In conferences dedicated to concurrency, there is a lot of interest in our tutorials on software testing and review of concurrent software. People come to the tutorial because they know that to get their system to work they need to deal with nasty concurrency bugs such as races and deadlocks. Whilst everyone, both in the industry and academia, acknowledge that the topic of concurrency testing is difficult, important and interesting, no workshop, previous to Parallel and Distributed Systems: Testing and Debugging (PADTAD), was dedicated to it and, consequently, a community of researchers was never formed.

This special issue of Concurrency and Computation: Practice and Experience is dedicated to the best papers of the first two PADTAD workshops. PADTAD was created to nurture a community dedicated to testing and debugging multi-threaded/concurrent/parallel applications. The people working on this topic publish in a number of domains, dedicated to parallelism, testing, formal verification, and others. The workshop arose out of a need to get all of these people working together, as no single approach is likely to solve the problem. The first paper in the special issue, ‘Towards a framework and a benchmark for testing tools for multi-threaded programs’, both explains the need for mutual cooperation and suggests an architecture on which different tools in the domain can collaborate.

To suggest solutions to the concurrency testing problem, the unique characteristics of bugs in the multi-threaded domain have to be studied. Two papers, ‘A method for verifying concurrent Java components based on an analysis of concurrency failures’ and ‘Distributed desk checking’, describe fault models for the concurrent domain. The first paper uses the fault models to propose a verification method for concurrent Java programs and the second presents a method to review such programs. Another infrastructure paper, ‘An instrumentation technique for online analysis of multithreaded programs’, describes instrumentation, which is a key problem in research on testing tools, and instrumentation of multi-threaded programs requires specific techniques and tool support.

The paper ‘MultiRace: efficient on-the-fly data race detection in multithreaded C++ programs’ describes an approach to race detection that addresses the computational costs, so that race detection can be used in commercial applications.

The paper ‘Choosing among alternative pasts’ presents the beginning of research on a novel and promising noise-generation technique for testing. Two further papers on the topic published elsewhere, ‘Fidgeting until the point of no return’ (published in the PADTAD proceedings) and ‘Choosing among alternative futures’ (published in the Haifa Verification Conference proceedings), elaborate on the approach and tool support. The last paper in this collection, ‘A methodology for early validation of cache coherence protocols based on relational databases’ is interesting in itself and because it shows the relevance of the technologies discussed in PADTAD to the hardware verification domain.
The first PADTAD workshop took place in Nice in April 2003 and was part of the *IEEE International Parallel and Distributed Processing Symposium* (IPDPS) and included 11 presentations. The second PADTAD workshop occurred in Santa Fe, NM, in April 2004 and was also part of IPDPS with 11 presentations. The third PADTAD took place in Haifa as part of the *IBM Verification Conference* and the fourth PADTAD is scheduled to be part of *ISSTA 2006*. The community is growing and getting better organized. The seven papers selected from the first two workshops provide a good overview for anyone interested in the field and also suggest interesting research topics.

I hope that you will enjoy this special issue and join our growing community.

SHMUEL UR

*IBM Haifa Research Laboratory,*

*Haifa 31905, Israel*