Prof. Bernhard Steffen, TU Dortmund (German)

BIO

Bernhard Steffen graduated in Mathematics (1983) and obtained a PhD in Computer Science (1987) from the Christian-Albrechts Universität Kiel (D), then he was Research Fellow at the Laboratory for Foundations of Computer Science (LFCS) in Edinburgh and Researcher at the University of Aarhus (Denmark). 1990 he became Associate Professor for Distributed Systems at RWTH Aachen, and 1993 Full Professor for Programming Systems at the University of Passau. Since 1997 he holds the Chair of Programming Systems and Compiler Construction at the University of Dortmund, where he was Dean of Computer Science between 2002 and 2006 and member of the Senate 2006/07.

He is author of over 400 internationally refereed papers concerning various aspects of formal (verification) methods and tools for program analysis, compiler optimization, model generation, testing, and service-oriented software development, and methods for the explanation of computational results (h-number 67). He has served on more than 150 Program Committees, over 15 times as chair, on numerous Steering Committees, and in the Editorial Boards of the *ACM Transactions on Programming Languages and Systems* (TOPLAS), Kluwer´s *Formal Methods in System Design*, as well as Springer´*s Software: Concepts and Tools* and *Innovations in Software and Systems Engineering: A NASA Journal*. Since 2004 he is editor of LNCS (Lecture Notes in Computer Science) for the sub-libraries *`Theoretical Computer Science´*, *`Programming Techniques and Software Engineering´* and *`Advanced Research in Computing and Software Science´*,

He has broad experience in the use of formal methods to support state of the art industrial software development of distributed cooperative systems through major academic and industrial projects (where he won the European IT Award in 1996, and a start-up competition in 2001) and consulting (he was founder and member of the Advisory Board of various start-up companies), as well as through his activity as an Advisory Board Member of ASTEC, a Swedish technology transfer initiative for *Advanced Software TEChnology*, and as a member of the International Scientific Advisory Board of the UK strategic research & training initiative in Large-Scale Complex IT Systems (LSCITS).

He is founder and Editor in Chief of *Software Tools for Technology Transfer* (STTT), Springer Verlag, co-founder and Steering Committee Member of TACAS, the Int. Conference on *Tools and Algorithms for the Construction and Analysis of Systems*, and co-founder of ETAPS, *the European Joint Conference on Theory and Pratice of Software*. In 2004 he co-founded ISoLA (*Int. Symposium on Leveraging Applications* of Formal Methods, Verification and Validation), and in 2010 the Challenge for the *Rigorous Examination of Reactive Systems*. Characteristic for RERS is its use of Benchmark problems that are generated by property-preserving transformation.

In 1989 he co-developed the Concurrency Workbench, one of the earliest formal analysis tools for distributed and parallel systems, 1991 he set the scene for Software Model Checking with his paper *Data Flow Analysis as Model Checking*, 1992 he presented the first functioning Model Checker for infinite-state systems with *Model Checking for Context-Free Processes*, in 2002 he obtained the *Most Influential PLDI Paper Award* for *Lazy Code Motion*, which is given 10 years later in retrospective, and in 2014 he received the CAV Artifact Award for the Open-Source LearnLib. In 2020 he presented a precise algebraic explainability methodology based on semantic preserving transformations of Random Forests.

Currently, his research focuses on the interplay of methods and tools capturing various meta-levels:

(1) At the Language Workbench-level, the design of Domain-specific languages for supporting low code development as well as languages that are designed for, e.g., verifiability, testability, learnability, explainability. The point here is to guarantee properties via meta-level reasoning.

(2) At the IDE-level, property-preserving transformation approaches to support, e.g., correctness by construction, where the full semantics needs to be preserved, systematic (verification) Benchmark generation, where the satisfaction of, e.g., pre/post conditions or LTL formulas is maintained, or aspect-specific view generation for diagnosis. The latter may even provide so-called model explanations or outcome explanations for computational structures like Random Forests.

(3) At runtime, behaviour/observation-based inference of models and computational structures comprising automata learning, but also other machine learning techniques providing, e.g., Random forests and Neutral Networks. This inference may be active, i.e., explicitly querying the artefact to be analysed, as typical for automata learning, or passive, relying an provided data sets, as typical, e.g., in most Neural Network scenarios.