Der Jobtransferdienst für IBM-Großrechner mit dem Betriebssystem VM

P. Holleczek, R. Kummer

Regionales Rechenzentrum Erlangen

(R R Z E)

1. Einleitung

Der Job-Transfer-(JT bzw. RJE = Remote Job Entry)-Dienst hat eine besondere Bedeutung als Kommunikationsdienst in der Forschungslandschaft.

Er erlaubt insbesondere einen wirtschaftlichen Zugriff auf Super-Rechner (z.B. CRAY, ETA) deren interaktiver Zugang oft stark eingeschränkt ist. Dabei führt der Job-Zugang klassischerweise nicht an den Super-Rechner selbst, sondern an einen größeren Universalrechner (z.B. DEC, CDC, IBM), der über genügend Hintergrund Speicherkapazität verfügt und der seinerseits mit Spezialprozeduren an den Super-Rechner angeschlossen ist.

Im Rahmen der Einführung von standardisierten Protokollen im Bereich des Deutschen Forschungs-Netz (DFN) kommen als erste Stufe für den Jobtransfer Protokolle zum Einsatz, die im Sinne des ISO/OSI Referenzmodells einschließlich der Transportschicht auf ISO-Standards, darüber jedoch auf einer nationalen Vereinbarung (DFN-RJE) beruhen. Dieser Dienst soll, sobald verfügbar, durch den ISO-Dienst JTM abgelöst werden.

Im vorliegenden Fall bestand die Aufgabe, den Dienst DFN-JT für das Betriebssystem IBM-VM zu implementieren. Bei dem Betriebssystem IBM-VM liegt, im Gegensatz zu dem bekannten "batch"-orientierten Betriebssystem MVS, das Schwergewicht auf einem interaktiven Zugang.

Als Programmiersprachen stehen PASCAL mit Betriebssystem-Aufrufen (Fa. IBM) und PEARL (Fa. Werum) zur Verfügung.

2. Aufgabenstellung

Im Rahmen eines gemeinsamen DFN-Projektes zwischen der GMD Darmstadt, dem ENC (European Networking Center) der Fa. IBM, Heidelberg, dem Rechenzentrum der Universität Stuttgart (RUS) und dem Regionalen Rechenzentrum der Universität Erlangen-Nürnberg (RRZE) wurde für IBM-Anlagen mit dem Betriebssystem VM ein Remote Job Entry- (RJE) und ein Filetransfer- (FT) Protokoll implementiert. Dabei entfiel an das RRZE die Aufgabe, das RJE-Protokoll zu realisieren. Da bislang noch keine "stabile" Normung für diesen Dienst vorliegt (JTM), wurde auf eine einfache Art eines RJE-Protokolls zurückgegriffen, das von der PIX-Arbeitsgruppe (/PHB85/) erstellt wurde.

Zur Entwicklung des Programms für RJE gehört auch die Festlegung einer Benutzer- und Operateurschnittstelle und der Anschluß eines BATCH-Verarbeitungssytems (Abwicklung von Aufträgen im Hintergrund /VMB85/) innerhalb des VM-Betriebssystems.

Der RJE-Dienst ist asymmetrisch. Er unterscheidet zwischen Jobs, die von einem Eingaberechner zu einem Verarbeitungsrechner und Output, der vom Verarbeitungsrechner zum Ausgaberechner transportiert wird. Der Verarbeitungsrechner muß die ankommenden Jobs an das BATCH-System weiterreichen, die fertigen Ergebnisse sammeln und an den Ausgaberechner zurückschicken. Eingaberechner und Ausgaberechner sind oft identisch.

Da in der Implementation in der Regel nicht zwischen Eingabe-/Ausgabe-/Verarbeitungsrechner unterschieden wird, muß das JT-Programm gleichzeitig folgende Datenströme verwalten:

Empfangen & Senden von Jobs vom bzw. zum Netz
Empfangen & Senden von Output vom bzw. zum Netz
Senden von Jobs zum BATCH-System
Empfangen von Output vom BATCH-System
Eingaben und Rückmeldung von bzw. zu einer Benutzerschnittstelle

Alle diese Vorgänge finden außerdem gleichzeitig für verschiedene logische Verbindungen statt.

Das JT-Programm hat also im hohen Maße viele Aufgaben gleichzeitig zu tun, die zudem meist asynchron angestoßen werden.

Es liegt daher nahe, zur Entwicklung des JT-Programms parallele Prozesse als Strukturierungshilfsmittel heranzuziehen, nach Möglichkeit unterstützt durch eine geeignete Programmiersprache.

Da das RJE-Protokoll lediglich die Schicht 7 (Anwendungsschicht) des ISO-Schichtenmodells darstellt, sind zumindest die Schichten 1 bis 4 vorauszusetzen. Auf die Schichten 5 (Sitzungsschicht) und 6 (Präsentationsschicht) konnte im Rahmen dieses Projektes verzichtet werden. Die Schicht 4 (Transportschicht, /T7084/) liegt für die VM-Systeme in verschiedenen Formen vor (s. Kap. 3).

3. Einbettung in die VM-Welt.

Das Grundbetriebssystem CP (Control Program, /CP87/) der IBM-Rechner der Serie 43xx, 93xx und 3090 ermöglicht die Bereitstellung von sogenannten Virtuellen Maschinen (VM). Darauf aufbauend findet innerhalb der Virtuellen Maschinen vor allem das Dialog- und den Programmablauf unterstützende Betriebssystem CMS (Conversational Monitor System, /CMS87/) Verwendung. Im folgenden wird also immer davon ausgegangen, daß sowohl das CP als auch das CMS zur Verfügung stehen.

Es laufen nicht alle die im Zusammenhang mit diesem Projekt entstandenen Programme in einer einzigen VM ab. Es ergibt sich folgendes Übersichtsbild für die verschiedenen Virtuellen Maschinen:

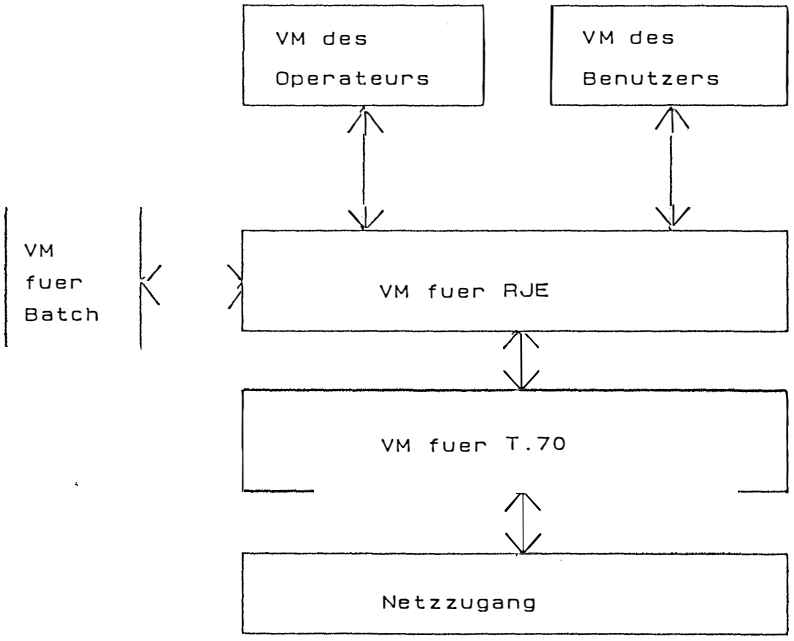

Bild 1: Übersicht über die verschiedenen Virtuellen Maschinen

Zur Beschreibung der Kommunikationen (Pfeile) zwischen den Virtuellen Maschinen sei auf das Kapitel 4.1 verwiesen.

Die einzelnen Virtuellen Maschinen lassen sich kurz, wie folgt, beschreiben:

- VM des Benutzers: Dem Benutzer des RJE stehen Kommandos zur Verfügung, um BATCH-Aufträge oder Listen an ferne Rechner zu senden und administrative Befehle einzugeben, wie z.B. den Status der Jobs abzufragen.

- VM des Operateurs: Bei einem Operateur handelt es sich um einen privilegierten Benutzer, der zwar die gleichen Kommandos wie ein normaler Benutzer absetzen kann, aber nicht der Beschränkung auf seine eigenen Jobs unterliegt. Er kann also alle Jobs von allen Benutzern in der RJE-Maschine administrieren.

- VM für BATCH: In dieser Virtuellen Maschine befindet sich das Programm zur Abwicklung von BATCH-Aufgaben. Dieses Programm trennt die Steueranweisungen von möglichen Eingabedaten, führt die Steueranweisungen aus und liefert die Ergebnisse des Auftrags an die RJE-Maschine zurück.

- VM für RJE: Wie bereits in Kapitel 2 erwähnt, wickelt die Maschine das DFN-RJE-Protokoll ab. Damit verbunden ist das Entgegennehmen, Speichern und Weiterleiten von BATCH-Aufträgen und -Ergebnisprotokollen aus dem DATEX-P-Netz, vom Benutzer und von der BATCH-Maschine. Außerdem müssen die Administrationsbefehle der Benutzer und des Operateurs ausgeführt werden.

- VM für T.70: Für Rechner mit Betriebssystem VM existieren zwei Programmpakete zur Abwicklung des Transportprotokolls:
  - Das Programm "TLMAIN" vom ENC Heidelberg (/ENC86/), das fertig gebunden vorliegt.
  - Die Makro-Sammlung "OTSS" (/OTSS87/) als offizielles Produkt der Firma IBM setzt wiederum die Makro-Sammlung "OSNS" (/OSNS87/) voraus. Dabei können die Prozeduren von OTSS nicht direkt genutzt werden, sondern müssen von einem in der Transportmaschine ablaufenden Anwendungsprogramm aufgerufen werden.

  Die genannten Transportprogramme erfüllen die CCITT-Norm (Comité Consultatif International Telegraphique et Telephonique) T.70 bzw. die Klasse 0 der ISO-Norm für das Transportprotokoll. Sie sind jeweils in PASCAL programmiert und fertig ausgetestet. Als zusätzliche Funktionenmenge sind Möglichkeiten zur Gebührenerfassung und zur Erstellung von Statistiken implementiert.

- Netzzugang: Um den Programmen für die Transportschicht (TLMAIN oder OTSS mit OSNS) den Zugang zu einem X.25-Netz (z.B. dem DATEX-P-Netz der Deutschen Bundespost) zu ermöglichen, sind verschiedene Vorrechner nötig. Wird das Programm TLMAIN benutzt, ist ein Rechner der Reihe Serie/1 notwendig. Bei der Verwendung von OTSS und OSNS ist ein Vorrechner des Typs 37X5 oder 3720 Voraussetzung. Als billigere Alternative dazu ist es auch möglich, den Zugang über den "Integrated Communications Adapter" (ICA) zu nutzen, so daß auf einen Vorrechner verzichtet werden kann.

4. Schnittstellen der RJE-Maschine zum vorhandenen Betriebssystem

4.1. Schnittstellen zur Kommunikation zwischen Virtuellen Maschinen bzw. zum Benutzer

Das Betriebssystem CP bietet zur Kommunikation zwischen Virtuellen Maschinen das IUCV-Paket (Inter-User Communications Vehicle, /CP87/) an, das den Austausch von sogenannten "Messages" (Nachrichten) erlaubt (Senden und Empfangen). Dadurch können Informationen von begrenzter Länge (z.B. 80 Zeichen einer Bildschirmzeile) von einer VM zu einer anderen gesendet werden:

MESSAGE userid ..... messagetext .....
mit userid ::= Name der Empfänger-VM.

Statt des CP-Kommandos "MESSAGE" wird das CMS-Kommando "TELL" verwendet, um auch Benutzer auf anderen über RSCS ("Remote Spool") erreichbaren Rechnern zu erreichen.

Müssen größere Datenmengen ausgetauscht werden, gibt es die Möglichkeit, Dateien von einer VM zu einer anderen zu senden. Um einen File senden zu können, benötigt man das CMS-Kommando SENDFILE (/CMS87/), für das Empfangen steht das CMS-Kommando RECEIVE zur Verfügung.

Mit Hilfe dieser Kommandos ist ein vollständiger Datenaustausch zwischen Virtuellen Maschinen sichergestellt. Wie diese Kommandos, die zunächst Dialogkommandos sind, vom RJE-Programm aus angesprochen werden, ist in Kapitel 4.2. behandelt.

Die Benutzer- und Operateurkommandos können von EXEC-Routinen, in REXX (Job-Control-Sprache /REX 83/) geschrieben, entgegengenommen und von diesen syntaktisch analysiert werden. Ist das Kommando syntaktisch richtig, wird es mit dem CMS-Kommando "TELL" and die RJE-Maschine gesendet.

4.2. Aufrufe des PEARL-Betriebssystems

Da die Implementierung in der Programmiersprache PEARL (/PEA78/, /PEA80/) erfolgt (siehe dazu Kapitel 5.2.1.) und sowohl CP als auch CMS die Sprache nicht unterstützen, existiert ein eigenes Laufzeit- und Betriebssystempaket für PEARL-Programme. Dieses Paket verwendet einige CMS-Makros bzw. CP-Prozeduraufrufe, die nachfolgend angeführt sind (/KNE86/):

- Makros für File-E/A
- Makros für Terminal-E/A
- Makros für Drucker-Ausgabe
- Supervisor-Calls und CP-Timer

5. Verwendete Spezifikationstechnik und Programmiersprachen

5.1. Spezifikationstechnik PASS

Am Regionalen Rechenzentrum Erlangen, und inzwischen auch am ENC, wird seit einiger Zeit mit der Spezifikationstechnik PASS (Parallel Activities Specification Scheme, /FLE84/) gearbeitet. Sie wurde in dem Arbeitstreffen "Spezifikationstechniken im DFN" (/SPE83/) im November 1983 in Darmstadt vorgestellt. Unter anderem wurde sie in dem DFN-Projekt zur Implementierung von Basis-Diensten (FT mit T.70, X.28/X.3/X.29, /BAS83/) eingesetzt.

Diese Methode ist graphisch orientiert und erlaubt es, parallele Prozesse und ihre Interaktionen über Botschaften oder gemeinsame Objekte in übersichtlicher Weise darzustellen. Eigenständige Prozesse, die unabhängig voneinander (parallel) ablaufen können, sind als Strukturierungshilfsmittel allgemein bewährt, da sie voneinander unabhängige Abläufe leicht darstellbar machen. Als Hilfsmittel, um die Kommunikation zwischen den Prozessen anschaulich zu beschreiben, dienen Botschaften. Sie erlauben durch ihre kompakte Schreibweise, unabhängig von Implementationsdetails, die Synchronisation und den Datenaustausch zwischen Prozessen transparent zu machen.

Für eine genaue Beschreibung sei auf die angegebene Literatur verwiesen.

Das gesamte RJE-Protokoll ist mit der Spezifikationstechnik PASS beschrieben.

Als Beispiel sei auf die Bilder 2 ("Kommunikationsstruktur" = Darstellung aller Prozesse mit ihrer Kommunikation) und 3 ("Ablaufsteuerung" = Feinstruktur eines Prozesses) in Kaptiel 6 verwiesen.

5.2. Verwendete Programmiersprachen

5.2.1. PEARL als Sprache paralleler Prozesse

Da die Methode PASS eine klare Darstellung paralleler Prozesse erlaubt, liegt es nahe, eine Programmiersprache zu verwenden, die ebenfalls die Darstellung paralleler Prozesse erlaubt, um eine möglichst einfache Abbildung der Spezifikation auf Programmcode zu erzielen. In dem in Kapitel 4.1. erwähnten Projekt ("Implementierung von Basis-Diensten") wurde bereits erfolgreich die Abbildung von PASS auf die Programmiersprache PEARL (/PEA78/, /PEA80/) durchgeführt.

Es ist möglich, die in der Spezifikation definierten Botschaften in PEARL mit Hilfe von Prozeduren und Semaphoren nachzubilden (siehe 6.1.). Sogenannte "Schreibprozeduren" sollen dabei das Senden nachbilden, während "Leseprozeduren" das Empfangen darstellen. Um den Datenaustausch zu synchronisieren, werden "REQUEST"- und "RELEASE"-Operationen auf Semaphoren verwendet.

Für IBM-Rechner mit den Betriebssystemen MVS und VM existiert von der Firma WERUM, Lüneburg, ein PEARL-Compiler und -Laufzeitsystem (/WER78/, /WER80/). Dabei werden PEARL-Programme von einem PL/I geschriebenen Compiler im Assemblercode übersetzt und dann mit Hilfe des Assembler-Übersetzers in Maschinencode transferiert. Anschließend werden übersetzte Programme mit der PEARL-Laufzeitbibliothek zu einem lade- und lauffähigen Modul gebunden.

5.2.2. Zugriff auf Systemroutinen über PASCAL-Prozeduren

Wie bereits in Kapitel 3.1 erwähnt, gibt es CP- bzw. CMS-Kommandos, um die Kommunikation zwischen Virtuellen Maschinen zu erzielen. Um einen möglichst bequemen Anschluß für die FT- und RJE-Protokollprogramme zur Verfügung zu stellen, wurde vom ENC im Rahmen der Implementierung des Transportprotokolls das Prozedurenpaket ("Message Passing Handler" MPH4 /ENC86/) realisiert. In diesem Paket sind die wichtigsten Aufrufe zur Kommunikation mit der VM von T.70 ("CONNECT REQUEST", "DATA REQUEST", etc.) und mit anderen Virtuellen Maschinen ("SEND_MESSAGE", "RECEIVE_MESSAGE", etc.) zusammengefaßt. Eine genauere Beschreibung der Prozeduraufrufe befindet sich in /HOL85/.

Für die Durchführung weiterer CMS- und CP-Kommandos gibt es in PASCAL/VS (/PAL85/) die Möglichkeit, die Prozedur "CMS" (/PAP85/) zu verwenden. Sie dient zum Absetzen von Kommandos in CMS-Umgebung und liefert in Form eines ganzzahligen Rückgabeparame-

ters die Information, ob die Aktionen erfolgreich abgeschlossen oder mit einem Fehler beendet wurden. Damit können auch ganze EXEC-Routinen, z.B. REXX geschrieben, abgewickelt werden.

Da das RJE-Protokoll in PEARL codiert wurde, andererseits das MPH4 in PASCAL programmiert ist, war es notwendig, einen sauberen Übergang von PEARL nach PASCAL zu schaffen. Da in unserem Fall lediglich PASCAL-Prozeduren von PEARL-Programmen aufgerufen werden, reduziert sich das Problem auf den Vergleich der Datenstrukturen und auf einen geeigneten Mechanismus, um von PEARL-Programmen aus PASCAL-Prozeduren aufzurufen und die Parameter zu übergeben.

Bei den Datenstrukturen wurde auf die Verwendung von Strukturen ("RECORDS") verzichtet, da der Aufbau in den beiden Programmiersprachen zu unterschiedlich ist. Die folgende Liste gibt eine Abbildung von PASCAL- auf PEARL-Datentypen wider.

PASCAL	PEARL	
INTEGER	FIXED (n)	mit 16 < = n < = 32
POINTER	REF	
PACKED ARRAY [1..2*n] OF CHAR	(n) CHAR(2)	mit n > = 1
PACKED ARRAY [1..2*n] OF CHAR	CHAR (2*n)	mit n > = 1
STRING (n-2)	CHAR (n)	mit n > = 3, CHAR(1) = 0 CHAR(2) = tatsächliche Länge, STRING (1..n-2) = CHAR (3..n)

Beim Aufruf von PASCAL-Prozeduren sind einige Konventionen zu beachten (siehe dazu auch den "Programmer's Guide", /PAP85/). Insbesondere bei der Registerbelegung und den damit verbundenen reservierten Speicherbereichen von globalen und lokalen Daten sind strenge Vorgaben zu befolgen. Da sich die Firma WERUM in ihrer Implementation des PEARL-Compilers und -Laufzeitsystems sehr genau daran gehalten hat (/KNE86/), gibt es an dieser Stelle keine Probleme. Trotzdem sind bei der Übergabe von Prozedurparametern einige Regeln zu beachten, wie z.B.

- Die Parameterübergabe kann nur per Adresse, also "IDENT" (PEARL) bzw. "VAR" (PASCAL), erfolgen.
- Da PEARL im ASCII-Alphabet arbeitet, PASCAL dagegen im EBCDIC, müssen bei Text- bzw. Zeichenanalysen Codeumwandlungstabellen verwendet werden.

Da die Prozedurschnittstelle "MPH4" für die RJE- und die FT-Implementierung einheitlich bleiben soll, wurde für den Anschluß dieser Prozeduren an PEARL eine Prozedurzwischenschicht "PMPH4" in PASCAL entwickelt, die die obigen Regeln beachtet.

6. Implementation

6.1. Umsetzung der Prozesse und Botschaften aus der Spezifikation in PEARL-Programme

Die in der Spezifikation angegebenen Prozesse sind direkt in PEARL-Prozesse umgesetzt worden, wobei die Sender- und Empfängerprozesse mehrfach vorkommen. Die in PASS formulierten Botschaften werden in PEARL durch Prozeduren simuliert.

In der nachfolgenden "Kommunikationsstruktur" sind die PASS-Prozesse bzw. PEARL-Prozesse angegeben:

PASS-Prozeß	PEARL-Prozeß/-Prozedur
DISTRI	DISTRI
EMPFANG	EMPF
RDRWAI	RDRWAI
SENDER	SENDER
SERVER	STRTTA
TIMER	TIMER
TYGET	TYGETE

Um den Zusammenhang zwischen den Ablaufdiagrammen bzw. den Benutzermaschinen und dem Programmcode herzustellen, wurden die Knoten- und Botschaftsnamen als Kommentare in die Programme übernommen.

Bild 2 zeigt die am RJE-Programm beteiligten Prozesse und die zwischen ihnen ausgetauschten Botschaften (Kommunikationsstruktur aus PASS).

Bild 2: Kommunikationsstruktur

mit i gerade (='EMPFANG'),
j ungerade (='SENDER')

6.2. Zustandsdiagramme der Prozesse und des Gedächtnisses

Jeder der spezifizierten bzw. programmierten Prozesse befindet sich zu bestimmten Zeitpunkten in einem definierten Zustand. Fast alle Prozesse wechseln dabei zwischen den Zuständen "nicht aktiv" und "aktiv". Besondere Zustände bzw. Zustandsübergänge haben nur die Prozesse SENDER und EMPFANG, wie das nachfolgende Bild, vereinfacht, als Ausschnitt aus einer PASS-"Ablaufsteuerung" zeigt.

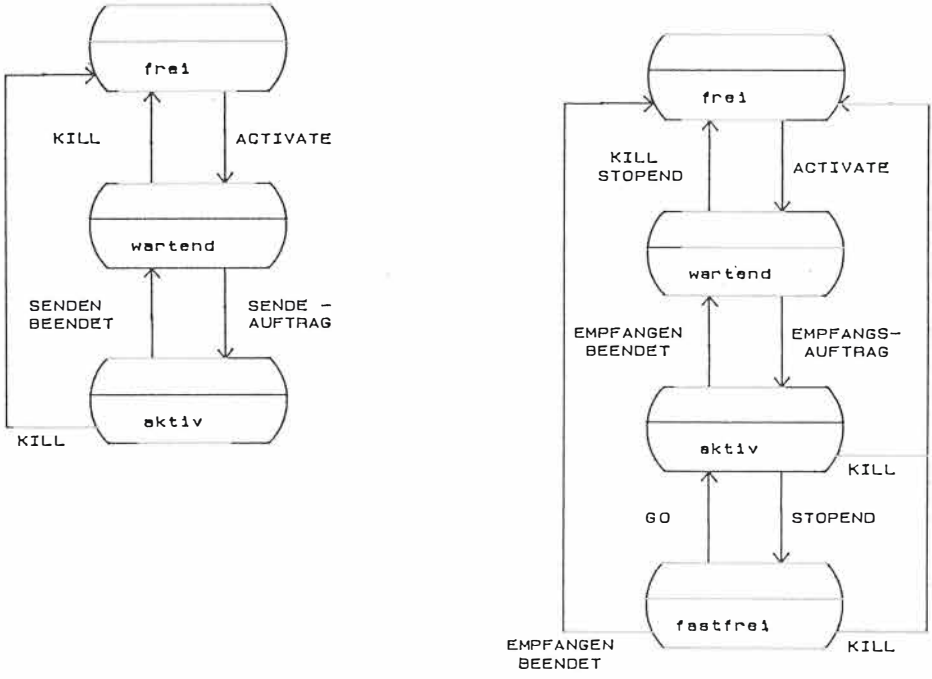

Bild 3: Zustandsdiagramm der Prozesse EMPFANG (links) und SENDER (rechts)

7. Erfahrungen

Die gewählte Implementationstechnik, d.h.

- die strikte Einhaltung des Software Life Cycle mit typ-down-Entwurf,
- die in der Spezifikations-Technik vorgesehenen Kommunikationsmittel zwischen parallelen Prozessen,
- die schematische Umsetzung der Spezifikation im Programm-Code,

erwies sich als außergewöhnlich erfolgreich. So konnte die vorgesehene Implementationszeit von 4 Mannjahren deutlich unterschritten werden, obwohl im Zusammenspiel von PEARL-/PASCAL-Laufzeitsystem und VM-Betriebssystem eine Reihe von Tücken vorborgen waren.

Mittlerweile ist RJE-VM an einer Reihe von Universitäten auf Rechner der unterschiedlichsten Größenordnung im Einsatz, angefangen z.B. von einer 4361 (ca. 1 MIPs) bis zu einer 3090-200 (ca. 30 MIPs). Das Programm hat sich bislang als stabil erwiesen, was die Wünsche nach einer immer komfortableren Benutzerschnittstelle nicht minderte.

8. Literatur

/BAS83/ Andres, Fleischmann, Hillmer, Holleczek, Kummer: "Pflichtenheft für die Basis-Dienste des Erlanger DFN-Anschlusses", Interner Arbeitsbericht Nr. 193, RRZE, Erlangen, 1983

/CMS87/ IBM Corporation: "VM/SP CMS Command and Macro Reference, Release 5", 1987

/CP87/ IBM Corporation: "VM/SP CP Command Reference for General Users, Release 5", 1987

/ENC86/ Haas, Kropp, Reinhardt, Schulz: "OSI Transport Layer Implementation for VM/SP", European Networking Center, Heidelberg, 1986

/FIS86/ J. Fischer: Private Mitteilungen, Erlangen, 1986

/FLE84/ A. Fleischmann: "Ein Konzept zur Darstellung und Realisierung von verteilten Prozeßautomatisierungssystemen", Mitteilungsblatt des RRZE, Erlangen, 1984

/HOL85/ R. Holliday: "MPH4 Functions and Internals", Heidelberg, 1985

/KNE86/ E. Kneuer: Private Mitteilungen, Lüneburg, Erlangen, 1986

/OSNS87/ IBM Corporation: "Open Systems Network Support (OSNS)", Release 2.0, Stuttgart, 1987

/OTSS87/ IBM Corporation: "Open Systems Transport and Session Suppport (OTSS)", Release 2.0, Stuttgart, 1987

/PAL85/ IBM Corporation: "PASCAL/VS - Language Reference Manual - Program Number: 5796-PNQ", 1985

/PAP85/ IBM Corporation: "PASCAL/VS - Programmer's Guide - Program Number: 5796-PNQ, 1985

/PEA78/ DIN 66253 Teil 2: "Programmiersprache PEARL, FULL PEARL", Beuth Verlag, Berlin, 1980

/PEA80/ DIN 66253 Teil 1: "Programmiersprache PEARL, BASIC PEARL", Beuth Verlag, Berlin, 1978

/PHB85/ Zentrale Projektleitung - DFN: "Protokollhandbuch Version 2", 1983

/REX83/ IBM-Corporation: "VM/SP System Product Interpreter Reference, Release 3", 1983

/SPE83/ Bauerfeld, Henken - ZPL-DFN: "Spezifikationstechniken im DFN", Tagungsband zum Arbeitstreffen am 28.-30. November 1983 in Darmstadt

/T7084/ ISO: "OSI - Connection Oriented Transport Protocol Specification, Draft International Standard ISO/DIS 8073", 1984

/VMB85/ IBM Corporation: "VM Batch Subsystem Program Description / Operations Manual", 1985

/WER78/ Werum, Windauer: "PEARL, Process and Experiment Automation Realtime Language", Vieweg & Sohn, Braunschweig, 1978

/WER80/ WERUM DV-Systeme GmbH: "PEARL Programming System for IBM 43xx with CMS - User's Guide", Lüneburg, 1986