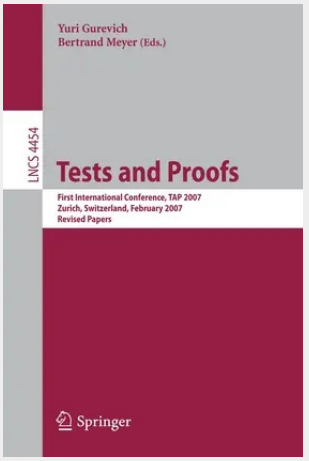
Meyer / Gurevich

Tests and Proofs

First International Conference, TAP 2007 Zurich, Switzerland, February 12-13, 2007 Revised Papers

To prove the correctness of a program is to demonstrate, through impeccable mathematical techniques, that it has no bugs. To test a program is to run it with the expectation of discovering bugs. These two paths to software reliability seem to diverge from the very start: if you have proved your program correct, it is fruitless to comb it for bugs; and if you are testing it, that surely must be a sign that you have given up on any hope to prove its correctness. Accordingly, proofs and tests have, since the onset of software engineering research, been pursued by distinct communities using different kinds of techniques and tools. Dijkstra's famous pronouncement that tests can only show the presence of errors — in retrospect, perhaps one of the best advertisements one can imagine for testing, as if "only" finding bugs were not already a momentous achievement! — didn't help make testing popular with provers, or proofs attractive to testers. And yet the development of both approaches leads to the discovery of common issues and to the realization that each may need the other. The emergence of model checking was one of the first signs that apparent contradiction may yield to complementarity; in the past few years an increasing number of research efforts have encountered the need for combining proofs and tests, dropping earlier dogmatic views of incompatibility and taking instead the best of what each of these software engineering domains has to offer.

To prove the correctness of a program is to demonstrate, through impeccable mathematical techniques, that it has no bugs. To test a program is to run it with the expectation of discovering bugs. These two paths to software reliability seem to diverge from the very start: if you have proved your program correct, it is fruitless to comb it for bugs; and if you are testing it, that surely must be a sign that you have given up on any hope to prove its correctness. Accordingly, proofs and tests have, since the onset of software engineering research, been pursued by distinct communities using different kinds of techniques and tools. Dijkstra's famous pronouncement that tests can only show the presence of errors — in retrospect, perhaps one of the best advertisements one can imagine for testing, as if "only" finding bugs were not already a momentous achievement! — didn't help make testing popular with provers, or proofs attractive to testers. And yet the development of both approaches leads to the discovery of common issues and to the realization that each may need the other. The emergence of model checking was one of the first signs that apparent contradiction may yield to complementarity; in the past few years an increasing number of research efforts have encountered the need for combining proofs and tests, dropping earlier dogmatic views of incompatibility and taking instead the best of what each of these software engineering domains has to offer.



53,49 €
49,99 € (zzgl. MwSt.)

Lieferfrist: bis zu 10 Tage

Artikelnummer: 9783540737698
Medium: Buch
ISBN: 978-3-540-73769-8
Verlag: Springer Berlin Heidelberg
Erscheinungstermin: 09.08.2007
Sprache(n): Englisch
Auflage: 2007
Serie: Lecture Notes in Computer Science
Produktform: Kartoniert
Gewicht: 359 g
Seiten: 217
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)