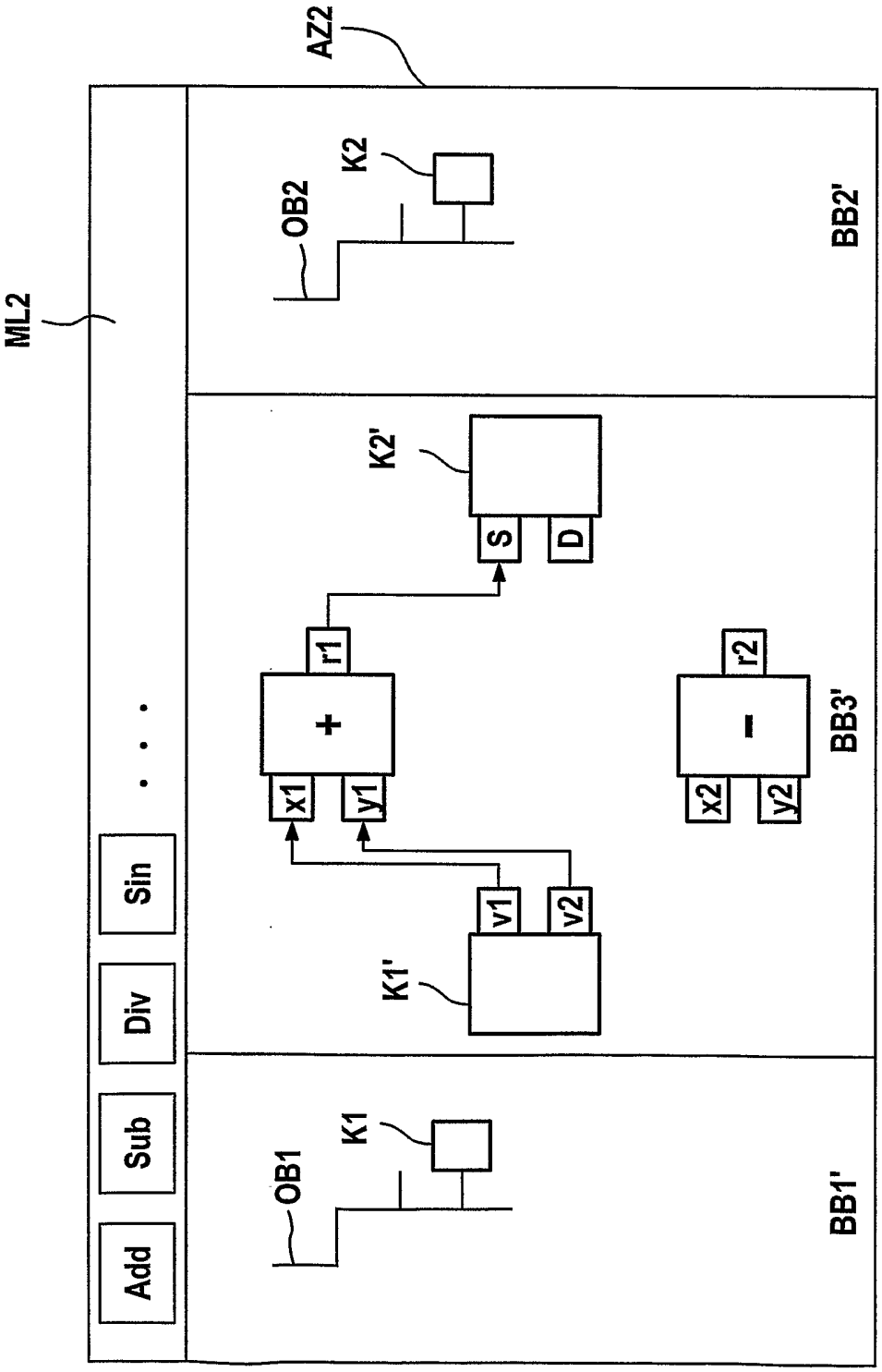

AZ2
ML2
OB2
K2
BB2'
K2'
S
D
r1
+
x1
y1
r2
-
x2
y2
BB3'
v1
v2
K1'
Sin
Div
Sub
Add
OB1
K1
BB1'

FIG 5

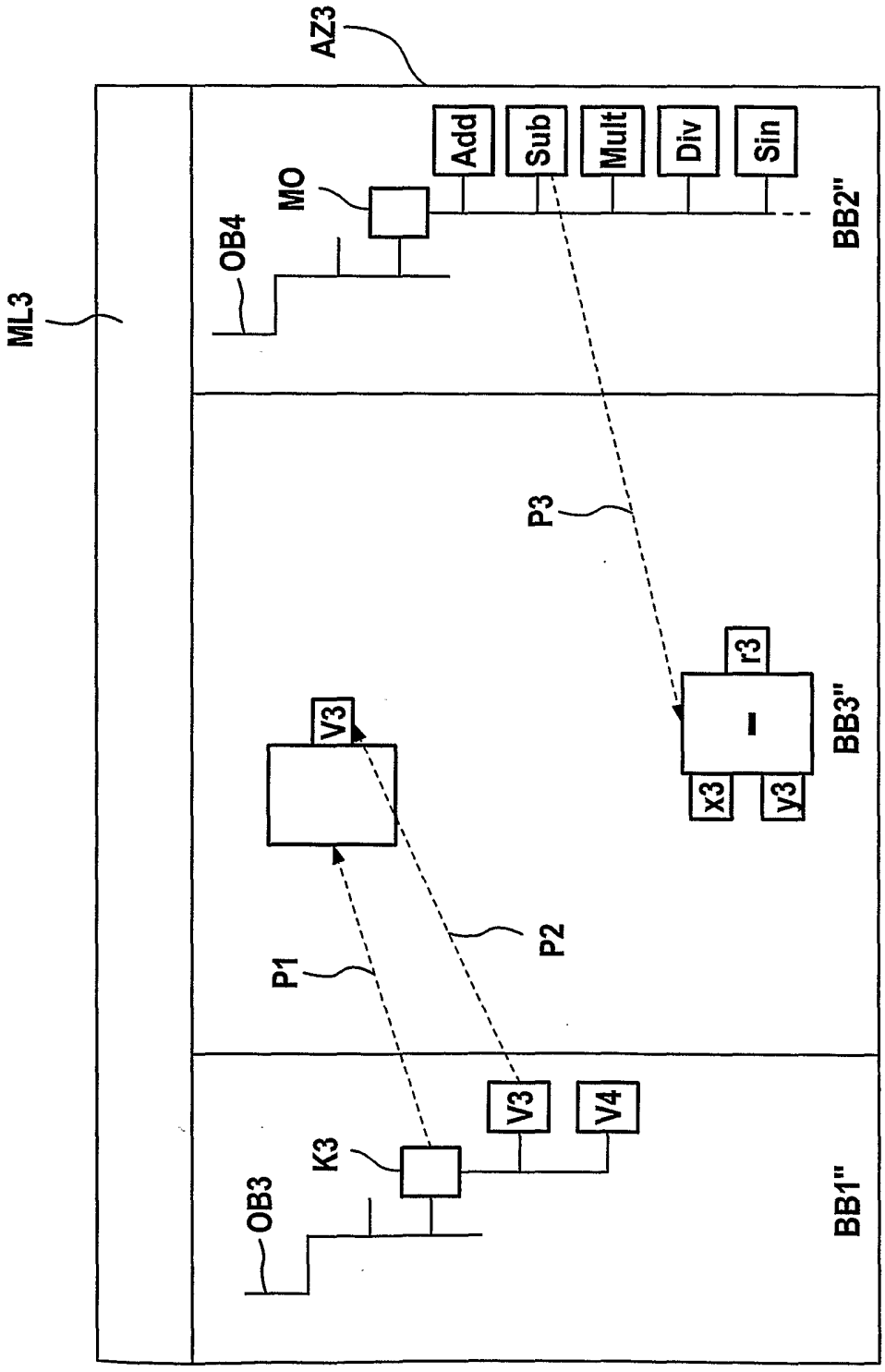

AZ3
ML3
Add
Sub
Mult
Div
Sin
MO
OB4
BB2"
P3
r3
-
x3
y3
BB3"
V3
P1
P2
V3
V4
K3
OB3
BB1"

FIG 6

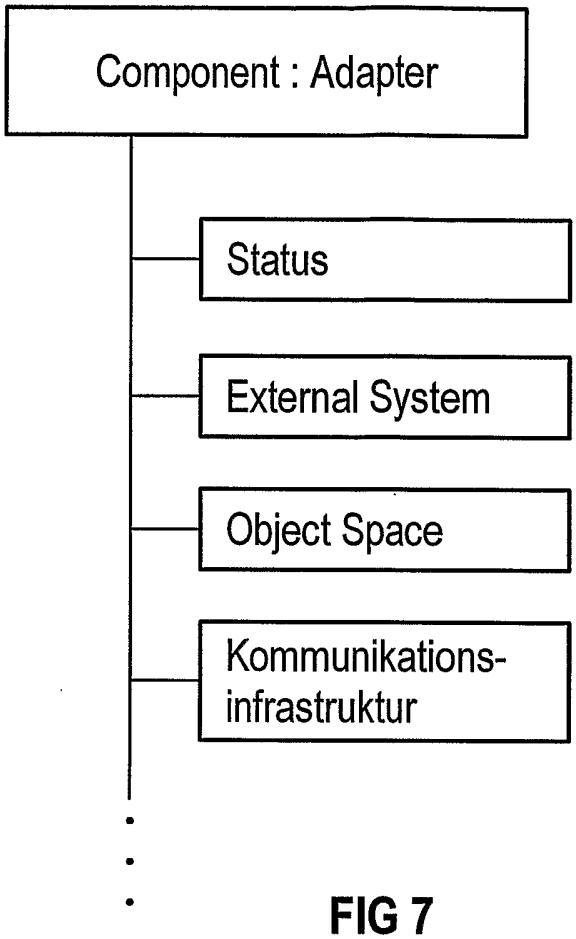

Component : Adapter
Status
External System
Object Space
Kommunikations-
infrastruktur

FIG 7

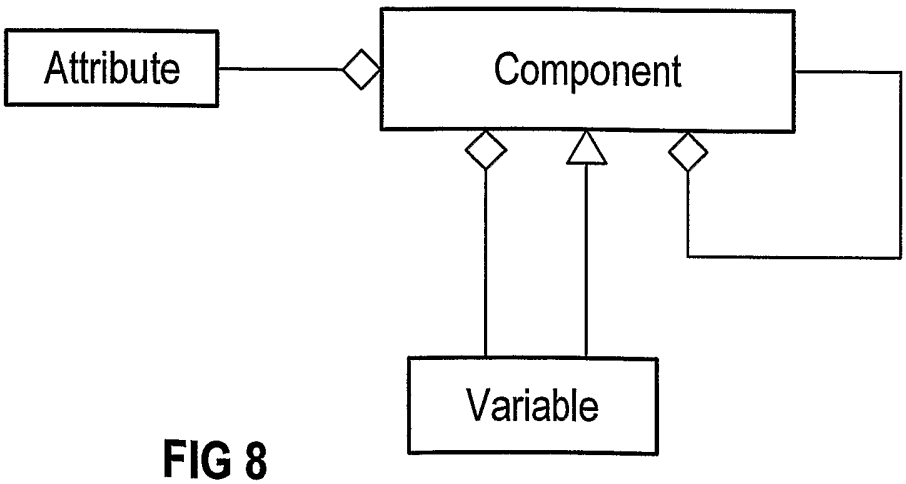

Attribute
Component
Variable

FIG 8