Handbook of Software Fault Localization

Foundations and Advances

Edited by
W. Eric Wong
T.H. Tse

IEEE PRESS
IEEE computer society
WILEY

Handbook of Software Fault Localization

IEEE Press
445 Hoes Lane
Piscataway, NJ 08854

About IEEE Computer Society

IEEE Computer Society is the world's leading computing membership organization and the trusted information and career-development source for a global workforce of technology leaders including: professors, researchers, software engineers, IT professionals, employers, and students. The unmatched source for technology information, inspiration, and collaboration, the IEEE Computer Society is the source that computing professionals trust to provide high-quality, state-of-the-art information on an on-demand basis. The Computer Society provides a wide range of forums for top minds to come together, including technical conferences, publications, and a comprehensive digital library, unique training webinars, professional training, and the Tech Leader Training Partner Program to help organizations increase their staff's technical knowledge and expertise, as well as the personalized information tool my Computer. To find out more about the community for technology leaders, visit http://www.computer.org.

IEEE/Wiley Partnership

The IEEE Computer Society and Wiley partnership allows the CS Press authored book program to produce a number of exciting new titles in areas of computer science, computing, and networking with a special focus on software engineering. IEEE Computer Society members receive a 35% discount on Wiley titles by using their member discount code. Please contact IEEE Press for details.

To submit questions about the program or send proposals, please contact Mary Hatcher, Editor, Wiley-IEEE Press: Email: mhatcher@wiley.com, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030-5774.

Handbook of Software Fault Localization

Foundations and Advances

Edited by

W. Eric Wong
Department of Computer Science, University of Texas at Dallas, Richardson, TX, USA

T.H. Tse
Department of Computer Science, The University of Hong Kong, Pokfulam, Hong Kong

This edition first published 2023

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada.

Library of Congress Cataloging-in-Publication Data

Names: Wong, W. Eric, editor. | Tse, T.H., editor. | John Wiley & Sons, publisher.
Title: Handbook of software fault localization : foundations and advances / edited by W. Eric Wong, T.H. Tse.
Description: Hoboken, New Jersey : Wiley, [2023] | Includes bibliographical references and index.
Identifiers: LCCN 2022037789 (print) | LCCN 2022037790 (ebook) | ISBN 9781119291800 (paperback) | ISBN 9781119291817 (adobe pdf) | ISBN 9781119291824 (epub)
Subjects: LCSH: Software failures. | Software failures–Data processing. | Debugging in computer science. | Computer software–Quality control.
Classification: LCC QA76.76.F34 H36 2023 (print) | LCC QA76.76.F34 (ebook) | DDC 005.1–dc23/eng/20220920
LC record available at https://lccn.loc.gov/2022037789
LC ebook record available at https://lccn.loc.gov/2022037790

Cover designed by T.H. Tse

Set in 9.5/12.5pt STIXTwoText by Straive, Pondicherry, India

Contents

Editor Biographies *xv*
List of Contributors *xvii*

1 Software Fault Localization: an Overview of Research, Techniques, and Tools *1*
W. Eric Wong, Ruizhi Gao, Yihao Li, Rui Abreu, Franz Wotawa, and Dongcheng Li
1.1 Introduction *1*
1.2 Traditional Fault Localization Techniques *14*
1.2.1 Program Logging *14*
1.2.2 Assertions *14*
1.2.3 Breakpoints *14*
1.2.4 Profiling *15*
1.3 Advanced Fault Localization Techniques *15*
1.3.1 Slicing-Based Techniques *15*
1.3.2 Program Spectrum-Based Techniques *20*
1.3.2.1 Notation *20*
1.3.2.2 Techniques *21*
1.3.2.3 Issues and Concerns *27*
1.3.3 Statistics-Based Techniques *30*
1.3.4 Program State-Based Techniques *32*
1.3.5 Machine Learning-Based Techniques *34*
1.3.6 Data Mining-Based Techniques *36*
1.3.7 Model-Based Techniques *37*
1.3.8 Additional Techniques *41*
1.3.9 Distribution of Papers in Our Repository *45*
1.4 Subject Programs *47*
1.5 Evaluation Metrics *50*
1.6 Software Fault Localization Tools *53*

1.7 Critical Aspects *58*
1.7.1 Fault Localization with Multiple Bugs *58*
1.7.2 Inputs, Outputs, and Impact of Test Cases *60*
1.7.3 Coincidental Correctness *63*
1.7.4 Faults Introduced by Missing Code *64*
1.7.5 Combination of Multiple Fault Localization Techniques *65*
1.7.6 Ties Within Fault Localization Rankings *67*
1.7.7 Fault Localization for Concurrency Bugs *67*
1.7.8 Spreadsheet Fault Localization *68*
1.7.9 Theoretical Studies *70*
1.8 Conclusion *71*
Notes *73*
References *73*

2 Traditional Techniques for Software Fault Localization *119*
Yihao Li, Linghuan Hu, W. Eric Wong, Vidroha Debroy, and Dongcheng Li
2.1 Program Logging *119*
2.2 Assertions *121*
2.3 Breakpoints *124*
2.4 Profiling *125*
2.5 Discussion *128*
2.6 Conclusion *130*
References *131*

3 Slicing-Based Techniques for Software Fault Localization *135*
W. Eric Wong, Hira Agrawal, and Xiangyu Zhang
3.1 Introduction *135*
3.2 Static Slicing-Based Fault Localization *136*
3.2.1 Introduction *136*
3.2.2 Program Slicing Combined with Equivalence Analysis *137*
3.2.3 Further Application *138*
3.3 Dynamic Slicing-Based Fault Localization *138*
3.3.1 Dynamic Slicing and Backtracking Techniques *144*
3.3.2 Dynamic Slicing and Model-Based Techniques *145*
3.3.3 Critical Slicing *148*
3.3.3.1 Relationships Between Critical Slices (CS) and Exact Dynamic Program Slices (DPS) *149*
3.3.3.2 Relationship Between Critical Slices and Executed Static Program Slices *150*
3.3.3.3 Construction Cost *150*

3.3.4 Multiple-Points Dynamic Slicing *151*
3.3.4.1 *BwS* of an Erroneous Computed Value *152*
3.3.4.2 *FwS* of Failure-Inducing Input Difference *152*
3.3.4.3 *BiS* of a Critical Predicate *154*
3.3.4.4 *MPSs:* Dynamic Chops *157*
3.3.5 Execution Indexing *158*
3.3.5.1 Concepts *159*
3.3.5.2 Structural Indexing *161*
3.3.6 Dual Slicing to Locate Concurrency Bugs *165*
3.3.6.1 Trace Comparison *165*
3.3.6.2 Dual Slicing *168*
3.3.7 Comparative Causality: a Causal Inference Model Based on Dual Slicing *173*
3.3.7.1 Property One: Relevance *174*
3.3.7.2 Property Two: Sufficiency *175*
3.3.8 Implicit Dependences to Locate Execution Omission Errors *177*
3.3.9 Other Dynamic Slicing-Based Techniques *179*
3.4 Execution Slicing-Based Fault Localization *179*
3.4.1 Fault Localization Using Execution Dice *179*
3.4.2 A Family of Fault Localization Heuristics Based on Execution Slicing *181*
3.4.2.1 Heuristic I *182*
3.4.2.2 Heuristic II *183*
3.4.2.3 Heuristic III *185*
3.4.3 Effective Fault Localization Based on Execution Slices and Inter-block Data Dependence *188*
3.4.3.1 Augmenting a Bad $\mathcal{D}^{(1)}$ *189*
3.4.3.2 Refining a Good $\mathcal{D}^{(1)}$ *190*
3.4.3.3 An Incremental Debugging Strategy *191*
3.4.4 Other Execution Slicing-Based Techniques in Software Fault Localization *193*
3.5 Discussions *193*
3.6 Conclusion *194*
Notes *195*
References *195*

4 Spectrum-Based Techniques for Software Fault Localization *201*
W. Eric Wong, Hua Jie Lee, Ruizhi Gao, and Lee Naish
4.1 Introduction *201*
4.2 Background and Notation *203*
4.2.1 Similarity Coefficient-Based Fault Localization *204*

4.2.2 An Example of Using Similarity Coefficient to Compute Suspiciousness *205*
4.3 Insights of Some Spectra-Based Metrics *210*
4.4 Equivalence Metrics *212*
4.4.1 Applicability of the Equivalence Relation to Other Fault Localization Techniques *217*
4.4.2 Applicability Beyond Fault Localization *218*
4.5 Selecting a Good Suspiciousness Function (Metric) *219*
4.5.1 Cost of Using a Metric *219*
4.5.2 Optimality for Programs with a Single Bug *220*
4.5.3 Optimality for Programs with Deterministic Bugs *221*
4.6 Using Spectrum-Based Metrics for Fault Localization *222*
4.6.1 Spectrum-Based Metrics for Fault Localization *222*
4.6.2 Refinement of Spectra-Based Metrics *227*
4.7 Empirical Evaluation Studies of SBFL Metrics *232*
4.7.1 The Construction of D^* *234*
4.7.2 An Illustrative Example *235*
4.7.3 A Case Study Using D^* *237*
4.7.3.1 Subject Programs *237*
4.7.3.2 Fault Localization Techniques Used in Comparisons *238*
4.7.3.3 Evaluation Metrics and Criteria *239*
4.7.3.4 Statement with Same Suspiciousness Values *240*
4.7.3.5 Results *241*
4.7.3.6 Effectiveness of D^* with Different Values of * *247*
4.7.3.7 D^* Versus Other Fault Localization Techniques *248*
4.7.3.8 Programs with Multiple Bugs *251*
4.7.3.9 Discussion *255*
4.8 Conclusion *261*
Notes *262*
References *263*

5 Statistics-Based Techniques for Software Fault Localization *271*
Zhenyu Zhang and W. Eric Wong
5.1 Introduction *271*
5.1.1 Tarantula *272*
5.1.2 How It Works *272*
5.2 Working with Statements *274*
5.2.1 Techniques Under the Same Problem Settings *275*
5.2.2 Statistical Variances *275*
5.3 Working with Non-statements *283*
5.3.1 Predicate: a Popular Trend *283*

5.3.2 BPEL: a Sample Application *285*
5.4 Purifying the Input *286*
5.4.1 Coincidental Correctness Issue *286*
5.4.2 Class Balance Consideration *287*
5.5 Reinterpreting the Output *288*
5.5.1 Revealing Fault Number *288*
5.5.2 Noise Reduction *291*
Notes *292*
References *293*

6 Machine Learning-Based Techniques for Software Fault Localization *297*
W. Eric Wong
6.1 Introduction *297*
6.2 BP Neural Network-Based Fault Localization *298*
6.2.1 Fault Localization with a BP Neural Network *298*
6.2.2 Reduce the Number of Candidate Suspicious Statements *302*
6.3 RBF Neural Network-Based Fault Localization *304*
6.3.1 RBF Neural Networks *304*
6.3.2 Methodology *305*
6.3.2.1 Fault Localization Using an RBF Neural Network *306*
6.3.2.2 Training of the RBF Neural Network *307*
6.3.2.3 Definition of a Weighted Bit-Comparison-Based Dissimilarity *309*
6.4 C4.5 Decision Tree-Based Fault Localization *309*
6.4.1 Category-Partition for Rule Induction *309*
6.4.2 Rule Induction Algorithms *310*
6.4.3 Statement Ranking Strategies *310*
6.4.3.1 Revisiting Tarantula *310*
6.4.3.2 Ranking Statements Based on C4.5 Rules *312*
6.5 Applying Simulated Annealing with Statement Pruning for an SBFL Formula *314*
6.6 Conclusion *317*
Notes *317*
References *317*

7 Data Mining-Based Techniques for Software Fault Localization *321*
Peggy Cellier, Mireille Ducassé, Sébastien Ferré, Olivier Ridoux, and W. Eric Wong
7.1 Introduction *321*
7.2 Formal Concept Analysis and Association Rules *324*
7.2.1 Formal Concept Analysis *325*

7.2.2 Association Rules *327*
7.3 Data Mining for Fault Localization *329*
7.3.1 Failure Rules *329*
7.3.2 Failure Lattice *331*
7.4 The Failure Lattice for Multiple Faults *336*
7.4.1 Dependencies Between Faults *336*
7.4.2 Example *341*
7.5 Discussion *342*
7.5.1 The Structure of the Execution Traces *342*
7.5.2 Union Model *343*
7.5.3 Intersection Model *343*
7.5.4 Nearest Neighbor *343*
7.5.5 Delta Debugging *344*
7.5.6 From the Trace Context to the Failure Context *344*
7.5.7 The Structure of Association Rules *345*
7.5.8 Multiple Faults *345*
7.6 Fault Localization Using *N*-gram Analysis *346*
7.6.1 Background *347*
7.6.1.1 Execution Sequence *347*
7.6.1.2 *N*-gram Analysis *347*
7.6.1.3 Linear Execution Blocks *349*
7.6.1.4 Association Rule Mining *349*
7.6.2 Methodology *350*
7.6.3 Conclusion *353*
7.7 Fault Localization for GUI Software Using *N*-gram Analysis *353*
7.7.1 Background *354*
7.7.1.1 Representation of the GUI and Its Operations *354*
7.7.1.2 Event Handler *356*
7.7.1.3 *N*-gram *356*
7.7.2 Association Rule Mining *357*
7.7.3 Methodology *357*
7.7.3.1 General Approach *358*
7.7.3.2 *N*-gram Fault Localization Algorithm *358*
7.8 Conclusion *360*
Notes *361*
References *361*

8 Information Retrieval-Based Techniques for Software Fault Localization *365*
Xin Xia and David Lo
8.1 Introduction *365*

8.2 General IR-Based Fault Localization Process *368*
8.3 Fundamental Information Retrieval Techniques for Software Fault Localization *369*
8.3.1 Vector Space Model *369*
8.3.2 Topic Modeling *370*
8.3.3 Word Embedding *371*
8.4 Evaluation Metrics *372*
8.4.1 Top-*k* Prediction Accuracy *372*
8.4.2 Mean Reciprocal Rank (MRR) *373*
8.4.3 Mean Average Precision (MAP) *373*
8.5 Techniques for Different Scenarios *374*
8.5.1 Text of Current Bug Report Only *374*
8.5.1.1 VSM Variants *374*
8.5.1.2 Topic Modeling *375*
8.5.2 Text and History *376*
8.5.2.1 VSM Variants *376*
8.5.2.2 Topic Modeling *378*
8.5.2.3 Deep Learning *378*
8.5.3 Text and Stack/Execution Traces *379*
8.6 Empirical Studies *380*
8.7 Miscellaneous *383*
8.8 Conclusion *385*
Notes *385*
References *386*

9 Model-Based Techniques for Software Fault Localization *393*
Birgit Hofer, Franz Wotawa, Wolfgang Mayer, and Markus Stumptner
9.1 Introduction *393*
9.2 Basic Definitions and Algorithms *395*
9.2.1 Algorithms for MBD *401*
9.3 Modeling for MBD *404*
9.3.1 The Value-Based Model *405*
9.3.2 The Dependency-Based Model *409*
9.3.3 Approximation Models for Debugging *413*
9.3.4 Other Modeling Approaches *416*
9.4 Application Areas *417*
9.5 Hybrid Approaches *418*
9.6 Conclusions *419*
Notes *420*
References *420*

10 Software Fault Localization in Spreadsheets *425*
Birgit Hofer and Franz Wotawa
10.1 Motivation *425*
10.2 Definition of the Spreadsheet Language *427*
10.3 Cones *430*
10.4 Spectrum-Based Fault Localization *431*
10.5 Model-Based Spreadsheet Debugging *435*
10.6 Repair Approaches *440*
10.7 Checking Approaches *443*
10.8 Testing *445*
10.9 Conclusion *446*
Notes *446*
References *447*

11 Theoretical Aspects of Software Fault Localization *451*
Xiaoyuan Xie and W. Eric Wong
11.1 Introduction *451*
11.2 A Model-Based Hybrid Analysis *452*
11.2.1 The Model Program Segment *452*
11.2.2 Important Findings *454*
11.2.3 Discussion *454*
11.3 A Set-Based Pure Theoretical Framework *455*
11.3.1 Definitions and Theorems *455*
11.3.2 Evaluation *457*
11.3.3 The Maximality Among All Investigated Formulas *461*
11.4 A Generalized Study *462*
11.4.1 Spectral Coordinate for SBFL *462*
11.4.2 Generalized Maximal and Greatest Formula in F *464*
11.5 About the Assumptions *465*
11.5.1 Omission Fault and 100% Coverage *465*
11.5.2 Tie-Breaking Scheme *467*
11.5.3 Multiple Faults *467*
11.5.4 Some Plausible Causes for the Inconsistence Between Empirical and Theoretical Analyses *468*
Notes *469*
References *470*

12 Software Fault Localization for Programs with Multiple Bugs *473*
Ruizhi Gao, W. Eric Wong, and Rui Abreu
12.1 Introduction *473*
12.2 One-Bug-at-a-Time *474*

12.3 Two Techniques Proposed by Jones et al. *475*
12.3.1 J1: Clustering Based on Profiles and Fault Localization Results *476*
12.3.1.1 Clustering Profile-Based Behavior Models *476*
12.3.1.2 Using Fault Localization to Stop Clustering *478*
12.3.1.3 Using Fault Localization Clustering to Refine Clusters *479*
12.3.2 J2: Clustering Based on Fault Localization Results *480*
12.4 Localization of Multiple Bugs Using Algorithms from Integer Linear Programming *481*
12.5 MSeer: an Advanced Fault Localization Technique for Locating Multiple Bugs in Parallel *483*
12.5.1 MSeer *485*
12.5.1.1 Representation of Failed Test Cases *485*
12.5.1.2 Revised Kendall tau Distance *486*
12.5.1.3 Clustering *488*
12.5.1.4 MSeer: a Technique for Locating Multiple Bugs in Parallel *494*
12.5.2 A Running Example *496*
12.5.3 Case Studies *499*
12.5.3.1 Subject Programs and Data Collections *499*
12.5.3.2 Evaluation of Effectiveness and Efficiency *501*
12.5.3.3 Results *503*
12.5.4 Discussions *510*
12.5.4.1 Using Different Fault Localization Techniques *510*
12.5.4.2 Apply MSeer to Programs with a Single Bug *510*
12.5.4.3 Distance Metrics *512*
12.5.4.4 The Importance of Estimating the Number of Clusters and Assigning Initial Medoids *514*
12.6 Spectrum-Based Reasoning for Fault Localization *514*
12.6.1 Barinel *515*
12.6.2 Results *517*
12.7 Other Studies *518*
12.8 Conclusion *520*
Notes *521*
References *522*

13 Emerging Aspects of Software Fault Localization *529*
T.H. Tse, David Lo, Alex Gorce, Michael Perscheid, Robert Hirschfeld, and W. Eric Wong
13.1 Introduction *529*
13.2 Application of the Scientific Method to Fault Localization *530*
13.2.1 Scientific Debugging *531*
13.2.2 Identifying and Assigning Bug Reports to Developers *532*

13.2.3 Using Debuggers in Fault Localization *534*
13.2.4 Conclusion *538*
13.3 Fault Localization in the Absence of Test Oracles by Semi-proving of Metamorphic Relations *538*
13.3.1 Metamorphic Testing and Metamorphic Relations *539*
13.3.2 The Semi-proving Methodology *541*
13.3.2.1 Semi-proving by Symbolic Evaluation *541*
13.3.2.2 Semi-proving as a Fault Localization Technique *542*
13.3.3 The Need to Go Beyond Symbolic Evaluation *543*
13.3.4 Initial Empirical Study *543*
13.3.5 Detailed Illustrative Examples *544*
13.3.5.1 Fault Localization Example Related to Predicate Statement *544*
13.3.5.2 Fault Localization Example Related to Faulty Statement *548*
13.3.5.3 Fault Localization Example Related to Missing Path *552*
13.3.5.4 Fault Localization Example Related to Loop *556*
13.3.6 Comparisons with Related Work *558*
13.3.7 Conclusion *560*
13.4 Automated Prediction of Fault Localization Effectiveness *560*
13.4.1 Overview of PEFA *561*
13.4.2 Model Learning *564*
13.4.3 Effectiveness Prediction *564*
13.4.4 Conclusion *564*
13.5 Integrating Fault Localization into Automated Test Generation Tools *565*
13.5.1 Localization in the Context of Automated Test Generation *566*
13.5.2 Automated Test Generation Tools Supporting Localization *567*
13.5.3 Antifragile Tests and Localization *568*
13.5.4 Conclusion *568*
Notes *569*
References *569*

Index *581*

Editor Biographies

W. Eric Wong, PhD, is a Full Professor, Director of Software Engineering Program, and the Founding Director of Advanced Research Center for Software Testing and Quality Assurance in Computer Science at the University of Texas at Dallas. He also has an appointment as a Guest Researcher with the National Institute of Standards and Technology, an agency of the US Department of Commerce.

Professor Wong's research focuses on helping practitioners improve software quality while reducing production cost. In particular, he is working on software testing, program debugging, risk analysis, safety, and reliability. He was the recipient of the ICST 2020 (The 13th IEEE International Conference on Software Testing, Verification, and Validation) Most Influential Paper award for his paper titled "Using Mutation to Automatically Suggest Fixes for Faulty Programs" published at ICST 2010. He also received a JSS 2020 (*Journal of Systems and Software*) Most Influential Paper award for his paper titled "A Family of Code Coverage-based Heuristics for Effective Fault Localization" published in Volume 83, Issue 2, 2010. The conference version of this paper received the Best Paper Award at the 31st IEEE International Computer Software and Applications Conference.

Professor Wong was the award recipient of the 2014 IEEE Reliability Society Engineer of the Year. In addition, he was the Editor-in-Chief of the *IEEE Transactions on Reliability* for six years ending on 31 May 2022. He has also been an Area Editor of Elsevier's *Journal of Systems and Software* since 2017. Dr. Wong received his MS and PhD in Computer Science from Purdue University, West Lafayette, IN, USA. More details can be found at Professor Wong's homepage https://personal.utdallas.edu/~ewong

T.H. Tse received his PhD in Information Systems from the London School of Economics and was a Visiting Fellow at the University of Oxford. He is an Honorary Professor in Computer Science with The University of Hong Kong after retiring from the full professorship in 2014. His research interest is in program testing and debugging. He has more than 270 publications, including a book in

the Cambridge Tracts in Theoretical Computer Science series, Cambridge University Press. He is ranked internationally as no. 2 among experts in metamorphic testing. The 2010 paper titled "Adaptive Random Testing: the ART of Test Case Diversity" by Professor Tse and team has been selected as the Grand Champion of the Most Influential Paper Award by the *Journal of Systems and Software*.

Professor Tse is a Steering Committee Chair of the IEEE International Conference on Software Quality, Reliability, and Security; and an Associate Editor of *IEEE Transactions on Reliability*. He served on the Search Committee for the Editor-in-Chief of *IEEE Transactions on Software Engineering* in 2013. He is a Life Fellow of the British Computer Society and a Life Senior Member of the IEEE. He was awarded an MBE by Queen Elizabeth II of the United Kingdom.

List of Contributors

Rui Abreu
Department of Informatics Engineering, Faculty of Engineering University of Porto, Porto, Portugal

Hira Agrawal
Peraton Labs, Basking Ridge, NJ, USA

Peggy Cellier
INSA, CNRS, IRISA, Universite de Rennes, Rennes, France

Vidroha Debroy
Department of Computer Science University of Texas at Dallas Richardson, TX, USA
and
Dottid Inc., Dallas, TX, USA

Mireille Ducassé
INSA, CNRS, IRISA, Universite de Rennes, Rennes, France

Sébastien Ferré
CNRS, IRISA, Universite de Rennes Rennes, France

Ruizhi Gao
Sonos Inc., Boston, MA, USA

Alex Gorce
School of Informatics, Computing, and Cyber Systems, Northern Arizona University, Flagstaff AZ, USA

Robert Hirschfeld
Department of Computer Science Universität Potsdam, Brandenburg Germany

Birgit Hofer
Institute of Software Technology Graz University of Technology, Graz Austria

Linghuan Hu
Google Inc., Mountain View CA, USA

Hua Jie Lee
School of Computing, Macquarie University, Sydney, NSW Australia

Dongcheng Li
Department of Computer Science University of Texas at Dallas Richardson, TX, USA

Yihao Li
School of Information and Electrical Engineering, Ludong University
Yantai, China

David Lo
School of Computing and Information Systems, Singapore Management University, Singapore

Wolfgang Mayer
Advanced Computing Research Centre, University of South Australia
Adelaide, SA, Australia

Lee Naish
School of Computing and Information Systems, The University of Melbourne
Melbourne, Australia

Michael Perscheid
Department of Computer Science
Universität Potsdam, Brandenburg
Germany

Olivier Ridoux
CNRS, IRISA, Universite de Rennes
Rennes, France

Markus Stumptner
Advanced Computing Research Centre, University of South Australia
Adelaide, SA, Australia

T.H. Tse
Department of Computer Science, The University of Hong Kong, Pokfulam
Hong Kong

W. Eric Wong
Department of Computer Science
University of Texas at Dallas
Richardson, TX, USA

Franz Wotawa
Institute of Software Technology, Graz University of Technology, Graz
Austria

Xin Xia
Software Engineering Application Technology Lab, Huawei, China

Xiaoyuan Xie
School of Computer Science, Wuhan University, Wuhan, China

Xiangyu Zhang
Department of Computer Science
Purdue University, West Lafayette
IN, USA

Zhenyu Zhang
Institute of Software, Chinese Academy of Sciences, Beijing, China

1

Software Fault Localization: an Overview of Research, Techniques, and Tools

W. Eric Wong[1], Ruizhi Gao[2], Yihao Li[3], Rui Abreu[4], Franz Wotawa[5], and Dongcheng Li[1]

[1] *Department of Computer Science, University of Texas at Dallas, Richardson, TX, USA*
[2] *Sonos Inc., Boston, MA, USA*
[3] *School of Information and Electrical Engineering, Ludong University, Yantai, China*
[4] *Department of Informatics Engineering, Faculty of Engineering, University of Porto, Porto, Portugal*
[5] *Institute of Software Technology, Graz University of Technology, Graz, Austria*

1.1 Introduction

Software fault localization, the act of identifying the locations of faults in a program, is widely recognized to be one of the most tedious, time-consuming, and expensive – yet equally critical – activities in program debugging. Due to the increasing scale and complexity of software today, manually locating faults when failures occur is rapidly becoming infeasible, and consequently, there is a strong demand for techniques that can guide software developers to the locations of faults in a program with minimal human intervention. This demand in turn has fueled the proposal and development of a broad spectrum of fault localization techniques, each of which aims to streamline the fault localization process and make it more effective by attacking the problem in a unique way. In this book, we categorize and provide a comprehensive overview of such techniques and discuss key issues and concerns that are pertinent to software fault localization.

Software is fundamental to our lives today, and with its ever-increasing usage and adoption, its influence is practically ubiquitous. At present, software is not just employed in, but is critical to, many security and safety-critical systems in industries such as medicine, aeronautics, and nuclear energy. Not surprisingly,

---

This chapter is an extension of Wong, W.E., Gao, R., Li, Y., Abreu, R., and Wotawa, F. (2016). A survey on software fault localization. *IEEE Transactions on Software Engineering* 42 (8): 707–740.

Handbook of Software Fault Localization: Foundations and Advances, First Edition.
Edited by W. Eric Wong and T.H. Tse.

Published 2023 by John Wiley & Sons, Inc.

this trend has been accompanied by a drastic increase in the scale and complexity of software. Unfortunately, this has also resulted in more software bugs, which often lead to execution failures with huge losses [1–3]. On 15 January 1990, the AT&T operation center in Bedminster, NJ, USA, had an increase of red warning signals appearing across the 75 screens that indicated the status of parts of the AT&T worldwide network. As a result, only about 50 percent of the calls made through AT&T were connected. It took nine hours for the AT&T technicians to identify and fix the issue caused by a misplaced break statement in the code. AT&T lost $60 to $75 million in this accident [4].

Furthermore, software faults in safety-critical systems have significant ramifications not only limited to financial loss, but also to loss of life, which is alarming [5]. On 20 December 1995, a Boeing 757 departed from Miami, FL, USA. The aircraft was heading to Cali, Colombia. However, it crashed into a 9800 feet mountain. A total of 159 deaths resulted; leaving only five passengers alive. This event marked the highest death toll of any accident in Colombia at the time. This accident was caused by the inconsistencies between the naming conventions of the navigational charts and the flight management system. When the crew looked up the waypoint "Rozo", the chart indicated the letter "R" as its identifier. The flight management system, however, had the city paired with the word "Rozo". As a result, when the pilot entered the letter "R", the system did not know if the desired city was Rozo or Romeo. It automatically picked Romeo, which is a larger city than Rozo, as the next waypoint.

A 2006 report from the National Institute of Standards and Technology (NIST) [6] indicated that software errors are estimated to cost the US economy $59.5 billion annually (0.6 percent of the GDP); the cost has undoubtedly grown since then. Over half the cost of fixing or responding to these bugs is passed on to software users, while software developers and vendors absorb the rest.

Even when faults in software are discovered due to erroneous behavior or some other manifestation of the fault(s),[1] finding and fixing them is an entirely different matter. Fault localization, which focuses on the former, i.e. identifying the locations of faults, has historically been a manual task that has been recognized to be time-consuming and tedious as well as prohibitively expensive [7], given the size and complexity of large-scale software systems today. Furthermore, manual fault localization relies heavily on the software developer's experience, judgment, and intuition to identify and prioritize code that is likely to be faulty. These limitations have led to a surge of interest in developing techniques that can partially or fully automate the localization of faults in software while reducing human input. Though some techniques are similar and some very different (in terms of the type of data consumed, the program components focused on, comparative effectiveness and efficiency, etc.), they each try to attack the problem of fault localization from a unique perspective, and typically offer both advantages and disadvantages relative

to one another. With many techniques already in existence and others continually being proposed, as well as with advances being made both from a theoretical and practical perspective, it is important to catalog and overview current state-of-the-art techniques in fault localization in order to offer a comprehensive resource for those already in the area and those interested in making contributions to it.

In order to provide a complete survey covering most of the publications related to software fault localization since the late 1970s, in this chapter, we created a publication repository that includes 587 papers published from 1977 to 2020. We also searched for Masters' and PhD theses closely related to software fault localization, which are listed in Table 1.1.

Table 1.1 A list of recent PhD and Masters' theses on software fault localization.

Author	Title	Degree	University	Year
Ehud Y. Shapiro [8]	Algorithmic Program Debugging	PhD	Yale University	1983
Hiralal Agrawal [9]	Towards Automatic Debugging of Computer Programs	PhD	Purdue University	1991
Hsin Pan [10]	Software debugging with dynamic instrumentation and test-based knowledge	PhD	Purdue University	1993
W. Bond Gregory [11]	Logic Programs for Consistency-based Diagnosis	PhD	Carleton University	1994
Benjamin Robert Liblit [12]	Cooperative Bug Isolation	PhD	The University of California, Berkeley	2004
Bernhard Peischl [13]	Automated Source-Level Debugging of Synthesizeable VHDL Designs	PhD	Graz University of Technology	2004
Haifeng He [14]	Automated Debugging using Path-based Weakest Preconditions	Master	University of Arizona	2004
Alex David Groce [15]	Error Explanation and Fault Localization with Distance Metrics	PhD	Carnegie Mellon University	2005

(*Continued*)

Table 1.1 (Continued)

Author	Title	Degree	University	Year
Emmanuel Renieris [16]	A Research Framework for Software-Fault Localization Tools	PhD	Brown University	2005
Daniel Köb [17]	Extended Modeling for Automatic Fault Localization in Object-Oriented Software	PhD	Graz University of Technology	2005
David Hovemeyer [18]	Simple and Effective Static Analysis to Find Bugs	PhD	University of Maryland	2005
Peifeng Hu [19]	Automated Fault Localization: a Statistical Predicate Analysis Approach	PhD	The University of Hong Kong	2006
Xiangyu Zhang [20]	Fault Localization via Precise Dynamic Slicing	PhD	The University of Arizona	2006
Rafi Vayani [21]	Improving Automatic Software Fault Localization	Master	Delft University of Technology	2007
Ramana Rao Kompella [22]	Fault Localization in Backbone Networks	PhD	University of California, San Diego	2007
Andreas Griesmayer [23]	Debugging Software: from Verification to Repair	PhD	Graz University of Technology	2007
Tao Wang [24]	Post-Mortem Dynamic Analysis For Software Debugging	PhD	Fudan University	2007
Sriraman Tallam [25]	Fault Location and Avoidance in Long-Running Multithreaded Applications	PhD	The University of Arizona	2007
Ophelia C. Chesley [26]	CRISP-A fault localization Tool for Java Programs	Master	Rutgers, The State University of New Jersey	2007
Shan Lu [27]	Understanding, Detecting and Exposing Concurrency Bugs	PhD	University of Illinois at Urbana-Champaign	2008

Table 1.1 (Continued)

Author	Title	Degree	University	Year
Naveed Riaz [28]	Automated Source-Level Debugging of Synthesizable Verilog Designs	PhD	Graz University of Technology	2008
James Arthur Jones [29]	Semi-Automatic Fault Localization	PhD	Georgia Institute of Technology	2008
Zhenyu Zhang [30]	Software Debugging through Dynamic Analysis of Program Structures	PhD	The University of Hong Kong	2009
Rui Abreu [31]	Spectrum-based Fault Localization in Embedded Software	PhD	Delft University of Technology	2009
Dennis Jefferey [32]	Dynamic State Alteration Techniques for Automatically Locating Software Errors	PhD	University of California Riverside	2009
Xinming Wang [33]	Automatic Localization of Code Omission Faults	PhD	The Hong Kong University of Science and Technology	2010
Fabrizio Pastore [34]	Automatic Diagnosis of Software Functional Faults by Means of Inferred Behavioral Models	PhD	University of Milan Bicocca	2010
Mihai Nica [35]	On the Use of Constraints in Automated Program Debugging – From Foundations to Empirical Results	PhD	Graz University of Technology	2010
Zachary P. Fry [36]	Fault Localization Using Textual Similarities	Master	The University of Virginia	2011
Hua Jie Lee [37]	Software Debugging Using Program Spectra	PhD	The University of Melbourne	2011

(*Continued*)

Table 1.1 (Continued)

Author	Title	Degree	University	Year
Vidroha Debroy [38]	Towards the Automation of Program Debugging	PhD	The University of Texas at Dallas	2011
Alberto Gonzalez Sanchez [39]	Cost Optimizations in Runtime Testing and Diagnosis	PhD	Delft University of Technology	2011
Jared David DeMott [40]	Enhancing Automated Fault Discovery and Analysis	PhD	Michigan State University	2012
Xin Zhang [41]	Secure and Efficient Network Fault Localization	PhD	Carnegie Mellon University	2012
Xiaoyuan Xie [42]	On the Analysis of Spectrum-based Fault Localization	PhD	Swinburne University of Technology	2012
Alexandre Perez [43]	Dynamic Code Coverage with Progressive Detail Levels	Master	University of Porto	2012
Raul Santelices [44]	Change-effects Analysis for Effective Testing and Validation of Evolving Software	PhD	Georgia Institute of Technology	2012
George. K. Baah [45]	Statistical Causal Analysis for Fault Localization	PhD	Georgia Institute of Technology	2012
Swarup K. Sahoo [46]	A Novel Invariants-based Approach for Automated Software Fault Localization	PhD	University of Illinois at Urbana-Champaign	2012
Birgit Hofer [47]	From Fault Localization of Programs written in 3rd level Language to Spreadsheets	PhD	Graz University of Technology	2013
Aritra Bandyopadhyay [48]	Mitigating the Effect of Coincidental Correctness in Spectrum-based Fault Localization	PhD	Colorado State University	2013

Table 1.1 (Continued)

Author	Title	Degree	University	Year
Shounak Roychowdhury [49]	A Mixed Approach to Spectrum-based Fault Localization Using Information Theoretic Foundations	PhD	The University of Texas at Austin	2013
Shaimaa Ali [50]	Localizing State-Dependent Faults Using Associated Sequence Mining	PhD	The University of Western Ontario	2013
Christian Kuhnert [51]	Data-driven Methods for Fault Localization in Process Technology	PhD	Karlsruhe Institute of Technology	2013
Dawei Qi [52]	Semantic Analyses to Detect and Localize Software Regression Errors	PhD	Tsinghua University	2013
William N. Sumner [53]	Automated Failure Explanation Through Execution Comparison	PhD	Purdue University	2013
Mark A. Hays [54]	A Fault-based Model of Fault Localization Techniques	PhD	University of Kentucky	2014
Sang Min Park [55]	Effective Fault Localization Techniques for Concurrent Software	PhD	Georgia Institute of Technology	2014
Gang Shu [56]	Statistical Estimation of Software Reliability and Failure-causing Effect	PhD	Case Western Reserve University	2014
Lucia [57]	Ranking-based Approaches for Localizing Faults	PhD	Singapore Management University	2014
Seok-Hyeon Moon [58]	Effective Software Fault Localization using Dynamic Program Behaviors	Master	Korea Advanced Institute of Science and Technology	2014

(*Continued*)

Table 1.1 (Continued)

Author	Title	Degree	University	Year
Yepang Liu [59]	Automated Analysis of Energy Efficiency and Performance for Mobile Applications	PhD	The Hong Kong University of Science and Technology	2014
Cuiting Chen [60]	Automated Fault Localization for Service-Oriented Software Systems	PhD	Delft University of Technology	2015
Matthias Rohr [61]	Workload-sensitive Timing Behavior Analysis for Fault Localization in Software Systems	PhD	Kiel University	2015
Ozkan Bayraktar [62]	Ela: an Automated Statistical Fault Localization Technique	PhD	The Middle East Technical University	2015
Azim Tonzirul [63]	Fault Discovery, Localization, and Recovery in Smartphone Apps	PhD	University of California Riverside	2016
Laleh Gholamosseinghandehari [64]	Fault Localization based on Combinatorial Testing	PhD	The University of Texas at Arlington	2016
Ruizhi Gao [65]	Advanced Software Fault Localization for Programs with Multiple Bugs	PhD	The University of Texas at Dallas	2017
Shih-Feng Sun [66]	Statistical Fault Localization and Causal Interactions	PhD	Case Western Reserve University	2017
Rongxin Wu [67]	Automated Techniques for Diagnosing Crashing Bugs	PhD	The Hong Kong University of Science and Technology	2017
Arjun Roy [68]	Simplifying dataleft fault detection and localization	PhD	University of California San Diego	2018

Table 1.1 (Continued)

Author	Title	Degree	University	Year
Yun Guo [69]	Towards Automatically Localizing and Repairing SQL Faults	PhD	George Mason University	2018
Nasir Safdari [70]	Learning to Rank Relevant Files for Bug Reports Using Domain knowledge, Replication and Extension of a Learning-to-Rank Approach	Master	Rochester Institute of Technology	2018
Dai Ting [71]	A Hybrid Approach to Cloud System Performance Bug Detection, Diagnosis and Fix	PhD	North Carolina State University	2019
George Thompson [72]	Towards Automated Fault Localization for Prolog	Master	North Carolina A&T State University	2020
Xia Li [73]	An Integrated Approach for Automated Software Debugging via Machine Learning and Big Code Mining	PhD	The University of Texas at Dallas	2020
Muhammad Ali Gulzar [74]	Automated Testing and Debugging for Big Data Analytics	PhD	University of California, Los Angeles	2020
Mihir Mathur [75]	Leveraging Distributed Tracing and Container Cloning for Replay Debugging of Microservices	Master	University of California, Los Angeles	2020

All papers in our repository[2] are sorted by year, and the result is displayed in Figure 1.1. As shown in the figure, the number of publications grew rapidly after 2001, indicating that more and more researchers began to devote themselves to the area of software fault localization over the last two decades.

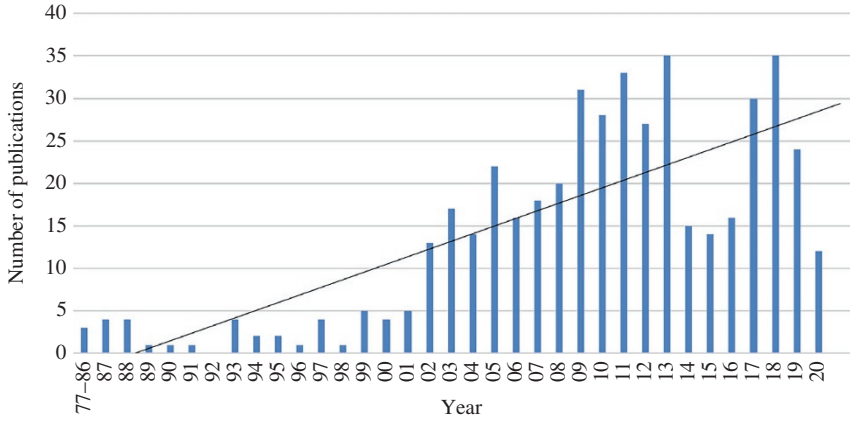

Figure 1.1 Papers on software fault localization from 1977 to 2020.

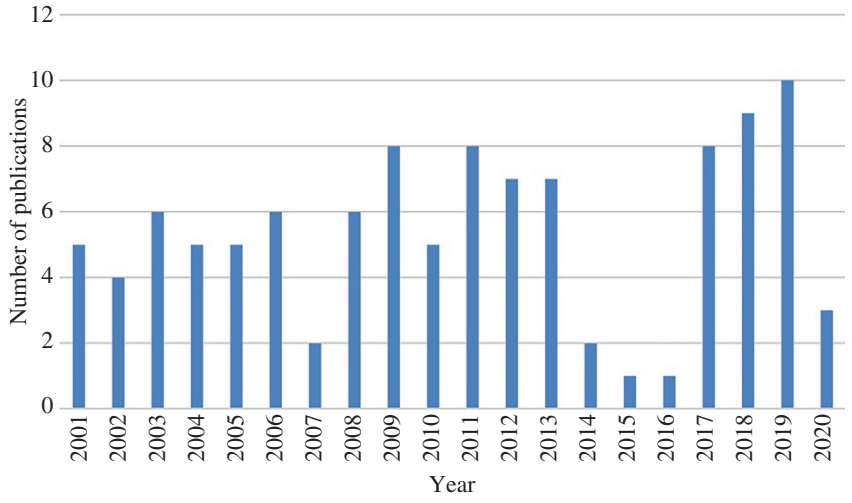

Figure 1.2 Publications on software fault localization in top venues from 2001 to 2020.

Also, as per our repository, Figure 1.2. gives the number of publications related to software fault localization that have appeared in top quality and leading journals and conferences that focus on Software Engineering – *IEEE Transactions on Software Engineering, ACM Transactions on Software Engineering and*