WELCHE VORTEILE BIETET DIE PROZESSRECHNERSPRACHE PEARL?

E. Joho

Stuttgart

Institut für Regelungstechnik und Prozeßautomatisierung
Universität Stuttgart

28. 07. 1980

PEARL (Process and Experiment Realtime Language) ist die zur Zeit modernste Prozeßrechner-Programmiersprache. PEARL-Systeme (d.h. PEARL-Compiler, PEARL-Betriebs- und Laufzeitsysteme, PEARL-Test und -Bediensysteme) werden heute von allen bedeutenden Prozeßrechensystem-Herstellern angeboten.
Um die Vorteile und die Bedeutung dieser seit 1969 entwickelten, inzwischen genormten (DIN 66253) Prozeßrechner-Programmiersprache abschätzen zu können, ist es zweckmäßig, sich einmal der wichtigsten Anforderungen, die an eine solche Sprache gestellt werden müssen, zu vergegenwärtigen:

- Forderung nach Echtzeitspracheigenschaften,
- Forderung nach Kommunikation zwischen Prozeßrechner und technischem Prozeß durch Ein-, Ausgabe von Prozeßdaten,
- Forderung, ein Programmsystem modular, in Ausbaustufen aufzubauen,
- Forderung nach anwenderfreundlicher Handhabbarkeit (methodische Erlernbarkeit der Sprache, Zuverlässigkeit und hoher Dokumentationswert der Programme).

Durch welche Sprachmittel und durch welche Grundkonzeption für den Aufbau eines Programmsystems PEARL diesen Anforderungen begegnet und ihnen gerecht wird, zeigen die folgenden Abschnitte:

Echtzeit-Sprachmittel

Zur Steuerung technischer Prozesse sind zeitlich parallele, voneinander unabhängige Abläufe von Programmteilen erforderlich. Diese müssen bei Auftreten spontaner Ereignisse im technischen Prozeß oder auch zu einplanbaren Zeiten angestoßen werden können. Dazu wird in PEARL die Task bereit gestellt, worunter man den Ablauf eines Programmstücks unter der Kontrolle des Betriebssystems versteht. Eine Task kann verschiedene Zustände einnehmen, die Steuerung der Übergänge von Taskzuständen wird durch die Tasksteueranweisungen vorgenommen. In Bild 1 ist dieser Zusammenhang in einem vereinfachten Taskzustandsdiagramm dargestellt. Werden Tasksteueranweisungen in Verbindung mit einer Art "Startbedingung" verwendet, um eine einmalige oder zyklische Einplanung von Tasks in Abhängigkeit von Zeitbedingungen oder Unterbrechungssignalen zu erreichen, spricht man vom "Scheduling". Ein Anwendungsbeispiel der Echtzeit-Sprachmittel zur Realisierung einer Zeitüberwachung ist in Bild 2 dargestellt. Als technischer Prozeß wird eine Paketverteilanlage betrachtet, die aus einer Eingangsstation (s. Bild 3) und sieben Verteilstationen besteht. Durch die Zeitüberwachung soll überprüft werden, ob ein Paket die Eingangsstation, nachdem diese in die Freigabestellung gebracht wurde, ordnungsgemäß verlassen hat oder eventuell eingeklemmt ist.

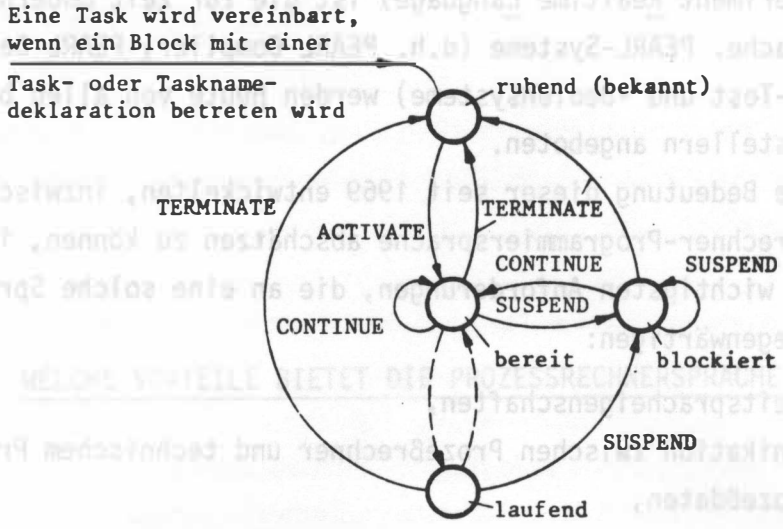

Bild 1: Taskzustände und Tasksteueranweisungen (die durch gestrichelte Pfeile dargestellten Übergänge werden durch das Betriebssystem veranlaßt)

```
174  ESSTEUERUNG: TASK GLOBAL;
                    :
232  SEND FREIGABEORGANSTELLSIGNAL TO FREIGABEORGAN; /*DIE EINGANGSSTATION WIR
233                                          IN DIE FREIGABESTELLUNG GEBRACHT*/
234  AFTER MAXAUSLAUFZEIT ACTIVATE AUSLAUFFEHLERMELDEN; /*NACH DER ZEITDAUER
235  "MAXAUSLAUFZEIT" WIRD DIE TASK "AUSLAUFFEHLERMELDEN" AKTIVIERT*/
236  WHEN PAKETAUSLAUFIRPT RESUME; /*DIE TASK "ESSTEUERUNG" WIRD SOLANGE ANGE-
237  HALTEN, BIS DER INTERRUPT "PAKETAUSLAUFINTERRUPT" EINTRIFFT*/
238  PREVENT AUSLAUFFEHLERMELDEN;
239  /*DIE ZUVOR EINGEPLANTE TASK "AUSLAUFFEHLERMELDEN" WIRD WIEDER AUSGEPLANT
                    :
313  END; /*ENDE DER TASK "ESSTEUERUNG*/
```

Bild 2: Programmierung einer Zeitüberwachung in PEARL

In Bild 2 tauchen zwei Tasksteueranweisungen auf, die in dem vereinfachten Zustandsdiagramm von Bild 1 noch nicht aufgeführt sind:
Zum einen die RESUME-Anweisung, durch die eine Task bis zum Eintreffen eines Interrupts (oder aber für eine bestimmte Zeitdauer) angehalten werden kann.
Zum anderen die PREVENT-Anweisung, durch die Einplanungen für eine Task gelöscht werden können. Im obigen Fall wird bei Eintreffen des Interrupts "PAKETAUSLAUFIRPT" (ausgelöst bei einem Dunkel-Hell-Übergang des Meldeorgans, also dann, wenn das Paket die Eingangsstation verlassen hat) die angehaltene Task "ESSTEUERUNG" fortgesetzt und danach über die PREVENT-Anweisung die zeitlich eingeplante Aktivierung der Task "AUSLAUFFEHLERMELDEN", die einen nicht ordnungsgemäßen Paketauslauf melden soll, wieder gelöscht.
Um eine Synchronisierung parallel ablauffähiger Programmteile zu erreichen, werden von PEARL sogenannte Semaphor-Variablen und Semaphor-Operationen bereitgestellt, durch die beispielsweise eine vorgegebene Reihenfolge von Tasks erzwungen, oder ein gleichzeitiger Zugriff auf Betriebsmittel (bspw. einen Schnelldrucker) oder Daten verhindert werden kann.

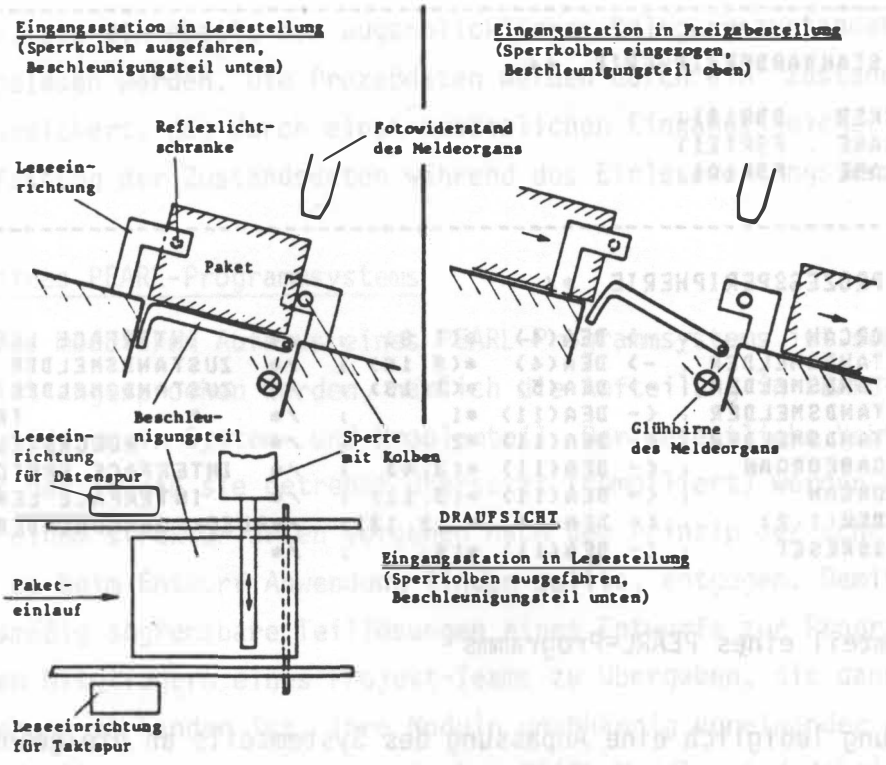

Bild 3: Eingangsstation der Paketverteilanlage

Sprachmittel zur Ein-/Ausgabe von Prozeßdaten

Vorab muß in diesem Abschnitt der prinzipielle Aufbau eines PEARL-Programms angesprochen werden. Ein PEARL-Programm ist in Moduln gegliedert (siehe dazu auch den nächsten Abschnitt "Modularer Aufbau eines PEARL-Programmsystems"), die wiederum in einen System- und einen Problemteil aufgeteilt werden können.
Im Systemteil wird die Konfiguration der Standard- und Prozeßperipherie beschrieben, sowie symbolische - vom Benutzer zu wählende - Namen an die systemdefinierten Datenstationen und deren Anschlüsse vergeben (Datenstationen können als abstrakte Gebilde verstanden werden, die aus der Sicht des PEARL-Programms als Quelle oder Senke für eine Datenübertragung erscheinen; die systemdefinierten Datenstationen sind dem Prozeßrechensystem a priori "bekannt", zumeist jedoch mit recht unanschaulichen Bezeichnungen, wie beispielsweise "SDR101" versehen). Als Beispiel ist in Bild 4 ein Ausschnitt des Systemteils des PEARL-Programms für die Automatisierung der Paketverteilanlage dargestellt.
Im Problemteil, der das eigentliche Automatisierungsprogramm enthält, werden zur Kommunikation zwischen Prozeßrechner und technischem Prozeß, d.h. für die Ein-, Ausgabe von Prozeßdaten, die im Systemteil angegebenen Benutzernamen (in Bild 4 die von einem Doppelpunkt gefolgten Namen) verwendet. Dadurch wird eine Lösung von dem anlagenspezifischen Teil des Automatisierungssystems erreicht. Der Vorteil dieses Konzepts liegt darin, daß bei einer Änderung der Prozeßperipherie und gleichbleibender

```
7  SYSTEM;
8
9  /*-------------------------------------------------------------------------*/
10
11  /*  STANDARDPERIPHERIE  */
12
13  DRUCKER : SDR101;
14  EINGABE : FSR1E1;
15  AUSGABE : FSR1A1;
16
17  /*-------------------------------------------------------------------------*/
18
19  /*  PROZESSPERIPHERIE  */
20
21  LESEORGAN         ; -> DEA(4)  *(1:8)   ;  /*    INTERFACE LESEORGAN     */
22  AZUSTANDSMELDER   ; -> DEA(4)  *(9:16)  ;  /*  ZUSTANDSMELDER.TEIL 1     */
23  BZUSTANDSMELDER   ; -> DEA(5)  *(1:13)  ;  /*  ZUSTANDSMELDER TEIL 2     */
24  TZUSTANDSMELDER   ; <- DEA(11) *1       ;  /*     *          TRENNSIGNAL */
25  RZUSTANDSMELDER   ; <- DEA(11) *2       ;  /*     *   RUECKSTELLSIGNAL    */
26  FREIGABEORGAN     ; <- DEA(11) *(3:4)   ;  /*  INTERFACE FREIGABEORGAN   */
27  LENKORGAN         ; <- DEA(11) *(5:11)  ;  /*    INTERFACE LENKORGAN     */
28  BAENDER(1:2)      ; <- DEA(11) *(12:13);   /*RUECKFUEHRBAENDER 1-5 & 6+7*/
29  ADRESSRESET       ; <- DEA(11) *14.     ;  /*                            */
30
```

Bild 4: Systemteil eines PEARL-Programms

Aufgabenstellung lediglich eine Anpassung des Systemteils an die geänderten Verhältnisse erforderlich ist, während der Problemteil des Programmsystems beibehalten werden kann. Dadurch war es beispielsweise möglich, einen Teil des Programmsystems für die Paketverteilanlage mit Hilfe eines einfachen Hardware-Simulationsgerätes (bei dem die Zustände der Lichtschranken durch Umlegen von Schaltern, die Ausgabe von Stellsignalen durch LEDs angezeigt wurden) statischen Tests zu unterziehen, lange bevor das Anlagenmodell fertiggestellt und damit die endgültige Anschlußbelegung festgelegt war, und danach durch leicht durchzuführende und überschaubare Änderungen des Systemteils diesen Programmteil voll zu übernehmen.
Um Prozeßdaten ein- bzw. auszugeben werden in PEARL die TAKE- bzw. SEND-Anweisungen benützt. Eine Anwendung dieser Anweisungen ist in Bild 5 enthalten, wobei auch der Zusammenhang mit den im Systemteil definierten Benutzernamen noch einmal klar ersichtlich wird.

```
                    :
371  SEND (NOT ZURUECKSETZEN) TO RZUSTANDSMELDER ;  /* RUECKSTELLSIGNAL
372                                                    ZURUECKNEHMEN    */
373  SEND TRENNEN TO ZUSTANDSMELDER ;  /*  ZWISCHENSPEICHER VOM EINGANGS-
374                                         SPEICHER TRENNEN  */
375
376  TAKE AVERTEILSTATIONSDATEN FROM AZUSTANDSMELDER ;
377  TAKE BVERTEILSTATIONSDATEN FROM BZUSTANDSMELDER ;
378
379  SEND (NOT TRENNEN) TO TZUSTANDSMELDER ;     /*  TRENNSIGNAL
380                                                  ZURUECKNEHMEN  */
381  SEND ZURUECKSETZEN TO RZUSTANDSMELDER ;  /*  EINGANGS- MIT ZWISCHEN-
382                                               SPEICHER VERBINDEN  */
                    :
```

Bild 5: Ein- bzw. Ausgabe von Prozeßdaten

In dem Beispiel von Bild 5 wird gezeigt, wie die Prozeßdaten "Zustandsdaten" der Paketverteilanlage, die ein Abbild des augenblicklichen Belegungszustandes der Anlage liefern, eingelesen werden. Die Prozeßdaten werden durch ein "Zustandsmeldegerät" zwischengespeichert, das durch einen zusätzlichen Eingangsspeicher auch die hardwaremäßige Erfassung der Zustandsdaten während des Einlesevorgangs ermöglicht.

Modularer Aufbau eines PEARL-Programmsystems

Das Grundprinzip des modularen Aufbaus eines PEARL-Programmsystems ist schon im vorangegangenen Abschnitt angesprochen worden, nämlich die Aufteilung in PEARL-Moduln und deren Aufteilung wiederum in System- und Problemteil. Der wesentliche Vorteil der PEARL-Moduln liegt darin, daß sie getrennt übersetzt (compiliert) werden können. PEARL kommt damit einem strukturierten Vorgehen nach dem Prinzip der schrittweisen Verfeinerung, wie es beim Entwurf Anwendung finden sollte, entgegen. Damit ist es möglich, funktionsmäßig abgrenzbare Teillösungen eines Entwurfs zur Programmierung in Moduln einzelnen Mitgliedern eines Projekt-Teams zu übergeben, die dann, wenn eine geeignete Testumgebung vorhanden ist, ihre Moduln unabhängig voneinander austesten können. Eine erforderliche Korrespondenz zwischen PEARL-Moduln wird dabei innerhalb eines Programmsystems über sogenannte globalen Größen hergestellt, die in allen Moduln benutzbar sind. Durch diese Eigenschaft von PEARL wird ein stufenweiser Ausbau bzw. die Erweiterung eines schon vorhandenen Automatisierungssystems wesentlich unterstützt. Der Grundgedanke der Modularität oder - anders benannt - der Blockstruktur läßt sich auch im inneren Aufbau eines PEARL-Moduls erkennen. Denn der Problemteil des Moduls wird seinerseits auch aus Blöcken gebildet, den Tasks, Prozeduren und den sogenannten BEGIN-END-Blöcken, wobei letztere sowohl in Tasks als auch in Prozeduren auftauchen und in sich selbst verschachtelt sein können. PEARL eignet sich daher besonders für die Implementation von Entwürfen, die mit top-down-orientierten Entwurfswerkzeugen erstellt wurden. Im konkreten Fall der Paketverteilanlage erfolgte der Entwurf des Automatisierungssystems rechnergestützt mit Hilfe des Software-Werkzeugs EPOS (Entwurfsunterstützendes Prozeß-orientiertes Spezifikationssystem).

Anwenderfreundliche Handhabbarkeit von PEARL, Zuverlässigkeit und Dokumentationswert der in PEARL erstellten Programme

PEARL verfügt über eine für den Anwender leicht überschaubare Menge von Sprachelementen, die ausreichend sind für alle Problemformulierungen der Prozeßautomatisierung. Durch das Sprachniveau von PEARL wird eine Abstraktion des verwendeten Prozeßrechners zu einer problembezogenen Maschine vollzogen, die vor allem die Mensch-Maschine-Schnittstelle vereinfacht. Als Folge der Trennung von anlagen- und problemspezifischem Teil ist zudem ein hohes Maß an Portabilität der PEARL-Programme gewährleistet, was

sich vor allem bei Projekten mit gleichgelagerter Aufgabenstellung und verschiedenen Rechnern bezahlt macht.
Die Summe all dieser Eigenschaften, sowie der hohe Dokumentationswert der PEARL-Programme (s. die angeführten Beispiele des Programmsystems zur Automatisierung einer Paketverteilanlage), ihre hohe Zuverlässigkeit - ein großer Teil möglicher Programmierfehler werden schon beim Compiliervorgang entdeckt, so daß die Wahrscheinlichkeit für noch verbleibende Fehler gering gehalten wird - machen diese Programmiersprache in der Hand eines Automatisierungsingenieurs zu einem Werkzeug, dem eine Assemblersprache nichts Gleichwertiges entgegenzusetzen vermag.

Literaturverzeichnis:

[1] Werum, H. und Windauer, H.: PEARL; Vieweg-Verlag, Braunschweig, 1978

[2] Kappatsch, A., Rieder, P. und Mittendorf, H.: PEARL; Oldenbourg-Verlag, München, 1979

[3] Lauber, R.: Prozeßautomatisierung I; Springer-Verlag, Berlin, 1976

[4] Lauber, R.: Möglichkeiten und Grenzen höherer Prozeßrechner-Programmiersprachen; Neue Technik Nr. 5 (1980), S. 29 - 34

[5] Eggert, H.: Praktischer Einsatz von PEARL am Beispiel der Brauereiautomatisierung. Referat auf dem "Aussprachetag Prozeßrechner", Dortmund, 27./28.3.1979

[6] Vergleich verschiedener Spezifikationsverfahren am Beispiel einer Paketverteilanlage (Herausgeber G. Hommel), Kernforschungszentrum Karlsruhe, Bericht KFK-PDV 186 (1980)

[7] Gottschalk, W.: Paketverteilanlage, Fallbeispiel des PDV-Arbeitskreises "Systematische Entwicklung von PDV-Systemen"

[8] Biewald, J. und Joho, E.: Was ist EPOS? Eigenschaften, Aufbau und Anwendung des Spezifikations- und Entwurfssystems EPOS: