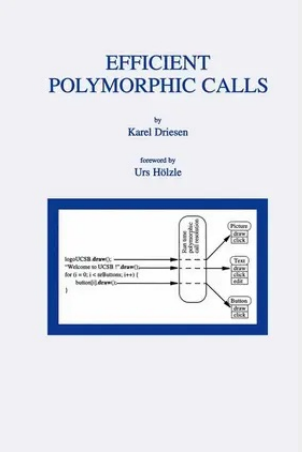
Driesen

Efficient Polymorphic Calls

The implementation of object-oriented languages has been an active topic of research since the 1960s when the first Simula compiler was written. The topic received renewed interest in the early 1980s with the growing popularity of object-oriented programming languages such as c++ and Smalltalk, and got another boost with the advent of Java. Polymorphic calls are at the heart of object-oriented languages, and even the first implementation of Simula-67 contained their classic implementation via virtual function tables. In fact, virtual function tables predate even Simula-for example, Ivan Sutherland's Sketchpad drawing editor employed very similar structures in 1960. Similarly, during the 1970s and 1980s the implementers of Smalltalk systems spent considerable efforts on implementing polymorphic calls for this dynamically typed language where virtual function tables could not be used. Given this long history of research into the implementation of polymorphic calls, and the relatively mature standing it achieved over time, why, one might ask, should there be a new book in this field? The answer is simple. Both software and hardware have changed considerably in recent years, to the point where many assumptions underlying the original work in this field are no longer true. In particular, virtual function tables are no longer sufficient to implement polymorphic calls even for statically typed languages; for example, Java's interface calls cannot be implemented this way. Furthermore, today's processors are deeply pipelined and can execute instructions out-of order, making it difficult to predict the execution time of even simple code sequences.

The implementation of object-oriented languages has been an active topic of research since the 1960s when the first Simula compiler was written. The topic received renewed interest in the early 1980s with the growing popularity of object-oriented programming languages such as c++ and Smalltalk, and got another boost with the advent of Java. Polymorphic calls are at the heart of object-oriented languages, and even the first implementation of Simula-67 contained their classic implementation via virtual function tables. In fact, virtual function tables predate even Simula-for example, Ivan Sutherland's Sketchpad drawing editor employed very similar structures in 1960. Similarly, during the 1970s and 1980s the implementers of Smalltalk systems spent considerable efforts on implementing polymorphic calls for this dynamically typed language where virtual function tables could not be used. Given this long history of research into the implementation of polymorphic calls, and the relatively mature standing it achieved over time, why, one might ask, should there be a new book in this field? The answer is simple. Both software and hardware have changed considerably in recent years, to the point where many assumptions underlying the original work in this field are no longer true. In particular, virtual function tables are no longer sufficient to implement polymorphic calls even for statically typed languages; for example, Java's interface calls cannot be implemented this way. Furthermore, today's processors are deeply pipelined and can execute instructions out-of order, making it difficult to predict the execution time of even simple code sequences.



160,49 €
149,99 € (zzgl. MwSt.)

Lieferfrist: bis zu 10 Tage

Artikelnummer: 9781461356752
Medium: Buch
ISBN: 978-1-4613-5675-2
Verlag: Springer
Erscheinungstermin: 26.10.2012
Sprache(n): Englisch
Auflage: 1. Auflage 2012
Serie: The Springer International Series in Engineering and Computer Science
Produktform: Kartoniert
Gewicht: 371 g
Seiten: 216
Format (B x H): 155 x 235 mm

Kundenservice Verlag Dr. Otto Schmidt KG
Neumannstraße 10, 40235 Düsseldorf | kundenservice@otto-schmidt.de | 0221-93738-997 (Inland)