Nachdruck aus dem Do-Post

Höhere Programmiersprachen im Programmentwicklungsprozeß

P. Elzer

Höhere Programmiersprachen sind nicht das einzige Mittel zur Lösung von Softwareproblemen. Da sie aber Auswirkungen auf mehrere Stufen des Softwareentwicklungsprozesses haben und durch ihren "stilbildenden" Charakter das Gesamtsystem stark beeinflussen, sollen sie hier stellvertretend für andere Programmierwerkzeuge betrachtet werden. Da es außerdem sehr viele höhere Programmiersprachen gibt, soll die Auswahl auf die beiden für systemtechnische Entwicklungen bedeutsamsten, nämlich "PEARL" und "ADA" beschränkt werden.

Die derzeit existierenden Softwareprobleme kann man grob einteilen in Massenprobleme, die aus einer übergroßen Nachfrage resultieren, der kein entsprechendes Produktionspotential gegenübersteht, und Qualitätsprobleme, die durch die wachsende Komplexität der Systeme entstehen. Abhilfemaßnahmen haben also zwei Zielen zu dienen:

- Erhöhung der Produktivität des Entwicklers
- Erhöhung der Arbeitsqualität.

Hierbei muß gleichzeitig auf zwei Ebenen vorgegangen werden:

- durch Managementmaßnahmen, wie etwa die geeignete Zergliederung des Gesamtproblems, Teamorganisation oder Entwurfs- und Entwicklungsrichtlinien;
- durch technologische Maßnahmen, die hauptsächlich in der Verfügbarmachung geeigneter Programmierwerkzeuge, einer "Software-" oder "Programmierumgebung" bestehen. Diese Klasse von Maßnahmen ist vergleichbar mit den klassischen Investitionen in Maschinen und Ausrüstung.

Diese Maßnahmen sollten folgende Hauptwirkungen haben:

- Abbau der Kommunikationslücke zwischen Anwendungsexperten und Programmentwicklern;
- Verringerung der für den Entwickler sichtbaren Komplexität des Systems;
- Entlastung von Routineaufgaben und Detailentscheidungen, die nicht zur Problemlösung beitragen, um geistige Kapazität für den eigentlich ingenieurmäßigen kreativen Anteil der Arbeit freizusetzen.

Zu einer solchen Programmierumgebung gehören etwa:

- Entwurfsverfahren
- Höhere Programmiersprachen und ihre Übersetzer
- Test- und Integrationshilfen
- Projektdatenbasis und Dokumentationsverfahren.

ENTSTEHUNG VON PEARL

PEARL (=Process and Experiment Automation Realtime Language) wurde in der Bundesrepublik in Zusammenarbeit von Rechnerherstellern, System- und Softwarehäusern, anwendender Industrie und Forschungsinstituten entwickelt. Sowohl die Entwicklung der Sprache selbst als auch die von Übersetzern dazu wurden vom BMBW (Bundesministerium für Bildung und Wissenschaft) und vom BMFT (Bundesministerium für Forschung und Technologie) gefördert. Der erste vollständige Sprachentwurf erschien 1973. 1978 erschien der Normentwurf DIN 66253, Teil 1, Basic PEARL. Dieser legt den Teil der Sprache PEARL fest,

der praktisch allen zur Zeit verfügbaren Implementationen gemeinsam ist. Der Gesamtumfang von PEARL wurde Anfang 1981 als DIN Normentwurf veröffentlicht.
Im Dezember 1979 wurde eine Organisation zur weiteren Betreuung von PEARL gegründet, der PEARL-Verein e.V. Seine Geschäftsstelle befindet sich im VDI-Haus in Düsseldorf.
PEARL-Programmiersysteme werden in der Bundesrepublik unter anderem von ATM, BBC, DEC, Dornier, Krupp-Atlas, mbp und Siemens angeboten.

EINSATZBEREICH VON PEARL

PEARL ist bewußt als eine Programmiersprache für den anwendenden Ingenieur konzipiert. Sie erlaubt dem Anwendungsfachmann, einen Großteil seiner Probleme selbst für den Computer aufzubereiten. Sie stützt sich auf existierende Betriebssysteme und andere Softwarehilfen und erlaubt so die Benutzung vom Rechnerhersteller standardmäßig gelieferter Programmierumgebungen. Es ist jedoch auch möglich, in sich geschlossene, ganz auf PEARL ausgerichtete Programmierumgebungen zu entwickeln. Dies führt sogar zu besonders guten Einsatzresultaten, wie später am Beispiel des bei Dornier entwickelten PEARL-Programmiersystems, gezeigt werden soll.
PEARL unterstützt vom Ansatz her also vorwiegend den Abbau der Kommunikationslücke zwischen Anwendungsspezialist und Programmentwickler und erlaubt eine Verringerung der Komplexität der Problembeschreibung auf relativ hohem, d.h. anwendungsorientiertem Niveau.

WESENTLICHE ELEMENTE VON PEARL

Die Beschreibung der Hardwarekonfiguration im sogenannten "Systemteil" beruht auf dem Konzept allgemeiner Netzwerke. Eine Programmzeile des Systemteils beschreibt ein Teilstück des Weges, den Information im Gesamtsystem nehmen kann, nämlich jeweils eine direkte Verbindung zwischen einer informationserzeugenden und der zugehörigen informationsverbrauchenden Komponente des Hardwaresystems. Vom Übersetzer werden aus diesen Teilstücken die gesamten Datenwege zusammengesetzt und auf ihre Realisierbarkeit geprüft.

Die Realzeitelemente stellen die herausragende Stärke von PEARL dar.

Sie erlauben:
- die Festlegung parallel ablauffähiger Programmstücke (Tasks),
- die Beeinflussung ihres Ablaufes durch: Start, Beendigung, Anhalten, Wiederanlauf,
- ihre Kopplung an externe Zeitbedingung oder Spontanereignisse (Interrupts),
- die gegenseitige Synchronisation solcher parallel ablaufender Programmstücke.

Die Ein-/Ausgabe von PEARL ist auf die Bedürfnisse der Automatisierungstechnik ausgerichtet. Ein allgemeines Modell erlaubt in Ergänzung zur statischen Beschreibung von Datenwegen im Systemteil ihre funktionelle Beschreibung durch Datenstationen ("Dations") und dazwischengeschaltete Interfaces. Die eigentlichen Ein-/Ausgabeanweisungen von PEARL sind anwendungsorientiert und unterteilen sich in folgende Klassen:
- zeichenorientiert, wie etwa bei FORTRAN
- wertorientiert, d.h. Transfer von Daten ohne Wandlung

Der algorithmische Teil von PEARL entspricht weitgehend den Möglichkeiten klassischer Programmiersprachen. Er enthält Elemente zur Verarbeitung von Daten der Arten (Typen): ganzzahlig, Gleitkomma, Bitketten, Zeichenketten. Diese können wiederum in den Modi "Konstante" oder "Variable" vorkommen und zu "Feldern" und "Strukturen" gruppiert werden. Über die Möglichkeiten der meisten klassischen Programmiersprachen hinaus gehen die Datentypen "Uhrzeit" und "Zeitdauer" und die Möglichkeiten zur Definition eigener Typen durch den Entwickler.
Die modulare Struktur von PEARL-Programmen bietet folgende Vorteile:
- Erleichterung von Test und Integration dadurch, daß nur einzelne Teile des Gesamtprogramms neu übersetzt werden müssen.
- Die Trennung von "Systemteil" und eigentlichem Programm ("Problemteil") erleichtert die Wiederverwendung von Programmen, falls sich nur die Hardwarekonfiguration ändert.

DORNIER-PEARL

Bei Dornier System wurde im Auftrag des BMVg ein PEARL-Programmiersystem entwickelt, das speziell auf die Anforderungen rechnergestützter Systeme im Bereich der Luft- und Raumfahrt und der Wehrtechnik zugeschnitten ist. Es wurde entsprechend folgenden Anforderungen entworfen:

- der erzeugte Code muß möglichst effizient sein,
- der Test integrierter Systeme muß unterstützt und erleichtert werden
- eine Programmbibliothek zur Entwurfsunterstützung muß aufgebaut werden können
- Systeme mit verteilten Rechnern müssen in PEARL programmiert werden können
- das Übersetzungssystem muß weitgehend unabhängig sowohl von der jeweiligen Übersetzungsmaschine als auch vom Zielrechner sein.

Das Dornier-PEARL Übersetzungssystem steht bisher für die Zielmaschinen Dornier-MUDAS 432 und AEG-Telefunken 8020 zur Verfügung. Eine Version für den Mikroprozessor INTEL 8086 ist in Vorbereitung. Es ist ablauffähig auf den Übersetzungsrechnern PDP-11/70 und AEG 8020.

ENTSTEHUNG VON ADA

"ADA" ist eine vom US-Verteidigungsministerium finanzierte Entwicklung. Der Name der Sprache ist nicht wie üblich ein Akronym, sondern der Vorname der "ersten Programmiererin der Welt": Augusta Ada Byron (1816-1862), später Lady Lovelace, die zeitweilig mit Charles Babbage, dem Erfinder der ersten programmgesteuerten Rechenmaschine der Welt, zusammenarbeitete.

Das Sprachentwicklungsprojekt begann 1975 unter dem Namen "DOD 1". In einer Reihe von Dokumenten wurden die technischen Anforderungen an die Sprache niedergelegt. Die Version "Ironman" vom Januar 1977 war die Grundlage für eine Ausschreibung der eigentlichen Sprachentwicklung. Zunächst wurden vier Sprachentwürfe im Wettbewerb entwickelt. Nach eingehenden Bewertungen auf internationaler Basis wurde im Mai 1979 der Entwurf von CII-Honeywell Bull ausgewählt und in den "Sigplan-Notices" der Association of Computing Machinery (ACM) veröffentlicht. Nach eingehenden Tests wurde die Sprache im Juli 1980 festgeschrieben. Testimplementationen sind in Arbeit. Erste Produktionscompiler werden für 1983 bis 1984 erwartet.

EINSATZBEREICH VON ADA

Ada wurde vorwiegend für die Programmierung großer militärischer Systeme entwickelt. Es wurde Wert darauf gelegt, daß möglichst wenig Unterstützungssoftware seitens des Rechnerherstellers zur Verfügung gestellt werden muß und daß möglichst viel davon in Ada selbst geschrieben werden kann. Großer Wert wurde auf Prüfarbeit und Zuverlässigkeit der entstehenden Programme gelegt. Übersetzbarkeit auf dem Zielrechner selbst war kein wesentliches Entwurfskriterium. Es wird vielmehr eine Entwicklungsumgebung angestrebt, in der der im Anwendungssystem selbst enthaltene Zielrechner ("embedded computer") von einem großen Entwicklungsrechner aus programmiert wird.

WESENTLICHES ELEMENT VON ADA

Entsprechend den Anforderungen wurde auch das Schwergewicht bei der Entwicklung auf sicherheitsrelevante Sprachelemente gelegt. Das Konzept der Typbildung und Modularisierung entspricht neuesten Forschungsergebnissen. Im Gegensatz zu den meisten anderen Programmiersprachen umfaßt die Definition eines "Typs" in Ada auch die darauf anwendbaren Operationen und den Wertebereich, der angenommen werden kann. Die Definition eines Moduls erlaubt die Spezifikation der Teile, die von außen beeinflußt werden können und solcher, die geschützt werden sollen. Damit sind weitgehende statische Prüfungen möglich, aber meist auch ein entsprechender Schreibaufwand verbunden.
Der Fehlerbehandlungsmechanismus erlaubt gezielte Reaktionen auf Programmfehlverhalten wie z.B. Multiplikationsüberlauf, Division durch Null, Bereichsüberschreitungen, etc. Die Regeln, nach denen betroffene Programmteile abgeschaltet oder Fehler weitergereicht werden dürfen, sind genau festgelegt.

Der Synchronisationsmechanismus, der die Kommunikation zwischen den natürlich auch in Ada vorhandenen parallel ausführbaren Programmteilen (Tasks) ermöglicht, wurde auch unter dem Sicherheitsaspekt entworfen. Es werden hier im Gegensatz zu den sonst üblichen Modellen nicht Daten zwischen Tasks ausgetauscht, sondern vorher definierte Programmteilstücke gemeinsam ausgeführt.

DIE SOFTWAREUMGEBUNG VON ADA

Bereits zu Beginn der Ada-Entwicklung war klar, daß der Erfolg der Sprache von der Verfügbarkeit einer entsprechenden Softwareumgebung abhängt. Es wurden deshalb schon weitgehend parallel zur Entwicklung der technischen Anforderungen an die Sprache auch solche an die Softwareumgebung erarbeitet. Im Rahmen einer Beobachterfunktion für die Bundesregierung arbeitete der Autor an den ersten Dokumenten dieser Art mit ("Pebbleman" und Pebbleman revised"). In diesen Dokumenten werden sowohl Organisationsvorschläge für die Betreuung der Sprache gemacht als auch Forderungen an die Komponenten der Softwareumgebung aufgestellt. Sie soll u.a. Compiler, Codegeneratoren, intelligenten Editor, Datenbasis, Testhilfen und Managementhilfen umfassen. In der neuesten Version dieses Dokumentes ("Stoneman") werden auch Vorstellungen entwickelt, wie eine solche Softwareumgebung stufenweise, aufbauend auf einer Minimalausstattung, entwickelt werden kann.