PEARL - Anwendungen bei der ESG

M. Fenzl, F. K. Nonhoff, W. Hirschel, München

Zusammenfassung

Es werden drei Projekte vorgestellt, die mit ATM-PEARL realisiert wurden. Die dabei in bezug auf PEARL gesammelten Erfahrungen werden diskutiert.

Summary

Three applications implemented in ATM-PEARL are presented. The common PEARL specific aspects are discussed.

1. Einleitung

Die PEARL-Aktivitäten der Firma ESG reichen ca. 10 Jahre zurück. Die Erfahrungen im Projekteinsatz von PEARL (seit 79) beruhen vorwiegend auf realzeitorientierten, nationalen militärischen Systemen.

Es werden drei Projekte mit unterschiedlichem Systemcharakter, Umfang und Entwicklungsrandbedingungen vorgestellt. Bei allen Anwendungen kommen ATM Rechner und ATM-PEARL zum Einsatz. Die beim Entwickeln der PEARL-Programmsysteme gewonnenen Erfahrungen werden diskutiert.

2. Übersicht über die Projekte JADE, HFLA, ADLER

2.1 Projekt JADE

Im Bereich der Deutschen Bucht und der angrenzenden Wasserstraßen entsteht derzeit das "Verkehrssicherungssystem Deutsche Bucht", das mit seinen technischen Komponenten wie Synthetikradar, Präzisionspeiler und Datenverarbeitung die Sicherheit des Schiffsverkehrs und die Verkehrssteuerung verbessern wird.
Innerhalb dieses Vorhabens wurde das Programmsystem für die Komponente "Schiffsdatenverarbeitung mit Datenverbund" realisiert, das folgende Hauptaufgaben umfaßt:

- Speicherung, Organisation und Darstellung von Daten für die Verkehrsüberwachung
- Dokumentation des Verkehrssystems über einen längeren Zeitraum
- Automatischer Austausch von Schiffsdaten zwischen den Schiffsdatenrechnern der Revierzentralen Wilhelmshaven, Bremerhaven, Cuxhaven und Brunsbüttel über DATEX-L.

Das System ist als Doppelrechnersystem für hot-stand-by-Betrieb mit teilweise umschaltbarer Peripherie konzipiert und hat folgende Hardware-Konfiguration (B i l d 1) :

- 2 ATM 80-30 mit 384 KByte, gekoppelt über Selektorkanal
- 2 Plattenspeicher (10 MByte)
- 2 Protokollfernschreiber
- 4 Datensichtgeräte
- 2 graphische Sichtgeräte
- 2 Magnetbandsysteme mit Doppelzugriff.

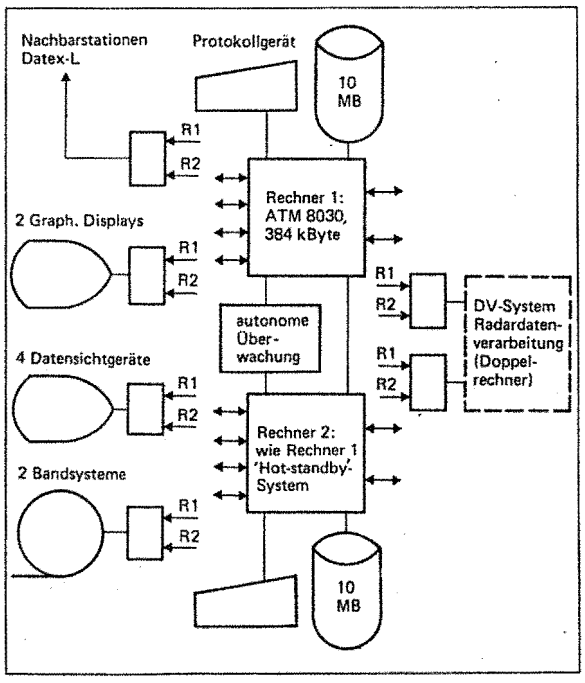


Bild 1. Hardware-Konfiguration JADE

Softwarekonzept
Die Konsistenz der Datenbasis im Doppelrechnersystem wird dadurch gewährleistet, daß alle, die Datenbasis und systemglobale Zustände verändernden Aufträge in Ringpuffern in beiden Rechnern abgelegt werden. Die Bearbeitung dieser Aufträge erfolgt sequentiell.

Das Anwendersystem ist in vier Subsysteme aufgeteilt:
- Ablaufsteuerung und Verwaltung der Schiffsdatenbestände
- Dialogsubsystem
- Datenverbundsteuerung
- Verkehrsdokumentation

Die Kommunikation der Subsysteme erfolgt über Botschaften und Ringpuffer. Jedes Subsystem enthält einen Monitor, der ihm zugeordnete Verarbeitungstasks koordiniert.

2.2 Projekt HFLA
Im Rahmen der Entwicklung HFlaAFüSys (Heeres-Flugabwehr-Aufklärungs- und Gefechtsführungssystem, kurz HFla) erstellt Firma ESG die Software für den Gerätesatz "Luftlagedatenverarbeitung" (LLDV) mit folgenden Hauptfunktionen:
- Erstellung einer Sensorluftlage aus den Informationen der lokalen Radaranlage
- Einarbeiten der eigenen Luftlage und der über Datenfunk empfangenen Luftlageinformation benachbarter Information in eine Gesamtluftlage (Multi-Radar-Tracking)
- Eliminierung von Festzielen
- Ortung von Störern

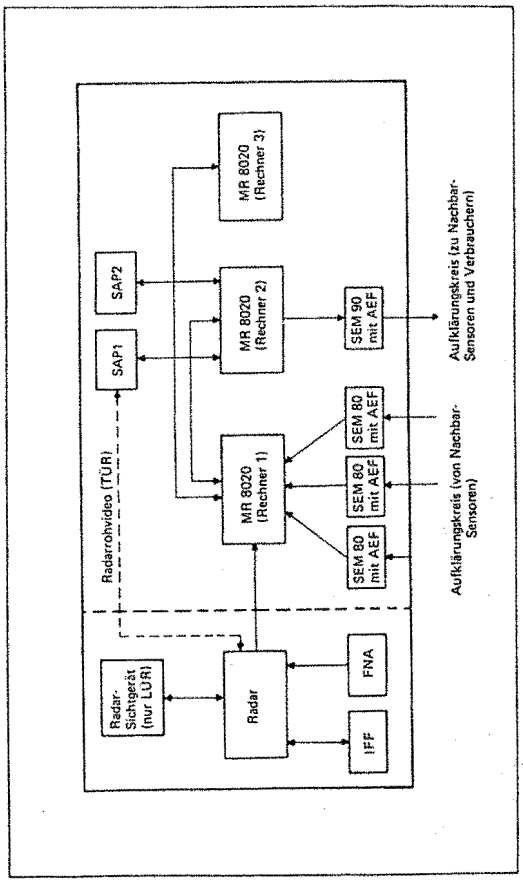
Die Track-Algorithmen und die Bedienung der Peripherie, nämlich:
- zwei autonome Systemarbeitsplätze (SAP)
- vier Datenfunkgeräte SEM 80/90 mit Anpaßeinheit Funk (AEF)
- Radarschnittstelle

induzieren eine hohe Rechnerlast. Drei MR8020, die über den Speicher-Bus gekoppelt sind, erbringen die geforderte Rechenleistung. Virtuell verfügen die Rechner über einen Hauptspeicher von 256 KByte. Der gemeinsame Speicherblock von 32 KByte liegt physikalisch im Rechner 1 (B i l d 2).

Softwarekonzept
Die logische Kopplung der drei Rechner erfolgt über einen Satz von Ringpuffern, die im gemeinsamen Speicherbereich angelegt sind. Da pro Ring nur ein Leser und ein Schreiber existiert, können die Zugriffe mit minimalem Aufwand implizit über Schreib- und Lesezeiger synchronisiert werden. Die Ablaufsteuerung erfolgt in den drei Rechnern einheitlich über Monitortasks, die die relevanten Ringpuffer nach vorgegebenen Prioritäten zyklisch abarbeiten und die zugehörigen Verarbeitungsprogramme anstoßen. Die Synchronisierung zwischen Monitor und Verarbeitungsprogrammen erfolgt über Semaphore.

2.3 Projekt ADLER
ADLER (Artillerie-Daten-Lage-Einsatz-Rechnerverbund) ist das Führungssystem für die Artillerie. Die Hauptaufgaben sind dabei:
- Führung des Feuerkampfes
- Planung und Leitung des Feuerkampfes
- Befehlsgebung
- Artilleristische Aufklärung
- Verbindung innerhalb ADLER und zu anderen Verbänden

Das Ziel, den Einsatz der Artillerie effizienter zu gestalten, wird erreicht durch Einführung von Rechnerunterstützung bei der Lösung von Teilaufgaben und Beschleunigung des Informationsaustausches über Datenfunk in einem gemäß den militärischen Führungsebenen gegliederten, hierarchischen Netz von Funkkreisen.



Hardware-Konfiguration Luftlagedatenverarbeitung

In ADLER gibt es Dienstposten mit unterschiedlichen funktionellen Anforderungen. Die für die verschiedenen Varianten benötigte Geräteausstattung wird aus logistischen Gründen aus einer einheitlichen ADLER-Zelle (B i l d 3) konfiguriert. Die Varianten bestehen aus einer, zwei oder vier Zellen. Die Geräteausstattung einer ADLER-Zelle ist wie folgt:

- Rechner MR8020 1 MByte
- 1 oder 2 Funkgeräte
- 1 oder 2 Bildschirme mit Tastatur
- Drucker
- Programmladegerät (Bubble-Speicher 1 MByte) Plattenspeicher (40 - 70 MByte; nur in Operationszentrale)
- Schnittstellen für Gefechtsstandskommunikation

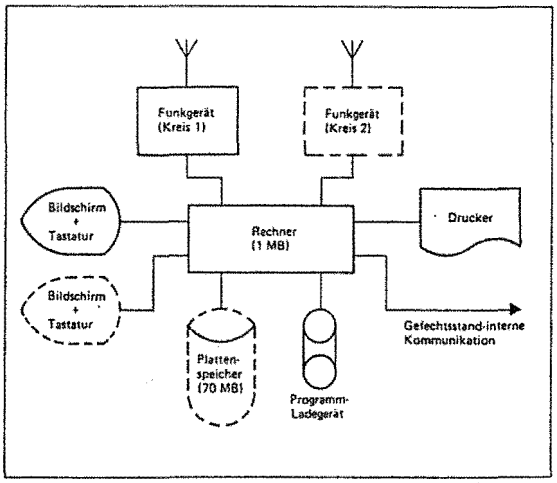


Bild 3. Konfiguration einer ADLER-Zelle

Software-Konzept

Neben Funktionsumfang und Randbedingungen bezüglich Hardware und Systemsoftware beeinflußten folgende Punkte die Wahl des Software-Konzepts wesentlich:

- N-Teilnehmersystem
  Jeder ADLER-Teilnehmer sollte (prinzipiell) Zugriff auf das gesamte Anwendungsspektrum haben, mit funktionsabhängigen Anwendungsprofilen.
- Leichte Konfigurierbarkeit der Software im Hinblick auf den Ablauf auf unterschiedlichen Hardware-Konfigurationen und verteilte Verarbeitung.
- Erweiterbarkeit
  Die Realisierung in mehreren Ausbaustufen erfordert ein die Modularisierung förderndes, von den Anwenderfunktionen unabhängiges Softwarekonzept.
- Die lange Lebensdauer militärischer Systeme stellt hohe Anforderungen in bezug auf Pflegbarkeit, Wartbarkeit und Zuverlässigkeit des Softwareprodukts.
- Die Anforderungen der Nutzer an das Realzeitverhalten erfordern eine sorgfältige Abbildung, insbesondere der systemnahen Software-Anteile, auf die Gegebenheiten der Hardware und Vorgaben für die Anwendung des PEARL-Sprachumfangs in der Implementierung.

Die Ablaufsteuerung des Mehrbenutzersystems erfolgt aus Effizienzgründen über Tabellen, die von verteilten Abwicklern abgearbeitet werden. Den Benutzern werden segmentierte Arbeitsspeicherbereiche zugeordnet (realisiert als PEARL-Datenmodul), in denen sie sich voneinander unabhängig entwickeln können. Außerdem wurde den Entwicklern ein Stack-Mechanismus, der Parameterübergabe und Aufrufschachtelung erleichtert, sowie eine Pufferverwaltung und Dateiverwaltung zur Verfügung gestellt.

Die Intertaskkommunikation wird über Botschaften abgewickelt. Das Konzept unterstützt asynchronen und synchronen Nachrichtenaustausch (Auftrag ohne, bzw. mit warten auf Quittung). Es gibt kurze Botschaften mit maximal 5 Byte Nutzinformation für den Empfänger sowie variabel lange Extensionen, die im Stack-Bereich der Arbeitsspeicher oder in Puffern abgelegt werden. Als Hilfsmittel für Integration und Test ist ein Task-Trace integriert.

Im Hinblick auf Modularität, Erweiterbarkeit und Flexibilität werden im Programmsystem Tasks, Benutzer und Geräte auf logische Bezeichner abgebildet. Für die Implementierung wurden streng formalisierte PEARL-Rahmenprogramme vom Design-Team festgelegt, die alle konzeptionell notwendigen Datenstrukturen und Zugriffsfunktionen auf Betriebsmittel enthielten.

Nach anfänglichen Widerständen bei den Implementierern (sie fühlten sich in ihrem schöpferischen Freiraum eingeschränkt) zeigten sich große Vorteile in bezug auf Implementierungsgeschwindigkeit und bei der Softwareintegration.

3. Programmkenngrößen

In T a b e l l e 1 sind einige Kenngrößen der Programmsysteme zusammengestellt. Bemerkenswert ist der hohe PEARL-Anteil in allen drei Anwendungen.

Kenngrößen \ Projekt	JADE	HFLA	ADLER
PEARL-Tasks	69	36	100
Anwendercode [KB]	500	350	1500
Betriebssystem [KB]	80	3 x 60	80
Tabellen	50	-	1600
Interrupts/sec.	50 - 100	200	50 - 100
PEARL-Anteil	93 %	95 %	98 %

Tabelle 1. Kenngrößen der Programmsysteme

4. Schalenmodell

Die Software wurde in hierarchischen Schalen strukturiert; diese Vorgehensweise bietet, ähnlich wie beim OSI-Modell der Kommunikation, Vorteile in bezug auf die Zuordnung von Subsystemen, Funktionskomplexen und Dienstleistungen innerhalb eines Softwaresystems und die Kommunikation der einzelnen Komponenten untereinander.
Bei der Realisierung von einheitlichen Daten- und Aufrufschnittstellen zwischen den Schichten und Programmen bietet PEARL sehr gute sprachliche Voraussetzungen. Blockstruktur und Modulkonzept unterstützen einen bei der Modularisierung und bei der Festlegung von Sichtbarkeitsgrenzen.
Die folgende T a b e l l e 2 gibt einen Oberblick über die Zuordnung von Funktionen zu Softwareschichten.

Projekt / Schicht	JADE	HFLA	ADLER
ANWENDER-SW (Hauptaufgaben)	Schiffsdatenverwaltung Meldungsverarbeitung Aufzeichnung/ Replay Verbundsteuerung	Sensorlufttlage Gesamtlufttlage Störerortung Falschziele/ Clutter	Feuerkampf Befehlsgebung Meldungsverarbeitung Aufklärung Lagebeurteilung Verbundsteuerung
PROZESS-SOFTWARE			
Ablaufsteuerung	x	x	x
Systeminitialis.	x	x	x
Systemüberwachung	x	x	x
Kommunikation	x	x	x
Datenhaltung	x		x
Dialog (MMI)	x		x
SYSTEMNAHE ANWENDER-SW			
Rechnerkopplung	x		x
Botschaftsdienst	x		x
Pufferverwaltung	x	x	x
Zugriffsroutinen	x	x	x

Tabelle 2. Zuordnung von Funktionen im Schalenmodell

5. Entwicklungsumgebung

Für die Softwareentwicklung steht eine umfangreiche DV-Konfiguration zur Verfügung, die über ein lokales Netz auf Ethernet-Basis (Bandbreite 10 MBaud) miteinander gekoppelt ist. Als Host-Rechner dienen Siemens-Rechner der Serie 75xx, die unter BS2000 laufen sowie VAX 750 mit den Betriebssystemen UNIX und VMS. Neben zahlreichen Netzterminals und Arbeitsplatzrechnern sind eine Reihe von projektspezifischen Rechnern angeschlossen.
B i l d 4 gibt am Beispiel ADLER einen Oberblick über die Zuordnung einzelner Entwicklungstätigkeiten auf die Rechner.
Die Entwicklung erfolgt unter AMDES, in dem UNIX-Host-Systeme mit ATM-Entwicklungsrechnern gekoppelt sind, die unter AMBOS laufen. Dies erlaubt eine komfortable Entwicklung unter UNIX. Die AMBOS-Maschinen dienen als Compile-Prozessoren sowie zum Generieren und Testen der PEARL-Programmsysteme.

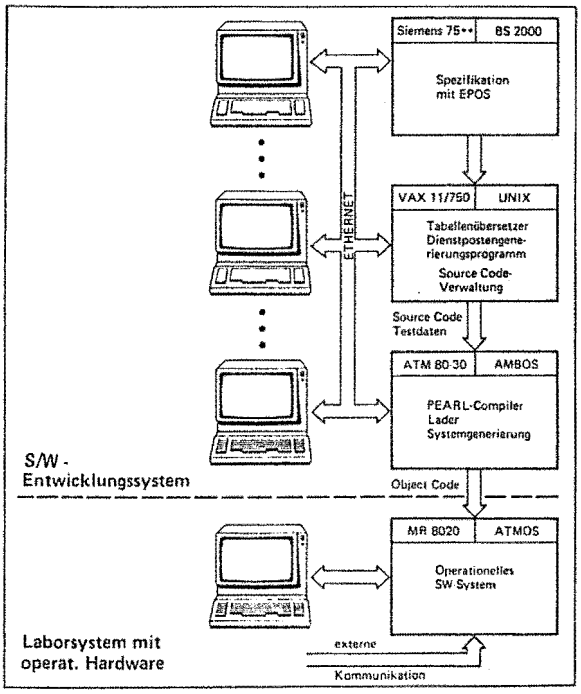


Bild 4. ADLER-Software-Entwicklungssystem

6. Projektübergreifende PEARL-Erfahrungen

Zunächst einige Bemerkungen zu Schwachpunkten:
Die meisten Mängel sind implementierungsabhängig, bzw. auf Hardware-Eigenschaften zurückzuführen.

- Der PEARL-Compiler ist nicht reentrant. Dies führt zu Wartezeiten in der Implementierungsphase.
- Größere PEARL-Module können häufig erst nach iterativer Anpassung von Compiler-Startparametern übersetzt werden, weil gewisse interne Listen linear abhängig sind.
- Das Laufzeitpaket ist zwar gut modularisiert, verlängert jedoch die Programmlaufzeiten.
- Manche Sprachkonstrukte wie GET, PUT, CAT, OPEN erzeugen viel Code.
- Es fehlt ein PEARL-Binder. Die globalen Querbezüge werden in Modul-Tasks abgelegt, die eine Prozeßnummer belegen und in Hauptspeichersystemen statisch Speicherplatz beanspruchen.
- Sprachliche Mängel sind in der Ausprägung der CASE-Anweisung und dem Fehlen von variablen RECORDS zu sehen.
- Für viele Prozessanwendungen wäre es wünschenswert, bei Task-Anweisungen Parameter mitgeben zu können.

Positive Erfahrungen mit PEARL:

- Sprachumfang

ATM-PEARL weist gegenüber Full PEARL gemäß DIN 66253-F nur geringe Einschränkungen auf, wie z. B.:

- o Keine expliziten Operatordeklarationen (Standardoperatoren)
- o Keine expliziten Interface-Deklarationen (Standardinterfaces)
- o Keine BOLT-Variable (nur Semaphore)
- o Keine Semaphor-Listen in Semaphoranweisungen
- o Keine dynamische Änderung von Taskprioritäten

Die Einschränkungen wirkten sich bei unseren Anwendungen nicht erschwerend aus und wurden ggf. auf Anwenderebene umgangen.

Bei größeren Programmsystemen ist es neben anderen Maßnahmen sinnvoll, den Entwicklern in der Anwendung von Sprachkonstrukten Vorgaben zu machen, z. B. um einen einheitlichen Programmierstil zu erzwingen, Integrationsproblemen vorzubeugen und Implementierungsschwächen (Code-aufwendige Sprachkonstrukte etc.) zu umgehen.
Folgende, über Basic PEARL hinausgehende Elemente erwiesen sich dabei als besonders nützlich:

- Typspezifikation (hauptsächlich mit Strukturattribut) trägt in Verbindung mit INCLUDE-Pragma sehr zur Programmiersicherheit bei
- Beliebige Strukturschachtelung erlaubt optimale Darstellung von Daten nach funktionellen Gesichtspunkten
- Identitätsspezifikation (SPC...IDENT(...)) zur Zuordnung von weiteren Zugriffsfunktionen bzw. Bezeichnern auf existierende Objekte
- Alle Deklarationsarten (außer TASK's und SEMA's) auf beliebiger Blocktiefe
- Keine spezielle Deklarationsreihenfolge
- Referenzobjekte
- Arrays in Strukturen
- Beliebige Arrayindexgrenzen (dynamisch)
- Zuweisung von Arrays und Strukturen als ganzes
- Beliebige Ausschnitte (SLICES) aus ARRAY's, Zeichen- und Bitketten
- Array und Strukturdisplays

Weitere positive Aspekte sind:

- Gute Unterstützung durch ATM beim Lösen von Problemfällen
- Interesse des Herstellers an Weiterentwicklungen, z. B. Code-Datentrennung für Anwenderprogramme und Einführung eines Optimierungslaufes im PEARL-Compiler
- Mehrstufiges INCLUDE-Pragma
- Große Anzahl von Fehlermeldungen durch Compiler und LZO
- Quellbezogenes PEARL-Testsystem (QPTS) bietet gute Hilfestellung beim Test von Anwendersystemen
- Erlernbarkeit/Akzeptanz
  PEARL ist leicht erlernbar (Aufwand 1 bis 4 Wochen). Die Erstellung optimierter Programme erfordert ca. 6 Monate Erfahrung. Die Akzeptanz der Sprache im Entwickler-Team ist sehr gut.
- Die Anwendung von PEARL fördert die Produktion von gut strukturierten, transparenten Programmsystemen. Dies beeinflußt die Kosten für Wartung und Pflege günstig.

Schlußbemerkung:
Zusammenfassend kann man sagen, daß sich PEARL als Implementierungssprache für Prozeßanwendungen in unserem Hause gut bewährt hat.
In das zur erfolgreichen Projektabwicklung nötige Software-Engineering läßt sich PEARL gut einbetten.

Anschrift des Autors:

Dr. M. Fenzl
Elektronik-System-Gesellschaft mbH
Vogelweideplatz 9
8000 München 80