Panel Discussion

What we know

From *W. E. Wong, R. Gao, Y. Li, R. Abreu and F. Wotawa, "A Survey on Software Fault Localization," in IEEE Transactions on Software Engineering, vol. 42, no. 8, pp. 707-740, Aug. 1 2016. doi: 10.1109/TSE.2016.2521368*

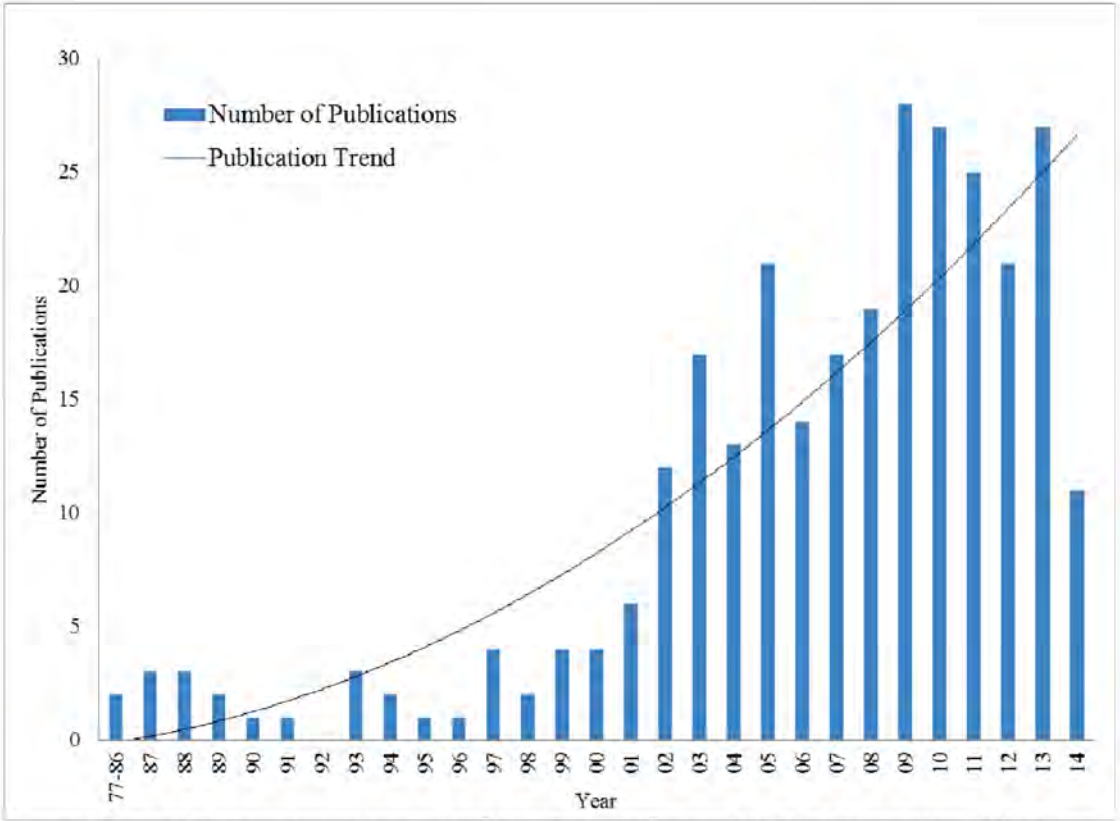


Figure 1. Papers on software fault localization from 1977 to November 2014

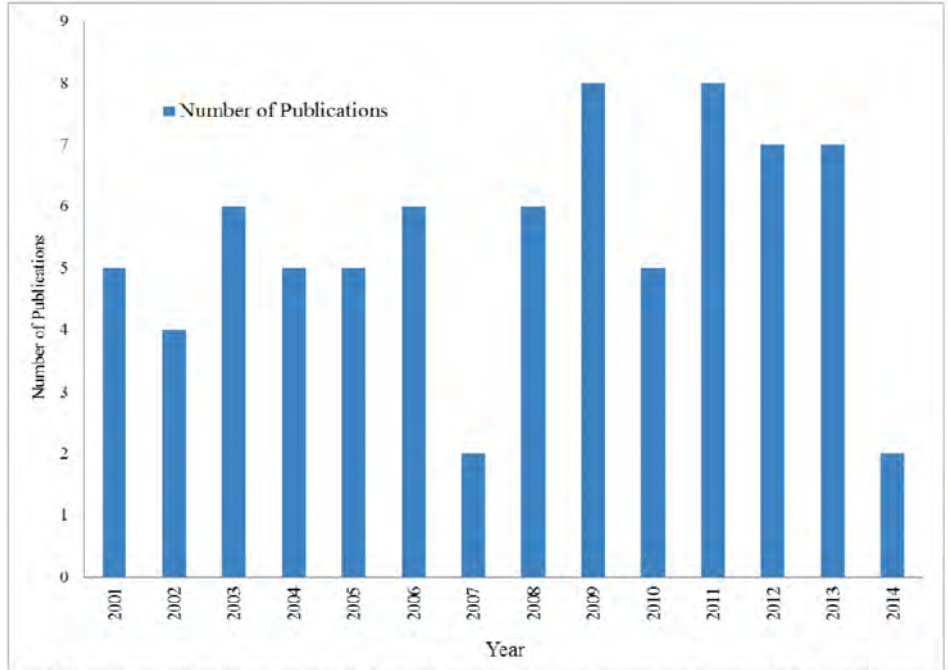


Figure 2. Publication of software fault localization in top venues from 2001 to November 2014

The beginnings

Program Slicing

MARK WEISER

Abstract—Program slicing is a method for automatically decomposing programs by analyzing their data flow and control flow. Starting from a subset of a program's behavior, slicing reduces that program to a minimal form which still produces that behavior. The reduced program, called a "slice," is an independent program guaranteed to represent faithfully the original program within the domain of the specified subset of behavior.

Some properties of slices are presented. In particular, finding *statement-minimal* slices is in general unsolvable, but using data flow analysis is sufficient to find approximate slices. Potential applications include automatic slicing tools for debugging and parallel processing of slices.

July 1984

ness shows up in testing, parallel processor distribution, maintenance, and especially debugging. A previous study showed experienced programmers mentally slicing while debugging, based on an informal definition of slice [22]. Our concern here is with 1) a formal definition of slices and their abstract

projections of behavior from the program being decomposed. Unlike procedures and data abstractions, slices are designed to be found automatically after a program is coded. Their usefulness shows up in testing, parallel processor distribution, maintenance, and especially debugging. A previous study showed experienced programmers mentally slicing while debugging, based on an informal definition of slice [22]. Our concern here is with 1) a formal definition of slices and their abstract properties, 2) a practical algorithm for slicing, and 3) some experience slicing real programs.

Manuscript received August 27, 1982; revised June 28, 1983. This work was supported in part by the National Science Foundation under Grant MCS-80-18294 and by the U.S. Air Force Office of Scientific Research under Grant F49620-80-C-001. A previous version of this paper was presented at the 5th International Conference on Software Engineering, San Diego, CA, 1981.

The author is with the Department of Computer Science, University of Maryland, College Park, MD 20742.

Definitions

This section considers programs without procedure calls. Procedures are discussed later. The first few definitions review the standard definitions of digraph, flowgraph, and computation in terms of state trajectory. Finally, a slice is defined as preserving certain projections from state trajectories.

The next few definitions simply establish a terminology for graphs, and restrict attention to programs whose control structure is single-entry single-exit ("hammock graphs").

Definition: A *digraph* is a structure $\langle N, E \rangle$, where N is a set

subset of V: $REF(n)$ is the set of variables whose values are used at n, and $DEF(n)$ is the set of variables whose values are changed at n.

A state trajectory of a program is just a trace of its execution which snapshots all the variable values just before executing each statement.

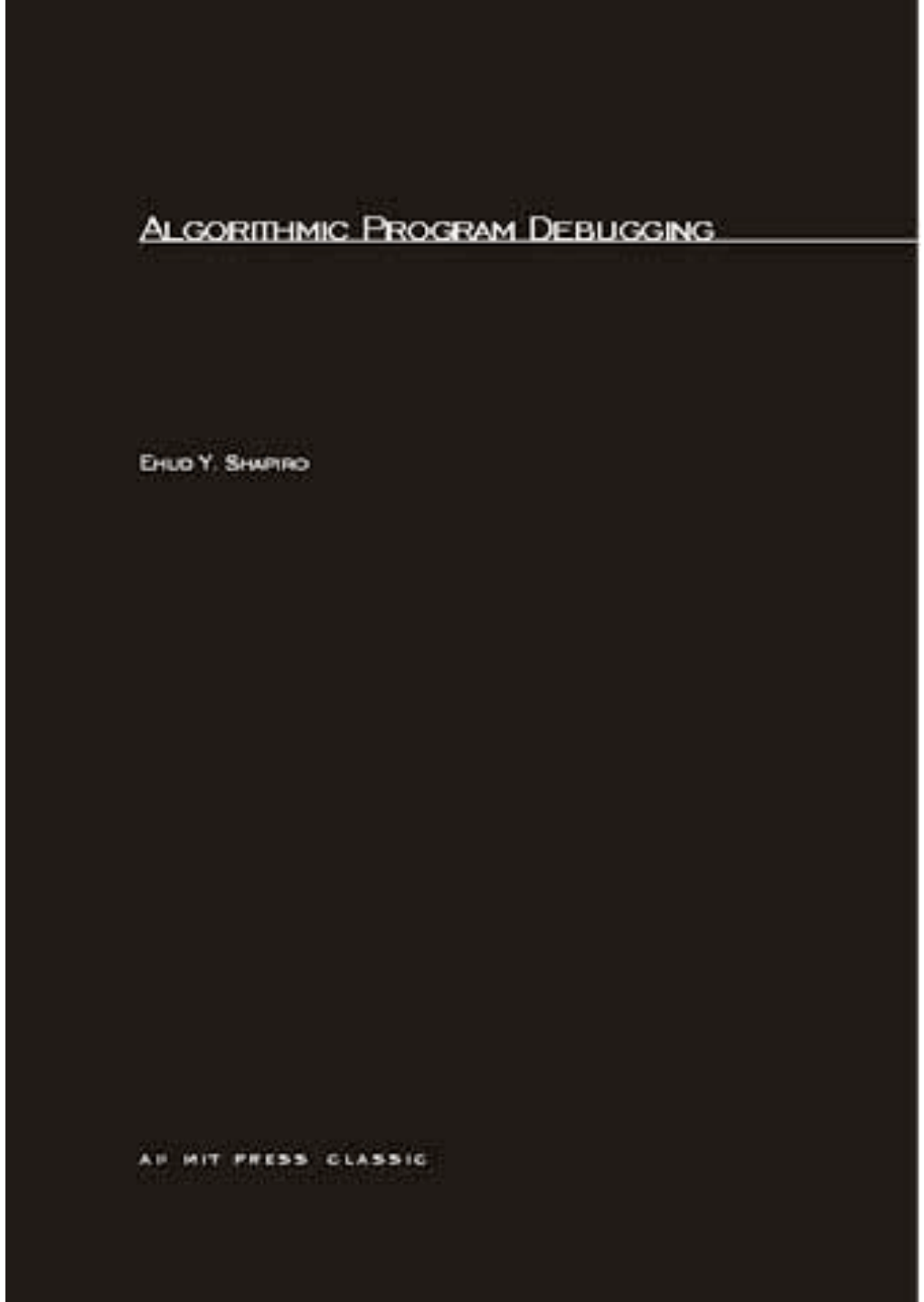
Definition: A *state trajectory* of length k of a program P is a finite list of ordered pairs

$$(n_1, s_1)(n_2, s_2) \cdots (n_k, s_k)$$

where each n is in N (the set of nodes in P) and each s is a function mapping the variables in V to their values. Each (n, s) gives the values of V immediately before the execution of n. Our attention will be on programs which halt, so infinite state trajectories are specifically excluded.

Slices reproduce a projection from the behavior of the original program. This projection must be the values of certain variables as seen at certain statements.

0098-5589/84/0700-0352$01.00



1982 ACM Distinguished Dissertation

There have been workshops

- Workshop on Algorithmic and Automatic Debugging AADEBUG (1993, 1995, 1997, 2000, 2003, 2005)
- International Workshop on Program Debugging IWPD (2010-2017)

… And tools

wala.sourceforge.net/wiki/index.php/Main_Page

koenig

Log in / create account

article | discussion | view source | history

Main Page

WALA

T. J. WATSON LIBRARIES FOR ANALYSIS

project information
- Main Page
- SourceForge Page

resources
- User Guide
- Tutorials
- Getting Started
- Javadoc
- Publications
- Recent Changes

developer information
- Contributing
- Support
- Mailing lists
- Search list archives
- Regression tests
- Browse repository
- Current Release

search

Go | Search

toolbox
- What links here
- Related changes
- Upload file
- Special pages

Contents [hide]

Welcome to the T.J. Watson Libraries for Analysis (WALA)

The T. J. Watson Libraries for Analysis (WALA) provide static analysis capabilities for Java bytecode and related languages and for JavaScript. The system is licensed under the Eclipse Public License, which has been approved by the OSI (Open Source Initiative) as a fully certified open source license. The initial WALA infrastructure was independently developed as part of the DOMO research project at the IBM T.J. Watson Research Center. In 2006, IBM donated the software to the community.

For recent updates on WALA, join the mailing list, or follow WALA on Twitter or Google+.

Core WALA Features

WALA features include:

- Java type system and class hierarchy analysis
- Source language framework supporting Java and JavaScript
- Interprocedural dataflow analysis (RHS solver)
- Context-sensitive tabulation-based slicer

jslice.sourceforge.net

koenig

JSlice

Latest version: 2.0 (April 15, 2008)

SOURCEFORGE.net

JSlice is dynamic slicing tool for Java programs. It collects and analyzes an execution trace (for slicing) in a compressed form.

- **What is dynamic slicing?**

  Dynamic slicing is a technique for program debugging and understanding. Given a program P, the programmer provides a slicing criterion of the form (I, L, V), where I is a program input, L is a set of some statement instances during execution of program P with input I, and V is a set of variables referenced by L. The purpose of slicing is to find out statements in P which have affected the values of V at L during execution, via dynamic control or data dependencies. So, if during program execution, the values of V at L were "unexpected", the corresponding slice can be inspected to explain the reason for the unexpected values. More on dynamic slicing can be found in research paper [2].

- **Why collect, and analyze an execution trace in compressed form?**

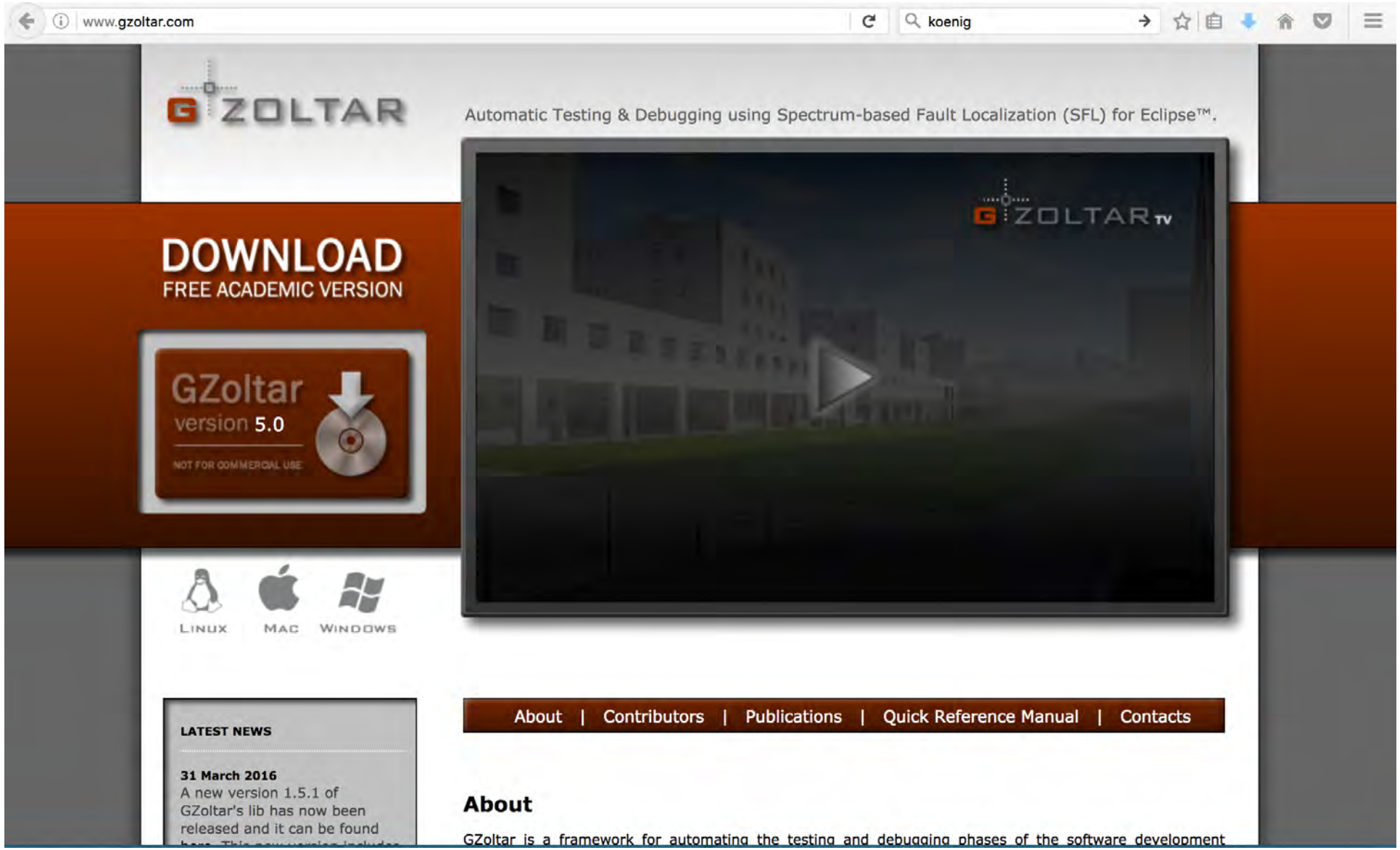
  Dynamic slicing is performed on an execution trace by detecting dynamic control and data dependencies. However, the size of an execution trace may be huge. Consequently, it is important to compactly represent the execution trace and perform program analysis (e.g. dynamic slicing) over the compact representation.

- **How does the compression scheme work in the tool?**

  The compactness of the trace representation is owning to several factors. First, bytecodes which do not correspond to memory access or control transfer are not traced. Second, the sequences of addresses used by memory access or control transfer bytecodes are stored separately. Since these sequences typically have high repetition of patterns, we exploit such repetition to save space. We modify a well-known lossless data compression algorithm called SEQUITUR [1] for this purpose. More technical details about this tool have been discussed in our paper [3].

- **Are there similar tools available?**

  To the best of our knowledge, there is no other dynamic slicing tool available for Java. If you are interested, you can find static slicing tools for C and Java here.


www.gzoltar.com
koenig
GZOLTAR
Automatic Testing & Debugging using Spectrum-based Fault Localization (SFL) for Eclipse™.
DOWNLOAD
FREE ACADEMIC VERSION
GZoltar
version 5.0
NOT FOR COMMERCIAL USE
GZOLTAR TV
LINUX
MAC
WINDOWS
LATEST NEWS
31 March 2016
A new version 1.5.1 of GZoltar's lib has now been released and it can be found
About | Contributors | Publications | Quick Reference Manual | Contacts
About
GZoltar is a framework for automating the testing and debugging phases of the software development

- But there is not so much use of these tools in practice
- Mainly research oriented

- But why?

Why are there not so many tools?

- Adaptation effort
  - “Every programming language is different”
- Maybe they are not so useful
  - Programmers know their code well (during development)
  - Quick response required
  - User interface should be well adapted

But there are niches!

- End user programming
  - Testing & debugging for spreadsheets
- Debugging support for nightly builds
  - Who to blame?
- Debugging support for maintenance
  - Who to blame?
  - Support programmer
- Teaching programming ako Tutoring Systems