Informationstechnik und Technische Informatik 35 (1993) 1 © R. Oldenbourg Verlag it+ti 1/93

Logische Grundlagen eines Triggerentwurfssystems

B. Reinwald und H. Wedekind, Universität Erlangen-Nürnberg

Dipl.-Inform. Berthold Reinwald arbeitet seit 1989 als wissenschaftlicher Mitarbeiter am Lehrstuhl für Datenbanksysteme der Universität Erlangen-Nürnberg. Hauptarbeitsgebiete: Aktive und verteilte Datenbanksysteme.

Prof. Dr. Hartmut Wedekind ist Professor für Informatik an der Universität Erlangen-Nürnberg.

„Die Erforschung der Logik bedeutet die Erforschung aller Gesetzmäßigkeit. Und außerhalb der Logik ist alles Zufall."
(L. Wittgenstein, T.6.3)

„Triggers considered harmful"
(A. Reuter)

Der Beitrag ist ein Plädoyer für eine triggerorientierte Programmierung auf der Basis des ECA-Konzepts. Dazu werden die begrifflichen und logischen Grundlagen des ECA-Konzepts erarbeitet. Der Schwerpunkt liegt dabei auf der Analyse und Synthese von Ereignissen, da sich der Ereignisbegriff als grundlegend für den Entwurf eines Triggersystems herausgestellt hat. Ein Fork- und ein Join-Schema stehen im Mittelpunkt einer Triggerkontrollogik zur Überwachung der anwendungstypischen Nebenläufigkeit. Für eine nebenwirkungsfreie Triggerverarbeitung wird das Transaktionskonzept herangezogen. Die Untersuchungen werden am Beispiel einer Paketverteilanlage veranschaulicht.

Logical Foundations of a Trigger Design System

The paper advocates trigger-oriented programming on the basis of the ECA-concept. The conceptual and logical foundations of the ECA-mechanisms are investigated. The event concept is of paramount importance in trigger-programming. Therefore the analysis and synthesis of events are elaborated. As far as trigger interferences are concerned a fork and join schema is developed in order to control parallel processing. The transaction concept is used to guarantee parallel trigger processing without side-effects. A parcel distribution system is outlined as an illustrative example.

1. Einleitung

Die klassische Programmierung geht davon aus, daß Kontrollfluß (Kontrollogik), Aktionsausführung (Verarbeitungsschritte) und Datenzugriff integriert ablaufen. In einem deterministischen Programm ist nach einer Aktionsausführung die Folgeaktion immer eindeutig determiniert. Eine isolierte Änderung von Komponenten zum Zwecke der Erstellung von Programmvarianten ist nur in einfachen Fällen möglich. Mit seinem berühmten Aufsatz über „Guarded Commands" war Dijkstra [Dijk 75] einer der ersten auf dem Gebiete der Programmierung, der konditionierte Aktionsteile als Regeln separat von einer Kontrollogik darstellte, um die Struktur nichtdeterministischer Programme zu demonstrieren. In solchen Programmen ist nach einer Aktionsausführung nicht bestimmt, welche Aktion als nächstes zur Ausführung kommt. Die regelorientierte Programmierung und ihre Sprachen sind eine Weiterentwicklung dieses Ansatzes. Auf dem Gebiet der Datenbanksysteme war es etwa zur gleichen Zeit Eswaran [Eswa 76], der konditionierte Datenzugriffe, sog. Trigger, zur Erhaltung der Datenkonsistenz einführte. Weiterentwickelte Datenbanksysteme wie z.B. das Postgres-System [StHP 88] verfügen mittlerweile über ein Triggersubsystem. Unter dem Gattungsbegriff „Aktive Datenbanksysteme" werden heute diese Entwicklungen zusammengefaßt. Insbesondere Dayal [Daya 88] hat vorgeschlagen, Trigger unter dem Aspekt der Sequenz *event, condition, action* (ECA) zu analysieren. „E" ist das auslösende Ereignis, „C" ist die auszuwertende Bedingung, die erfüllt sein muß, um die Aktion „A" ausführen zu können. Um das Verständnis für die Unterscheidung von E und C zu verbessern, wird herausgestellt, daß durch ein Ereignis (E) festgelegt wird, wann ein Trigger angewendet werden kann (Zeitbezug), während die Bedingung (C) bestimmt, was dann vorausgesetzt werden muß (Sachbezug). Folgen der Sachbezug (C) und die Aktionsausführung ohne Wartezeiten unmittelbar auf das Auftreten eines Ereignisses, so spricht man von einem synchronen Verlauf. Gibt es Wartezeiten, so liegt ein asynchroner Verlauf vor.

Beispiele für die Verwendung des ECA-Konzeptes lassen sich in den verschiedensten Anwendungsbereichen finden: Prozeßsteuerung und -überwachung in Fertigungssystemen, Lagerverwaltung, Börse, Software-Entwicklungsumgebungen, Netzwerk-Management, Datenbankanwendungen etc. Nach-