

Diss. ETH No. 14755

The Active Object System Design and Multiprocessor Implementation

A dissertation submitted to the
SWISS FEDERAL INSTITUTE OF TECHNOLOGY ZURICH
(ETH Zurich)

for the degree of
Doctor of Technical Sciences

presented by
Pieter Johannes Muller
M.Sc., Stellenbosch University
born 11th July 1968
citizen of the Republic of South Africa

accepted on the recommendation of

Prof. Dr. J. Gutknecht, examiner
Prof. Dr. N. Wirth, co-examiner

2002

Abstract

Today's pervasive operating systems are large agglomerations of software with growing resource requirements in every new release; both for their development and their operation. Fortunately, with the Oberon project Wirth and Gutknecht have demonstrated that it is possible to develop interesting, practical systems with light resource requirements.

We describe the design and implementation of a generally applicable system more powerful than Oberon, but still comparatively simple and transparent. It is based on a lean and efficient extensible kernel intended for varied hardware environments — servers, workstations, embedded control systems, and small mobile devices.

The kernel primarily acts as a runtime environment for the Active Oberon language, which uses an *active object* concept for concurrency, and *exclusive regions*, combined with a general *await* statement, for synchronization. In a related project the kernel was reused as the basis for a Java virtual machine, demonstrating its flexibility.

The system was implemented for Intel IA-32 symmetric multiprocessor machines, where it schedules active objects to run in parallel, but it also runs on singleprocessor machines. A student has ported it to the Intel StrongARM processor, popular in mobile devices.

This work makes three main contributions. It resulted in the release of a reliable and flexible system that is powerful enough for real applications. It relates experience with active objects and the elegant *await* synchronization statement. Finally, it serves as a new example of the *lean software* approach to system design. Arguably, by focusing on the essential, such systems can perform their task more securely and reliably than conventional operating systems.

Zusammenfassung

Die heute verbreitetsten Betriebssysteme sind grosse Ansammlungen von Softwareelementen, und bei jeder neuen Version wachsen die Ressourcenanforderungen, sowohl für die Entwicklung als auch für den Betrieb. Wie Wirth und Gutknecht indessen mit dem Oberon-Projekt belegen konnten, ist die Entwicklung interessanter, praktischer, mit wenig Ressourcen auskommender Systeme durchaus möglich.

In der vorliegenden Arbeit werden Design und Implementierung eines allgemein einsetzbaren Systems beschrieben, das leistungsfähiger als Oberon ist, bei vergleichbarer Einfachheit und Transparenz. Das System basiert auf einem schlanken und zugleich leistungsfähigen, erweiterbaren Kern, der sich für verschiedenste Hardwareumgebungen wie Server, Workstations, eingebettete Kontrollsysteme und kleine Mobilgeräte eignet.

Der Kern dient hauptsächlich als Laufzeitumgebung für die Active-Oberon-Sprache, welche die Konzepte *aktive Objekte* für Nebenläufigkeit und *exklusive Regionen*, kombiniert mit einem allgemeinen *await*-Statement, für Synchronisierung verwendet. Die Flexibilität des Kerns wurde in einem verwandten Dissertationsprojekt aufgezeigt, bei dem er als Grundlage für eine Java virtuelle Maschine verwendet wurde.

Das System wurde für Intel IA-32 symmetrische Multiprozessor-Rechner implementiert, wo es aktive Objekte parallel ablaufen lässt, funktioniert aber genauso auf Einprozessor-Maschinen. In einem Studentenprojekt wurde es auf den Intel StrongARM Prozessor portiert, der häufig bei Mobilgeräten eingesetzt wird.

Folgende Hauptbeiträge werden in der vorliegenden Studie präsentiert: Ein zuverlässiges und flexibles Betriebssystem wurde entwickelt, das leistungsfähig genug ist für echte Anwendungen. Im weiteren wird über die gemachten Erfahrungen mit aktiven Objekten und dem elegan-

ten *await*-Synchronisierungsstatement berichtet. Schliesslich wird ein neuartiges Beispiel des *lean software* Ansatzes zum Systemdesign aufgezeigt. Derart entwickelte Systeme dürften, indem sie sich auf das Wesentliche konzentrieren, ihre Aufgabe sicherer und zuverlässiger erfüllen als herkömmliche Betriebssysteme.