Formal Methods and Testing

An Outcome of the FORTEST Network
Revised Selected Papers
Preface

With the growing significance of computer systems within industry and wider society, techniques that assist in the production of reliable software are becoming increasingly important. The complexity of many computer systems requires the application of a battery of such techniques. Two of the most promising approaches are formal methods and software testing. Traditionally, formal methods and software testing have been seen as rivals. Thus, they largely failed to inform one another and there was very little interaction between the two communities. In recent years, however, a new consensus has developed. Under this consensus, these approaches are seen as complementary. This has introduced the prospect of collaboration between individuals and groups in these fields and work such as that contained in this book.

This book, which came out of the Formal Methods and Testing (FORTEST) network [3], includes 12 peer-reviewed chapters on ways in which formal methods and software testing complement each other. FORTEST was formed as a network established under UK EPSRC funding that investigated the relationships between formal (and semi-formal) methods and software testing. In particular, it was concerned with ways in which these areas complement each other. This is an exciting area of research that has led to a number of significant results produced in both industry and academia. While the EPSRC funding for FORTEST was for a fixed period only, FORTEST is now a subject group of two BCS Special Interest Groups: Formal Aspects of Computing Science (BCS FACS) and Special Interest Group in Software Testing (BCS SIGIST).

FORTEST has considered a wide range of testing activities, including static testing techniques (such as model-checking and theorem proving) as well as dynamic testing. Its members have investigated a wide range of problems of interest to the testing and formal methods communities. Examples include: automating (black-box) test generation; producing tests likely (or guaranteed) to detect certain types of fault; using static analysis to support software testing; methods for reasoning about test effectiveness; producing tests that are likely to find faults in systems formally refined from specifications; and using tests to explore assumptions underlying proofs. While much of the work in this area has concerned the problem of generating tests from formal specifications and models, there is a much wider range of ways in which these fields might interact. In particular, there is the interesting question of the relationships between static testing and dynamic testing.

The relationship between formal methods and software testing has received significant attention since the late 1980s. Seminal work during this period includes the paper by Dick and Faivre on testing from a VDM specification [4] and work on testing from algebraic specifications (see, for example, [2, 6]), the latter introducing a framework. More recently, the term model-based testing has