MSR + ICPC 2015

Nevena Milojković
SCG

MSR 2015

- Keynote: Radu Marinescu
  - we need to bring academia and industry more close; academics need to produce real tools for real projects

- **Code Ownership and Software Quality: A replication Study**
  - ownership = number of commits
  - 90% of ownership is good
  - more contributors = lower quality
    - => developers are not social people :)

- **The Uniqueness of Changes: Characteristics and Applications**
  - some changes are more frequent, and some are unique
  - unique are more error prone
  - they built a recommendation system which recommends changes that the similar code as selected has experienced in the past

- **Co-evolution of infrastructure and Source Code - An Empirical Study**

  - RQs: How many infrastructure files does a system have? How many of them change per month? How large are these changes?

  - Answers: Almost as source code and test files. Median value of 0.28. Comparable to build files.

  - They are tightly coupled with the productions and test changes.

- **Why Power Laws? An Explanation from Fine Grained Code Changes**

  - distribution of many software system measures follow a power law distribution

  - simple small changes produce big distributions

- **Using Developer-Interaction Trails to Triage Change Requests**
  - developed tool iHDev which recommends developer to fix the issue
  - the developers who interacted with the source code relevant to a given change request are most likely to best assist with its resolution.
- **Studying Developers Copy and Paste Behaviour**
  - data of the 20 000 Eclipse users
  - C&P is usually within the same file, and is common across different programming languages
  - need for clone detection techniques that can detect clones across different programming languages

- **Unveiling Exception Handling Bug Hazards in Android based on GitHub and Google Code Issues**

  - goal is to investigate whether from stack trace info one can reveal bug hazards in relation to exception handling code

  - NullPointer is the most common thrown exception.

  - Developers often wrap exceptions.

    - Authors propose to throw explanatory exceptions.

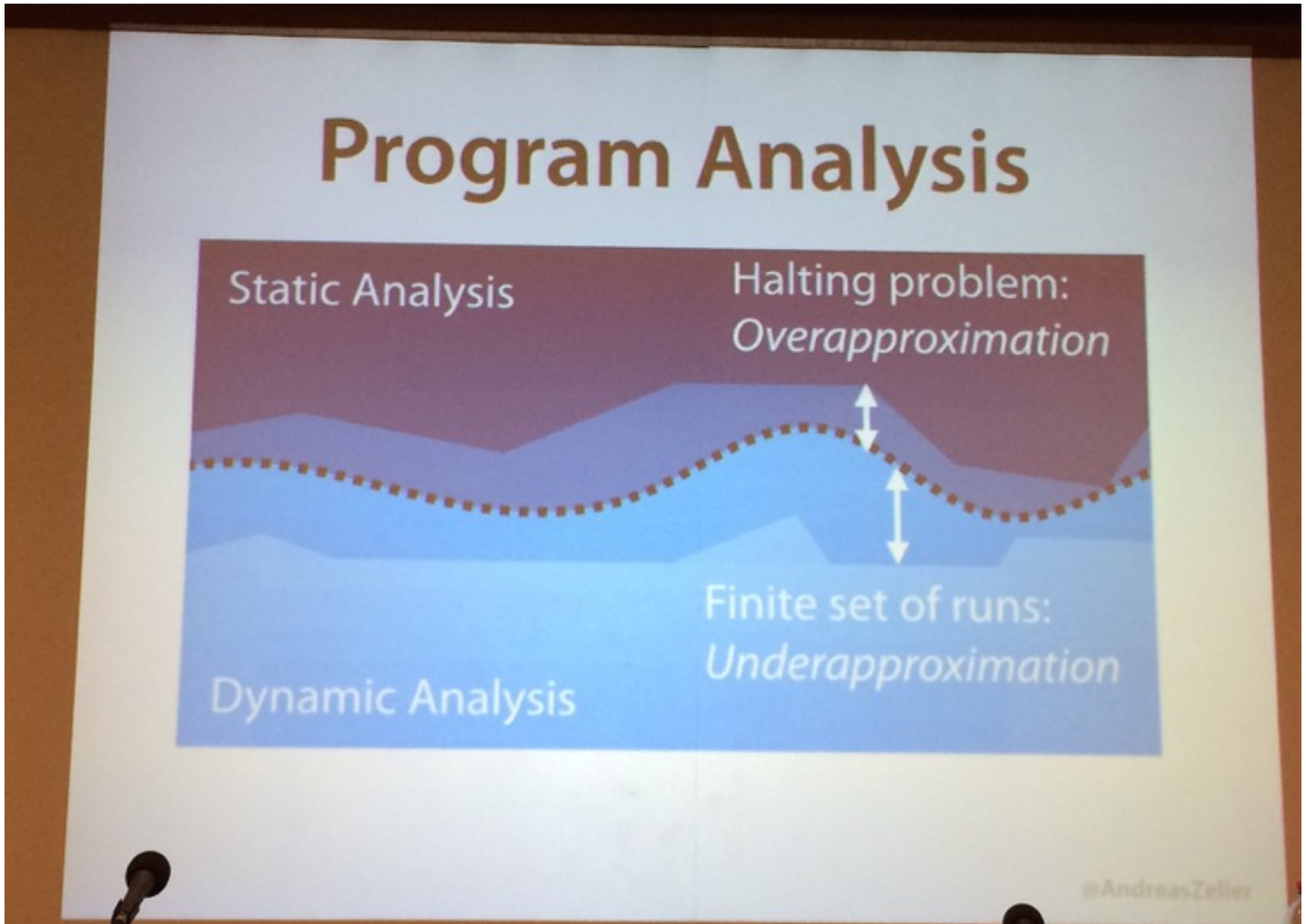
- **Mining StackOverflow to Filter out Off-topic IRC Discussion**

- Filter out chats on IRC, by using Stackoverflow as the positive example, and Youtube comments (top popular) as the negative examples to get rid of junk.

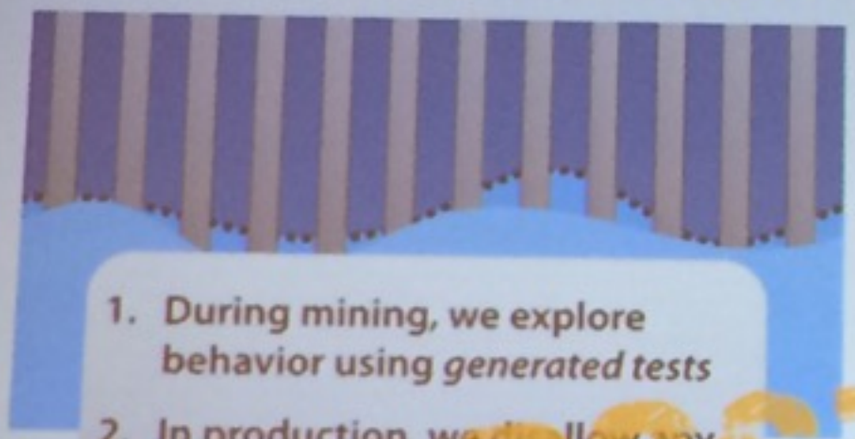
- **An Empirical Study of Architectural Change in Open-Source Software Systems**
  - ARCADE, an automated workbench for software architecture recovery and analysis
  - FOSS versioning scheme is not an accurate indicator of architecture change
  - The package structure is not a complete representation of the system's architecture
- **Are These Bugs Really 'Normal'?**
  - a number of techniques to predict the severity of the bug: "severe", "normal", "minor"
  - normal bug - 80%
  - many normal bug reports are not normal
- **Do Bugs Foreshadow Vulnerabilities? A Study of the Chromium Project (MSR Best paper 2015)**
  - No

ICPC 2015

- Andreas Zeller (keynote speaker)

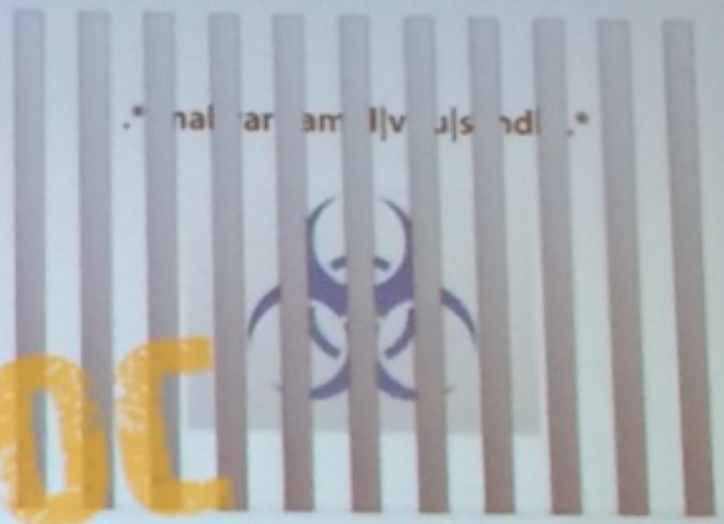


Test Complement Exclusion

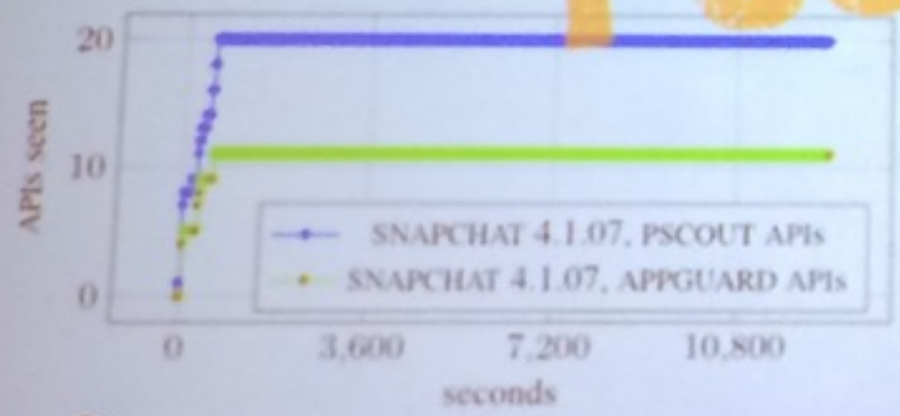


1. During mining, we explore behavior using *generated tests*
2. In production, we disallow any behavior *not seen during testing*

Disclose or Die



Sensitive APIs seen



APIs quickly saturate • Few false alarms

Test Complement Exclusion

prevents *unexpected behavior changes*	prevents *latent malware*	closes *backdoors and exploits*
fully automatic	works on adverse and obscure code	produces *guarantees from testing*

See paper: https://dibt.unimol.it/ICPC15/Keynote.html

- **Discovering Loners and Phantoms in Commit and Issue Data**
  - Link issue to commits.
  - 50% of issue are linked to commits, and 74% of commits are linked to an issue.
  - Loners = missing the link between issue and commits, even though they should be connected
  - Phantoms = one issue linked to more commits.
- **I Know What You Did Last Summer - An Investigation of How Developers Spend Their Time**
  - RQ 1) How much time developers spend understanding? 70%.
  - RQ 2) How much time developers spend in fiddling in the UI? 14%.
  - RQ 3) Correlation between times of leaving the IDE and thinking process? The number of time intervals spent outside the IDE increases the understanding time.

- **Generating Reproducible and Replayable Bug Reports from Android Application Crashes**

  - Bug reports are long, and time-consuming.

  - Authors developed CrashDroid, a tool that reproduce bug reports which include offline workflow and online dataflow.

- **Synonym Suggestion for Tags on Stack Overflow**

  - How to build synonyms for tags?

  - One of the approaches: megaphone:

  - behaior - behaviour, hierarchy - heirarchy

  - covers 50% of results of all strategies

- **“Concise and Consistent Naming” - the most influential paper from IWPC’05.**
  - Comments in German.
  - Theory needs to be put into practice.
- **Fault Localization during System testing**
  - Test and coding teams do not interfere a lot. When test fails, what’s the problem? Or even better where? Tests or code?
  - Created a tool to localise the problem.
  - They managed to identify the faults in the system.
  - Named Tarantula technique.

- **The Plague Doctor: A Promising Cure for the Window Plague (ERA)**
  - Something like AutomnLeaves
  - an automatic interaction profiler that monitors all the fine-grained interactions of the developer with the IDE
- **A Survey of the Forms of Java Reference Names**
  - the majority of names use the components suggested in naming conventions
  - 18% of field names: abbreviation expansion or spell and neologism checking