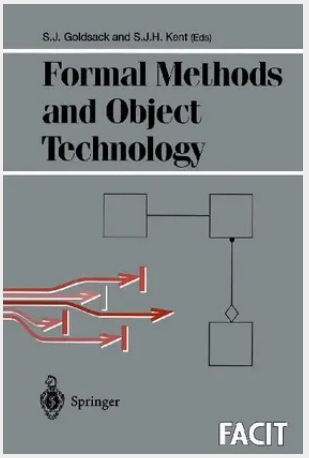
Goldsack / Kent

Formal Methods and Object Technology

Rationale Software engineering aims to develop software by using approaches which en able large and complex program suites to be developed in a systematic way. However, it is well known that it is difficult to obtain the level of assurance of correctness required for safety critical software using old fashioned program ming techniques. The level of safety required becomes particularly high in software which is to function without a break for long periods of time, since the software cannot be restarted and errors can accumulate. Consequently programming for mission critical systems, for example, needs to address the requirements of correctness with particular care. In the search for techniques for making software cheaper and more reliable, two important but largely independent influences have been visible in recent years. These are: • Object Technology • Formal Methods First, it has become evident that objects are, and will remain an important concept in software. Experimental languages of the 1970's introduced various concepts of package, cluster, module, etc. giving concrete expression to the importance of modularity and encapsulation, the construction of software com ponents hiding their state representations and algorithmic mechanisms from users, exporting only those features (mainly the procedure calling mechanisms) which were needed in order to use the objects. This gives the software com ponents a level of abstraction, separating the view of what a module does for the system from the details of how it does them.

Rationale Software engineering aims to develop software by using approaches which en able large and complex program suites to be developed in a systematic way. However, it is well known that it is difficult to obtain the level of assurance of correctness required for safety critical software using old fashioned program ming techniques. The level of safety required becomes particularly high in software which is to function without a break for long periods of time, since the software cannot be restarted and errors can accumulate. Consequently programming for mission critical systems, for example, needs to address the requirements of correctness with particular care. In the search for techniques for making software cheaper and more reliable, two important but largely independent influences have been visible in recent years. These are: ¿ Object Technology ¿ Formal Methods First, it has become evident that objects are, and will remain an important concept in software. Experimental languages of the 1970's introduced various concepts of package, cluster, module, etc. giving concrete expression to the importance of modularity and encapsulation, the construction of software com ponents hiding their state representations and algorithmic mechanisms from users, exporting only those features (mainly the procedure calling mechanisms) which were needed in order to use the objects. This gives the software com ponents a level of abstraction, separating the view of what a module does for the system from the details of how it does them.



106,99 €
99,99 € (zzgl. MwSt.)

Lieferfrist: bis zu 10 Tage

Artikelnummer: 9783540199779
Medium: Buch
ISBN: 978-3-540-19977-9
Verlag: Springer
Erscheinungstermin: 26.04.1996
Sprache(n): Englisch
Auflage: Softcover Nachdruck of the original 1. Auflage 1996
Serie: Formal Approaches to Computing and Information Technology (FACIT)
Produktform: Kartoniert
Gewicht: 587 g
Seiten: 368
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)