(19) Republik Österreich Patentamt

(10) Nummer: AT 006 351 U1

(12) GEBRAUCHSMUSTERSCHRIFT

(21) Anmeldenummer: GM 8055/02

(22) Anmeldetag: 17. 5.2001

(42) Beginn der Schutzdauer: 15. 7.2003
Längste mögliche Dauer: 31. 5.2011
(45) Ausgabetag: 25. 8.2003

(51) Int.Cl.[7] : G06F 13/00

(67) Umwandlung aus Patentanmeldung: 790/2001

(73) Gebrauchsmusterinhaber:

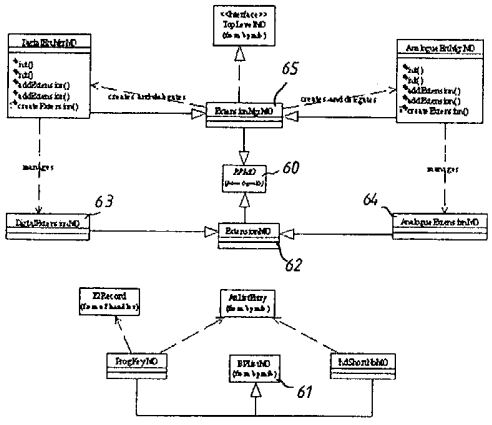
ERICSSON ENTERPRISE GMBH
A-1121 WIEN (AT).

(72) Erfinder:

BEDA ENDRE
WIEN (AT).
GRUY PETER
MAUERBACH, NIEDERÖSTERREICH (AT).
TODOROV VALENTIN
WIEN (AT).

(54) VERFAHREN ZUR PROGRAMMIERUNG BZW. ZUM MANAGEMENT VON KOMMUNIKATIONSNETZEN

(57) Verfahren zur Programmierung bzw. zum Management von Kommunikationsnetzen, insbesondere von Nebenstellenanlagen (1), in einer Umgebung, welche abstrakte Datentypen (52), insbesondere Managed Objects (27) als Abstraktion der Ressourcen (6) des Kommunikationsnetzes bereitstellt, die vorzugsweise in einer Sammlung, insbesondere in einer Management Information Base (MIB) vorliegen, wobei die Datentypen (52) bzw. die Managed Objects (27) über ein Management Interface mit dem Kommunikationsnetz kommunizieren bzw. Funktionsdaten (25) austauschen können, und wobei die Datentypen (52) bzw. die Managed Objects (27) in einer Programmiersprache implementiert sind und Managementkommandos (52') für den Zugriff auf die Datentypen (52) bzw. die Managed Objects (27) bereitstellen, wobei ein Interpretierer und Zugriffskommandos (51) auf die Datentypen (52) bzw. die Managed Objects (27) bereitgestellt werden, die Zugriffskommandos (51) an den Interpretierer übergeben werden, die Zugriffskommandos (51) durch den Interpretierer automatisch in ein bzw. mehrere Managementkommandos (51') übersetzt werden und die entsprechenden Managementkommandos (51') an den Datentypen (52) bzw. den Managed Objects (27) automatisch ausgeführt werden.



AT 006 351 U1

Die Erfindung betrifft eine Programmlogik zur Programmierung bzw. zum Management von Kommunikationsnetzen, insbesondere von Nebenstellenanlagen, in einer Umgebung, welche abstrakte Datentypen, insbesondere Managed Objects als Abstraktion der Ressourcen des Kommunikationsnetzes bereitstellt, die vorzugsweise in einer Sammlung, insbesondere in einer Management Information Base (MIB) vorliegen, wobei die Datentypen bzw. die Managed Objects über ein Management Interface mit dem Kommunikationsnetz kommunizieren bzw. Funktionsdaten austauschen können, und wobei die Datentypen bzw. die Managed Objects in einer Programmiersprache implementiert sind und Managementkommandos für den Zugriff auf die Datentypen bzw. die Managed Objects bereitstellen.

Eine solche Programmlogik ist insbesondere im Bereich von Nebenstellenanlagen bekannt. Die Programmierung erfolgt über den Zugriff auf die Managed Objects, d.h. über den Zugriff auf ein im Computer gespeichertes Abbild der Funktionsdaten der Nebenstellenanlage. Der Zugriff erfolgt über eine Benutzeroberfläche, die die unmittelbare Eingabe bzw. Veränderung des Abbildes der Funktionsdaten erlaubt.

Nachteilig bei der bekannten Programmlogik erweist sich die Schwierigkeit, öfters anfallende Programmier- bzw. Konfigurationsarbeiten automatisch durchzuführen. Die im Stand der Technik bekannte Methode, wiederkehrende Benutzereingaben auf der Benutzeroberfläche in Form von Makros abzuspeichern, stößt insbesondere bei Änderungen in der Struktur der Benutzeroberfläche auf ihre Grenzen, da dann ältere Makros nicht mehr verwendet werden können. Dies macht die Programmierung dem raschen technologischen Fortschritt und somit häufigen Anpassungen unterworfener Telekommunikationsanlagen äußerst kompliziert und zeitlich aufwendig.

Es ist Aufgabe der vorliegenden Erfindung, eine Programmlogik der eingangs angeführten Art anzugeben, welche diese Nachteile beseitigt und die automatische, von einer bestimmten Benutzeroberfläche unabhängige Programmierung von Telekommunikationsanlagen ermöglicht.

Erfindungsgemäß wird dies dadurch erreicht, daß ein Interpretierer und Zugriffskommandos auf die Datentypen bzw. die Managed Objects bereitgestellt werden, die Zugriffskommandos an den Interpretierer übergeben werden, die Zugriffskommandos durch den Interpretierer automatisch in ein bzw. mehrere Managementkommandos übersetzt werden und die entsprechenden Managementkommandos an den Datentypen bzw. den Managed Objects automatisch ausgeführt werden.

Dadurch ergibt sich die Möglichkeit, das Kommunikationsnetz ausschließlich über die Zugriffskommandos und somit unabhängig von einer Benutzeroberfläche zu programmieren. Die Zugriffskommandos können in jedem Texteditor erstellt werden. Dies erlaubt die schnelle Anpassung bestehender Konfigurationsprogramme an neue Hardware-Einheiten des Kommunikationsnetzes bzw. der Nebenstellenanlage. Die Zugriffskommandos können auch nach Integration neuer Ressourcen in das Kommunikationsnetz oder nach der Erstellung einer völlig neuen Benutzeroberfläche weiter verwendet werden. Zusätzlich ergibt

sich beim erfindungsgemäßen Zugriff auf die Managed Objects der Vorteil, daß die Hardware-Struktur des Kommunikationsnetzes eine Entsprechung in der Struktur des Datenmodells findet. Die Konfiguration der Ressourcen wird so erheblich übersichtlicher. Dies erlaubt eine einfachere und schnellere Entwicklung von Konfigurationsprogrammen. Durch den objektorientierten Ansatz können beispielsweise Fehler in der Programmierung leichter gefunden werden. Weiters stellen die objektorientierten Datenmodelle eine beträchtliche Erleichterung bei der Erstellung neuer und bei der Wartung bestehender Konfigurationsprogramme dar, sodaß das Kommunikationsnetz leichter und schneller programmiert werden kann. Der Zusammenschluß aller die Ressourcen darstellenden Datenmodelle in einer Sammlung erlaubt das einfache und schnelle Auffinden der Datenmodelle und somit die schnellere Ausführung der Zugriffskommandos. Desweiteren ergibt sich die dadurch Möglichkeit, dem Benutzer eine Auswahl aller derzeit Kommunikationsnetz integrierter Hardware-Einheiten anzuzeigen. Dies trägt zu einer übersichtlicheren und einfacheren Benutzung bei.

Gemäß einer weiteren Variante der Erfindung kann vorgesehen sein, daß bei der Übersetzung der Zugriffskommandos zusätzliche Benutzerangaben eingelesen werden und diese bei der Übersetzung in die Managementkommandos berücksichtigt werden. Dies ermöglicht es, Zugriffskommandos abhängig von den realen Ressourcen anzupassen und abhängig von der individuellen Situation unterschiedlich auszuführen. Damit ergibt sich die Möglichkeit einer bedeutend flexibleren Programmierung des Kommunikationsnetzes.

Nach einer anderen Ausführungsform der Erfindung kann vorgesehen sein, daß mehrere Zugriffskommandos in ein Kommandoskript zusammengefaßt werden. Dies erlaubt die schnelle, automatische Abarbeitung einer großen Anzahl von Zugriffskommandos bei öfters anfallenden Konfigurationsarbeiten.

Gemäß einer weiteren Variante der Erfindung kann vorgesehen sein, daß Kommandoskripts in ein Skriptdepot geladen bzw. aus diesem abgerufen werden. Dadurch können häufig verwendete Kommandoskripts schneller aufgefunden werden.

Nach einer anderen Ausführungsform der Erfindung kann vorgesehen sein, daß Zugriffskommandos zunächst automatisch durch Auslesen der Datentypen bzw. der Managed Objects eines bereits programmierten Kommunikationsnetzes erzeugt werden und in einem speziellen Kommandoskript, einem sogenannten Template zusammengefaßt werden und dieses Template zur Programmierung eines weiteren Kommunikationsnetzes verwendet wird. Dadurch kann ein neu installiertes Kommunikationsnetz, z.B. eine neue Nebenstellenanlage in einem Schritt durch Übernahme der Programmstruktur einer bestehenden Anlage automatisch programmiert werden.

In Weiterbildung der Erfindung kann vorgesehen sein, daß die Übersetzung der Zugriffskommandos und die Ausführung der Managementkommandos durch einen Server-Prozess umgesetzt werden und Benutzereingaben von einem Client-Prozess eingelesen werden, wobei der Client-Prozess und der Server-Prozess über eine Schnittstelle miteinander kommunizieren. Dadurch wird erreicht, daß der Client-Prozess und Server-Prozess auf örtlich getrennten Computern laufen können. Dies ist besonders bei der Fernwartung von Telekom-

munikationsanlagen vorteilhaft, da die Wartung einer örtlich entfernten Telekommunikationsanlage über einen lokal laufenden Client-Prozess erfolgen kann, der mit einem auf einem mit der Telekommunikationsanlage verbundenen Computer laufenden Server-Prozess kommuniziert.

Nach einer anderen Ausgestaltung der Erfindung kann vorgesehen sein, daß der Client-Prozess und der Server-Prozess über ein Netzwerkprotokoll, insbesondere über das im Internet relevante TCP/IP Protokoll miteinander kommunizieren. Dies bietet den Vorteil, daß die Verbindung der Computer, auf denen der Client-Prozess und der Server-Prozess laufen über ein Datennetzwerk wie das Internet erfolgen kann.

In weiterer Ausgestaltung der Erfindung kann vorgesehen sein, daß die Kommunikation zwischen dem Client-Prozess und dem Server-Prozess über einen Web-Server erfolgt. Dies erlaubt es, die Fernwartung einer Telekommunikationsanlage über einen beispielsweise als Browser-Applikation laufenden Client-Prozess durchzuführen.

In weiterer Ausgestaltung der Erfindung kann vorgesehen sein, daß der Status des Server-Prozesses regelmäßig vom Client-Prozess abgefragt wird. Damit kann der Benutzer auch bei Einwegverbindungen, bei denen keine Anfrage des Server-Prozesses an den Client-Prozess möglich sind, sofort über den Status des Server-Prozesses informiert werden und gegebenenfalls zur Eingabe entsprechender Anweisungen aufgefordert werden.

Die Erfindung betrifft weiters ein Verfahren zur Programmierung bzw. zum Management von Kommunikationsnetzen, insbesondere von Nebenstellenanlagen, in einer Umgebung, welche abstrakte Datentypen, insbesondere Managed Objects als Abstraktion der Ressourcen des Kommunikationsnetzes bereitstellt, die vorzugsweise in einer Sammlung, insbesondere in einer Management Information Base (MIB) vorliegen, wobei die Datentypen bzw. die Managed Objects über ein Management Interface mit dem Kommunikationsnetz kommunizieren bzw. Funktionsdaten austauschen können, und wobei die Datentypen bzw. die Managed Objects in einer Programmiersprache implementiert sind und Managementkommandos für den Zugriff auf die Datentypen bzw. die Managed Objects bereitstellen.

Solche Verfahren sind insbesondere im Bereich von Nebenstellenanlagen bekannt. Die Programmierung erfolgt über den Zugriff auf die Managed Objects, d.h. über den Zugriff auf ein im Computer gespeichertes Abbild der Funktionsdaten der Nebenstellenanlage. Der Zugriff erfolgt über eine Benutzeroberfläche, die die unmittelbare Eingabe bzw. Veränderung des Abbildes der Funktionsdaten erlaubt.

Nachteilig bei den bekannten Programmierverfahren erweist sich die Schwierigkeit, öfters anfallende Programmier- bzw. Konfigurationsarbeiten automatisch durchzuführen. Die im Stand der Technik bekannte Methode, wiederkehrende Benutzereingaben auf der Benutzeroberfläche in Form von Makros abzuspeichern, stößt insbesondere bei Änderungen in der Struktur der Benutzeroberfläche auf ihre Grenzen, da dann ältere Makros nicht mehr verwendet werden können. Dies macht die Programmierung dem raschen technologischen

Fortschritt und somit häufigen Anpassungen unterworfener Telekommunikationsanlagen äußerst kompliziert und zeitlich aufwendig.

Es ist Aufgabe der vorliegenden Erfindung, ein Verfahren der eingangs angeführten Art anzugeben, das diese Nachteile beseitigt und die automatische, von einer bestimmten Benutzeroberfläche unabhängige Programmierung von Telekommunikationsanlagen ermöglicht.

Erfindungsgemäß wird dies dadurch erreicht, daß ein Interpretierer und Zugriffskommandos auf die Datentypen bzw. die Managed Objects bereitgestellt werden, die Zugriffskommandos an den Interpretierer übergeben werden, die Zugriffskommandos durch den Interpretierer automatisch in ein bzw. mehrere Managementkommandos übersetzt werden und die entsprechenden Managementkommandos an den Datentypen bzw. den Managed Objects automatisch ausgeführt werden.

Dadurch ergibt sich die Möglichkeit, das Kommunikationsnetz ausschließlich über die Zugriffskommandos und somit unabhängig von einer Benutzeroberfläche zu programmieren. Die Zugriffskommandos können in jedem Texteditor erstellt werden. Dies erlaubt die schnelle Anpassung bestehender Konfigurationsprogramme an neue Hardware-Einheiten des Kommunikationsnetzes bzw. der Nebenstellenanlage. Die Zugriffskommandos können auch nach Integration neuer Ressourcen in das Kommunikationsnetz oder nach der Erstellung einer völlig neuen Benutzeroberfläche weiter verwendet werden. Zusätzlich ergibt sich beim erfindungsgemäßen Zugriff auf die Managed Objects der Vorteil, daß die Hardware-Struktur des Kommunikationsnetzes eine Entsprechung in der Struktur des Datenmodells findet. Die Konfiguration der Ressourcen wird so erheblich übersichtlicher. Dies erlaubt eine einfachere und schnellere Entwicklung von Konfigurationsprogrammen. Durch den objektorientierten Ansatz können beispielsweise Fehler in der Programmierung leichter gefunden werden. Weiters stellen die objektorientierten Datenmodelle eine beträchtliche Erleichterung bei der Erstellung neuer und bei der Wartung bestehender Konfigurationsprogramme dar, sodaß das Kommunikationsnetz leichter und schneller programmiert werden kann. Der Zusammenschluß aller die Ressourcen darstellenden Datenmodelle in einer Sammlung erlaubt das einfache und schnelle Auffinden der Datenmodelle und somit die schnellere Ausführung der Zugriffskommandos. Desweiteren ergibt sich die dadurch Möglichkeit, dem Benutzer eine Auswahl aller derzeit Kommunikationsnetz integrierter Hardware-Einheiten anzuzeigen. Dies trägt zu einer übersichtlicheren und einfacheren Benutzung bei.

Gemäß einer weiteren Variante der Erfindung kann vorgesehen sein, daß bei der Übersetzung der Zugriffskommandos zusätzliche Benutzerangaben eingelesen werden und diese bei der Übersetzung in die Managementkommandos berücksichtigt werden. Dies ermöglicht es, Zugriffskommandos abhängig von den realen Ressourcen anzupassen und abhängig von der individuellen Situation unterschiedlich auszuführen. Damit ergibt sich die Möglichkeit einer bedeutend flexibleren Programmierung des Kommunikationsnetzes.

Nach einer anderen Ausführungsform der Erfindung kann vorgesehen sein, daß mehrere Zugriffskommandos in ein Kommandoskript zusammengefaßt werden. Dies erlaubt

die schnelle, automatische Abarbeitung einer großen Anzahl von Zugriffskommandos bei öfters anfallenden Konfigurationsarbeiten.

Gemäß einer weiteren Variante der Erfindung kann vorgesehen sein, daß Kommandoskripts in ein Skriptdepot geladen bzw. aus diesem abgerufen werden. Dadurch können häufig verwendete Kommandoskripts schneller aufgefunden werden.

Nach einer anderen Ausführungsform der Erfindung kann vorgesehen sein, daß Zugriffskommandos zunächst automatisch durch Auslesen der Datentypen bzw. der Managed Objects eines bereits programmierten Kommunikationsnetzes erzeugt werden und in einem speziellen Kommandoskript, einem sogenannten Template zusammengefaßt werden und dieses Template zur Programmierung eines weiteren Kommunikationsnetzes verwendet wird. Dadurch kann ein neu installiertes Kommunikationsnetz, z.B. eine neue Nebenstellenanlage in einem Schritt durch Übernahme der Programmstruktur einer bestehenden Anlage automatisch programmiert werden.

In Weiterbildung der Erfindung kann vorgesehen sein, daß die Übersetzung der Zugriffskommandos und die Ausführung der Managementkommandos durch einen Server-Prozess umgesetzt werden und Benutzereingaben von einem Client-Prozess eingelesen werden, wobei der Client-Prozess und der Server-Prozess über eine Schnittstelle miteinander kommunizieren. Dadurch wird erreicht, daß der Client-Prozess und Server-Prozess auf örtlich getrennten Computern laufen können. Dies ist besonders bei der Fernwartung von Telekommunikationsanlagen vorteilhaft, da die Wartung einer örtlich entfernten Telekommunikationsanlage über einen lokal laufenden Client-Prozess erfolgen kann, der mit einem auf einem mit der Telekommunikationsanlage verbundenen Computer laufenden Server-Prozess kommuniziert.

Nach einer anderen Ausgestaltung der Erfindung kann vorgesehen sein, daß der Client-Prozess und der Server-Prozess über ein Netzwerkprotokoll, insbesondere über das im Internet relevante TCP/IP Protokoll miteinander kommunizieren. Dies bietet den Vorteil, daß die Verbindung der Computer, auf denen der Client-Prozess und der Server-Prozess laufen über ein Datennetzwerk wie das Internet erfolgen kann.

In weiterer Ausgestaltung der Erfindung kann vorgesehen sein, daß die Kommunikation zwischen dem Client-Prozess und dem Server-Prozess über einen Web-Server erfolgt. Dies erlaubt es, die Fernwartung einer Telekommunikationsanlage über einen beispielsweise als Browser-Applikation laufenden Client-Prozess durchzuführen.

In weiterer Ausgestaltung der Erfindung kann vorgesehen sein, daß der Status des Server-Prozesses regelmäßig vom Client-Prozess abgefragt wird. Damit kann der Benutzer auch bei Einwegverbindungen, bei denen keine Anfrage des Server-Prozesses an den Client-Prozess möglich sind, sofort über den Status des Server-Prozesses informiert werden und gegebenenfalls zur Eingabe entsprechender Anweisungen aufgefordert werden.

Weiters sind Computer und ein Computerprogramm bekannt, das auf einem computergeeigneten Medium gespeichert ist bzw. das in den internen Speicher eines Computers geladen werden kann.

Diese Computer und Computerprogramme sind im Zusammenhang mit der Programmierung von Telekommunikationsanlagen aus dem Stand der Technik bekannt, weisen aber allesamt die weiter oben im Zusammenhang mit einer bekannten Programmierlogik genannten Nachteile bei der automatischen Ausführung von Konfigurationsarbeiten auf.

Es ist folglich eine weitere Aufgabe der Erfindung, ein Computersystem der angeführten Art vorzustellen, das diese Nachteile beseitigen und das die automatische, von einer bestimmten Benutzeroberfläche unabhängige Programmierung von Telekommunikationsanlagen ermöglicht.

In diesem Zusammenhang betrifft die Erfindung ein Computersystem auf welchem eine Anwendung läuft. Die der Erfindung zugrunde liegende Aufgabe wird dadurch gelöst, daß es die Anwendung die Programmierung bzw. das Management eines Kommunikationsnetzes gemäß einer Programmlogik nach einem der Ansprüche 1-9 umsetzt.

In weiterer Folge betrifft die Erfindung ein Computersystem, auf welchem ein Client-Prozess läuft, der mit einem, vorzugsweise auf einem anderen Computersystem laufendem Server-Prozess über eine Schnittstelle kommuniziert. Die der Erfindung zugrunde liegende Aufgabe wird dadurch gelöst, daß der Server-Prozess die Übersetzung der Zugriffskommandos und die Ausführung der Managementkommandos zur Programmierung bzw. zum Management eines Kommunikationsnetzes gemäß einer Programmlogik nach einem der Ansprüche 6-9 umsetzt.

In weiterer Folge betrifft die Erfindung ein Computersystem auf welchem ein Server-Prozess läuft, der mit einem, vorzugsweise auf einem anderen Computersystem laufendem Client-Prozess über eine Schnittstelle kommuniziert. Die der Erfindung zugrunde liegende Aufgabe wird hier dadurch gelöst, daß der Server-Prozess die Übersetzung der Zugriffskommandos und die Ausführung der Managementkommandos zur Programmierung bzw. zum Management eines Kommunikationsnetzes gemäß einer Programmlogik nach einem der Ansprüche 6-9 umsetzt.

Die Erfindung wird unter Bezugnahme auf die beigeschlossenen Zeichnungen, in welchen besonders bevorzugte Ausführungsbeispiele dargestellt sind, näher beschrieben. Dabei zeigt:

Fig. 1 eine Prinzip-Skizze der Hardware-Struktur eines typischen Kommunikationsnetzes bzw. einer typischen Nebenstellenanlage 1;

Fig. 1a den Aufbau einer typischen Nebenstellenanlage 1;

Fig. 1b den Aufbau einer typischen Hardware-Einheit 6;

Fig. 2 eine Prinzip-Skizze des Aufbaus der Steuerungssoftware einer typischen Nebenstellenanlage 1;

Fig. 3 eine weitere Prinzip-Skizze der Software-Struktur einer typischen Nebenstellenanlage 1 beim Anschluß an ein externes Computersystem 24.

Fig. 3a die Struktur einer im Stand der Technik bekannten Programmierumgebung für Kommunikationsnetze;

Fig. 4 die Struktur eines erfindungsgemäßen Datentyps 52;

Fig. 4a eine vereinfachte Skizze der hierarchischen Struktur mehrerer erfindungsgemäßer Datentypen 52;

Fig. 5 die Struktur einer weiteren im Stand der Technik bekannten Programmierumgebung für Kommunikationsnetze;

Fig. 5a eine weitere Skizze der hierarchischen Struktur mehrerer erfindungsgemäßer Datentypen 52;

Fig. 6 die Struktur einer erfindungsgemäßen Programmierumgebung für Kommunikationsnetze;

Fig. 7 ein Objekt-Netz einer erfindungsgemäßen Skript-Umgebung für die Programmierung von Kommunikationsnetzen;

Fig. 8a den zeitlichen Ablauf bei der Abarbeitung eines erfindungsgemäßen Kommandoskripts 50;

Fig. 8b den Ablauf der Wechselwirkung mit einem Benutzer 41 für den Fall, daß eine Eingabe des Benutzers 41 erforderlich ist; und

Fig. 8c den Ablauf der Wechselwirkung mit einem Benutzer 41 für den Fall, daß ein Fehler bei der Ausführung eines Kommandoskripts 50 aufgetreten ist.

1 Einleitung

1.1 Kommunikationsnetze

Die Erfindung steht im Zusammenhang mit der Wartung von Kommunikationsnetzen wie z.B. Telekommunikationsanlagen bzw. Nebenstellenanlagen 1 in Firmennetzen.

Mit Telekommunikationsanlage wird dabei ein Zusammenschluß von mehreren Hardware-Einheiten bzw. Ressourcen 6 zu einem zentral steuerbaren System in der Telekommunikation bezeichnet. Solche Telekommunikationsanlagen sind in den verschiedensten Bereichen der Telekommunikation eingesetzt. Als Beispiel seien hier genannt die Integration von Telefondiensten mit Computerdiensten (engl: Computer Telephony Integration, CTI), die mobile Kommunikation im Haus, die Bereitstellung von Nebenstellenfunktionen in UMTS Netzen (Mobile Enterprise Projekte), oder die Sprachübertragung über Datenkanäle (engl: Voice over IP) in lokalen Netzen (engl: Lokal Area Network, LAN). In diesen Fällen umfassen die Hardware-Einheiten auch die einzelnen Server und Zugriffsknoten (engl: access nodes).

Der Begriff Kommunikationsnetz ist noch etwas weiter und bezeichnet im folgenden jedweden Zusammenschluß von mehreren Ressourcen 6 zu einem zentral steuerbaren System. In diesem Sinn sind auch allgemeine Systeme von miteinander kommunizierenden Ressourcen 6 wie z.B. ein Computernetz oder auch ein einzelner Computer als Kommunikationsnetz im Sinn der Erfindung zu verstehen. Als Ressourcen 6 sind in diesem Sinn sowohl physikalische Gegenstände wie beispielsweise Hardware-Einheiten oder Teile einer Kommunikationsausrüstung als auch logische Gegenstände wie z.B. Programme oder Kommunikationsverbindungen zu sehen.

1.2 Nebenstellenanlagen

Als ein spezielles Beispiel eines Kommunikationsnetzes, das in den Anwendungsbereich der Erfindung fällt, sollen im folgenden Nebenstellenanlagen 1 näher beschrieben werden. Diese stellen jedoch nur eines von vielen möglichen Beispielen für ein Kommunikationsnetz dar. Das erfindungsgemäße Verfahren kann zur Programmierung bzw. zum Management von allgemeinen Kommunikationsnetzen verwendet werden.

Im allgemeinen umfassen Nebenstellenanlagen 1 Hardware-Einheiten bzw. Ressourcen 6 unterschiedlicher Kategorien. So umfassen typische Nebenstellenanlagen neben den eigentlichen Telefonapparaten bzw. sonstigen Endgeräten wie Displays, Kameras, Mikrofone oder sonstigen Ein-/Ausgabegeräten im allgemeinen noch weitere, für den Betrieb der Nebenstellenanlage 1 wichtigen Anlagen, wie z.B. Computeranschlüsse, etc.

Dabei existieren Nebenstellenanlagen 1 in sehr verschiedenen Größenordnungen. Angefangen von kleinen Anlagen mit etwa zehn Anschlüssen für den Gebrauch in Wohnungen und Einfamilienhäusern, über firmeninterne Anlagen, wie z.B. die ASB 150 02 BusinessPhone Systeme der Ericsson GmbH mit bis zu etwa 300 Teilnehmern, existieren Nebenstellenanlagen 1 bis hin zu großen, im folgenden als Corporate Network bezeichneten, Anlagen mit bis zu 10000 in unterschiedlichen Netzen verteilten Ressourcen 6.

1.2.1 Hardware

Im allgemeinen weisen Nebenstellenanlagen 1 eine Hardware-Struktur auf, die der in Fig. 1 skizzierten ähnlich ist. Dabei erfolgt die Steuerung der Anlage über eine zentrale Funktionskarte, welche unter anderem einen zentralen Prozessor CPU 2, die Systemuhr 5, einen Bus-Controller 7 und die Stromversorgung 8 enthält. Ein weiteres Kernstück der zentralen Hardware ist das Koppelfeld 7' (engl. switch), welches das Durchschalten der einzelnen Eingänge und Ausgänge der Nebenstellenanlage 1 kontrolliert.

Diese zentrale Hardware kommuniziert über ein Bussystem 3 mit der regionalen Hardware bzw. den einzelnen Ressourcen 6. Als regionale Hardware können unterschiedliche Funktionskarten bzw. Ressourcen 6 mit dem System verbunden werden. Sie enthalten meist einen regionalen Prozessor 2' zur Steuerung der Funktionskarte sowie Schnittstellen zu Nebenstellen, Übertragungskanälen etc.

Fig. 1a zeigt etwas ausführlicher die typische Struktur eines BusinessPhone Systems. Die Nebenstellenanlage1 umfaßt neben einer zentralen CPU 2 mehrere Hardware-Einheiten bzw. Ressourcen 6. Fig. 1a zeigt dabei lediglich die Anschlußbelegung. Den unterschiedlichen Anschlüsse zur CPU-D4 entspricht eine reale zentrale CPU 2. Die Hardware-Einheiten sind insbesondere durch Funktionskarten wie z.B. gedruckten Schaltkreisen, Leiterplatten etc. gebildet. Die Funktionskarten weisen Anschlüsse bzw. Schnittstellen zu den unterschiedlichen Endgeräten wie Fax- und Telefonapparaten oder auch zu Übertragungskanälen wie ISDN-Abschlüssen oder Ethernet-Anschlüssen auf.

Die Hardware-Einheiten bzw. Ressourcen 6 sind im allgemeinen unterschiedlich ausgestaltet und weisen je nach Verwendungszweck unterschiedliche Funktionsumfänge auf, wobei mehrere Ressourcen 6 der selben Kategorie in einer Nebenstellenanlage 1 integriert werden können. Beispielsweise können mehrere gleichartige Funktionskarten für den Anschluß analoger Telefonapparate vorgesehen sein. Weiters können Zwischenknoten vorgesehen sein, die die Steuerung von weiteren Hardware-Einheiten bzw. Ressourcen 6 gewährleisten und somit eine physikalische und logische Baumstruktur der Nebenstellenanlage definieren.

Die einzelnen Hardware-Einheiten bzw. Ressourcen 6 kommunizieren in Fig. 1a über das Bussystem 3. Das Bussystem 3 umfaßt beispielsweise einen Synchronisationsbus (Clock-Bus), einen Funktions-Kontrollbus (FC-Bus), einen System-Kommunikationsbus (SC-Bus) und einen PCM-Bus zur Sprachübertragung.

Die Verbindung der Hardware-Einheiten bzw. Ressourcen 6 zu weiteren Endgeräten erfolgt über allgemeine Leitungen 4. Als Leitungen 4 gelten in diesem Sinn gewöhnliche analoge Telefonleitungen, ISDN Leitungen oder auch spezielle Steuerleitungen. Die Steuerleitungen für O&M Aufgaben bzw. für Applikationen können beispielsweise über bekannte V.24 Verbindungen realisiert sein. Bei digitalen Netzen kommt vorzugsweise die X.21 Empfehlung zur Anwendung. Darüberhinaus sind andere Leitungen 4 wie Glasfaserkabel bzw. Lichtwellenleiter etc. möglich.

Statt über ein Bussystem 3, bzw. zusätzlich dazu kann auch die Verbindung der einzelnen Hardware-Einheiten untereinander über die Leitungen 4 erfolgen. Die Verbindung zwischen den Hardware-Einheiten bzw. Ressourcen 6 bzw. zwischen größeren Gruppen von Hardware-Einheiten bzw. Ressourcen 6 in Corporate Network Umgebungen kann auch über Funk bzw. Satellitenübertragungssysteme erfolgen. Weiters kann die Verbindungsstruktur ringförmig (Token-Ring) mit fest vorgegebener Umlaufrichtung oder sternförmig sein. Es können unterschiedliche Busstrukturen vorgesehen sein. Insbesondere bei verteilten Systemen stellt die Verbindungsstruktur der Nebenstellenanlage 1 für gewöhnlich eine Mischstruktur aus mehreren Topologien dar.

Fig. 1b zeigt den Aufbau einer Hardware-Einheit des weiter oben beschriebenen BusinessPhone Systems. Die skizzierte Karte ELU-A 61 erlaubt die Verbindung mit bis zu 16 analogen Telefonapparaten 8. Die Karte enthält neben einem Bus-Controller für das Bussystem 3 einen regionalen Prozessor 2' sowie RAM und EPROM Speicherbausteine 9, 10.

Weiters ist ein analog/digital PCM Dekoder (DSLAC) 11 bzw. eine Codek-Funktion und ein DTMF Empfänger 12 vorgesehen.

1.2.2 Software

Die Struktur der Systemsoftware einer Nebenstellenanlage ist in Fig. 2 skizziert. Die Systemsoftware ist in Abstimmung mit der Hardware strukturiert. Ein zentrales Programm 13 kontrolliert die Schaltlogik und übernimmt die allgemeinen Steuerungsaufgaben der Anlage. Dieses zentrale Programm 13 ist in kleinere Programm-Einheiten 14 mit eigenen Datenbereichen 15 unterteilt, welche jeweils für sich klar definierte Aufgaben übernehmen, wie. z.B. die Steuerung der Telefonanschlüsse oder der Anschlüsse zu den externen Übertragungswegen, die Auswertung der Gesprächsstatistiken etc.

Die Funktionalität der Hardware-Einheiten bzw. Ressourcen 6 wird jeweils von einem oder mehreren regionalen Programmen 16 gesteuert, welche ebenfalls wieder in regionale Programm-Einheiten 17 mit eigenen Datenbereichen 18 unterteilt sein können.

Die Kommunikation der Programm-Einheiten 14, 17 untereinander erfolgt über den Austausch von Nachrichten 19 (Signalen). Dabei übernimmt ein zentrales Betriebssystem 20 die Verteilung der Nachrichten 19 und gegebenenfalls die Bereitstellung und Überwachung einer oder mehrerer Warteschlangen 21 für die Nachrichten 19.

Es sind aber auch Strukturen möglich, bei denen der Datenaustausch unabhängig von einem zentralen Betriebssystem 20 erfolgt. Die für den Signalaustausch erforderliche Logik ist in diesen Fällen in allen regionalen Programm-Einheiten 17 mit implementiert.

Besonders vorteilhaft ist es, wenn die Kommunikation der Programm-Einheiten 14, 17 untereinander über genau definierte, offene Schnittstellen 23' erfolgt. Dies ermöglicht die Integration von neuen Hardware-Einheiten bzw. Ressourcen 6 unterschiedlicher Hersteller in die Telekommunikationsanlage 1 (Multi-Vendor), wobei die Schnittstelle 23' die Kommunikation mit den auf den neuen Hardware-Einheiten bzw. Ressourcen 6 laufenden regionalen Programm-Einheiten 17 ermöglicht.

2 Programmierung von Kommunikationsnetzen

2.1 Zugriff auf die Funktionsdaten

Zunächst soll der Begriff der Programmierung näher erläutert werden. Unter Programmierung ist auf keinen Fall die Erstellung der eben beschriebenen Systemsoftware einer Nebenstellenanlage 1 zu verstehen. Vielmehr wird mit Programmierung eines Kommunikationsnetzes der Zugriff auf die Funktionsdaten des Kommunikationsnetzes bzw. der einzelnen Ressourcen 6 zur Definition bzw. Veränderung der Funktionalität einzelner Ressourcen 6 oder des gesamten Kommunikationsnetzes bezeichnet. Im Sinne des weiter unten beschriebenen OSI System Management ist unter Programmierung somit auch das Management bzw. das System Management des Kommunikationsnetzes zu verstehen.

Der Begriff Funktionsdaten 25 steht dabei sowohl für Programme, welche in die verschiedenen Hardware-Einheiten bzw. Ressourcen 6 geladen werden können, beispielsweise zentrale und regionale Programme 13, 16 und Programm-Einheiten 14, 17, als auch für sonstige, die Funktionalität der Hardware-Einheiten bzw. Ressourcen 6 beeinflussende Variablen, wie z.B. die zentralen und regionalen Datenbereiche 15, 18. Der Begriff programmierbare Funktionsdaten 25 gibt an, daß diese Daten softwaremäßig geändert werden können.

Bei gängigen Nebenstellenanlagen 1 ist die Systemsoftware transparent, d.h. die Nebenstellenanlage 1 bietet ein Interface bzw. eine Schnittstelle 23 an, über welches die Ressourcen 6 bzw. die Funktionsdaten 25 über ein bestimmtes Protokoll angesprochen werden können.

Dieser aus Sicht des Anwenders unmittelbare Zugriff auf die Ressourcen 6 stellt die einfachste Form der Programmierung in der Terminologie der Erfindung dar. Bei diesem einfachsten Verfahren zur Programmierung der Nebenstellenanlage 1 muß der Anwender nicht über die interne Datenverarbeitung der Nebenstellenanlage informiert sein, was eine von der Systemsoftware unabhängige Programmierung der Kommunikationsanlage erlaubt.

Nachteilig bei diesem einfachen Verfahren zur Programmierung ergibt sich unter anderem, daß der Anwender sehr eng an die physikalische Struktur der einzelnen Ressourcen 6 gebunden ist. Weiters kann die Programmierung der Nebenstellenanlagen 1 nur online, d.h. bei unmittelbar angeschlossener Nebenstellenanlage 1 erfolgen.

2.2 Zugriff auf ein externes Abbild der Funktionsdaten

Aus diesem Grund erfolgt die Programmierung gängiger Nebenstellenanlagen 1 meist nicht unmittelbar, sondern über den Zugriff auf ein externes Abbild 26 der Funktionsdaten 25. Bei diesem in Fig. 3 skizzierten Verfahren werden die Ressourcen 6 nicht direkt angesprochen. Bei der Programmierung wird in einem ersten Schritt das externe Abbild 26 der Funktionsdaten 25 verändert und dieses Abbild 26 in einem späteren Schritt über die Schnittstelle 23 auf die Nebenstellenanlage 1 übertragen.

Auf diese Weise kann die Programmierung auch offline, d.h. ohne unmittelbare Verbindung mit der Nebenstellenanlage 1 anhand des Abbildes 26 erfolgen. Die Funktionsdaten 25 können zu einem späteren Zeitpunkt,, wenn die Nebenstellenanlage 1 angeschlossen ist, übertragen werden. Ein weiterer Vorteil ergibt sich aus dem wesentlich einfacheren Zugriff auf die Daten. Da sämtliche Funktionsdaten 25 in Form des Abbildes 26 vorliegen, kann der Zugriff über geeignete Datenbankroutinen oder bekannte Oberflächen erfolgen und somit wesentlich einfacher und schneller als über das von der Schnittstelle 23 bereitgestellte Protokoll. Der Zugriff wird somit gegenüber der direkten Programmierung wesentlich erleichtert.

Die Bereitstellung eines eigenen Abbildes 26 aller programmierbaren Funktionsdaten 25 für die Nebenstellenanlage 1 auf dem externen Computersystem 24 ist für die Programmierung aber auch z.B. für die Sicherung der Funktionsdaten 25 vorteilhaft.

Besonders vorteilhaft für die Durchführung des eben beschriebenen Verfahrens ist es, wenn die Funktionsdaten 25 für die einzelnen Hardware-Einheiten bzw. Ressourcen 6 wie in Fig. 3 skizziert in einem oder mehreren zentralen Speicherbereichen 22 abgelegt sind. Über die Schnittstelle 23 können diese Funktionsdaten 25 von einem externen Computersystem 24 somit besonders einfach gelesen bzw. neue Funktionsdaten 25 von dem externen Computersystem 24 auf die Nebenstellenanlage 1 übertragen werden. Die Abbildung der Funktionsdaten 25 in einem zentralen Speicherbereich 22 innerhalb der Telekommunikationsanlage 1 ist aber nicht zwingend. Es ist beispielsweise auch möglich, daß die Funktionsdaten 25 unmittelbar von den Speicherbausteinen, z.B. von RAM oder EEPROM Speicherbausteinen 9, 10 der Hardware-Einheiten bzw. Ressourcen 6 auf das externe Computersystem 24 übertragen bzw. von diesem auf die Hardware-Einheiten bzw. Ressourcen 6 der Nebenstellenanlage 1 überspielt werden.

Das externe Computersystem 24 ist üblicherweise eine Datenverarbeitungsanlage bzw. ein Computer, Großrechner, Notebook, etc. Es kann sich aber auch lediglich um ein System programmierbarer Logikbausteine, wie eine speicherprogrammierbare Steuerung, oder aber lediglich um ein Speichersystem wie eine externe Festplatte oder ähnliches handeln.

Die Schnittstelle 23, über die die gespeicherten Funktionsdaten 25 mit dem externen Computersystem 24 ausgetauscht werden, erlaubt, daß die Telekommunikationsanlage mit unterschiedlicher Software kommuniziert.

Die zentrale Programmierung erfolgt gemäß dem eben beschriebenen Verfahren über den Zugriff auf das Abbild 26 der programmierbaren Funktionsdaten 25. Die in einem zentralen Speicherbereich 22 gespeicherten Funktionsdaten 25 werden zunächst auf einen externen Datenträger übertragen. Das externe Abbild 26 der programmierbaren Funktionsdaten 25 kann anschließend modifiziert werden. Die Programmierung der Hardware-Einheiten bzw. Ressourcen 6 erfolgt dadurch, daß einzelnen Variablen, d.h. Speicherbereichen innerhalb des Abbilds 26 der programmierbaren Funktionsdaten 25, bestimmte Werte zugewiesen werden. In einem nächsten Schritt kann das modifizierte Abbild 26 der programmierbaren Funktionsdaten 25 in den zentralen Speicherbereich 22 zurück geladen werden.

Der Zugriff auf das Abbild 26 erfolgt, wie in Fig. 3a skizziert, beispielsweise über eigene Benutzeroberflächen 30. Mit Hilfe dieser Benutzeroberfläche 30 kann jeder Variable der gewünschte Wert zugewiesen werden.

Diese dialoggesteuerte Dateneingabe ist bei kleinen Systemen durchaus ausreichend. Bei Systemen mit einer großen Anzahl an Hardware-Einheiten bzw. Ressourcen 6 ist die Zuweisung der entsprechenden Werte für alle Variablen aller Hardware-Einheiten bzw. Ressourcen 6 jedoch zu umständlich. Insbesondere bei Nebenstellenanlagen mit mehreren hundert Hardware-Einheiten bzw. Ressourcen 6 wie im BusinessPhone Bereich bzw. im Corporate Network Bereich mit bis zu 10000 untereinander vernetzten Hardware-Einheiten bzw. Ressourcen 6 ist eine effizientere Methode der Dateneingabe unumgänglich.

Aus diesem Grund existieren bei bekannten Systemen Benutzeroberflächen 30 mit Aufnahmefunktionen, die sämtliche Aktionen des Benutzers aufzeichnen. Das Funktionsprinzip dieser dialogbasierten Programmierung ist ebenfalls in Fig. 3a skizziert. Das Abbild 26 aller programmierbaren Funktionsdaten 25 der Telekommunikationsanlage 1 wird in der Datenbank TS verwaltet, welche über eine graphische Benutzeroberfläche (GUI) 30 angesprochen wird. Die vom Benutzer durchgeführten Kommandos und Eingaben können von einem Makrorecorder (MR) 29 aufgezeichnet werden. Durch das Vorsehen dieses Makrorecorders 29 ist es möglich, die Eingaben des Benutzers, die während der Konfiguration einer Hardware-Einheit 6 getätigt worden sind, aufzuzeichnen und als Makro abzuspeichern. Für die Bereitstellung der Funktionsdaten 25 für eine andere Hardware-Einheit der selben Kategorie kann das gespeicherte Makro abgerufen werden. Dabei gibt das Makro genau an, in welches Feld des Dialogs bzw. der Benutzeroberfläche 30 welcher Wert eingetragen wird. Weiters werden sonstige Aktionen des Benutzers, beispielsweise das Drücken einer bestimmten Funktionstaste auf der Tastatur, aufgezeichnet.

Bei diesen im Stand der Technik bekannten Methoden zur Automatisierung häufig vorkommender Programmierarbeiten werden Sitzungen (engl: sessions) von der Benutzeroberfläche 30 aufgezeichnet, während die betreffenden Aktionen vom Benutzer ausgeführt werden. Diese Sitzungen werden in einer Datei gespeichert und können später abgerufen und ausgeführt werden.

Ein typisches Makro für die weiter oben beschriebene Nebenstellenanlage 1 ist beispielsweise im folgenden angegeben.

```
! AUSTRIA: Trunk session: 2 wire tieline
! .comments .
trunk:1990:0000:F=ENTER
IDBOX:Identity:0000:S=
1600:1607:0000:M=12 kHz
1600:1609:0000:M=6
1600:1610:0000:M=-7
1600:1611:0000:M=0
1600:1612:0000:M=3
1600:1613:0000:M=0
1600:0000:0000:F=FN6
```

Nachteilig bei dieser bekannten Methode der Makroprogrammierung ist, daß die Makros sich ausschließlich auf die Dialogmaske der Benutzeroberfläche 30 beziehen. Bei einer Änderung der Dialogmaske, was beispielsweise bei der Bereitstellung einer neuen Hardware-Einheit mit neuen Funktionen notwendig sein kann, können die alten Makros nicht mehr verwendet werden.

2.3 Abstraktion der Funktionsdaten

Als besonders vorteilhaft für die Programmierung von größeren Kommunikationsnetzen erweist sich die Repräsentation jeder Ressource 6 in Form eines eigenen abstrakten Datentyps 52. Dabei wird ein eigener Datentyp 52, beispielsweise in Form einer Klassendefinition für jede Kategorie der verwendeten Hardware-Einheiten bzw. Ressourcen 6 bereitgestellt. Für jede reale Ressource 6 wird anschließend ein Exemplar 53 des Datentyps 52 erzeugt.

Diese Vorgehensweise erlaubt eine einfache Strukturierung der Funktionsdaten 25 aller Hardware-Einheiten bzw. Ressourcen 6 des gesamten Kommunikationsnetzes. Insbesondere wird der Zugriff auf die Hardware-Einheiten bzw. Ressourcen 6 durch die eindeutige Zuordnung zwischen der Kategorie der Hardware-Einheiten bzw. Ressourcen 6 und dem zugehörigen Datentyp 52 deutlich erleichtert.

Für die oben beschriebene Nebenstellenanlage 1 kann beispielsweise ein Datentyp 52 für alle Hardware-Einheiten bzw. Ressourcen 6 der Kategorie "analoger Anschluß einer Nebenstelle" auf den unterschiedlichen analogen karten ELU-A 61 bereitgestellt werden. Für den Zugriff auf eine reale Nebenstelle muß ein Exemplar 53 des Datentyps 52 erzeugt werden (Fig. 4).

Üblicherweise folgt die Repräsentation des Datentyps 52 einer objektorientierten Klassenvereinbarung. Dies bietet den Vorteil, daß die Prozedurvereinbarungen für die Operationen mit den Objekten in die Typvereinbarung gelegt werden können. Die Klassen umfassen eine Attributvereinbarung und eine Methodenvereinbarung. Diese formale Definition eines Datentyps 52 dient als Vorlage, aus der ein Exemplar 53 des Datentyps 52 zur Laufzeit erzeugt wird. Die Klasse definiert die Eigenschaften des Datentyps 52 und die Methoden, die zur Steuerung des Objektverhaltens verwendet werden. Vorzugsweise sind die einzelnen Klassendefinitionen in Klassenbibliotheken zusammengefaßt und in Klassenbibliotheksdateien abgespeichert.

Für gewöhnlich sind die Datentypen 52 weiters in einer Klassenhierarchie zusammengefaßt. Damit können gemeinsame Eigenschaften ähnlicher Datentype 52 leicht zusammengefaßt werden und es können Codeverdopplungen zwischen ähnlichen Klassen weitgehend vermieden werden. So sind beispielsweise die Gemeinsamkeiten aller Klassen in einer Wurzelklasse zusammengefaßt, weitere Hauptklassen umfassen beispielsweise die Gemeinsamkeiten aller Kontroller, Telefonkarten etc. (Fig. 4a).

Die Repräsentation der Kategorien der Hardware-Einheiten 6, d.h. die Bereitstellung geeigneter Datentypen 52, kann durch die erwähnte Definition objektorientierter Klassen oder aber auch durch die Definition anderer abstrakter Datentypen gängiger modularer Programmiersprachen erfolgen. In modularen Programmiersprachen können die abstrakte Datentypen 52 durch Module, beispielsweise über *struct* oder *typedef* Anweisungen, realisiert werden. Auch in diesem Fall wird für jede reale Ressource 6 ein Exemplar 53 des abstrakten Datentyps 52 erzeugt.

2.3.1 MO Managed Objects

Besonders im Telekombereich geht die Entwicklung in die Richtung einer möglichst umfassenden Standardisierung der Programmierung bzw. des Managements der Netze und der Dienste. Dabei sollen Managementsysteme möglichst interoperable Software-Werkzeuge bereitstellen, die es ermöglichen, heterogene Multivendorsysteme einfach und zentral zu verwalten. Insbesondere im Corporate Network Bereich mit bis zu 10000 untereinander vernetzten Hardware-Einheiten bzw. Ressourcen 6 ist für die zentrale Konfiguration der Hardware-Einheiten bzw. Ressourcen 6 eine einheitliche Managementumgebung von großer Bedeutung. Im Zuge dieser Bemühungen nach Vereinheitlichung wurde in den ITU-T Empfehlungen der X.700-Reihe die OSI Systems Management Norm definiert, welche Verfahren und Mechanismen für das Management von Kommunikationsnetzen festlegt. Die OSI-Managementumgebung stellt dabei die Mittel zur Kontrolle, Koordination und Beobachtung der Ressourcen 6 zur Verfügung. In diesem Zusammenhang wird insbesondere auf den Inhalt der Dokumente ITU-T X.700, ITU-T X.720 und ITU-T X.722 bezug genommen.

Dem OSI Systems Management liegt eine objektorientierte Betrachtungsweise zugrunde, die auf sogenannten Managed Objects 27 (MO) aufbaut. Diese MOs 27 sind ganz allgemein als Abstraktion einer physikalischen oder logischen Ressource 6 zu verstehen, also beispielsweise von bestimmten Speicherbausteinen als auch von Softwaremodulen etc. Ein Managed Object 27 wird durch seine Attribute, die Operationen, die an ihm ausgeführt werden, die von ihm ausgesendeten Meldungen und durch die Relationen zu anderen MOs 27 beschrieben. Somit ist ein dem OSI Systems Management entsprechendes MO 27 als eine spezielle Form eines weiter oben beschriebenen Datentyps 52 zu sehen, durch welchen physikalische oder logische Ressourcen 6 einer bestimmten Kategorie dargestellt werden können.

Für die beschriebene Nebenstellenanlage 1 existiert beispielsweise ein eigenes MO 27 für alle Ressourcen 6 der Kategorie "analoger Anschluß einer Nebenstelle". In diesem einfachsten Fall repräsentiert ein MO 27 alle Managementaspekte einer physikalischen Ressource 6. Es können aber auch übergeordnete Hierarchien, wie z.B. eine gesamte ELU-A Karte, durch ein MO 27 Objekt repräsentiert sein.

Besonders vorteilhaft erweist sich im Sinne der obigen Überlegungen die Definition der einzelnen MOs 27 als objektorientierte Klassen, insbesondere als Java-Klassen, welche eine weitgehende Unabhängigkeit vom verwendeten Betriebssystem ermöglichen. Die MOs 27, welche der weiter oben genannten vorzugsweise verwendeten Klassenhierarchie folgen, teilen sich beispielsweise in ein Wurzel-MO, mehreren Hauptbereichs-MOs (engl: top-level) für die Controller, die Directorys, Extension Manager, Funktionskarten etc. und einer Anzahl von Betriebsmittel-MOs auf.

2.3.2 Managementkommandos

Der Zugriff auf die MOs 27 erfolgt über eigene Managementkommandos 51'. Die Definitionen der Managementkommandos 51' werden vorzugsweise in die objektorientierte Definitionen der Datentypen 52 bzw. der MOs 27 integriert, beispielsweise über eine

Methodendefinition der Java-Klassen der MOs 27. Es ist aber auch möglich, die Managementkommandos 51' über eine allgemeine Schnittstellendefinitionssprache (IDL) zu definieren.

2.3.3 Instanz

Für jede reale Ressource 6 wird ein Exemplar 53 bzw. eine Instanz des entsprechenden Datentyps 52 bzw. MO Objektes 27 erzeugt. Die Exemplare 53 repräsentieren die Hardware-Einheiten bzw. Ressourcen 6 der Nebenstellenanlage 1. Beispielsweise wird für jede ELU-A Karte 61 ein Exemplar 53 des ELU-A Datentyps 52 bzw. des ELU-A MOs 27 erzeugt. Somit kann über das entsprechende Exemplar 53 auf jede Ressource 6 der Nebenstellenanlage 1 zugegriffen werden. Damit sind auch sämtliche Funktionsdaten 25 einer Ressource 6 über das entsprechende Exemplar der der Kategorie der Ressource 6 entsprechenden objektorientierten Klasse zugänglich.

Bei der Verwendung anderer abstrakter Datentypen 52 wird das Exemplar 53 auf entsprechende Weise durch die Deklaration einer dem Datentyp 52 entsprechenden Struktur erzeugt.

2.3.4 Management Interface

Die den einzelnen Ressourcen 6 zugeordneten Exemplare 53 der Datentypen 52 bzw. der MOs 27 müssen von speziellen Übersetzungsprogramme zur Übersetzung des abstrakten Abbildes 26 der Funktionsdaten 25 in geeignete, durch die Nebenstellenanlage 1 lesbare Daten, ansprechbar sein. Nur so ist gewährleistet, daß die durch Zugriff auf die MOs 27 geänderten Funktionsdaten 25 in die Nebenstellenanlage 1 übertragen werden können. Diese Übersetzungsprogramme bilden das sogenannte Management Interface und stellen eine Verallgemeinerung der Schnittstelle 23 dar. Die Übertragung zwischen MO 27 und Kommunikationsnetz bzw. Nebenstellenanlage 1 erfolgt für den Benutzer transparent. Somit kann die MIB im wesentlichen als ein Proxy angesehen werden, welcher den proprietären Datenaustausch mit dem Kommunikationsnetz verbirgt.

Die Kommunikation zwischen den MO Objekten 27 und den realen Betriebsmittel in den Hardware-Einheiten bzw. Ressourcen 6 ist dabei sowohl für den Programmierer der Nebenstellenanlage 1 als auch für den Anwendungsprogrammierer transparent. Während die Sitzungen bei bekannten Systemen mit den Datenstrukturen in den Hardware-Einheiten bzw. Ressourcen 6 eng verbunden sind, macht das MO Objekt 27 bzw. die mit ihm verbundenen Zugriffskommandos 51 diese Strukturen für den Programmierer transparent.

2.3.5 MIB

Sämtliche MO Objekte 27 sind in einem mit Management Information Base (MIB) bezeichneten Objektkatalog 28 bzw. einer Sammlung zusammengefaßt. Wenn eine Hardware-Einheit bzw. Ressource 6 geladen wird, wird das ihr zugerechnete Managed Objekt 27 in dem Objektkatalog MIB 28 gespeichert. Dies kann online oder offline erfolgen, d.h. unabhängig davon, ob die Telekommunikationsanlage 1 mit dem externen Computersystem

24 verbunden ist oder nicht. Damit sind sämtliche programmierbaren Funktionsdaten 25 der Telekommunikationsanlage 1 in dem Objektkatalog MIB 28 wiedergegeben. Dieser Objektkatalog MIB 28 stellt somit ein abstraktes Abbild 26 der programmierbaren Funktionsdaten 25 dar.

Fig. 5 zeigt die Prinzip-Skizze eines Programmierverfahrens mit Hilfe der beschriebenen Abstraktion der Ressourcen 6 durch zugehörige Datentypen 52 bzw. MOs 27. Die Management Information Base MIB stellt dabei ein abstraktes Abbild aller physikalischen und logischen Ressourcen 6 des Kommunikationsnetzes bereit. Gleichzeitig werden die möglichen Managementkommandos 51' auf die MOs 27 bereitgestellt.

Der Zugriff auf die MOs 27 erfolgt über eine graphische Oberfläche 30', wobei vorzugsweise eine eigene Zwischenschicht zur Kommunikation zwischen der graphischen Oberfläche 30'und der MIB vorgesehen ist. Diese im folgenden mit Presentation Layer bezeichnete Zwischenschicht ist für die Ausführung der Managementkommandos 51' zuständig.

Wie im Fall der Makroprogrammierung können auch hier Makrorekorder vorgesehen sein, welche die Eingaben des Anwenders, d.h. der für die Programmierung der Nebenstellenanlage 1 verantwortlichen Person aufzeichnen und somit die Automatisierung von wiederkehrenden Programmieraufgaben erlauben.

Nachteilig aus der Sicht des Anwenders, ergibt sich aber auch hier, daß durch eine Änderung der MIB, beispielsweise durch die Bereitstellung neuer MOs und somit eine Änderung der graphischen Benutzeroberfläche 30' vorhandene Konfigurationsmakros nicht mehr benutzt werden können.

Als Beispiel für die Struktur einer MIB ist in Fig. 5a eine MIB für die weiter oben beschriebene Nebenstellenanlage 1 skizziert. Das Datenmodell für die BusinessPhone Nebenstellenanlage 1 ist durch die BPMIB (BusinessPhone MIB) dargestellt. Es basiert auf der allgemeinen MIB Umgebung. Das in Fig. 5a gezeigte Klassendiagramm kann allgemein als Beispiel für die Implementierung einzelner MIBs innerhalb der generischen MIB Umgebung gesehen werden.

Alle die Nebenstellenanlage 1 beschreibenden MOs 27 sind von BPMO 60 oder BPListMO 61 abgeleitet, welche wiederum von MOClassInstanceImpl abgeleitet sind. MOClassInstanceImpl stellt die notwendige allgemeine Funktionalität bereit wie die Struktur des MIB Baumes, die Navigation im Baum, Suchverfahren, allgemeine Attribute und Zugriffsmethoden bzw. Managementkommandos 51' (set()/get()).

BPMO 60 stellt die der Nebenstellenanlage 1 eigene Funktionalität bereit, hauptsächlich die Wechselwirkung mit den unteren Schichten des OSI Modells, insbesondere dem Data Layer und Methoden für den direkten Zugriff auf diese Schichten. BPListMO 61 ist eine weitere Spezialisierung von BPMO 60 und umfaßt hauptsächlich Funktionen für die Leistungsoptimierung. Dieses MO fügt spezielle Funktionen zur Modellierung von Ressourcen hinzu, welche Attributsammlungen wie Tabellen etc. aufweisen und welche durch ein BMPO 60 nicht effektiv beschrieben werden können.

Alle MOs 27, welche Ressourcen 6 der Nebenstellenanlage 1 repräsentieren, sind entweder von BPMO 60 oder BPListMO 61 abgeleitet. Das Beispiel zeigt die MOs für unterschiedliche Kategorien von Nebenstellen. Die Basisklasse ExtensionMO 62 ist unterteilt um verschiedene MOs, hier DigitalExtensionMO 63 und AnalogueExtensionMO 64 zu definieren, welche die unterschiedlichen Kategorien der Nebenstellen - digital oder analog - repräsentieren. Auf ähnliche Art kann eine kabellose Nebenstelle modelliert werden.

Das ExtensionMgrMO 65 und seine Unterklassen sind MOs welche die Wechselwirkung zwischen Nebenstellen und Hardwarekarten, Directory und anderen Komponenten des Systems darstellen.

Die allgemeine MIB Umgebung ist für Anwendungen in verteilten Systemen entworfen, mit mehreren Kommunikationsnetzen, welche jeweils eine MIB bereitstellen. Aus Gründen der Rechenleistung und des Speicherplatzbedarfes definieren gängige Nebenstellenanlagen 1 die sie repräsentierenden MIBs nicht selbst sondern stellen lediglich Schnittstellen 23 bereit um die einzelnen Ressourcen 6 anzusprechen. Die MIB wird dementsprechend auf einem von der Nebenstellenanlage 1 getrennten Rechner 24 bereitgestellt.

Es ist aber auch möglich, die MIB unmittelbar innerhalb der Nebenstellenanlage 1 bzw. innerhalb des Kommunikationsnetzes zu integrieren. Das Management bzw. die Programmierung kann dann über ein Netzwerk (LAN, WAN) mit Hilfe eines standardisierten Protokolls erfolgen. Besonders vorteilhaft ist in diesem Zusammenhang die Verwendung des Inter-ORB Protokolls (IIOP), da dieses die Verwendung von Industrie Standards wie CORBA (Common Object Request Broker Architekture) in verteilten Systemen ermöglicht. Mit dem Protokoll IIOP, kann ein CORBA basiertes Programm, welches in einer beliebigen Sprache geschrieben ist und auf einem beliebigen Computer, unter einem beliebigen Betriebssystem läuft, mit einem anderen CORBA basierten Programm auf einem anderen Computer, Betriebssystem etc. kommunizieren. Dadurch können Programme auf MOs 27 zugreifen, wobei Programm und MOs 27 in unterschiedlichen Sprachen implementiert sind.

3 Configuration Script Language

Die vorliegende Erfindung hat zur Aufgabe, den weiter oben im Zusammenhang mit der Makroprogrammierung beschriebenen Nachteil zu beseitigen und eine Umgebung zu schaffen, die dem Anwender die größtmögliche Flexibilität bei der Programmierung gewährt. Dabei sind zwei Aspekte zu beachten.

Der Anwender soll einerseits bei der Programmierung des Kommunikationsnetzes nicht mit den Details der Implementierung der MIB konfrontiert sein. Die MOs 27 sind in einer Programmiersprache definiert und bieten beispielsweise Objektklassen in Java oder C++ an. Es ist dem Programmierer eines Kommunikationsnetzes nicht zuzumuten, auf die einzelnen MOs 27 in den entsprechenden Programmiersprachen zuzugreifen und entsprechende Programme in Java, C++ oder sonstigen Programmiersprachen zu erstellen. In diesem

Sinn ist -in diese Richtung gehen auch die bekannten graphischen Oberflächen- dem Anwender eine Umgebung zur Verfügung zu stellen, die die Ebene der Programmiersprachen für ihn abschirmt.

Andererseits soll eine größtmögliche Unabhängigkeit von notwendigerweise Veränderungen unterworfenen graphischen Oberflächen 30' gegeben sein und der Anwender die größtmögliche Flexibilität bei der Programmierung des Kommunikationsnetzes erhalten.

Diesen beiden Anforderungen wird durch die Bereitstellung einer Skriptsprache Rechnung getragen, welche beim erfindungsgemäßen Programmierverfahren verwendet wird.

3.1 Zugriffskommandos

Hierfür werden zunächst eigene Zugriffskommandos 51 auf die MOs 27 zur Verfügung gestellt, welche unabhängig von der Programmiersprache sind, in welcher die MOs 27 definiert sind. Durch diese Unabhängigkeit der Zugriffskommandos 51 von der Programmiersprache der MOs 27 kann der Programmierer die Nebenstellenanlage 1 besonders einfach und ohne genaue Kenntnis der Implementierung der einzelnen MOs 27 konfigurieren. Diese können in Java, C++ oder anderen Programmiersprachen definiert sein; für den Programmierer erfolgt der Zugriff auf die MOs 27 immer über die Zugriffskommandos 51.

Der Zugriff auf das abstrakte Abbild 26 der Funktionsdaten 25 einer Ressource 6 einer bestimmten Kategorie erfolgt somit beim erfindungsgemäßen Verfahren nicht unmittelbar über Managementkommandos 51' sondern mittelbar über Zugriffskommandos 51 auf ein MO Objekt 27 bzw. auf ein Exemplar 53 des dieser Kategorie entsprechenden Datentyps 52.

Fig. 6 zeigt ein Blockbild eines erfindungsgemäßen Verfahrens zur Programmierung der Nebenstellenanlage 1, welche den Unterschied zu dem in Fig. 5 skizzierten bekannten Verfahren verdeutlicht.

Unterschiedlich zum bekannten Verfahren, existiert bei dem in Fig. 6 skizzierten Verfahren ein Interpretierer bzw. ein Script-Engine 31 als eigenes Programmodul, welche dem Programmierer Zugriffskommandos 51 zur Verfügung stellt. Diese Zugriffskommandos 51 erlauben gegenüber dem in Fig. 5 skizzierten Verfahren eine flexible Programmierung des Kommunikationsnetzes und die Erstellung von Kommandoskripts, welche unabhängig von der graphischen Oberfläche 30' sind. Dieses einfache und effiziente Verfahren zur Programmierung bietet insbesondere bei der Eingabe der Funktionsdaten bei einem neuen System oder einer neu installierten Nebenstellenanlage 1 entscheidende Vorteile für den Anwender.

Das Script-Engine 31 bildet in Fig. 6 eine vom restlichen System getrennte Umgebung, welche die Zugriffskommandos 51 bereitstellt und die Managementkommandos 51' ausführt.

Aufgrund der vom Programmierer eingegebenen Zugriffskommandos 51 werden die entsprechenden Managementkommandos 51' durchgeführt. Die Zugriffskommandos 51 lehnen sich deshalb vorzugsweise eng an die Managementkommandos 51' an. Zugriffskommandos 51 können beispielsweise bereitgestellt werden zum Erzeugen oder Vernichten einer Instanz bzw. eines Exemplars 53 eines MOs 27, zum Setzen oder Abfragen bestimmter Werte der Exemplare 53, zum Kopieren oder Verschieben der Werte eines Exemplars 53 eines MOs 27 zu einem anderen Exemplar etc.

Das erfindungsgemäße Verfahren erlaubt es, die Zugriffskommandos 51 unmittelbar, beispielsweise über eine graphische Benutzeroberfläche (GUI) 30' einzugeben. Die interaktive Programmierung eines bestimmten Parameters bzw. die Bereitstellung der programmierbaren Funktionsdaten 25 einer Hardware-Einheit 6 besteht beispielsweise darin, das der Hardware-Einheit zugehörige MO Objekt 27 zu finden, dessen Attribute auszulesen, und diese in einer graphischen Benutzeroberfläche 30' anzuzeigen. Je nach den Eingaben des Benutzers können anschließend Attribute des MO Objekts 27 geändert oder Methoden ausgeführt werden.

Unterschiedlich zu bekannten Systemen der Makroprogrammierung bewirken die MO Objekte 27 aber eine vollständige Unabhängigkeit der Zugriffskommandos 51 von der Benutzeroberfläche 30'. Die Implementierung einer völlig neuen Benutzeroberfläche 30' ändert vorhandenen Zugriffskommandos 51 nicht.

Eine Online-Verbindung mit der Telekommunikationsanlage 1 bzw. mit den Hardware-Einheiten bzw. Ressourcen 6 ist nicht notwendig, um die Zugriffskommandos 51 ausführen zu können. Auch wenn das System offline ist, werden die Zugriffskommandos 51 an dem Abbild 26 der programmierbaren Funktionsdaten 25 ausgeführt. Die Funktionsdaten 25 können später zu den Hardware-Einheiten bzw. Ressourcen 6 übertragen werden.

Für das erfindungsgemäße Verfahren zur Programmierung eines Kommunikationsnetzes, sind somit die folgenden Schritte wesentlich: Ein erster Schritt besteht in der Bereitstellung geeigneter abstrakter Datentypen 52, insbesondere von Managed Objects 27 als Abstraktion der Ressourcen 6 des Kommunikationsnetzes. Jedes Datentyp 52 repräsentiert die Art der programmierbaren Funktionsdaten 25 einer Kategorie von Hardware-Einheiten bzw. Ressourcen 6. Unter Bereitstellung ist in diesem Zusammenhang sowohl das zur-Verfügungstellen bereits implementierter Datentype 52 als auch die Neuimplementierung geeigneter Datentype 52 für neu in das Kommunikationsnetz aufgenommene Kategorien von Hardware-Einheiten bzw. Ressourcen 6 zu verstehen. Die Datentypen 52 bzw. die Managed Objects 27 müssen über ein Management Interface mit dem Kommunikationsnetz kommunizieren bzw. Funktionsdaten 25 austauschen können und in einer Programmiersprache implementiert sind und Managementkommandos 51' für den Zugriff auf die Datentypen 52 bzw. die Managed Objects 27 bereitstellen. Ein weiterer Schritt besteht in der Erzeugung eines Exemplars 53 des

Datentyps 52 für jede Hardware-Einheit 6 der durch das Datentyp 52 repräsentierten Kategorie. In weiterer Folge erfolgt der Zugriff auf die Funktionsdaten 25 der Hardware-Einheiten bzw. Ressourcen 6 einer Kategorie durch Zugriffskommandos 51 auf ein Exemplar 53 des dieser Kategorie entsprechenden Datentyps 52. Hierfür werden erfindungsgemäß ein Interpretierer und Zugriffskommandos 51 auf die Datentypen 52 bzw. die Managed Objects 27 bereitgestellt, die Zugriffskommandos 51 an den Interpretierer übergeben, die Zugriffskommandos 51 durch den Interpretierer automatische in ein bzw. mehrere Managementkommandos 51' übersetzt und die entsprechenden Managementkommandos 51' an den Datentypen 52 bzw. den Managed Objects 27 automatisch ausgeführt.

3.2 Kommandoskripts

Bei einer besonders bevorzugten Programmiermethode werden mehrere Zugriffskommandos 51 in Kommandoskripts 50 zusammengefaßt. Dies erlaubt die schnelle, automatische Abarbeitung einer großen Anzahl von Zugriffskommandos 51 bei öfters anfallenden Konfigurationsarbeiten.

Auch die automatische Abarbeitung der Zugriffskommandos 51 eines Kommandoskripts 50 erfolgt bei der in Fig. 6 skizzierten Skript-Umgebung durch den Interpretierer, das Script-Engine (SE) 31.

Vorzugsweise handelt es sich bei der durch die Zugriffkommandos 51 gebildeten Skriptsprache um eine objektbasierte Skriptsprache, die Klassen in der weiter oben beschriebenen Form anbietet. Die Skriptsprache basiert somit auf der verwalteten Objektdarstellung der programmierbaren Funktionsdaten 25 der Hardware-Einheiten 6. Das erfindungsgemäße Verfahren weist durch die Bereitstellung der Skriptsprache den zusätzlichen Vorteil auf, daß das die Funktionsdaten 25 automatisch und unmittelbar aus den Kommandoskripts 50 erzeugt werden können.

Dadurch, daß die erfindungsgemäße Darstellung der Hardware-Einheiten bzw. Ressourcen 6 durch MO Objekte 27 die vollständige Unabhängigkeit von der Benutzeroberfläche 30' gewährleistet, kann selbst die Implementierung einer völlig neuen Benutzeroberfläche 30' existierende Kommandoskripts 50 nicht beeinflussen.

Vor allem die Integration neuer Hardware in bestehende Telekommunikationsanlagen wird durch das erfindungsgemäße Verfahren bedeutend erleichtert. Es ist lediglich die Erstellung eines neuen MOs 27 bzw. eines neuen Datentyps 52 für die Kategorie der neu verwendeten Hardware-Einheit 6 notwendig. Bestehende Kommandoskripts 50 können weiter verwendet werden. Weiters können neue Zugriffskommandos 51 zur Verwaltung der neuen Hardware-Einheiten bzw. Ressourcen 6 in die bestehenden Kommandoskripts 50 aufgenommen werden. Insbesondere im Bereich größerer Anlagen, bei denen oft Endgeräte bzw. Hardware-Einheiten bzw. Ressourcen 6 verschiedener Hersteller zusammengeschlossen sind (Multi-Vendor Anlagen) stellt dies einen bedeutenden Vorteil gegenüber den im Stand der Technik bekannten Programmierverfahren dar. Im Unterschied zum bekannten Stand der

Technik ist die Programmstruktur erweiterbar, d.h. neue Objektarten können in bestehenden Kommandoskripts 50 verwendet werden.

Eine Online-Verbindung mit der Telekommunikationsanlage 1 bzw. mit den Hardware-Einheiten bzw. Ressourcen 6 ist auch hier nicht notwendig, um die Kommandoskripts 50 ablaufen lassen zu können. Wenn das System offline ist, kann das gesamte Kommandoskript 50 mit dem Abbild 26 der programmierbaren Funktionsdaten 25 ausgeführt werden.

Besonders vorteilhaft ist weiters, daß die Skriptsprache nicht nur für Nebenstellenanlagen 1 sondern auch für anderen Kommunikationsnetze auf besonders einfache Art verwendet werden kann. Einzige Voraussetzung für die Verwendung mit anderen Kommunikationsnetzen ist, daß geeignete, die Hardware repräsentierender Datentypen 52 bzw. in einer MIB zusammengefaßte MO Objekte 27 bereitgestellt sind, die Datentypen 52 bzw. MOs 27 in einer Programmiersprache implementiert sind, und die MOs 27 über ein Management Interface mit dem Kommunikationsnetz kommunizieren können.

Das erfindungsgemäße Verfahren kann im Prinzip zur Programmierung beliebiger Telekommunikationsanlagen dienen. Beispielsweise kann bei der Überspielung der Software auf alle Mobiltelefone eines Netzbetreibers für jede Bauart der Mobiltelefone ein eigenes Managed Objekt 27 und für jedes Mobiltelefon ein entsprechendes Exemplar 53 erzeugt werden. Die Verbindung zwischen den einzelnen Hardware-Einheiten - in diesem Fall den Mobiltelefonen - und der zentralen Steuerung erfolgt über Funk über definierte Schnittstellen 23'.

Bei der in Fig. 6 skizzierten Skript-Umgebung können verschiedene Kommandoskripts 50 in einem zentralen Archiv bzw. in dem Skript-Depot 32 verwaltet werden. Das Archiv hat vorzugsweise eine Verzeichnisstruktur zum Speichern von unterschiedlichen Skript-Typen. So können z.B. die folgenden Skript-Typen existieren: Werk-Skripts, welche vom Benutzer weder geändert noch gelöscht werden können, benutzerdefinierte Skripts, und Templates, eine spezielle Art von Skripts, welche aufgrund der bestehenden Programmierung einer Hardware-Einheit 6 automatisch generiert werden können und zur Installation einer neuen Hardware-Einheit 6 verwendet werden können. Bei der dargestellten Verwendung eines Kommandoskripts 50 zur Durchführung einer Programmieraufgabe kann ein Kommandoskript 50 von dem mit Skript-Depot 32 bezeichneten Datenverbund geladen und anschließend vom Script-Engine 31 ausgeführt werden.

Das Skript-Depot 32 bietet hierfür beispielsweise die Methoden *view*, *edit*, *delete*, *export*, *install* zur Manipulation der Kommandoskripts 50 an. Die Umsetzung des Kommandoskripts 50 besteht wieder darin, die notwendigen MO Objekte 27 zu finden und an Ihnen die Programmkommandos 51 des geladenen Kommandoskripts 50 auszuführen.

Das erfindungsgemäße Verfahren bzw. die erfindungsgemäße objektorientierte Skriptsprache erlaubt somit die einfache Wiederverwendung von bereits bestehenden Pro-

grammen und Programmteilen. Dadurch kann der Entwicklungsaufwand reduziert und die Entwicklungszeit verkürzt werden. Durch die Wiederverwendung von Code werden Duplikate vermieden. Wartungsarbeiten können daher auf kleine Teile, beispielsweise einzelne Datentypen 52, beschränkt werden und wirken sich dann auf alle Kommandoskripts 50 aus, in denen diese Teile verwendet werden.

Der Aufbau eines erfindungsgemäßen Kommandoskriptes 50 soll nun an einem Beispiel genauer erläutert werden. Im folgenden ist als Beispiel ein dem weiter oben genannten Makro entsprechendes erfindungsgemäßes Kommandoskript 50 für die weiter oben beschriebene Nebenstellenanlage 1 angegeben.

```
// [DESC] AUSTRIA : Trunk session 2 wire tieline
// . comments .
TrunkMO aa;
aa.userDIRentry();
aa.set (1607, "1",     // allowed 12=1 or 16=0
        1609, "106",   // range 98 - 125
        1610, "1",     // range -12.0 - +6.0 in steps of 0.1
        1611, "1",     // range -12.0 - +6.0 in steps of 0.1
        1612, "1",     // range 0 - 7
        1613, "1",     // no comment
       );
aa.mark2copy(1607,1609,1610,1611,1612,1613)
```

Die Kommandoskripts 50 sind in einer sehr einfachen Skriptsprache geschrieben und als Volltext Dateien gespeichert. Dadurch sind sie jederzeit leicht und schnell mit jedem Texteditor editierbar.

Allgemein umfaßt ein Kommandoskript 50 die folgenden Schritte bzw. Zugriffskommandos 51:

Ein erster Schritt besteht in der Erzeugung eines Exemplars 53 eines Managed Objekts 27. Alle Hardware-Einheiten bzw. Ressourcen 6 wie z.B. Nebenstellen (engl: extensions), Amtsleitungen oder Querverbindungen (engl: trunks) und Betriebsmittel (engl:facilities) werden als MO Objekte 27 modelliert. Für die Arbeit mit einer Amtsleitung genügt die Erzeugung eines Exemplars 53 eines Amtsleitung-Objektes (engl: TrunkMO), beispielsweise durch das Zugriffskommando 51 "*TrunkMO aa*;". Alle weiteren Operationen werden an diesem Exemplar 53 *aa* des Amtsleitung-Objektes ausgeführt. In einem Kommandoskript können mehrere Exemplare 53 von unterschiedlichen MOs 27 definiert werden.

In einem nächsten Schritt können Eingaben des Benutzers eingelesen werden. Die zusätzlichen Benutzerangaben werden bei der Übersetzung der Zugriffskommandos 51

eingelesen und bei der Übersetzung in die Managementkommandos 51' berücksichtigt. Beispielsweise sind vom Benutzer die Verzeichnisnummer und die Nummer der Amtsleitung einzugeben. Dies wird durch die Verwendung des Zugriffskommandos 51 *userDIRentry()* bewirkt. Nach der Definition eines Amtsleitung-Exemplars *aa* kann die entsprechende *userDIRentry()* Funktion durch das Zugriffskommando 51 "*aa.userDIRentry()*" aufgerufen werden.

In einem nächsten Schritt können einzelnen Variablen der Exemplare 53 der MO Objekte 27 bestimmte Werte zugewiesen werden. Dies erfolgt über den Aufruf der Funktion *set()* des entsprechenden Exemplars 53. Der Aufruf hat eine Liste der Kommandos und der entsprechenden Werte zu enthalten. Alle relevanten Kommandos eines MO Objektes 27 sind auf diese Weise ansprechbar.

Üblicherweise werden die Daten zu weiteren Nebenstellen oder Amtsleitungen kopiert. Zu diesem Zweck kann die Funktion *mark2copy()* verwendet werden. Die der Funktion übergebenen Parameter stellen die Liste der zu kopierenden Daten dar. Beispielsweise kann zum Kopieren bestimmter konfigurierten Funktionsdaten 25 der Befehl das Zugriffskommando 51 "*aa.mark2copy(1607,1608)*" verwendet werden. Dieser Befehl kopiert die Daten der Variablen *1607* und *1608*. Die *copy()* Funktion kann sinnvollerweise nur auf Exemplare 53 von MOs 27 desselben Typs angewendet werden. Ein Auswahlfenster zeigt beispielsweise alle möglichen Amtsleitung-Objekte an, zu denen die Daten kopiert werden können. Die notwendigen Objekte können selektiert und die Daten durch Drücken eines *Submit*-Buttons kopiert werden.

3.3 Templates

Bei einer besonders bevorzugten Ausführungsform des erfindungsgemäßen Programmierverfahrens ist vorgesehen, ausgehend von einer bereits programmierten Nebenstellenanlage 1 den aktuellen Wert sämtlicher programmierbarer Funktionsdaten 25 der Nebenstellenanlage 1 bzw. des Kommunikationsnetzes auszulesen. Dies geschieht ganz einfach durch das automatische Auslesen sämtlicher in den Exemplaren 53 der Datentypen 52 bzw. der MOs 27 gespeicherten Werte der Funktionsdaten 25.

Für jedes Exemplar 53 und jeden Wert wird ein entsprechendes Zugriffskommando 51 generiert, welches das entsprechenden Exemplare 53 generiert und den entsprechenden Wert wie bei der bereits programmierten Nebenstellenanlage 1 setz.

Dies ermöglicht beim Vorhandensein einer fertig programmierten Nebenstellenanlage 1, automatisch ein Kommandoskript 50, ein sogenanntes Template, zu erstellen, welches alle Informationen über die programmierbaren Funktionsdaten 25 der Nebenstellenanlage 1 beinhaltet. Das erzeugte Template kann durch das Script-Engine 31 abgearbeitet werden und so zur Programmierung einer weiteren Nebenstellenanlage 1 verwendet werden.

Dies stellt einen bedeutenden Vorteil bei der Programmierung einer neuen Nebenstellenanlage 1 dar. Der Programmierer erstellt ein Template aufgrund einer bereits fertig programmierten Nebenstellenanlage 1. Dies geschieht automatisch durch die Skript-Umgebung. Zur Programmierung der neuen Nebenstellenanlage 1 muß lediglich das erstellte Template auf der neuen Nebenstellenanlage 1 ausgeführt werden.

Dieses Verfahren kann sowohl bei gleichartigen als auch bei unterschiedlichen Nebenstellenanlagen 1 verwendet werden.

3.4 Implementierung

Im folgenden wird die Implementierung des Interpretierers bzw. der Zugriffskommandos 51 näher beschrieben.

Die Grammatik der Skriptsprache selbst kann in üblicher Weise durch eine kontextfreie Grammatik definiert werden, d.h. durch die Angabe von Regeln zur Konstruktion syntaktischer Gruppierungen aus Einzelelementen. Die Regeln werden zweckmäßigerweise in der üblichen "Backus-Naur Form" (BNF) angegeben.

Vorzugsweise werden getrennte Grammatiken für einen lexikalischen Analysator (engl: lexical analyser) und einen syntaktischen Analysator (engl: parser) in BNF definiert. Die BNF Form für die Lexer und die Parser Grammatik einer erfindungsgemäßen Skriptsprache lauten beispielsweise:

3.4.1 Lexer Grammar

```
COMMENT:        "/" (SL_COMMENT |  ML_COMMENT);
protected
SL_COMMENT:     "/" ("[DESC]"|)    (ESC2|'"')* SLC2      ;
protected
SLC2:           '\n' | '\r';
protected
ML_COMMENT:     "*"
                ({ LA(2)!='/' }? '*'
                |   '\r' '\n'
                |   '\r'
                |   '\n'
                |   ~('*'|'\n'|'\r')| ':'
                )*
                "*/";
WS:             (' '
                |   '\t'
                |   '\n'
                |   '\r');

LPAREN:         '(';
RPAREN:         ')';
LCURLYPAREN:    '{';
RCURLYPAREN:    '}';
STAR:           '*';
MINUS:          '-';
PLUS:           '+';
SEMI:           ';' ;
ASSIGN:         '=' ;
```

```
COMMA:          ',' ;
DOT:            '.' ;
CHAR_LITERAL:   '\'' (ESC|~'\'') '\'' ;
STRING_LITERAL: '"' (ESC2)* '"' ;
DIGIT:          '0'..'9';
protected
HEX_DIGIT:      ('0'..'9'|'A'..'F'|'a'..'f');
INT:            (DIGIT)+ ;
protected
ESC:            '\\'
                (   'n'
                |   'r'
                |   't'
                |   'b'
                |   'f'
                |   '"'
                |   '\''
                |   '\\'
                |   ('u')+ HEX_DIGIT HEX_DIGIT HEX_DIGIT HEX_DIGIT
                |   ('0'..'3')
                    (
                    ('0'..'7')
                          (
                          '0'..'7'
                          )?
                    )?
                |   ('4'..'7')
                    (
                          ('0'..'9')
                    )?
                );
protected
ESC2:           'a'..'z'|'A'..'Z'|'_'|'0'..'9'
                |'#'|'?'|'!'|':'|';'|','|'.'|'+'|'-
                '|'%'|'*'|'/'|'\\'|'='|'('|')'|'['|']'|' '|"\t"
                |"'";
IDENTIFIER
options {
                testLiterals = true;
}
                :         ('a'..'z'|'A'..'Z'|'_')
                ('a'..'z'|'A'..'Z'|'_'|'0'..'9')* ;
```

3.4.2 Parser Grammar

```
programm:       programm_block EOF;
programm_block: ( build_block SEMI )*
build_block:        decl
                |   statement;
//--------------------------------------------------------------------
//              Declarations
//--------------------------------------------------------------------
type:           IDENTIFIER;
decl:           var_decl;
var_decl:       type var_list;
var_list:       variable
                (COMMA variable)*;
//--------------------------------------------------------------------
//              Statements
//--------------------------------------------------------------------
statement:      statement_assign
                |   statement_with
                |   statement_func;
lvalue:         variable;
statement_assign: lvalue ASSIGN expression;
statement_with: "with" LPAREN variable RPAREN LCURLYPAREN! programm_block
                RCURLYPAREN;
statement_func: function_call;
```

```
//------------------------------------------------------------------
//                  Expressions
//------------------------------------------------------------------
value:                  STRING_LITERAL
                      | lvalue
                      | INT
                      | function_call;

variable:               IDENTIFIER;

function_call:          basic_func_call (DOT (basic_func_call
                      | variable))*
                      | variable (DOT(basic_func_call|variable))+;
basic_func_call:      IDENTIFIER LPAREN! expresion_list RPAREN!;
expresion_list:       (expression ( COMMA expression)*)?;
expression:           value;
```

Die Umsetzung der Grammatiken in lauffähige Analysatoren kann beispielsweise durch das bekannte Software-Werkzeuge *antlr* erfolgen, welches ein Werkzeug zur Analyse von LL(k) Sprachen ist, und die Generierung von Analysatoren in unterschiedlichen Sprachen ermöglicht. Insbesondere können die für den Analysator notwendigen Java Klassen sehr einfach durch *antlr* generiert werden. Zur Erzeugung der notwendigen Klassen aus den Grammatiken können aber auch andere gängige Klassen-Generatoren, insbesondere Generatoren für Java-Klassen, verwendet werden.

Zusätzlich ist die Definition einer eigenen Grammatik für den Analysator der hierarchischen Struktur der MOs 27 bzw. der Datentypen 52 vorteilhaft. Auch hier kann der schon angesprochene *antlr* Generator für Analysatoren (engl: translator generator) verwendet werden.

3.4.3 Script-Tool

Der Interpretierer bzw. das Script-Engine 31 ist vorzugsweise in einer objektorientierten Programmiersprache wie Java oder C++ implementiert.

Fig. 7 zeigt das Objektnetz eines erfindungsgemäßen Interpretierers in einer Skript-Umgebung. Das Diagramm verdeutlicht die Struktur der Skript-Umgebung. In dem dargestellten Objektnetz deutet die Raute eine Zusammenfassung an. Die Zahlen drücken aus, wie viele Objekte an einer Beziehung beteiligt sind. Die Pfeile zeigen Spezialisierungen an.

Die Script-Factory (ScFactory) 35 erzeugt und verwaltet alle aktiven Script-Engines 31 (ScEngine). Es existiert nur eine aktive Script-Factory 35. Die Script-Factory 35 folgt dem Singleton Muster. Jeder Befehl zum Laden oder Ausführen eines Kommandoskripts 50 fordert von der Script-Factory 35 ein neues Exemplar eines Script-Engines 31 an. Durch diese Architektur können mehrere Script-Engines 31 gleichzeitig erzeugt werden und so beispielsweise mehrere Nebenstellenanlagen gleichzeitig programmiert werden.

Das Script-Engine 31 arbeitet die Kommandoskripts 50 ab. Die zwei Hauptmethoden sind *load()* und *run()* für das Laden und Ausführen von Kommandoskripts 50. Die *load()* Methode bedingt die Interaktion mit dem -hier nicht dargestellten- Skript-Depot 32.

Das entsprechende Kommandoskript 50 wird im Skript-Depot 32 lokalisiert und in den Speicher geladen. Die *run()* Methode erlaubt es, ein im Speicher befindliches Kommandoskript 50 abzuarbeiten. Weiters erlaubt das Script-Engine 31 den Status des Prozesses abzufragen und weiterzugeben, Benutzereingaben zu verlangen, wenn dies notwendig ist, und beim Auftreten von Fehlern die entsprechenden Fehlermeldungen weiterzugeben.

Der Skript-Datahandler (ScDataHandler) 36 erlaubt den Zugriff des Script-Engines 31 auf die verfügbaren Script-Typen und stellt auf Anfrage des Script-Engines 31 ein Exemplar eines bestimmten Script-Typs 37 (ScType) zur Verfügung. Alle in der Skriptsprache verfügbaren Typen (Integer, String, MOs, spezialisierte MOs wie ExtensionMO und TrunkMO) erben die Attribute von Skript-Type 37. Die Skript-Typen sind durch ihre Typen-Kommandos characterisiert, d.h. für jede Methode eines Skript-Typs existiert eine von SCTypeCMD 38 abgeleitete Klasse Cmd 39, welche die einzelnen Zugriffskommandos 51 definiert.

Das Script-Engine 31 und die Zugriffskommandos 51 können natürlich in einer beliebigen Programmiersprache implementiert sein. Vorzugsweise sind die Zugriffskommandos 51 jedoch in derselben Programmiersprache wie die die Nebenstellenanlage 1 repräsentierenden MOs 27 bzw die Management Information Base implementiert, also beispielsweise in Java oder C++. Dadurch wird die Implementierung wesentlich vereinfacht.

Bei einer besonders bevorzugten Ausführungsform erfolgt die Definition des Script-Engines 31 über eine Schnittstellendefinitionssprache (engl: Interface Definition Language IDL) für die Verwendung des erfindungsgemäßen Programmierverfahrens in verteilten Systemen. Beispielsweise kann hierfür der Industriestandard CORBA verwendet werden. Die einzelnen Klasse werden hierbei in IDL definiert und die entsprechenden Java Klassen bzw. Stubs über einen sprachspezifischen Präprozessor generiert. Damit ergibt sich der weiter oben genannte Vorteil der Interoperabilität von in verschiedenen Programmiersprachen geschriebenen Programmen.

3.5 Ablauf

Das in Fig. 8a gezeigte Diagramm illustriert den zeitlichen Ablauf der Interaktion zwischen den einzelnen Klassen bei der Abarbeitung eines Kommandoskripts 50 durch ein Ausführungsmodul der eben beschriebenen Skript-Umgebung.

SkriptMO 40 ist eine statische Klasse, welche dazu dient, die Nebenstellenanlage 1 zu identifizieren, für welches das Kommandoskript 50 ausgeführt werden soll. Das Managed Objekt SkriptMO 40 repräsentiert ein bestimmtes Softwaremodul, die Skript-Umgebung. ScriptMO 40 ist jedoch kein erfindungsgemäßer Datentyp 52, welcher eine Ressource 6 des Kommunikationsnetzes darstellt. ScriptMO 40 stellt vielmehr die Schnittstelle der Skript-Umgebung zur Außenwelt dar.

SciptMO 40 fordert von Script-Factory 35 ein neues Script-Engine 31 an. Anschließend wird die *load()* Methode des Script-Engines 31 aufgerufen und der Name des Kommandoskripts 50 übergeben. Das Script-Engine 31 greift auf das Script-Repository bzw. das Skript-Depot 32 (hier nicht gezeigt) zu, um die gesuchte Skript-Datei zu finden und in den Speicher zu laden.

Wenn das Kommandoskript 50 erfolgreich in den Speicher geladen werden konnte, wird die *run()* Methode des Script-Engines 31 aufgerufen, wodurch die eigentliche Ausführung des Kommandoskripts 50 gestartet wird und die einzelnen Zugriffskommandos 51 ausgeführt werden. Die Übersetzung (Interpretierung) des Kommandoskripts 50 geschieht in Zusammenwirken mit dem Objektkatalog der Management Information Base (MIB) 28. Das Script-Engine 31 lädt die notwendigen MOs 27 aus der MIB 28 und führt die ihnen zugeordneten spezifizierten Methoden aus, beispielsweise die *set()* Methode für einzelne Parameter.

Wenn die Ausführung des Kommandoskripts 50 vollendet ist, gibt das Script-Engine 31 eine Nachricht an das ScriptMO 40 weiter, welches wiederum den Benutzer 41 benachrichtigt.

Die beschriebene Architektur stellt lediglich eine von vielen möglichen Realisierungen eine erfindungsgemäßen Skript-Umgebung dar. Es sind beispielsweise auch sehr viel einfachere Strukturen möglich. Das Vorsehen der statischen klasse SkriptMO 40 erweist sich allerdings als besonders vorteilhaft, weil das Laden und die Ausführung der einzelnen Kommandoskripts 50 für den Benutzer 41 transparent erfolgt.

Die beschriebene Struktur bzw. Architektur der Skript-Umgebung bewirkt, daß die Skriptausführung nicht unmittelbar mit der Benutzeroberfläche 30' zusammenhängt. Dies ermöglicht es insbesondere, die Skriptausführung und die Bereitstellung der Benutzeroberfläche 30' in zwei voneinander getrennten Programmodulen zu implementieren. Dadurch wird eine Client-Server Struktur erreicht, wobei der Server-Prozess 55 die Ausführung der Kommandoskripts 50 kontrolliert und ein oder mehrere Benutzer 41 sich jeweils über einen Client-Prozess 56 mit dem Server-Prozess 55 verbinden können.

Somit ist es möglich, daß der Server-Prozess 55 die Ausführung der Zugriffskommandos 51 umsetzt und Benutzereingaben lediglich über einen Client-Prozess 56 erfolgen. Der Client-Prozess 56 und der Server-Prozess 55 kommunizieren dabei über eine definierte Schnittstelle 57 miteinander.

Client-Prozess 56 und Server-Prozess 55 können auf dem selben oder auf örtlich getrennten Computern laufen. Dies ist besonders bei der Fernwartung von Telekommunikationsanlagen 1 vorteilhaft. Beispielsweise kann der Client-Prozess 56 durch ein Java-Applet oder eine sonstige Browser-Application bereitgestellt sein. Der Benutzer 41 kann über den Client-Prozess Eingaben durchführen, die den Server-Prozess 55 dazu veranlassen, bestimmte Kommandoskripts 50 zur Wartung der Telekommunikationsanlage 1 auszuführen.

Vorzugsweise erfolgt die Verbindung zwischen Client-Prozess 56 und Server-Prozess 55 über das Internet und die dafür gängigen Protokolle wie TCP/IP. Es ist aber auch möglich, lokale Netzprotokolle wie NetBui oder Protokolle für verteilte Dateisysteme wie beispielsweise NFS zu verwenden. Bei einem besonders bevorzugten Verfahren erfolgt die Kommunikation zwischen dem Client-Prozess 56 und dem Server-Prozess 55 über einen Web-Server 34.

Die Computerprogramme für den Client-Prozess 56 und den Server-Prozess 55 bzw. für die gesamte Skript-Umgebung können lokal im Speicher des Computers abgelegt sein, oder auf einem computergeeigneten Medium wie Disketten oder CD-Roms gespeichert sein, bzw. bei Bedarf über Datenträger oder Datenleitungen, insbesondere über Internet, übermittelt werden. Die letzte Variante ist besonders vorteilhaft, weil es dadurch möglich ist, das Programm für den Client-Prozess 56 bzw. für die Benutzeroberfläche 30' erst auf Anfrage eines Benutzers 41 auf den Computer des Benutzers 41, beispielsweise in Form eines Java-Applets, zu übertragen. Dem Benutzer 41 steht somit stets die neueste Version der Benutzeroberfläche 30' zur Verfügung.

Wie bereits erwähnt ist es weiters möglich, die die Nebenstellenanlage 1 repräsentierende MIB direkt in die Nebenstellenanlage 1 zu integrieren. in diesem Fall umfaßt der Server 55 lediglich den Interpretierer, bzw. im Diagramm in Fig. 8a die Module ScriptMO 40 Script-Factory 35 und Skript-Engine 31.

Fig. 8b skizziert einen möglichen Ablauf der Wechselwirkung des Benutzers 41 mit der Skript-Umgebung. In dem meisten Fällen ist die Wechselwirkung mit dem Benutzer 41 unbedingt notwendig um wesentliche Parameter für die Ausführung der Kommandoskripts 50 zur Verfügung zu stellen. Die Skriptsprache erlaubt beispielsweise parametrisierte Kommandoskripts 50. Die Zugriffskommandos 51 in dem Kommandoskript 50 können beispielsweise an einem mit einer eindeutigen Identifikationsnummer gekennzeichneten MO Objekt 27 ausgeführt werden. Weiters benötigt die *copy()* Anweisung Angaben über die Hardware-Einheiten, zu denen die Daten kopiert werden sollen.

Bei der gängigen Zugangsform über das Internet ist keine Anfrage an den Benutzer 41 durch die Skript-Umgebung möglich. Aus diesem Grund muß der Client-Prozess 56 regelmäßig den Status des Server-Prozesses 55 abfragen und bei einer Eingabeaufforderung entsprechend antworten. In Fig. 8b wird der Status des ScriptMO 40 Objekts regelmäßig vom Client-Prozess 56 abgefragt. Die Abarbeitung des Kommandoskripts 50 wird vom Script-Engine 31 durchgeführt. Sobald eine Benutzereingabe verlangt wird, verständigt das Script-Engine 31 hiervon das ScriptMO Objekt 40. Dies bewirkt eine Statusänderung im SkriptMO Objekt 40, welche bei der nächsten Abfrage des Client-Prozesses 56 registriert wird. Daraufhin stellt der Client-Prozess 56 dem Benutzer 41 die entsprechende Eingabemaske zur Verfügung, nimmt die Benutzereingabe an und leitet diese an SkriptMO 40 weiter.

Die Fehlerbehandlung erfolgt auf analoge Weise (Fig. 8c). Beim Auftritt eines Fehlers während der Ausführung eines Kommandoskripts 50 wird dieses angehalten und der Benutzer 41 über den Fehler informiert. Die Übertragung, z.B. auf einen Webbrowser, erfolgt durch regelmäßige Abfrage des Status des Script-Engines 31. Der Status kann beispielsweise sein:

- Idle: Das Kommandoskript 50 ist noch nicht gestartet.
- Running: Das Kommandoskript 50 wird ausgeführt.
- Input required: Das Kommandoskript 50 forderte eine Eingabe vom Benutzer an. Der Client-Prozess 56 kann daraufhin abfragen, um welche Art der Benutzereingabe es sich handelt, dem Benutzer 41 eine entsprechende Eingabemaske präsentieren, und die Daten dem Server-Prozess 55 bzw. dem Script-Engine 31 zur Verfügung stellen. Daraufhin wird die Ausführung des Kommandoskripts 50 wieder aufgenommen.
- Error: Ein Fehler ist aufgetreten. Die Ausführung des Kommandoskripts 50 wird unterbrochen. Der Client-Prozess 56 kann zusätzliche Informationen über die Art des Fehlers anfordern und diese dem Benutzer 41 anzeigen.

Auch die Fehlerbehandlung kann somit über einen Web Client erfolgen. Weiters kann der Zugriff auf andere Elemente der Skript-Umgebung wie beispielsweise auf das Skript-Depot 32 über eine Weboberfläche erfolgen. Die Verwaltung der Kommandoskripts 50 ist vorzugsweise ebenfalls web-basiert, d.h. die Verwaltung kann über eine Benutzerschnittstelle, welche beispielsweise in einem Webbrowser läuft, über das Internet erfolgen. Die Schnittstelle zwischen Skript-Umgebung und Benutzeroberfläche 30' wird dabei vorzugsweise über ein eigenes Programmodul, den Presentation Layer 33 gebildet.

Bei Fernwartung von sensiblen Telekommunikationsanlagen 1 stellt die Datensicherheit einen besonders wichtigen Aspekt dar. Übliche Sicherheitskonzepte beinhalten in diesem Zusammenhang die Möglichkeit sicherer Datenübertragung über das Internet, weiters die Möglichkeit des geschützten Zugriffs auf das Skript-Depot 32, die Vergabe von Berechtigungen für das Ausführen von Kommandoskripts 50, und von Berechtigungen für das Ausführen einzelner Zugriffskommandos 51.

Das erfindungsgemäße Verfahren kann zur Programmierung von Nebenstellenanlagen 1 aber auch von anderen Kommunikationsnetzen, wie Rechnernetzen, oder dem weiter oben genannten Verbund von Mobiltelefonen oder auch einzelnen Computern dienen. Die Vorteile der erfindungsgemäßen objektorientierten Skriptsprache können somit für die Wartung von unterschiedlichen Kommunikationsnetzen ausgenützt werden.

ANSPRÜCHE

1. Programmlogik zur Programmierung bzw. zum Management von Kommunikationsnetzen, insbesondere von Nebenstellenanlagen (1), in einer Umgebung, welche

abstrakte Datentypen (52), insbesondere Managed Objects (27) als Abstraktion der Ressourcen (6) des Kommunikationsnetzes bereitstellt, die vorzugsweise in einer Sammlung, insbesondere in einer Management Information Base (MIB) vorliegen,

wobei die Datentypen (52) bzw. die Managed Objects (27) über ein Management Interface mit dem Kommunikationsnetz kommunizieren bzw. Funktionsdaten (25) austauschen können,

und wobei die Datentypen (52) bzw. die Managed Objects (27) in einer Programmiersprache implementiert sind und Managementkommandos (51') für den Zugriff auf die Datentypen (52) bzw. die Managed Objects (27) bereitstellen,

dadurch gekennzeichnet, daß:

ein Interpretierer und Zugriffskommandos (51) auf die Datentypen (52) bzw. die Managed Objects (27) bereitgestellt werden,

die Zugriffskommandos (51) an den Interpretierer übergeben werden,

die Zugriffskommandos (51) durch den Interpretierer automatisch in ein bzw. mehrere Managementkommandos (51') übersetzt werden und

die entsprechenden Managementkommandos (51') an den Datentypen (52) bzw. den Managed Objects (27) automatisch ausgeführt werden.

2. Programmlogik nach Anspruch 1, **dadurch gekennzeichnet**, daß bei der Übersetzung der Zugriffskommandos (51) zusätzliche Benutzerangaben eingelesen werden und diese bei der Übersetzung in die Managementkommandos (51') berücksichtigt werden.

3. Programmlogik nach Anspruch 1 oder 2, **dadurch gekennzeichnet**, daß mehrere Zugriffskommandos (51) in ein Kommandoskript (50) zusammengefaßt werden.

4. Programmlogik nach Anspruch 3, **dadurch gekennzeichnet**, daß Kommandoskripts (50) in ein Skriptdepot (32) geladen bzw. aus diesem abgerufen werden.

5. Programmlogik nach Anspruch 3 oder 4, **dadurch gekennzeichnet**, daß Zugriffskommandos (51) zunächst automatisch durch Auslesen der Datentypen (52) bzw. der Managed Objects (27) eines bereits programmierten Kommunikationsnetzes erzeugt werden und in einem speziellen Kommandoskript (50), einem sogenannten Template zusammengefaßt werden und dieses Template zur Programmierung eines weiteren Kommunikationsnetzes verwendet wird.

6. Programmlogik nach einem der Ansprüche 1-5, **dadurch gekennzeichnet**, daß die Übersetzung der Zugriffskommandos (51) und die Ausführung der Managementkommandos (51') durch einen Server-Prozess (55) umgesetzt werden und Benutzereingaben von einem Client-Prozess (56) eingelesen werden, wobei der Client-Prozess (56) und der Server-Prozess (55) über eine Schnittstelle (57) miteinander kommunizieren.

7. Programmlogik nach Anspruch 6, **dadurch gekennzeichnet**, daß der Client-Prozess (56) und der Server-Prozess (55) über ein Netzwerkprotokoll, insbesondere über das im Internet relevante TCP/IP Protokoll miteinander kommunizieren.

8. Programmlogik nach Anspruch 6 oder 7, **dadurch gekennzeichnet**, daß die Kommunikation zwischen dem Client-Prozess (56) und dem Server-Prozess (55) über einen Web-Server (34) erfolgt.

9. Programmlogik nach einem der Ansprüche 6 bis 8, **dadurch gekennzeichnet**, daß der Status des Server-Prozesses (55) regelmäßig vom Client-Prozess (56) abgefragt wird.

10. Verfahren zur Programmierung bzw. zum Management von Kommunikationsnetzen, insbesondere von Nebenstellenanlagen (1), in einer Umgebung, welche

    abstrakte Datentypen (52), insbesondere Managed Objects (27) als Abstraktion der Ressourcen (6) des Kommunikationsnetzes bereitstellt, die vorzugsweise in einer Sammlung, insbesondere in einer Management Information Base (MIB) vorliegen,

    wobei die Datentypen (52) bzw. die Managed Objects (27) über ein Management Interface mit dem Kommunikationsnetz kommunizieren bzw. Funktionsdaten (25) austauschen können,

    und wobei die Datentypen (52) bzw. die Managed Objects (27) in einer Programmiersprache implementiert sind und Managementkommandos (51') für den Zugriff auf die Datentypen (52) bzw. die Managed Objects (27) bereitstellen,

    dadurch gekennzeichnet, daß:

ein Interpretierer und Zugriffskommandos (51) auf die Datentypen (52) bzw. die Managed Objects (27) bereitgestellt werden,

die Zugriffskommandos (51) an den Interpretierer übergeben werden,

die Zugriffskommandos (51) durch den Interpretierer automatisch in ein bzw. mehrere Managementkommandos (51') übersetzt werden und

die entsprechenden Managementkommandos (51') an den Datentypen (52) bzw. den Managed Objects (27) automatisch ausgeführt werden.

11. Verfahren nach Anspruch 10, **dadurch gekennzeichnet,** daß bei der Übersetzung der Zugriffskommandos (51) zusätzliche Benutzerangaben eingelesen werden und diese bei der Übersetzung in die Managementkommandos (51') berücksichtigt werden.

12. Verfahren nach Anspruch 10 oder 11, **dadurch gekennzeichnet,** daß mehrere Zugriffskommandos (51) in ein Kommandoskript (50) zusammengefaßt werden.

13. Verfahren nach Anspruch 12, **dadurch gekennzeichnet,** daß Kommandoskripts (50) in ein Skriptdepot (32) geladen bzw. aus diesem abgerufen werden.

14. Verfahren nach Anspruch 12 oder 13, **dadurch gekennzeichnet,** daß Zugriffskommandos (51) zunächst automatisch durch Auslesen der Datentypen (52) bzw. der Managed Objects (27) eines bereits programmierten Kommunikationsnetzes erzeugt werden und in einem speziellen Kommandoskript (50), einem sogenannten Template zusammengefaßt werden und dieses Template zur Programmierung eines weiteren Kommunikationsnetzes verwendet wird.

15. Verfahren nach einem der Ansprüche 10 bis 14, **dadurch gekennzeichnet,** daß die Übersetzung der Zugriffskommandos (51) und die Ausführung der Managementkommandos (51') durch einen Server-Prozess (55) umgesetzt werden und Benutzereingaben von einem Client-Prozess (56) eingelesen werden, wobei der Client-Prozess (56) und der Server-Prozess (55) über eine Schnittstelle (57) miteinander kommunizieren.

16. Verfahren nach Anspruch 15, **dadurch gekennzeichnet,** daß der Client-Prozess (56) und der Server-Prozess (55) über ein Netzwerkprotokoll, insbesondere über das im Internet relevante TCP/IP Protokoll miteinander kommunizieren.

17. Verfahren nach Anspruch 15 oder 16, **dadurch gekennzeichnet,** daß die Kommunikation zwischen dem Client-Prozess (56) und dem Server-Prozess (55) über einen Web-Server (34) erfolgt.

18. Verfahren nach einem der Ansprüche 15 bis 17, **dadurch gekennzeichnet,** daß der Status des Server-Prozesses (55) regelmäßig vom Client-Prozess (56) abgefragt wird.

19. Computersystem (24) auf welchem eine Anwendung läuft, **dadurch gekennzeichnet**, daß die Anwendung die Programmierung bzw. das Management eines Kommunikationsnetzes gemäß einer Programmlogik nach einem der Ansprüche 1-9 umsetzt.

20. Computersystem (24') auf welchem ein Client-Prozess (56) läuft, der mit einem, vorzugsweise auf einem anderen Computersystem (24) laufendem Server-Prozess (55) über eine Schnittstelle (57) kommuniziert, **dadurch gekennzeichnet**, daß der Server-Prozess (55) die Übersetzung der Zugriffskommandos (51) und die Ausführung der Managementkommandos (51') zur Programmierung bzw. zum Management eines Kommunikationsnetzes gemäß einer Programmlogik nach einem der Ansprüche 6-9 umsetzt.

21. Computersystem (24) auf welchem ein Server-Prozess (55) läuft, der mit einem, vorzugsweise auf einem anderen Computersystem (24') laufendem Client-Prozess (56) über eine Schnittstelle (57) kommuniziert, **dadurch gekennzeichnet**, daß der Server-Prozess (55) die Übersetzung der Zugriffskommandos (51) und die Ausführung der Managementkommandos (51') zur Programmierung bzw. zum Management eines Kommunikationsnetzes gemäß einer Programmlogik nach einem der Ansprüche 6-9 umsetzt.

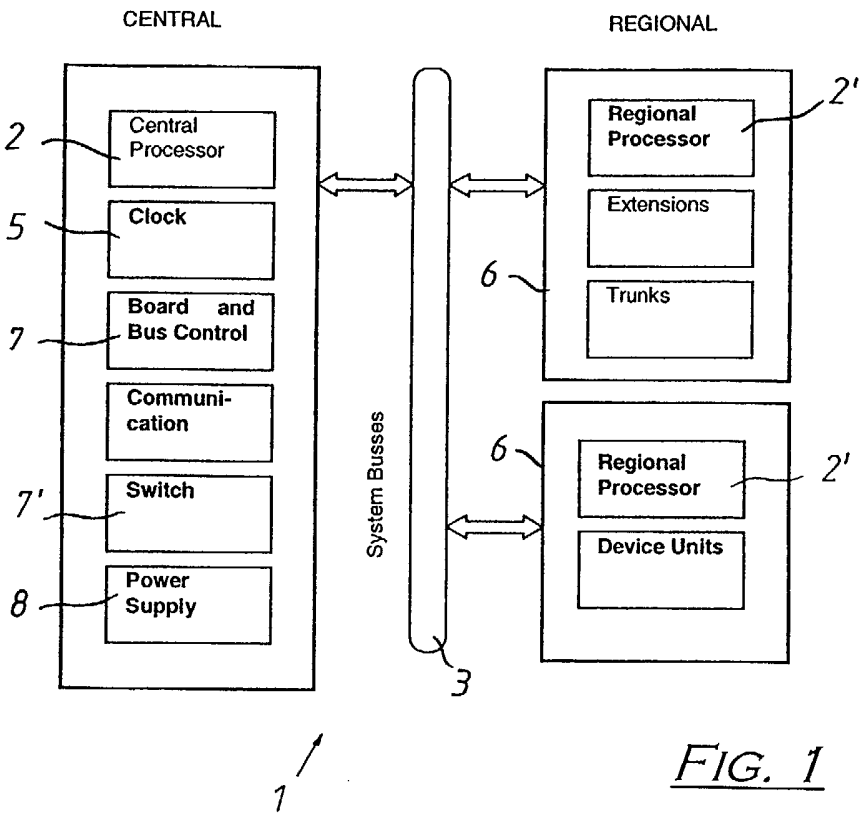

CENTRAL
REGIONAL
2
Central Processor
5
Clock
7
Board and Bus Control
Communi-cation
7'
Switch
8
Power Supply
System Busses
3
2'
Regional Processor
Extensions
6
Trunks
6
2'
Regional Processor
Device Units
1

FIG. 1

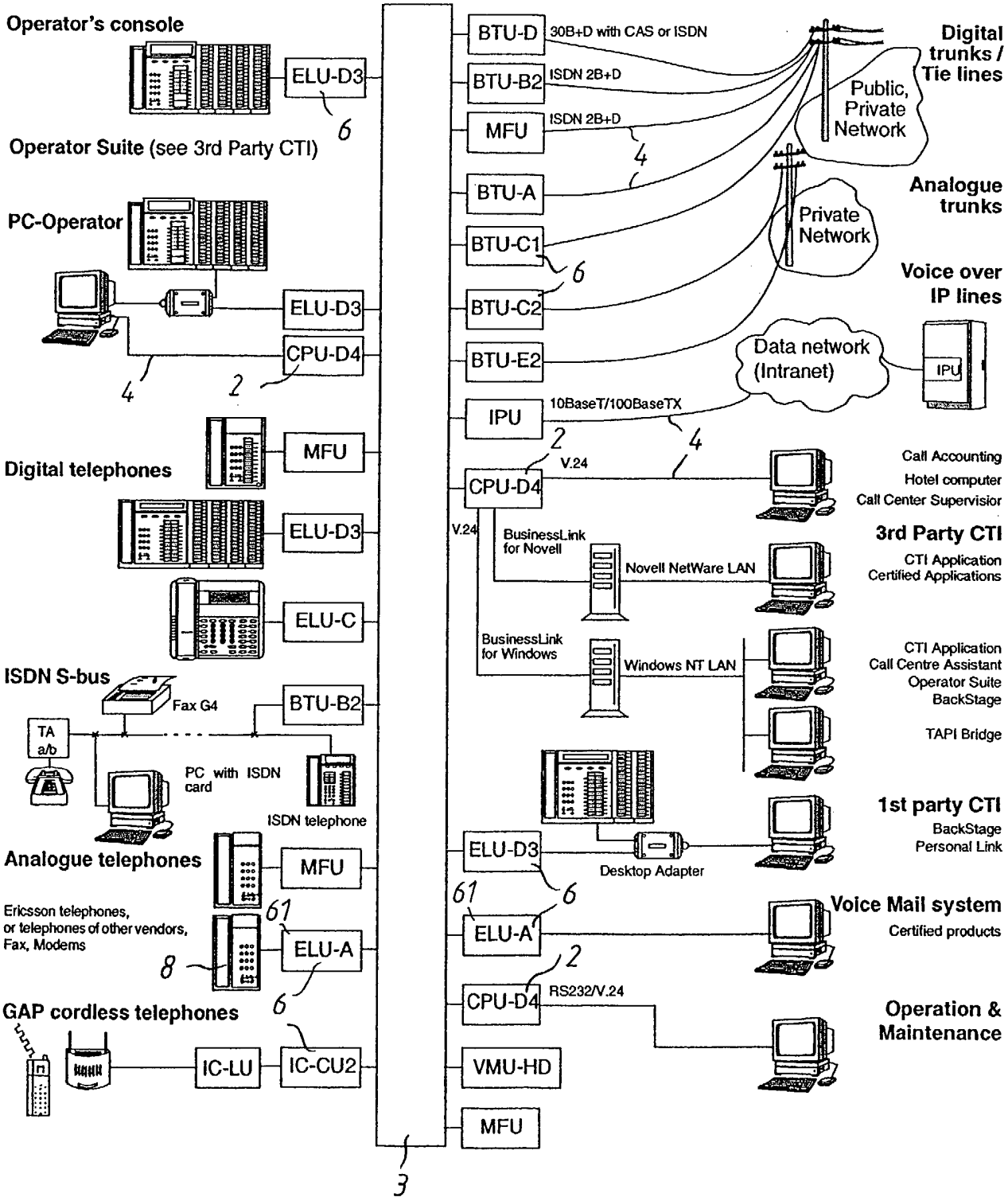

Operator's console
ELU-D3
6
Operator Suite (see 3rd Party CTI)
PC-Operator
ELU-D3
CPU-D4
4
2
Digital telephones
MFU
ELU-D3
ELU-C
ISDN S-bus
Fax G4
BTU-B2
TA
a/b
PC with ISDN card
ISDN telephone
Analogue telephones
MFU
Ericsson telephones, or telephones of other vendors, Fax, Modems
61
ELU-A
8
6
GAP cordless telephones
IC-LU
IC-CU2
3
BTU-D
30B+D with CAS or ISDN
BTU-B2
ISDN 2B+D
MFU
ISDN 2B+D
4
BTU-A
BTU-C1
6
BTU-C2
BTU-E2
IPU
10BaseT/100BaseTX
2
4
CPU-D4
V.24
V.24
BusinessLink for Novell
Novell NetWare LAN
BusinessLink for Windows
Windows NT LAN
ELU-D3
Desktop Adapter
61
6
ELU-A
2
CPU-D4
RS232/V.24
VMU-HD
MFU
Public, Private Network
Private Network
Data network (Intranet)
IPU
Digital trunks / Tie lines
Analogue trunks
Voice over IP lines
Call Accounting
Hotel computer
Call Center Supervisior
3rd Party CTI
CTI Application
Certified Applications
CTI Application
Call Centre Assistant
Operator Suite
BackStage
TAPI Bridge
1st party CTI
BackStage
Personal Link
Voice Mail system
Certified products
Operation & Maintenance

FIG. 1A

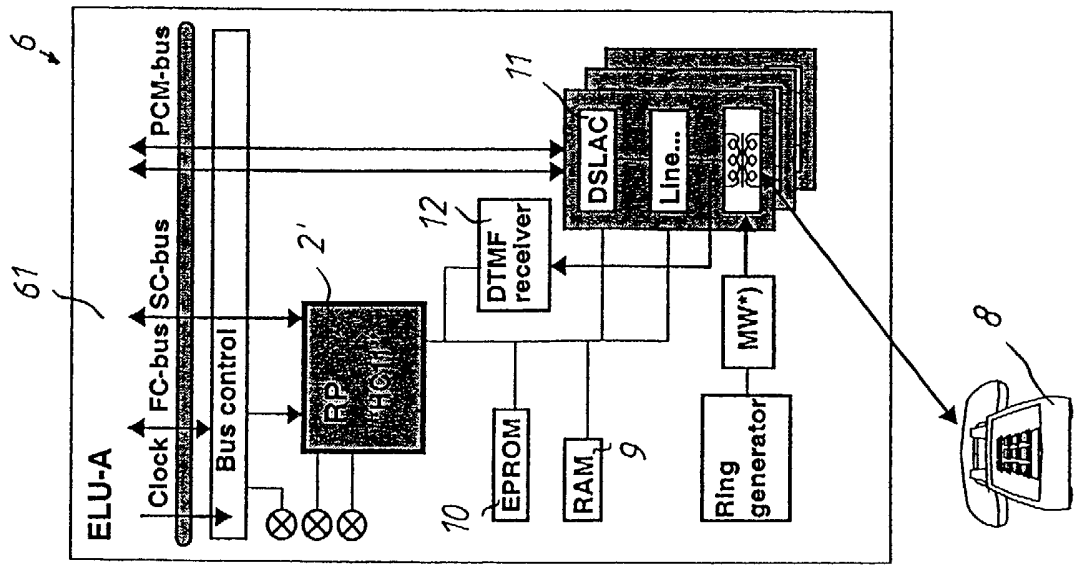

6
61
ELU-A
Clock
FC-bus
SC-bus
PCM-bus
Bus control
2'
RP
"HOM"
12
DTMF receiver
11
DSLAC
Line...
10
EPROM
RAM
9
MW*)
Ring generator
8

FIG. 1B

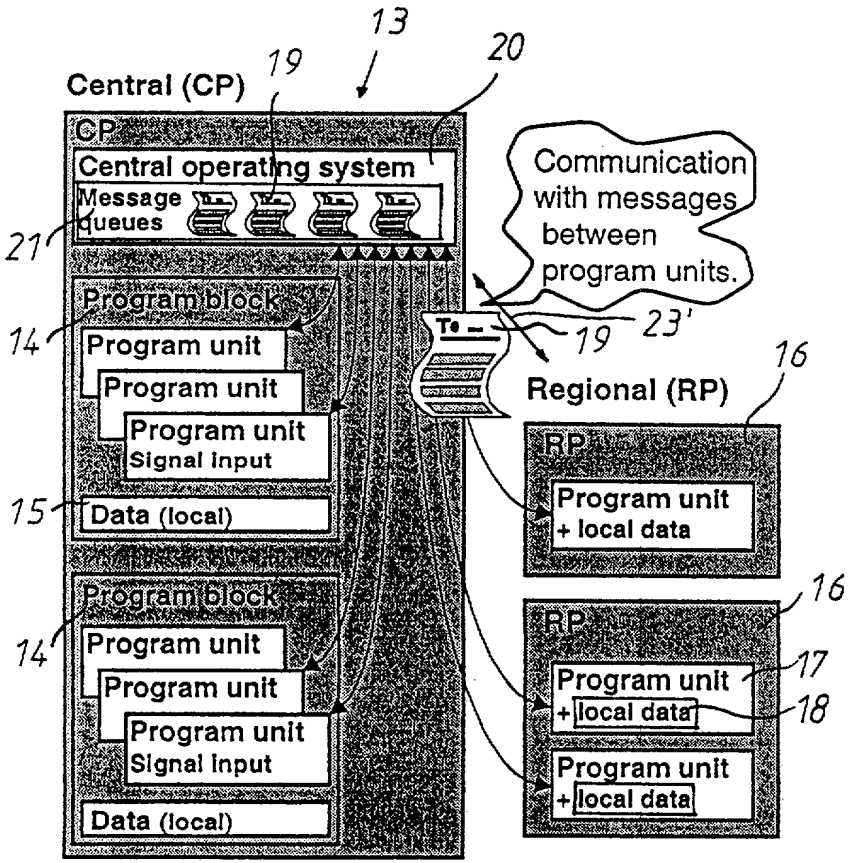

Central (CP)
13
19
20
CP
Central operating system
Message queues
21
Communication with messages between program units.
Program block
14
Program unit
Program unit
Program unit
Signal input
15
Data (local)
19
23'
Regional (RP)
16
RP
Program unit
+ local data
Program block
14
Program unit
Program unit
Program unit
Signal input
Data (local)
16
RP
Program unit
+ local data
17
18
Program unit
+ local data

FIG. 2

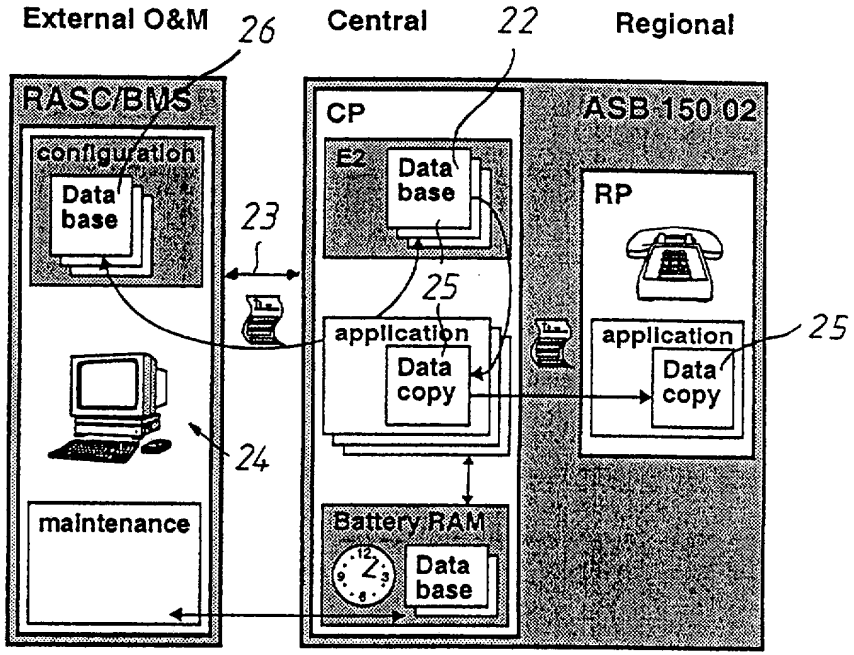

External O&M
26
Central
22
Regional
RASC/BMS
configuration
Data base
23
CP
E2
Data base
ASB 150 02
RP
25
application
Data copy
application
Data copy
25
24
maintenance
Battery RAM
Data base

FIG. 3

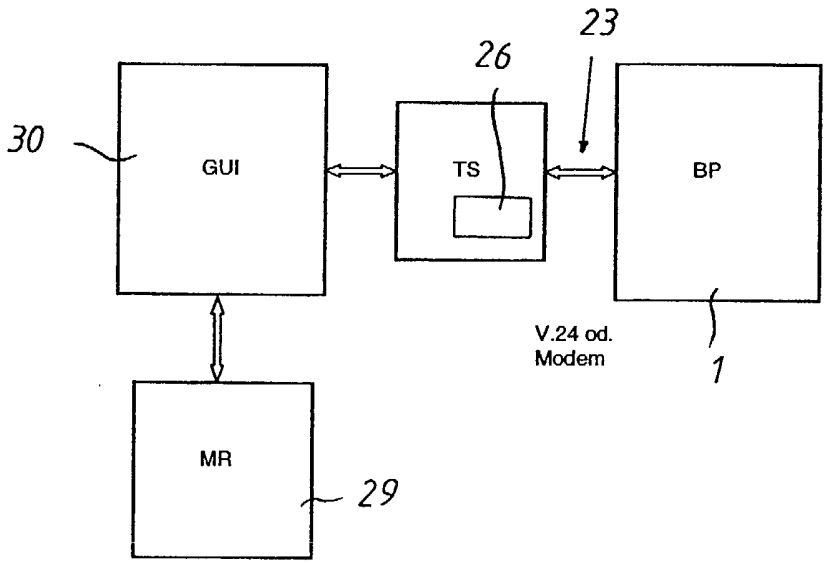


FIG. 3A

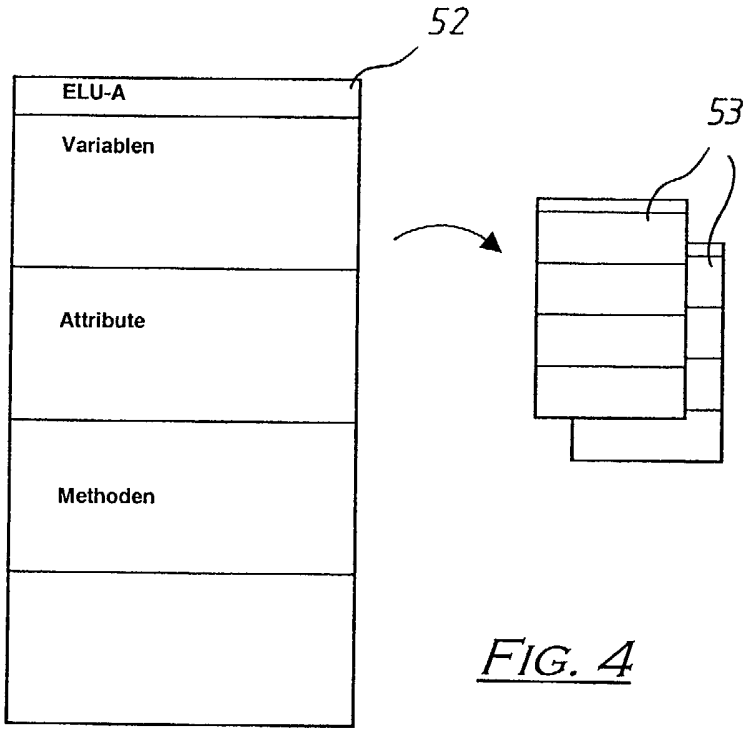

52
ELU-A
Variablen
Attribute
Methoden
53

FIG. 4

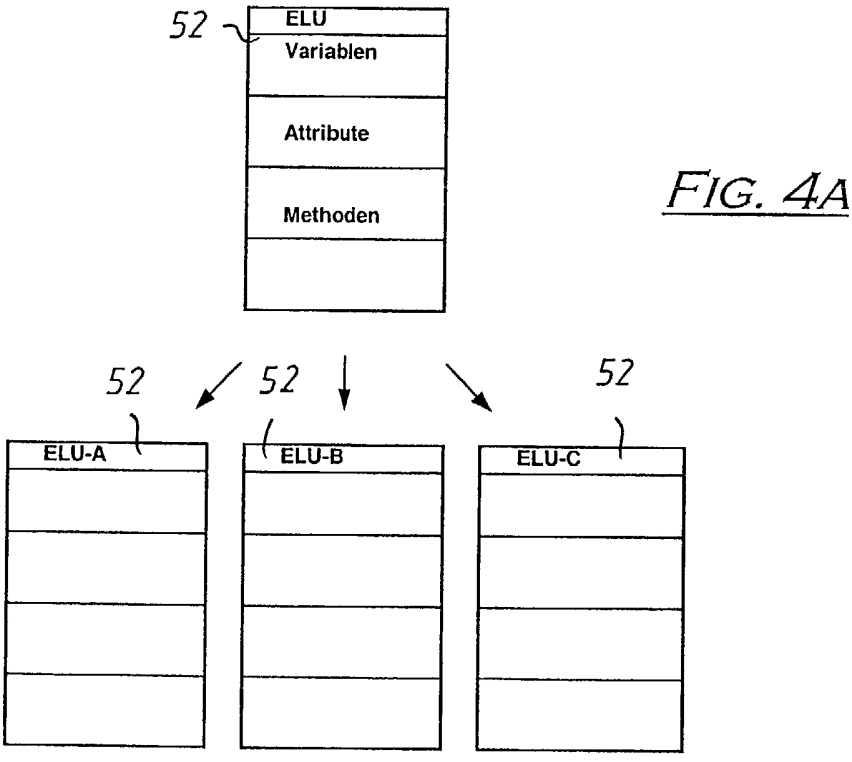

52
ELU
Variablen
Attribute
Methoden
52
52
52
ELU-A
ELU-B
ELU-C

FIG. 4A

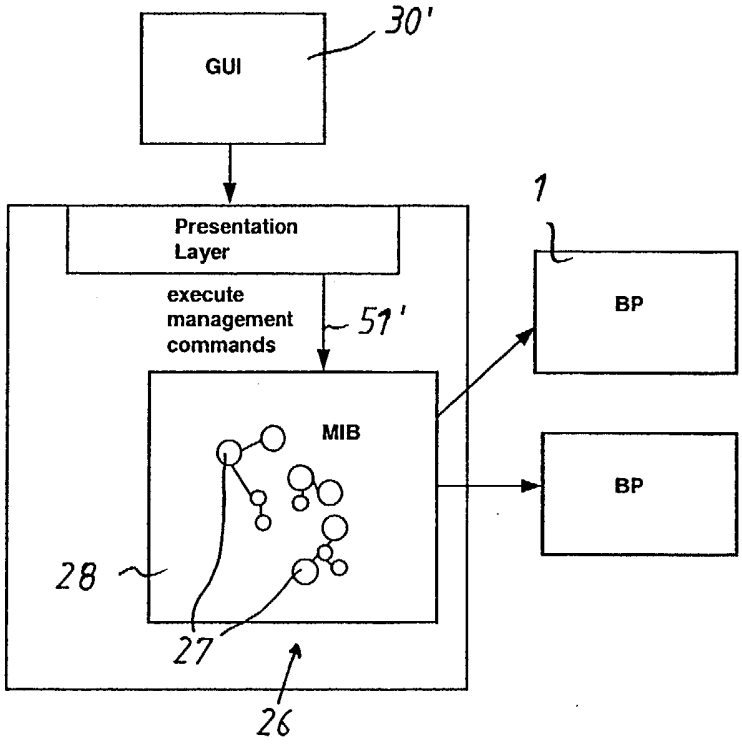

GUI
30'
Presentation Layer
execute management commands
51'
1
BP
BP
MIB
28
27
26

FIG. 5

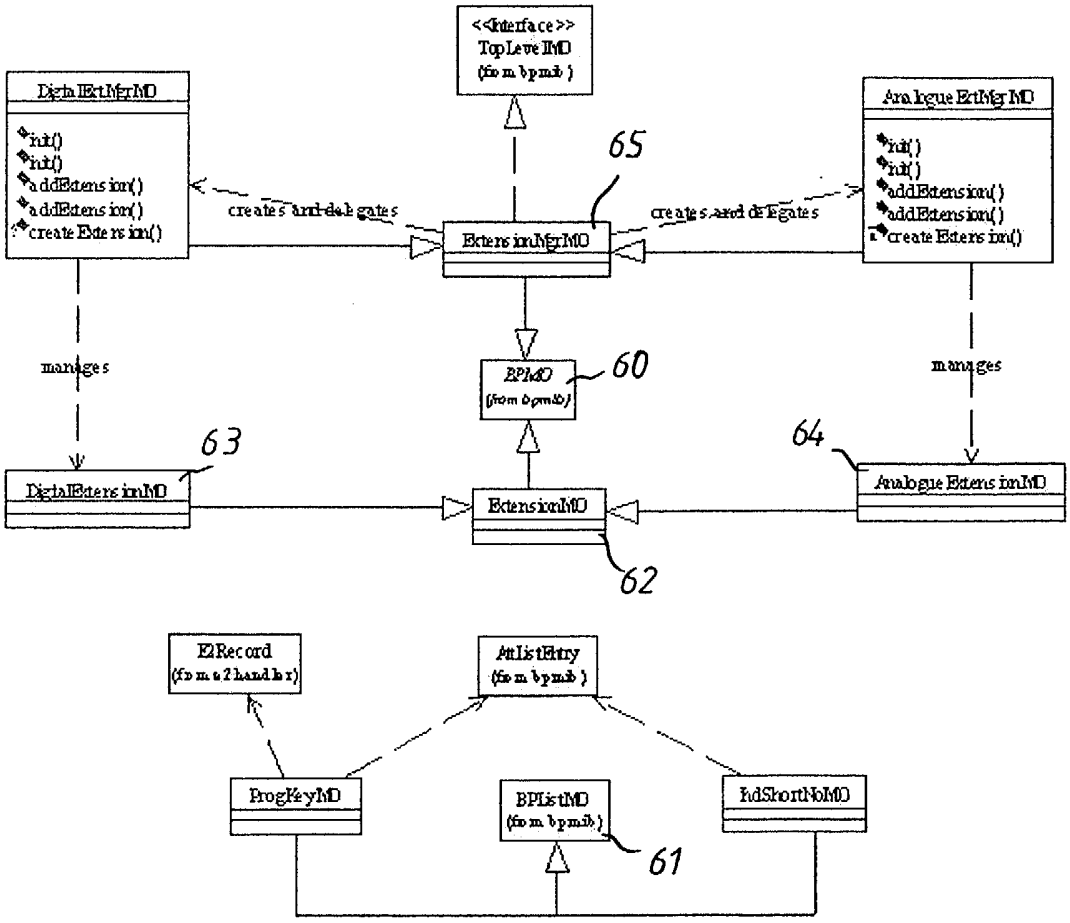

<<Interface>>
TopLevelIMD
(from bpmib)
DigitalExtMgrMD
init()
init()
addExtension()
addExtension()
createExtension()
creates and deletes
ExtensionMgrMD
65
creates and deletes
AnalogueExtMgrMD
init()
init()
addExtension()
addExtension()
createExtension()
manages
BPMD
(from bpmib)
60
manages
63
64
DigitalExtensionMD
ExtensionMD
AnalogueExtensionMD
62
E1Record
(from e2handler)
AttListEntry
(from bpmib)
ProgKeyMD
BPListMD
(from bpmib)
61
IndShortNoMD

FIG. 5A

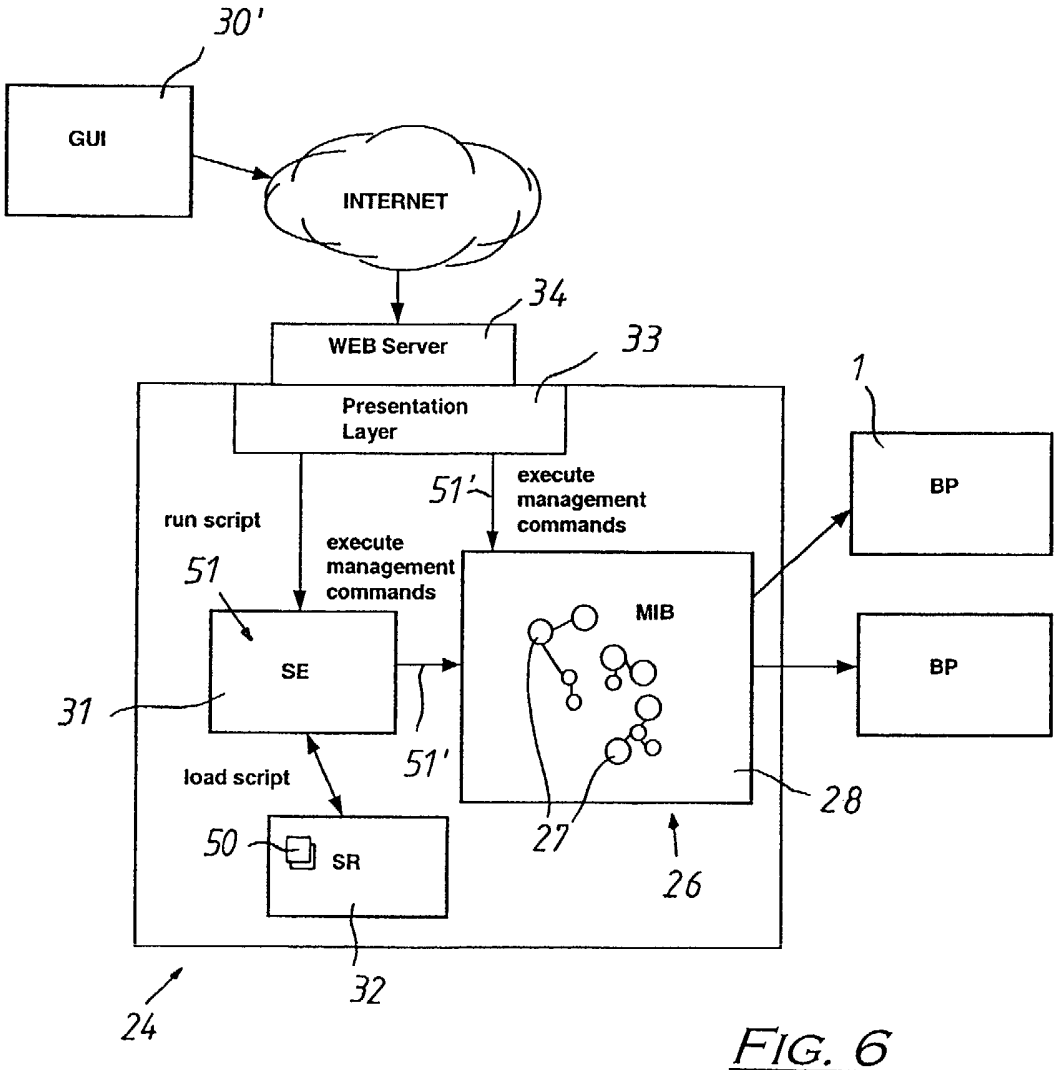

30'
GUI
INTERNET
34
WEB Server
33
Presentation Layer
1
BP
51'
execute management commands
run script
execute management commands
51
MIB
SE
BP
31
51'
load script
28
50
SR
27
26
32
24

FIG. 6

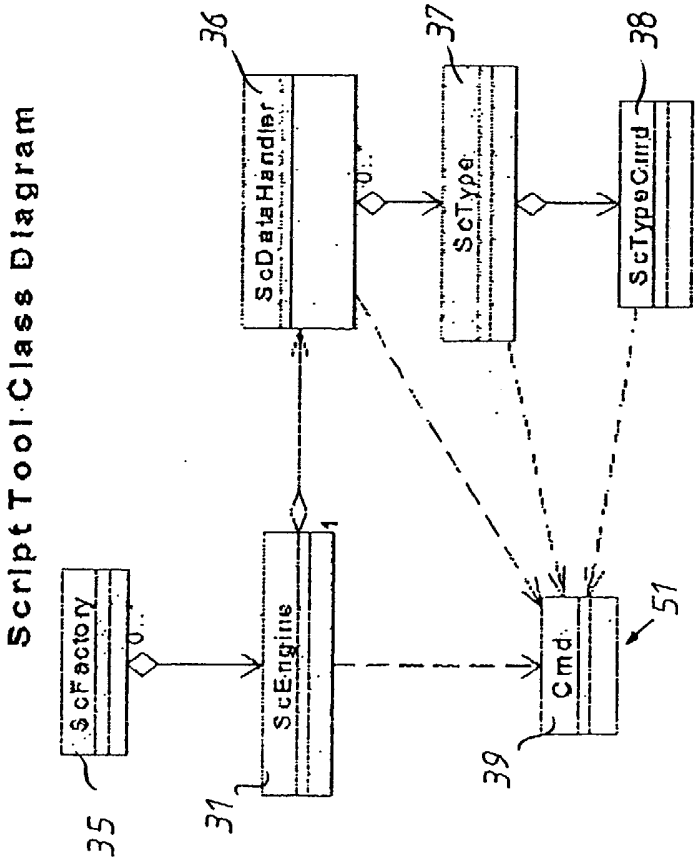

Script Tool Class Diagram
ScFactory
ScEngine
ScDataHandler
ScType
ScTypeCard
Cmd
35
31
36
37
38
39
51

FIG. 7

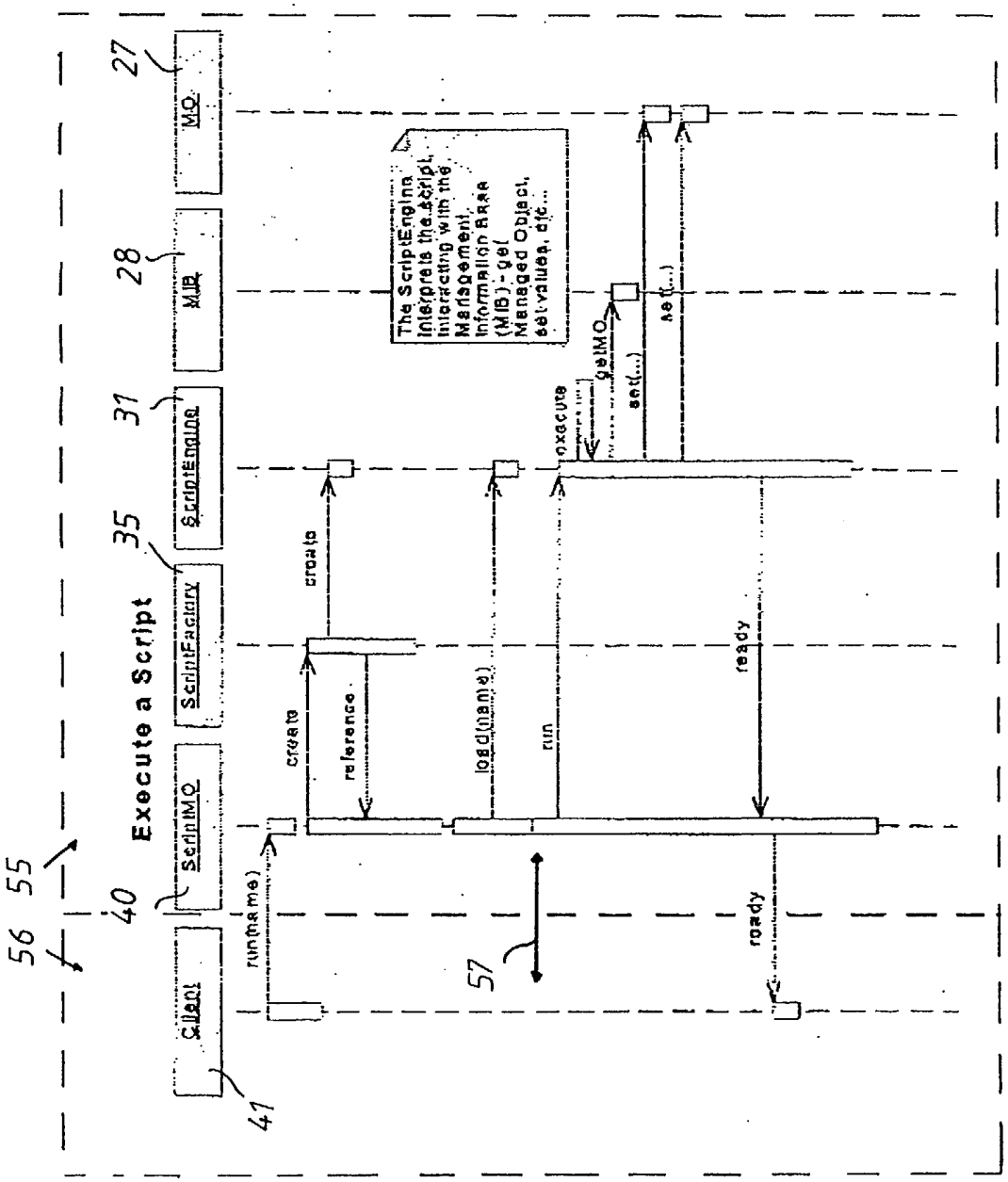

Execute a Script
Client
ScriptMO
ScriptFactory
ScriptEngine
MIB
MO
runName()
create
reference
loadName()
run
ready
getMO
set(...)
execute
The ScriptEngine interprets the script, interacting with the Management Information BASE (MIB) - get Managed Object, set values, etc...
55
56
57
40
41
35
31
28
27

FIG. 8A

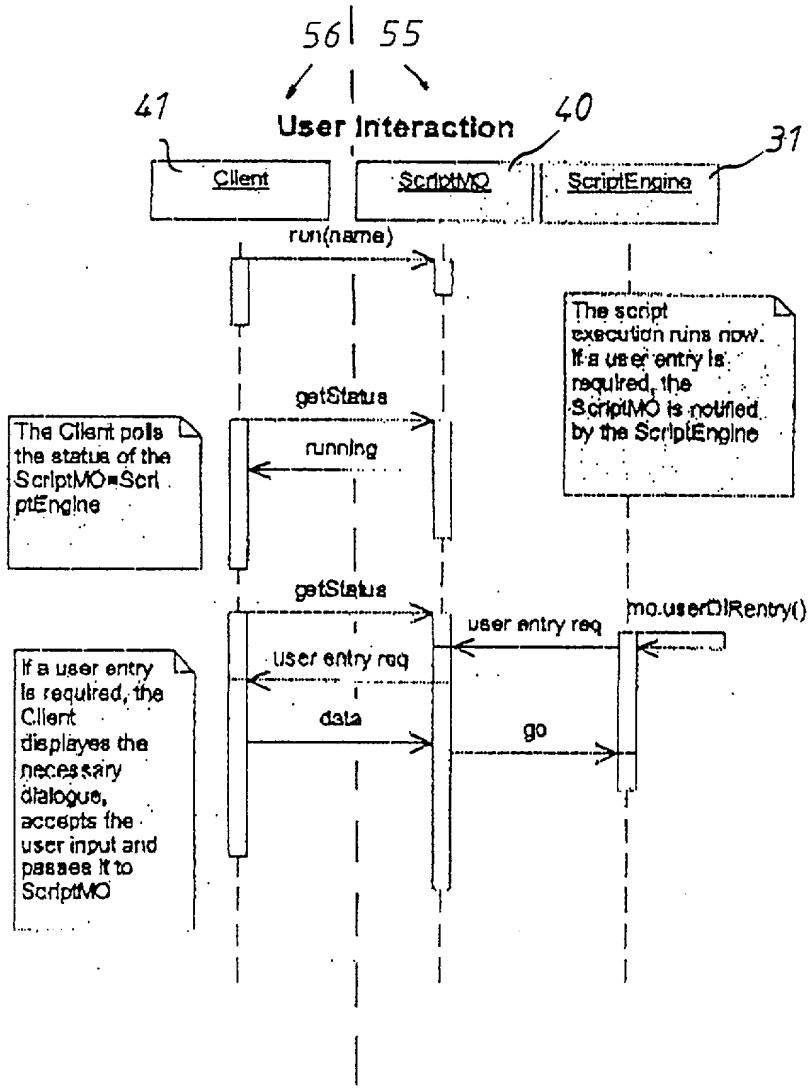

56
55
41
User Interaction
40
31
Client
ScriptMO
ScriptEngine
run(name)
The script execution runs now. If a user entry is required, the ScriptMO is notified by the ScriptEngine
getStatus
The Client polls the status of the ScriptMO=ScriptEngine
running
getStatus
mo.userDIRentry()
user entry req
user entry req
If a user entry is required, the Client displayes the necessary dialogue, accepts the user input and passes it to ScriptMO
data
go

FIG. 8B

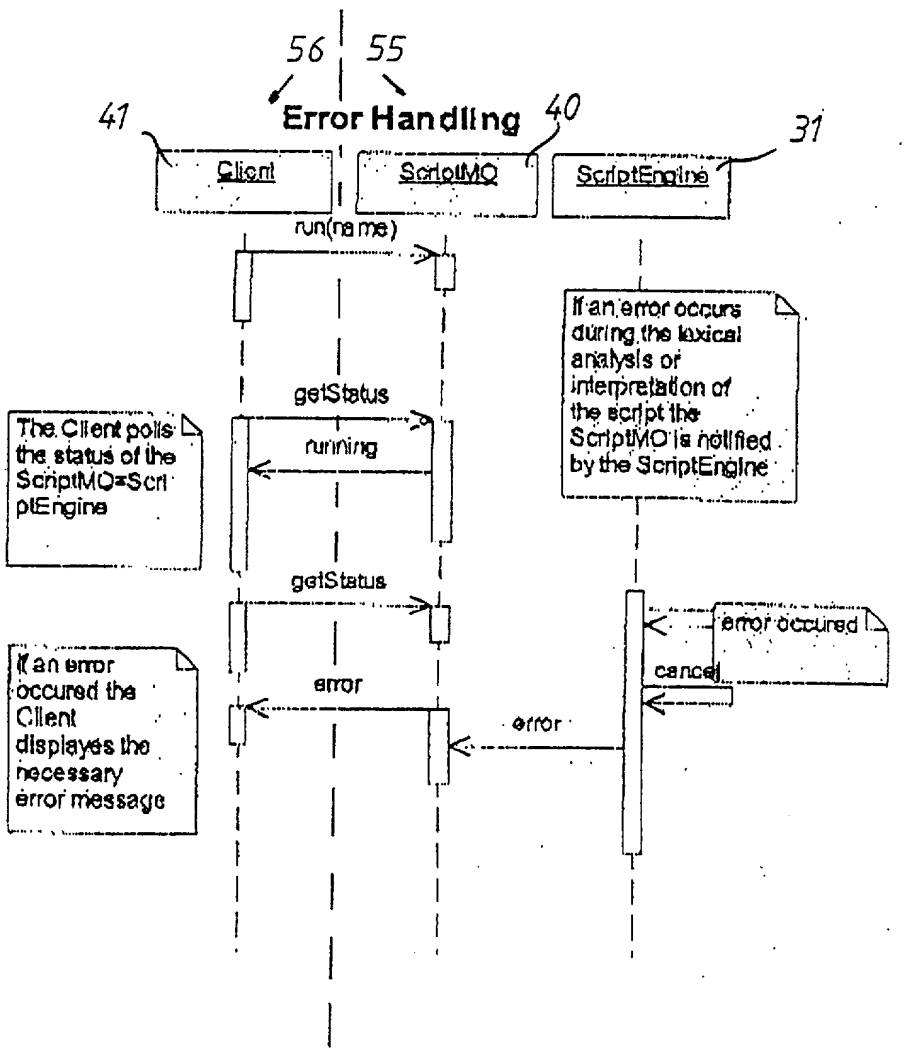

56
55
41
Error Handling
40
31
Client
ScriptMO
ScriptEngine
run(name)
If an error occurs during the lexical analysis or interpretation of the script the ScriptMO is notified by the ScriptEngine
getStatus
The Client polls the status of the ScriptMO=ScriptEngine
running
getStatus
error occured
cancel
If an error occured the Client displayes the necessary error message
error
error

FIG. 8C

ÖSTERREICHISCHES PATENTAMT

Recherchenbericht zu GM 8055/2002

Klassifikation des Anmeldungsgegenstands gemäß IPC*): G 06 F 13/00, G 06 F 15/16
Recherchierter Prüfstoff (Klassifikation):
Konsultierte Online-Datenbank:

Dieser Recherchenbericht wurde zu den **am 12. Juni 2002 eingereichten** Ansprüchen erstellt.

Die in der Gebrauchsmusterschrift veröffentlichten Ansprüche könnten im Verfahren geändert worden sein (§ 19 Abs. 4 GMG), sodass die Angaben im Recherchenbericht, wie Bezugnahme auf bestimmte Ansprüche, Angabe von Kategorien (X, Y, A), nicht mehr zutreffend sein müssen. In die dem Recherchenbericht zugrundeliegende Fassung der Ansprüche kann beim Österreichischen Patentamt während der Amtsstunden Einsicht genommen werden.

Kategorie*)	Bezeichnung der Veröffentlichung: Ländercode*), Veröffentlichungsnummer, Dokumentart (Anmelder), Veröffentlichungsdatum, Textstelle oder Figur soweit erforderlich	Betreffend Anspruch
Y	US 5 317 742 A (Bapat) 31. Mai 1994 (31.05.94) Beschreibung, Spalte 1, Zeilen 26-30,35-37,38-46, Fig. 1	1,10
A		2-9,11-21
Y	EP 0 909 058 A1 (SUN MICROSYSTEMS INC) 14. April 1999 (14.04.99) Beschreibung Spalte 1, Zeilen 59-63 und Spalte 2, Zeilen 11-20	1,10
A		2-9,11-21

Datum der Beendigung der Recherche:	Prüfer:
23. Juli 2002	Dr. MAYER

*) Bitte beachten Sie die Hinweise auf dem Erläuterungsblatt!

☐ Fortsetzung siehe Folgeblatt

ÖSTERREICHISCHES PATENTAMT

Erläuterungen zum Recherchenbericht

Die **Kategorien** der angeführten Dokumente dienen in Anlehnung an die Kategorien der Entgegenhaltungen bei EP- bzw. PCT-Recherchenberichten nur zur raschen Einordnung des ermittelten Stands der Technik. Sie stellen keine Beurteilung der Erfindungseigenschaft dar:

"**A**" Veröffentlichung, die den **allgemeinen Stand der Technik** definiert.

"**Y**" Veröffentlichung **von Bedeutung**: der Anmeldungsgegenstand kann nicht als auf erfinderischer Tätigkeit beruhend betrachtet werden, wenn die Veröffentlichung mit einer oder mehreren weiteren Veröffentlichungen dieser Kategorie in Verbindung gebracht wird und diese **Verbindung für einen Fachmann naheliegend** ist.

"**X**" Veröffentlichung **von besonderer Bedeutung**: der Anmeldungsgegenstand kann allein aufgrund dieser Druckschrift nicht als neu bzw. auf erfinderischer Tätigkeit beruhend betrachtet werden.

"**P**" Dokument, das **von besonderer Bedeutung** ist (Kategorie „X"), jedoch **nach dem Prioritätstag** der Anmeldung **veröffentlicht** wurde.

"**&**" Veröffentlichung, die Mitglied derselben **Patentfamilie** ist.

Ländercodes:

AT = Österreich; **AU** = Australien; **CA** = Kanada; **CH** = Schweiz; **DD** = ehem. DDR; **DE** = Deutschland; **EP** = Europäisches Patentamt; **FR** = Frankreich; **GB** = Vereinigtes Königreich (UK); **JP** = Japan; **RU** = Russische Föderation; **SU** = Ehem. Sowjetunion; **US** = Vereinigte Staaten von Amerika (USA); **WO** = Veröffentlichung gem. PCT (WIPO/OMPI); weitere Codes siehe **WIPO ST. 3**.

Die genannten Druckschriften können in der Bibliothek des Österreichischen Patentamtes während der Öffnungszeiten (Montag bis Freitag von 8 bis 12 Uhr 30, Dienstag von 8 bis 15 Uhr) unentgeltlich eingesehen werden. Bei der von der Teilrechtsfähigkeit des Österreichischen Patentamts betriebenen Kopierstelle können **Kopien** der ermittelten Veröffentlichungen bestellt werden.

Auf Bestellung gibt die von der Teilrechtsfähigkeit des Österreichischen Patentamts betriebene Serviceabteilung gegen Entgelt zu den im Recherchenbericht genannten Patentdokumenten allfällige veröffentlichte "**Patentfamilien**" (den selben Gegenstand betreffende Patentveröffentlichungen in anderen Ländern, die über eine gemeinsame Prioritätsanmeldung zusammenhängen) bekannt.

Auskünfte und Bestellmöglichkeit zu diesen Serviceleistungen erhalten Sie unter der Telefonnummer

01 / 534 24 - 738 bzw. 739;

Schriftliche Bestellungen:

per FAX Nr. 01 / 534 24 – 737 oder per E-Mail an Kopierstelle@patent.bmvit.gv.at