Zur Darstellung und Implementierung von kommunikationsorientierten Prozessen

Author(s):
Mumprecht, Eduard

Publication Date:
1980

Permanent Link:

https://doi.org/10.3929/ethz-a-000205361 →

Diss. ETH Nr. 6534

Zur Darstellung und Implementierung von kommunikationsorientierten Prozessen

Abhandlung

zur Erlangung des Titels eines
Doktors der Technischen Wissenschaften

der
EIDGENÖSSISCHEN TECHNISCHEN HOCHSCHULE ZÜRICH

vorgelegt von

Eduard Mumprecht
Dipl. El.-Ing. ETH
geboren am 13. Juli 1948
von Zürich und Herzogenbuchsee BE

Angenommen auf Antrag von
Prof. Dr. N. Wirth, Referent
Prof. Dr. K. Bauknecht, Korreferent

Zürich 1980

Abstract

On Representation and Implementation Methods
for Communicating Processes

The problem of designing communication subsystems for distributed processing is addressed.

Various programming concepts, methods and tools are discussed with emphasis on their usefulness in producing telecommunications software.

A finite state automaton based module is then proposed in conjunction with existing high level programming systems to overcome certain problems of representing syntactical structures within a concurrent environment.

Implementation aspects of that module are explained; its use is shown by some practically applied examples on various levels of computer architecture.

8. ZUSAMMENFASSUNG

8.1. Aufgabenstellung

Das Zusammenarbeiten verschiedener Komponenten eines Computersystems erfordert Kommunikation. Mit zunehmender Tendenz in Richtung "verteilte Systeme" und wachsender Komplexität der zuweilen "intelligent" genannten Funktionseinheiten steigen auch die Anforderungen an die Kommunikationsmechanismen.

Die vorliegende Arbeit befasste sich mit den Darstellungs- und Programmiermethoden zur Implementierung kommunikationsorientierter Prozesse. Das Hauptziel war dabei, dass die der Kommunikation inhärenten Strukturen beim Entwurf und der Implementierung von Subsystemen nicht verschwinden, sondern auf allen Abstraktionsebenen erkennbar bleiben. Weil sich Kommunikationsprozesse nicht so einfach aus ihrer Umgebung herauslösen lassen, besteht so wenigstens die Möglichkeit, deren Fehlverhalten am laufenden System zu beobachten und allfällige Modifikationen problemnah einzubringen.

8.2. Schwerpunkte der Arbeit

Die Arbeit gliedert sich in drei Abschnitte:

1) Untersuchung von Konzepten zur Interprozesskommunikation

Die zentralen Begriffe "Prozess" und "Kommunikation" wurden erläutert und in Beziehung gesetzt zu den verschiedenen Betrachtungsweisen der Informationsübertragung. Dabei fand im speziellen die Datenübertragung zwischen Prozessen aus verschiedenen Adressräumen Beachtung. Anhand einfacher Uebertragungsverfahren (Protokolle) wurde die Problematik ihrer Darstellung und Implementierung gezeigt und die Möglichkeiten verschiedener Programmiermethoden und Modelle behandelt.

2) Herleitung eines implementationsfreundlichen Modells für einen kommunikationsorientierten Prozess.

Es wurde ein Modell für einen Prozess vorgeschlagen, das sich zur problemnahen Formulierung seines Kommunikationsverhaltens eignet. Das Modell basiert auf einem Mealy-Automaten und gestattet eine recht einfache und sichere Implementierung kommunikationsorientierter Prozesse. Die Anwendung des "Funnel" genannten Modells erscheint dabei als spezielle Programmiermethode.

3) Anwendung des Modells an praktischen Beispielen.

Das Konzept des Modells wurde einerseits als interpretierendes System durch Mikroprogrammierung in die Umgebung einer höheren Programmiersprache implementiert. So konnte darauf aufbauend eine konkrete Anwendung der "Funnel"-Methode auf dem Computersystem B-1700 gezeigt werden. Anderseits wurde die Anwendbarkeit der Methodik auf verschiedenen Stufen der Computerarchitektur diskutiert. Eine etwas erweiterte Form des Modells wurde dann als Konzept "Datenflussgraph - Kontrollflussgraph" für den Entwurf und die

Implementierung eines Kommunikationspfades im Grenzbereich Hardware/Software verwendet.

8.3. Resultate und Erkenntnisse

Das Monitor-Konzept von Hoare und Brinch-Hansen hat wesentlich zum Verständnis des Problems der "kooperierenden sequentiellen Prozesse" beigetragen. Die Anwendung des Konzepts in Elementen von realen Programmiersprachen befriedigt hingegen nicht alle Ansprüche bei der Realisierung grösserer, hierarchisch gegliederter Systeme. Ein Monitor erscheint als statisch gruppierte Datenstruktur mit zugehörigen Zugriffsprozeduren, deren Aktivierung einschlägigen Regeln unterworfen ist. Das Verwalten der Datenstruktur ist dabei Teilaufgabe der Zugriffsoperationen. Ein Monitor liesse sich nun in einer erweiterten Auffassung als Prozess bezeichnen, wenn die interne Datenverwaltung ein Eigenleben entwickelt. Und genau das trifft zu, wenn die Daten innerhalb des Monitors zwischen verschiedenen Adressräumen übertragen werden müssen. Ein Protokoll wäre demnach als erweiterter abstrakter Datentyp mit Eigenleben aufzufassen.

Als Prozess zur Datenübertragung muss ein Protokoll implementiert werden können. Dabei sind die Ebenen der Programmierung für die Kommunikationspartner voneinander getrennt und möglicherweise verschiedenartig. Gesamthaft lässt sich ein Protokoll als Syntax einer Sprache auffassen und liesse sich deshalb mit den dafür bekannten Methoden der Informatik behandeln. Bei Protokollen zeigen sich aber einige Schwierigkeiten, wenn sich "Parallelität" und "Hierarchie" nicht zu vertragen scheinen.

Das vorgeschlagene Automatenmodell "Funnel" mildert die Problematik der Darstellung und Programmierung von Protokollen gleich auf drei Arten:

1) Die Strukturen der Kommunikationsprozeduren müssen nicht invertiert werden; die Syntax ist unmittelbar (als Graph) Element des "Funnel"-Moduls.

2) "Funnel"-Module lassen sich als Prozeduren (Subroutinen) verschachteln.

3) "Funnel"-Module lassen als Coroutinen quasiparallele Abläufe zu (Prozess-Konzept).

Sowohl das Konzept "Funnel" als auch dessen Implementierung als interpretatives System hat sich als sehr brauchbar erwiesen, indem bei der Anwendung vor allem die dem Kommunikationsverhalten eines Prozesses eigenen Strukturen auch zur Laufzeit erhalten bleiben.

Die Anwendung des "Funnel"-Konzeptes als Programmiermethode unterscheidet sich von den durch die blockstrukturierten höheren Programmiersprachen geförderten Techniken folgendermassen: Die Entwicklung der "strukturierten Programmierung" hat sich hauptsächlich auf die Möglichkeiten zur Kontrollfluss - Steuerung konzentriert. Dabei sind die Regeln der Datendarstellung und -verwaltung stark an den Kontrollfluss gebunden worden.

Das erweiterte "Funnel"-Konzept hebt die Idee des Datenflusses hervor. Dabei wird letztlich der Programmkontrollfluss eines Systems durch den Datenfluss bestimmt. Die beiden geschilderten Programmiertechniken sollen sich aber ergänzen und nicht als Alternativen angesehen werden.

8.4. Offene Arbeitsrichtungen

Das Entwerfen von Protokollen mit den üblichen Hilfsmitteln zur Darstellung und Implementierung von gewöhnlichen Programmen birgt grössere Unsicherheitsfaktoren über die funktionelle Richtigkeit des Produktes in sich. Es wären deshalb formal strengere Darstellungen von Protokollen nützlich, um diese analytisch behandeln zu können.

Der Aufbau von Datennetzen erfordert genormte Schnittstellen und Verfahren. Wünschenswert wäre ein Baukastensystem von Protokoll - Elementen, die erstens zueinander passen und zweitens portabel implementierbar sind. Bis dahin wird noch einige Arbeit auf den Gebieten höherer Protokolle nötig sein, besonderes Gewicht kommt dabei der internationalen Standardisierung zu.

Auf dem Gebiete der Systemprogrammierung wäre ein Durchbruch im Sinne der "Steelman Requirements" (Ref. DoD.77) wünschbar.

* * *