Formal Specification can help with code update, refactoring, etc.

The programs proofs have to be maintained throughout code evolutions:

it ensures non-regression of the code.

FIXING CODING ISSUES, LIKE CORRECTING BUFFER OVERFLOWS, ARE REFACTORINGS

Automatic code generation is the solution of the maintainability / performance problem. (It doesn't matter if the generated code is completely unreadable, because you never have to look at it.)

1. Debugging & fault localization is still in the 70s

2. Formal methods has an important role to play

3. The shortage of common benchmarks & infrastructure makes research harder than it has to be.

4 The soup was better than the wrap.

Wireless

VCC or ~~Freeman~~

Access Code: ~~icsme2014~~

~~icsmereg2014~~

CORRECT ACCESS CODE:

icsme2014

1. HAVE CODE ANALYSIS TOOLS FOR THE DISCOVERY OF SECURITY VULNERABILITIES EVOLVED SUFFICIENTLY IN TERMS OF FUNCTIONALITY, FLEXIBILITY AND EASE OF USE TO BECOME MANDATORY PARTS OF THE SOFTWARE CERTIFICATION PROCESS ?

2. WOULD THE WIDESPREAD ADOPTION OF STRONGLY-TYPED LANGUAGES BE A BETTER INVESTMENT IN TERMS OF SOFTWARE SECURITY THAN THE INTEGRATION OF CODE ANALYSIS TOOLS IN THE SOFTWARE DEVELOPMENT PROCESS ?

3. WHY AREN'T THERE MORE SOFTWARE SECURITY PRESENTATIONS AT SCAM ??

Three provocative statements/questions

from Dr. Syrine Tlili (ESII) & José M. Fernandez (POLYTECHNIQUE MONTRÉAL)

1) Have code analysis Tools for the discovery of security vulnerabilities evolved sufficiently in terms of functionality, flexibility and ease of use to become mandatory parts of the software certification process?

2) ~~Is~~ The widespread adoption of strongly-Typed languages a better investment in Terms of software security Than The integration of code analysis Tools in the software development process? (Would)

3) Why aren't There more software security presentations at SCAM?

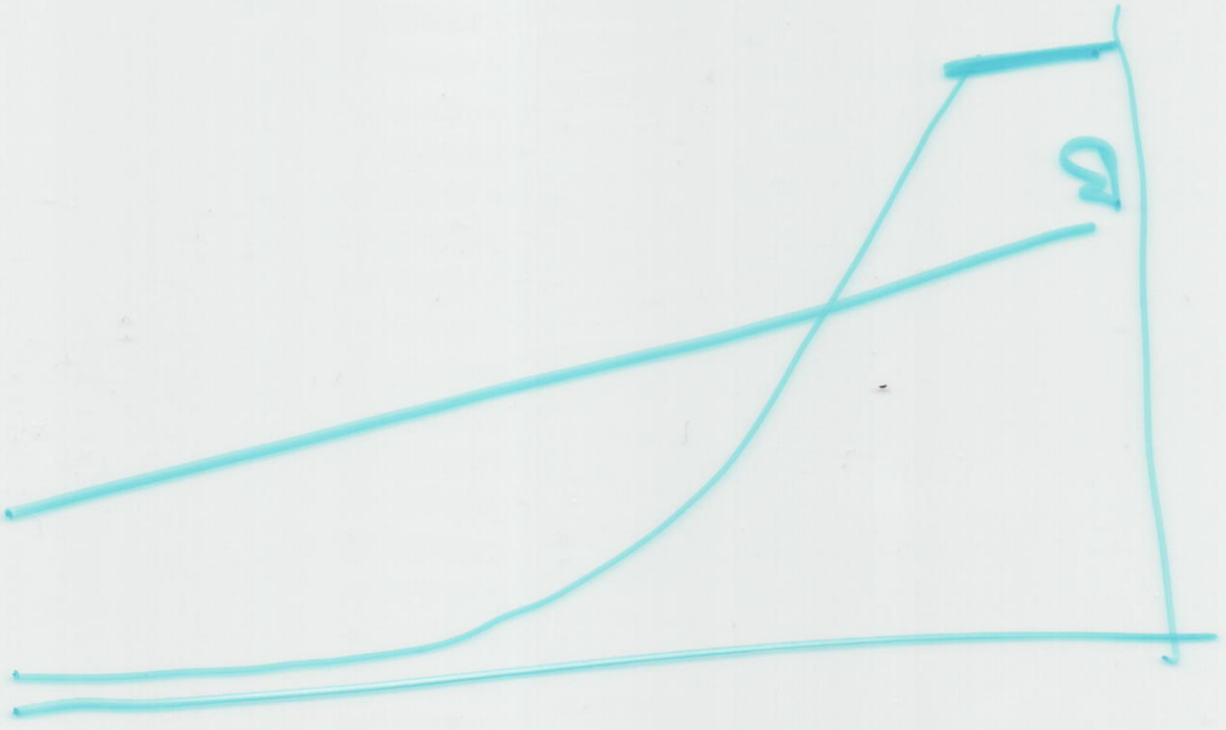
The inaccuracy of forward dynamic slicing is same with the inaccuracy of backward dynamic slicing.

It's time to focus slicing on I/O rather than values in a trace.

slicing SOA
slicing distributed processes

XML Pictures Json
Packets Files
Outputs Messages

Test Suite Augmentation is irrelevant to alleviating the limitation of dynamic analysis that execution set used does not fully represent the behavior of a program.



Practical + effective slicing / static deep analysis is on the horizon!* Rejoice!

* The horizon is an imaginary line in the distance that you can approach but never reach.**

** So is lightweight an option? Can we be more opportunistic + pragmatic in our research?

you name it (V)ORBS can slice it.

name ~ semantics

~ registration

(V)ORBS solves the undefinable

People working on dynamic slicing are too attached to the source code. Dynamic slicing is fundamentally a dynamic problem.

ICSME 2015

31st International Conference on Software Maintenance and Evolution
Bremen, Germany
Sep 29 – Oct 1, 2015
2015.icsme.org

The IEEE International Conference on Software Maintenance and Evolution is a renowned forum for researchers and practitioners from academia and industry to present and discuss the most recent innovations, trends, experiences, and challenges in software maintenance and evolution.

Topics of interest

- Maintenance and evolution processes
- Reverse engineering and re-engineering
- Software refactoring and restructuring
- Software migration and renovation
- Software and system comprehension
- Software repository analysis and mining
- Code cloning and provenance
- Concept and feature location
- Change and defect management
- Evolution of non-code artefacts
- Software testing
- Software quality assessment
- Run-time evolution and dynamic configuration
- Human aspects of software evolution

Research Track
Abstract submission: March 27, 2015
Paper submission: April 1, 2015
Notification: June 15, 2015

Other Tracks
Early Research Achievements
Industry Experience
Tool Demos
Doctoral Symposium

Conference
Sep 29 – Oct 1, 2015

General Chair
Rainer Koschke, *University of Bremen*

Program Chairs
Jens Krinke, *University College London*
Martin Robillard, *McGill University*

ICSME 2015 Call for Papers – Research Papers

We invite high quality submissions describing significant, original, and unpublished results. We solicit submissions relating to all aspects of software maintenance and evolution.

ICSME is a selective conference, but welcomes innovative ideas that are well presented and timely even if the findings are preliminary. All submissions must position themselves within the existing literature, describe the relevance of the results to specific software engineering goals, and include a clear motivation and presentation of the work.

To establish a consistent set of expectations in the review process, the authors are asked, as part of the online submission process, to identify their papers with one or more of the following categories: Analytical, Empirical, Technological, Methodological, Perspectives.

All papers are full papers, and papers may belong to more than one category. Note that papers from any research area can fall into any of these categories, as the categories are constructed surrounding methodological approaches, not research topics.

Evaluation

All submissions that meet the criteria and fit the scope of the conference will be reviewed by three members of the Program Committee. Submissions will be evaluated on the basis of soundness, importance of contribution, originality, quality of presentation, and appropriate comparison to related work.
Submitted papers must comply with IEEE plagiarism policy and procedures. Papers submitted to ICSME 2015 must not have been published elsewhere and must not be under review or submitted for review elsewhere while under consideration for ICSME 2015.

How to Submit

All submitted papers must conform to the IEEE Conference Publishing Services (CPS) formatting instructions. All submissions must be in PDF. Papers must be submitted electronically by the stated deadline. The deadline is firm and not negotiable.

Submissions must be submitted online via EasyChair:
https://www.easychair.org/conferences/?conf=icsme2015.

CO-EVOLUTION

- MAINTAINABILITY

- DEPLOYMENT/DELIVERY PIPELINES

- WHICH SUGGESTIONS SHOULD WE GIVE TO TESTERS?

BREAKING CHANGES MUST BE
FIRST CLASS CITIZENS

CAN WE USE EXISTING MINING ~~REPOs~~ APPROACHES TO IMPROVE CONTINUOUS PIPELINES?

WHAT ARE THE CHALLENGES?

BUG REPORTS SHOULD NEVER BE CLOSED WITHOUT A FIX

BREAKING CHANGES MUST BE FIRST-CLASS CITIZENS

- Devs don't Wanna read details in the commit messages.
- We should not clone exception handling code.
- Are we still using C-code?
- Reused Library is generates a new version and should be pushed back

1. Should the checked exceptions be removed from Java?
Was it a mistake?

2. Who should be responsible for exception handling?
language designers or
software programmers?

Is the parsers for un-preprocessed C programs useful?

It may be difficult to analyze control flows and data flows.

Applications are limited?

Which is a better strategy?

frequent but low cost updates

OR

jumping to the latest only when required

Developers do not like writing descriptive commit messages

so....
the messages should be generated automatically

There are many reasons we go "soundy", but none of the real vulnerabilities are detected by automate tool. So, How far should we go for soundness?

DOES YOUR COMMUNITY FOCUS TOO MUCH ON "BIG-O"?

HAVE WE OVERLOOKED THE AVERAGE CASE AT THE EXPENSE OF THE WORST CASE?

The first work that is

- on demand
- parallel
- use inta information

Can you think on uses?

LINES OF CODE
AND McCabe V(G) COMPLEXITY
METRICS ARE
TOO BORING.

CAN WE COMPUTE THINGS
IN TERMS OF
Object Creation Expressions
Depth of Inheritance Tree
Distinct Receiver Types
etc., etc.?

- Hostile code matters

- Analyzing ("messing with") code that doesn't want to be analyzed ("messed with") can be
  - fun
  - interesting
  - useful

- Existing software analysis techniques are generally of very limited use for reasoning about obfuscated code.

CAN WE USE EXISTING MINING REPOS APPROACHES TO IMPROVE CONTINUOUS PIPELINES? (DEVOPS)

WHAT ARE THE CHALLENGES?

Do we need IPDA?

What are the uses of IPDA?

Developers are not good writers (or lazy?) of descriptive commit messages

so....

the messages should be generated automatically