MINING SOFTWARE DATA TO HELP DEVELOPERS: CHALLENGES AND PERSPECTIVES

Massimiliano Di Penta
University of Sannio, Italy
dipenta@unisannio.it, @maxdipenta

UNIVERSITY OF... WHAT?

FAQ when people met me for the first time at a conference

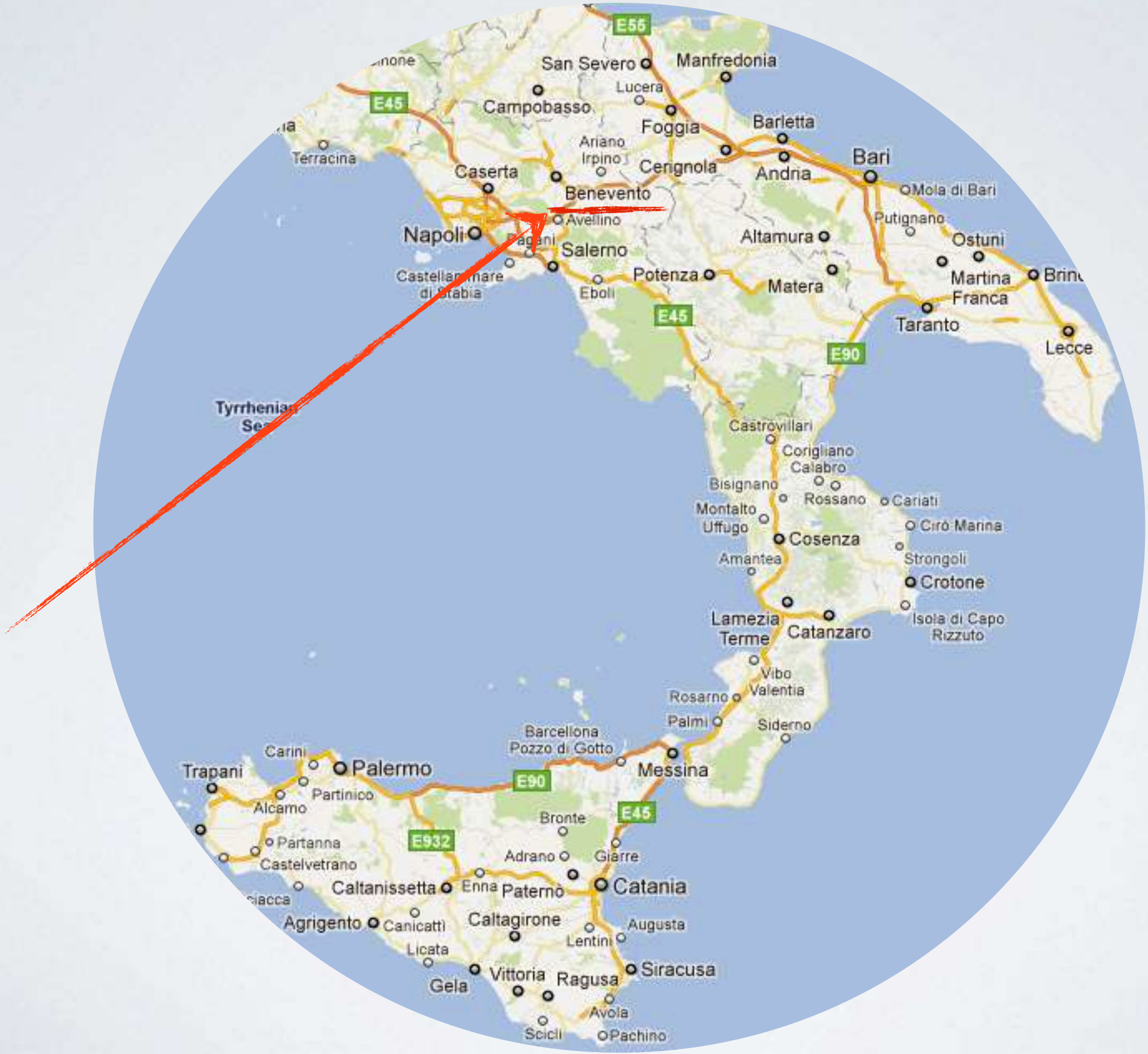
UNIVERSITY OF... WHAT?

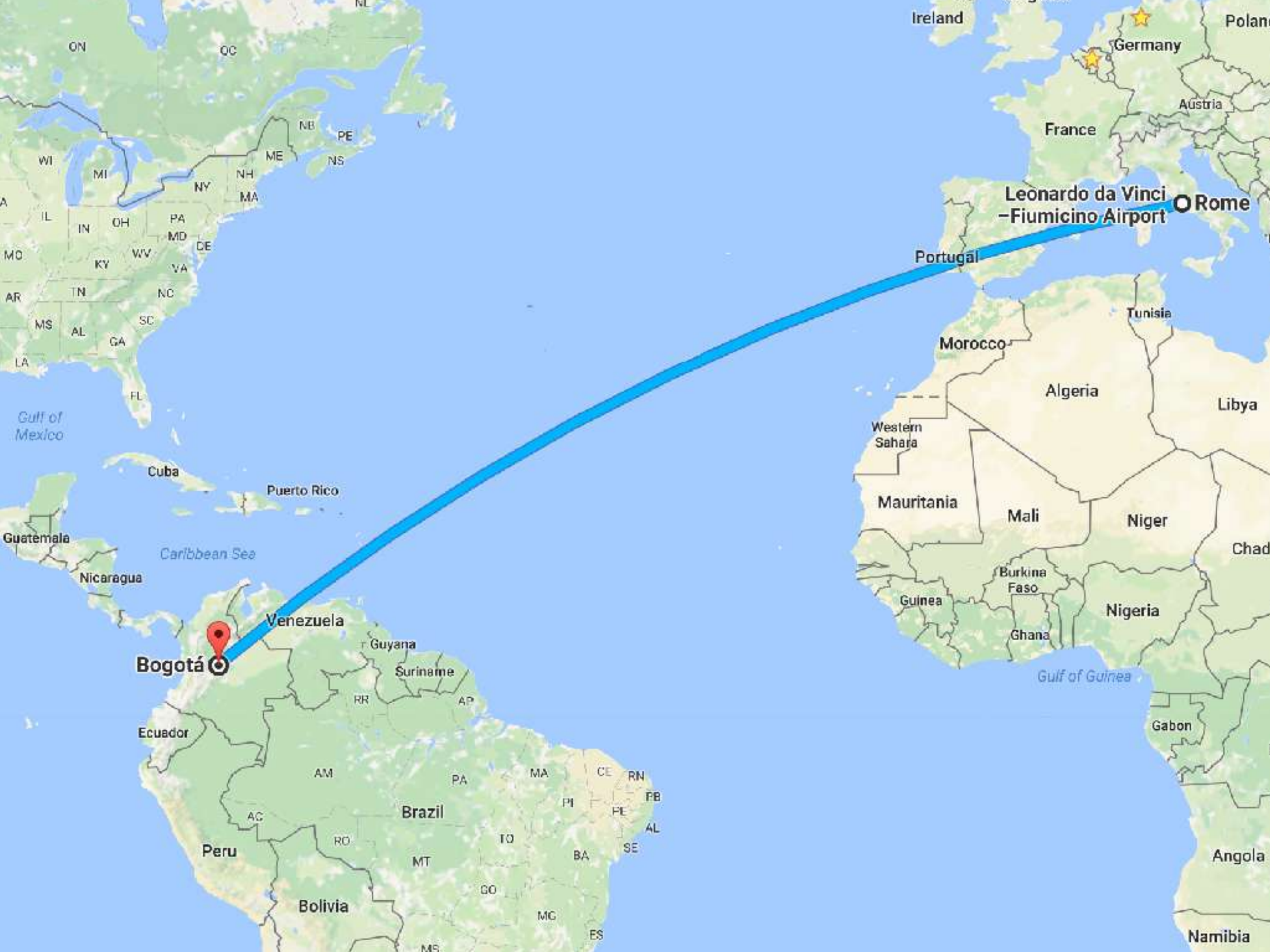
UNIVERSITY OF... WHAT?

UNIVERSITY OF... WHAT?

UNIVERSITY OF... WHAT?

UNIVERSITY OF... WHAT?




Leonardo da Vinci
–Fiumicino Airport
Rome
Bogotá
Ireland
Germany
France
Austria
Portugal
Tunisia
Morocco
Algeria
Libya
Western
Sahara
Mauritania
Mali
Niger
Chad
Burkina
Faso
Guinea
Nigeria
Ghana
Gulf of Guinea
Gabon
Angola
Namibia
Cuba
Puerto Rico
Guatemala
Caribbean Sea
Nicaragua
Venezuela
Guyana
Suriname
Ecuador
Brazil
Peru
Bolivia
Gulf of
Mexico
ON
QC
NL
NB
PE
NS
ME
NH
MA
NY
MI
WI
IL
IN
OH
PA
MD
DE
WV
VA
KY
TN
NC
SC
GA
AL
MS
LA
AR
MO
FL
RR
AP
AM
PA
MA
CE
RN
PB
PE
AL
SE
PI
BA
TO
MT
GO
MG
ES
MS
AC
RO

MAIN RESEARCH INTERESTS

- Software evolution
- Software analytics
- Empirical software engineering
- Continuous Integration
- Software testing

TALK OUTLINE

Introduction to mining software repositories

Recommender systems

Challenges

Key ingredients, summary, takeaways

INTRODUCTION TO MINING SOFTWARE REPOSITORIES

MINING SOFTWARE REPOSITORIES

software complexity

fault prediction

effort estimation

change propagation

change patterns

evolution visualization

Versioning

Emails

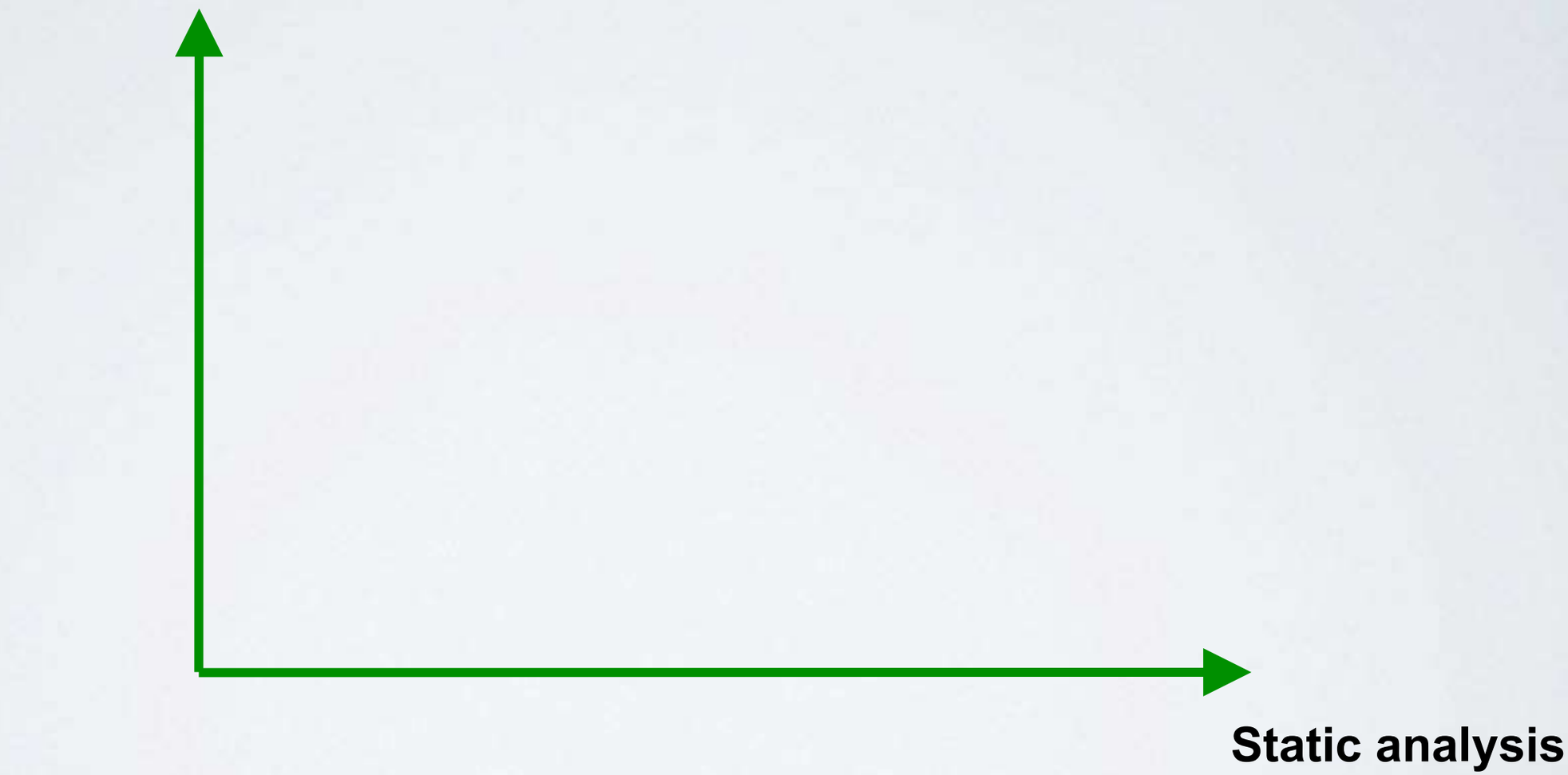
Issue Reports

data extraction

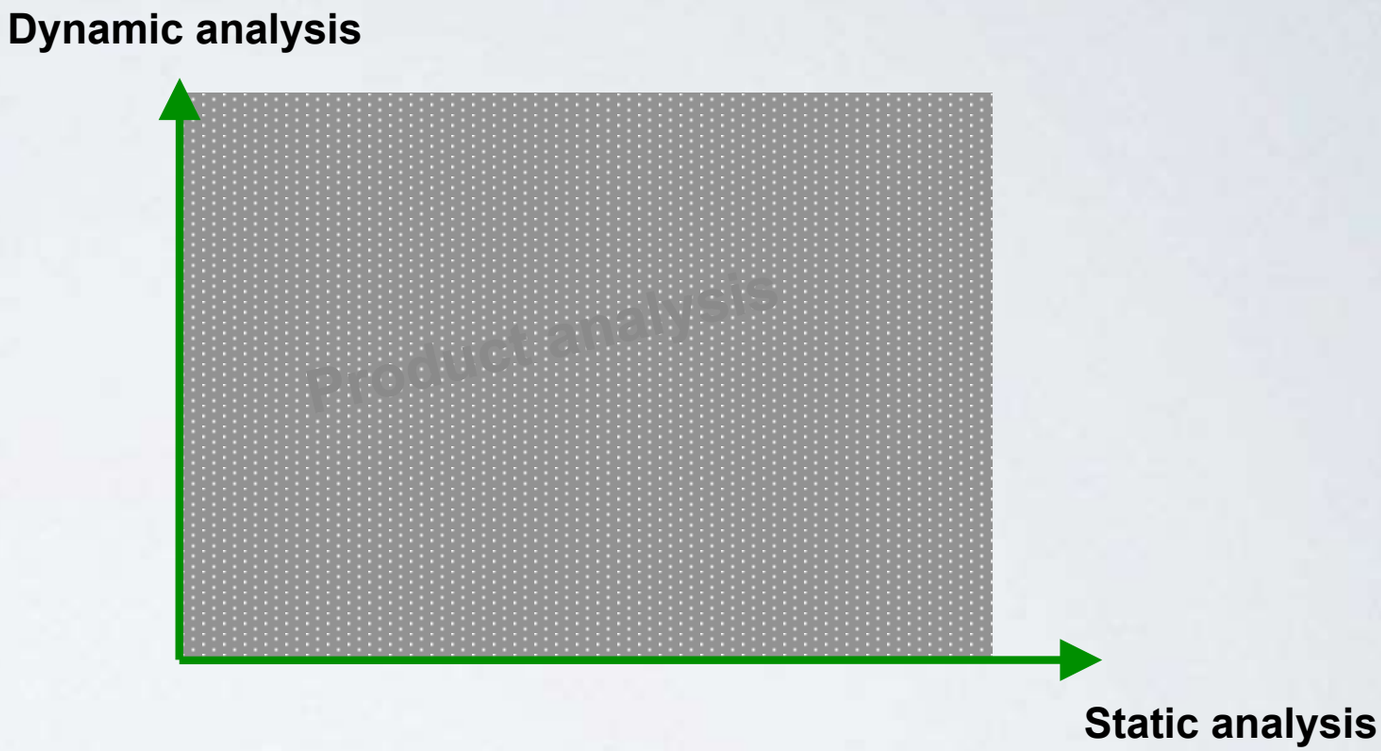
Software History

Knowledge inference
Classification
Association rules
Clustering

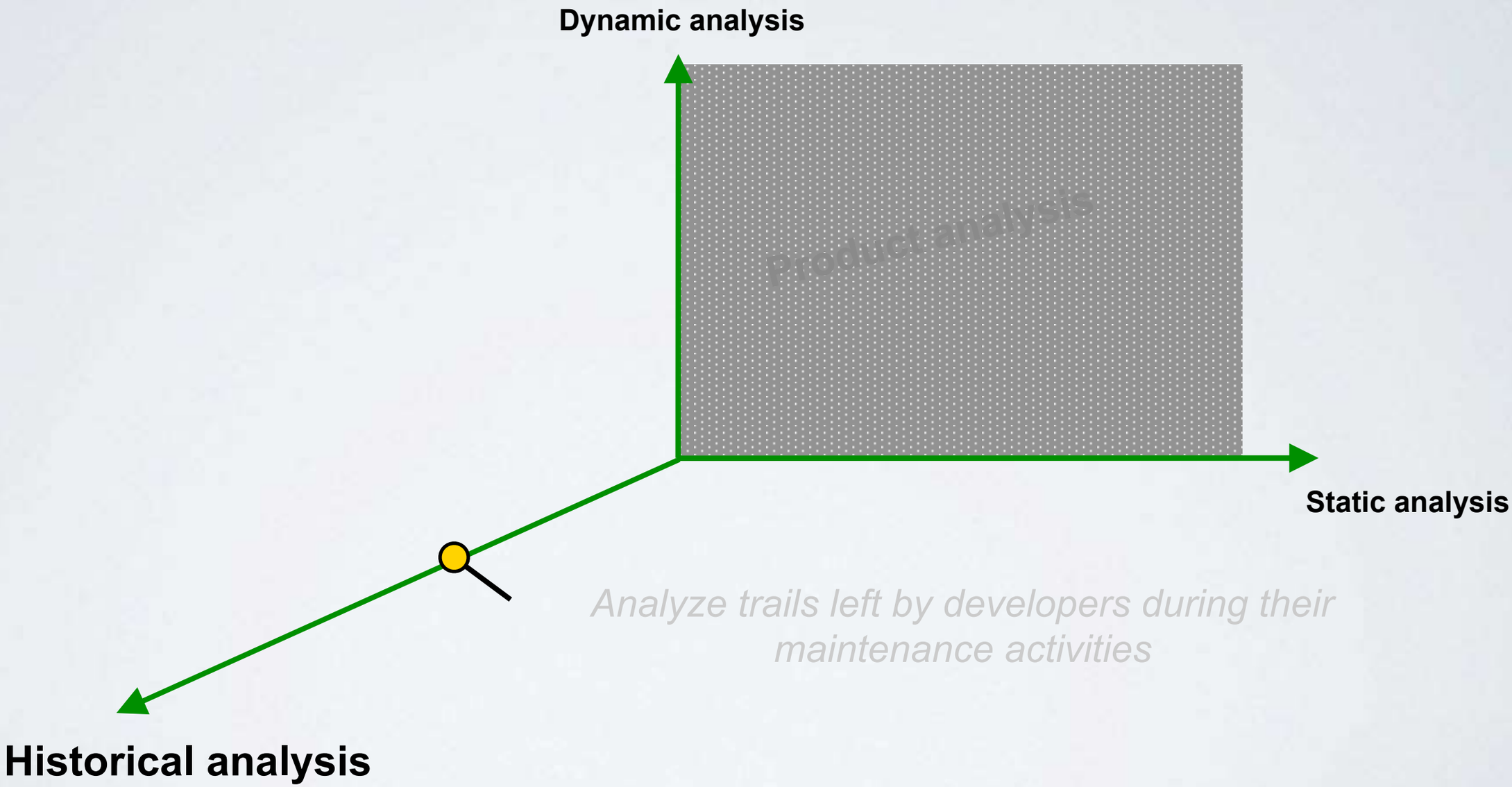
HISTORICAL ANALYSIS



HISTORICAL ANALYSIS

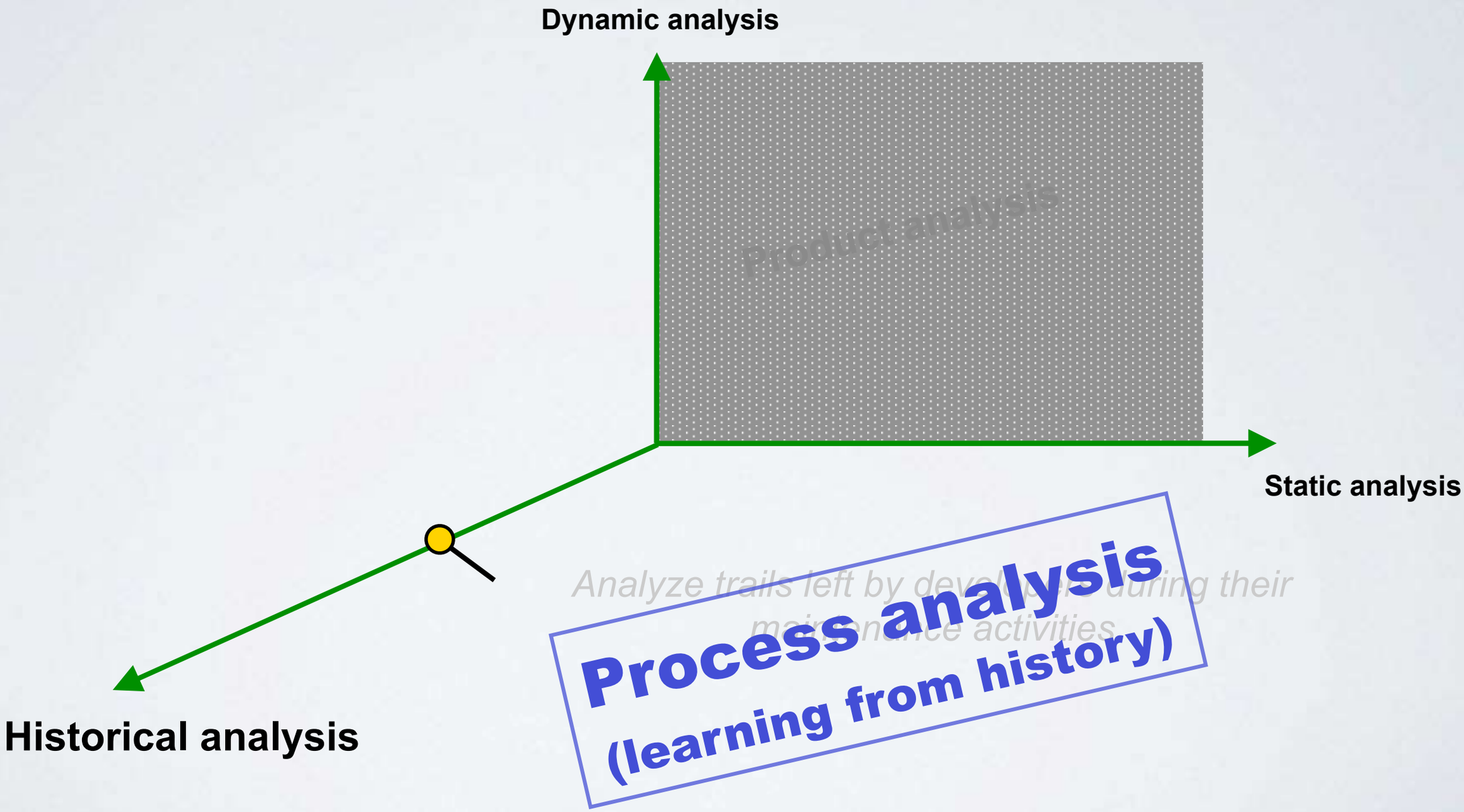


HISTORICAL ANALYSIS



Analyze trails left by developers during their maintenance activities

HISTORICAL ANALYSIS



HISTORICAL ANALYSIS

Static and dynamic analysis do not capture information such as:

How does an artifact change during the time?

When was it changed?

Who changed it?

Why was it changed?

What artifacts changed together?

OK, SO WHAT?

RECOMMENDER SYSTEMS FOR SOFTWARE ENGINEERS

MORE SERIOUSLY…

"A software application that provide information items estimated to be valuable for a software engineering task in a given context"

Martin P. Robillard, Robert J. Walker, Thomas Zimmermann:
Recommendation Systems for Software Engineering. IEEE Software 27(4): 80-86 (2010)

EXAMPLES

CHANGE IMPACT ANALYSIS

Mining Version Histories to Guide Software Changes

Thomas Zimmermann tz@acm.org
Peter Weißgerber weissger@st.cs.uni-sb.de
Stephan Diehl diehl@acm.org
Andreas Zeller zeller@acm.org

Saarland University, Saarbrücken, Germany

Abstract

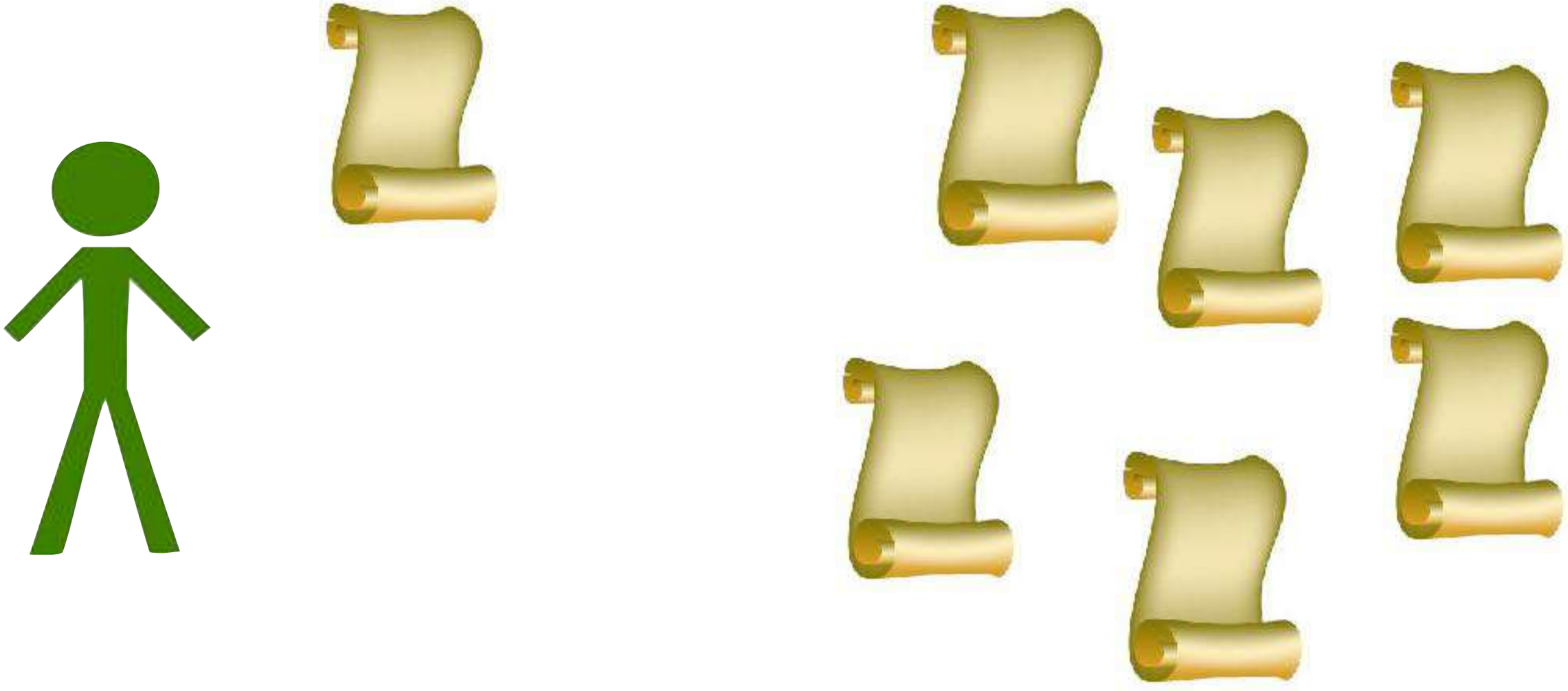
We apply data mining to version histories in order to guide programmers along related changes: "Programmers who changed these functions also changed…". Given a set of existing changes, such rules (a) suggest and predict likely further changes, (b) show up item coupling that is indetectable by program analysis, and (c) prevent errors due to incomplete changes. After an initial change, our ROSE prototype can correctly predict 26% of further files to be changed—and 15% of the precise functions or variables. The topmost three suggestions contain a correct location with a likelihood of 64%.

each time some programmer extended the fKeys[] array, she also extended the function that sets the preference default values. If the programmer now wanted to commit her changes *without* altering the suggested location, ROSE would issue a warning.

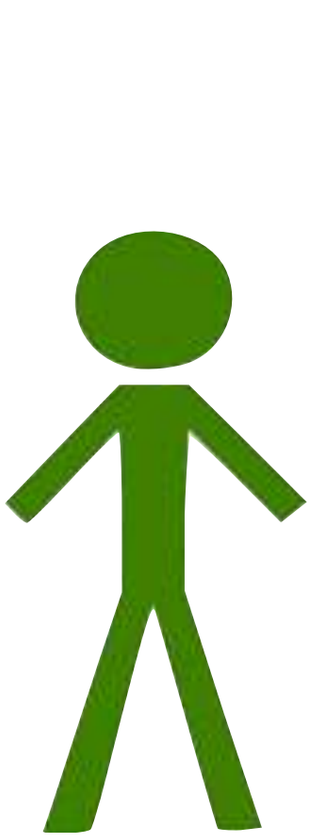
Detect coupling indetectable by program analysis. As ROSE operates uniquely on the version history, it is able to detect coupling between items that cannot be detected by program analysis—including coupling between items that are not even programs. In Figure 1, position 3 on the list is an ECLIPSE HTML documentation file with a confidence of 0.75—suggesting that after adding the new preference, the documentation

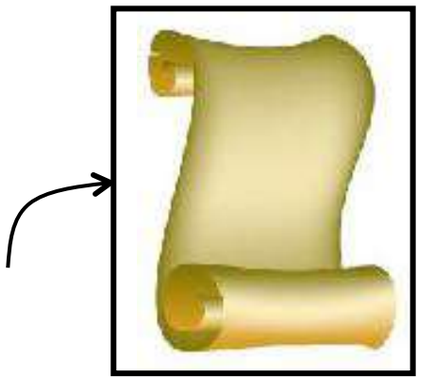
Thomas Zimmermann, Peter Weißgerber, Stephan Diehl, Andreas Zeller: Mining Version Histories to Guide Software Changes. ICSE 2004: 563-572

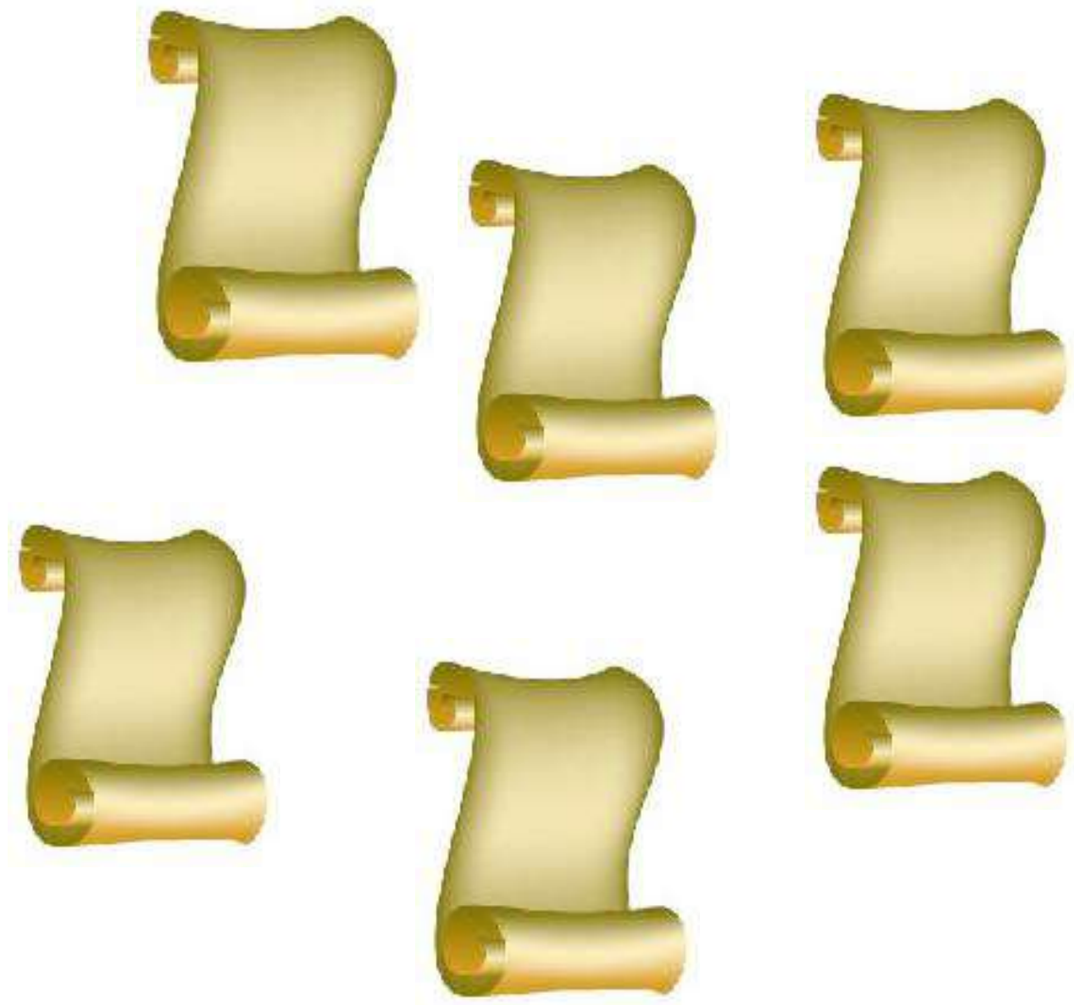
CHANGE IMPACT ANALYSIS



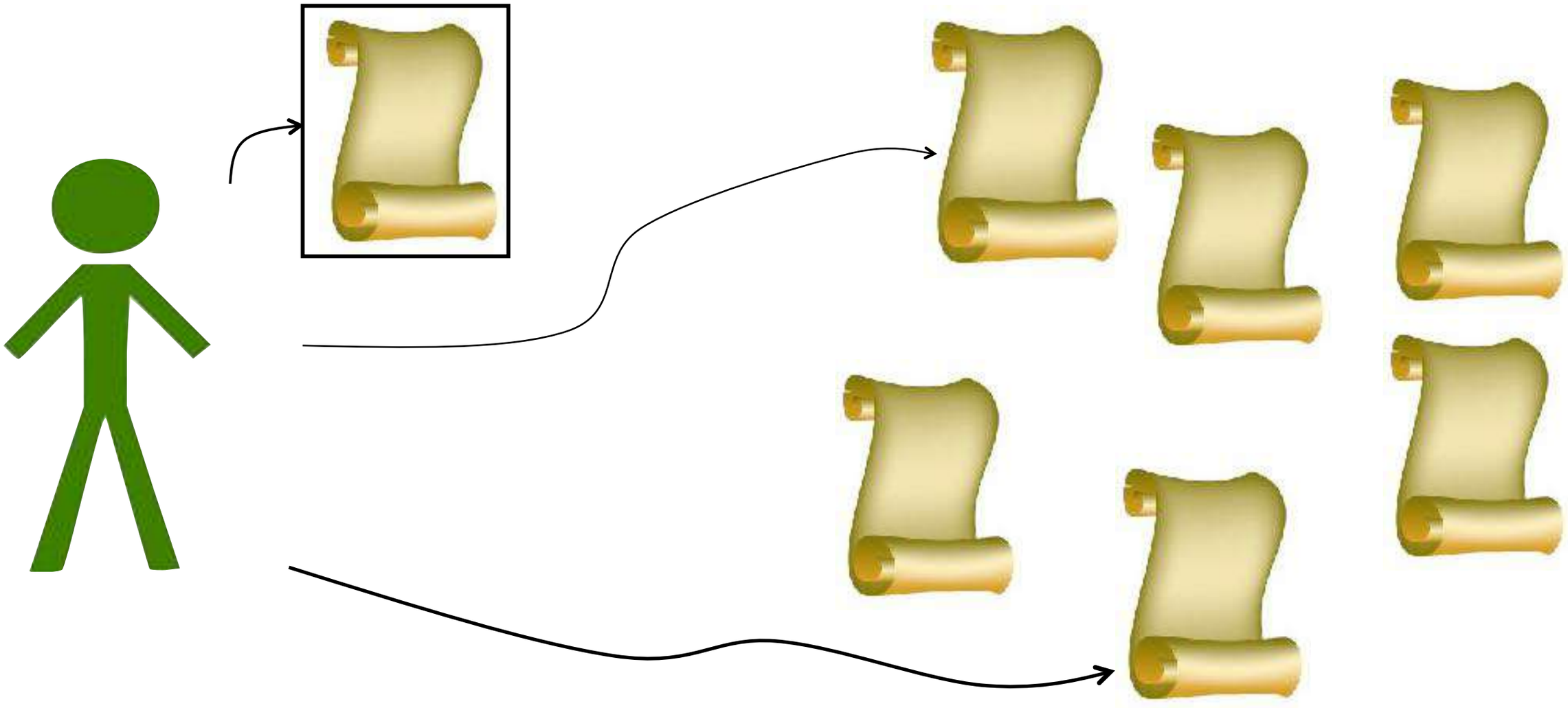
CHANGE IMPACT ANALYSIS







CHANGE IMPACT ANALYSIS



HOW DOES IT WORK?

amazon Try Prime

Books object criented software engineering

Departments Browsing History Massimi ...s Amazon.com Today's Deals Gift Cards & Registry Sell Help EN Hello, Massim Account &

Books Advanced Search New Releases NEW! Amazon Charts Best Sellers & More The New York Times® Best Sellers Children's Books Textbooks Textbook Rentals Sell Us Your Books Best Books of the

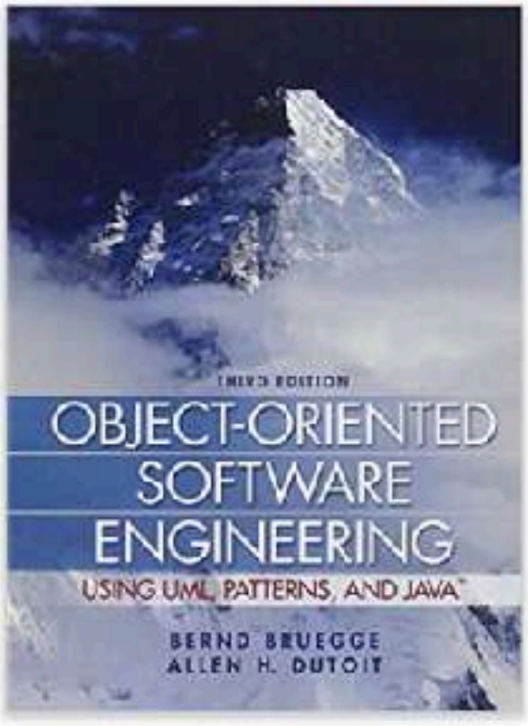
‹ Back to search results for "object oriented software engineering"

Object-Oriented Software Engineering Using UML, Patterns, and Java (3rd Edition) 3rd Edition

by Bernd Bruegge (Author), Allen H. Dutoit (Author)

★★★½☆ 25 customer reviews

Look inside



Kindle	Hardcover	Paperback	Other Sellers
from $29.10	$26.76 - $133.99	$34.99	from $26.76

Rent $26.76

Buy used $55.97

Buy new **$133.99**

In stock.
Usually ships within 2 to 3 days.
Ships from and sold by BRILANTI BOOKS.

This item ships to Campobasso, Italy.

List Price: ~~$190.80~~ Save: $56.81 (30%)

13 New from $91.49

$133.99 + $3.99 shipping

Add to Cart

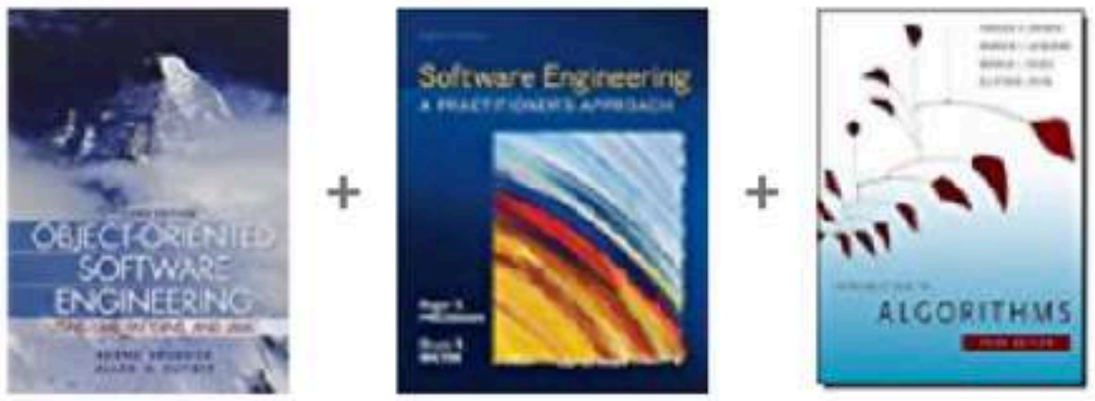
Turn on 1-Click ordering

Ship to:

Massimiliano Di P - 86100

HOW DOES IT WORK?

Frequently bought together



Total price: $326.04

Add all three to Cart

Add all three to List

These items are shipped from and sold by different sellers. Show details

- **This item:** Object-Oriented Software Engineering Using UML, Patterns, and Java (3rd Edition) by Bernd Bruegge Hardcover $133.99
- Software Engineering: A Practitioner's Approach (Irwin Computer Science) by Roger S. Pressman Hardcover $103.98
- Introduction to Algorithms, 3rd Edition (MIT Press) by Thomas H. Cormen Hardcover $88.07

DEFECT PREDICTION










DEFECT PREDICTION










DEFECT PREDICTION










DEFECT PREDICTION










DEFECT PREDICTION










MORE DURING THE TUTORIAL….

BUG TRIAGING

Who Should Fix This Bug?

John Anvik, Lyndon Hiew and Gail C. Murphy
Department of Computer Science
University of British Columbia
{janvik, lyndonh, murphy}@cs.ubc.ca

ABSTRACT

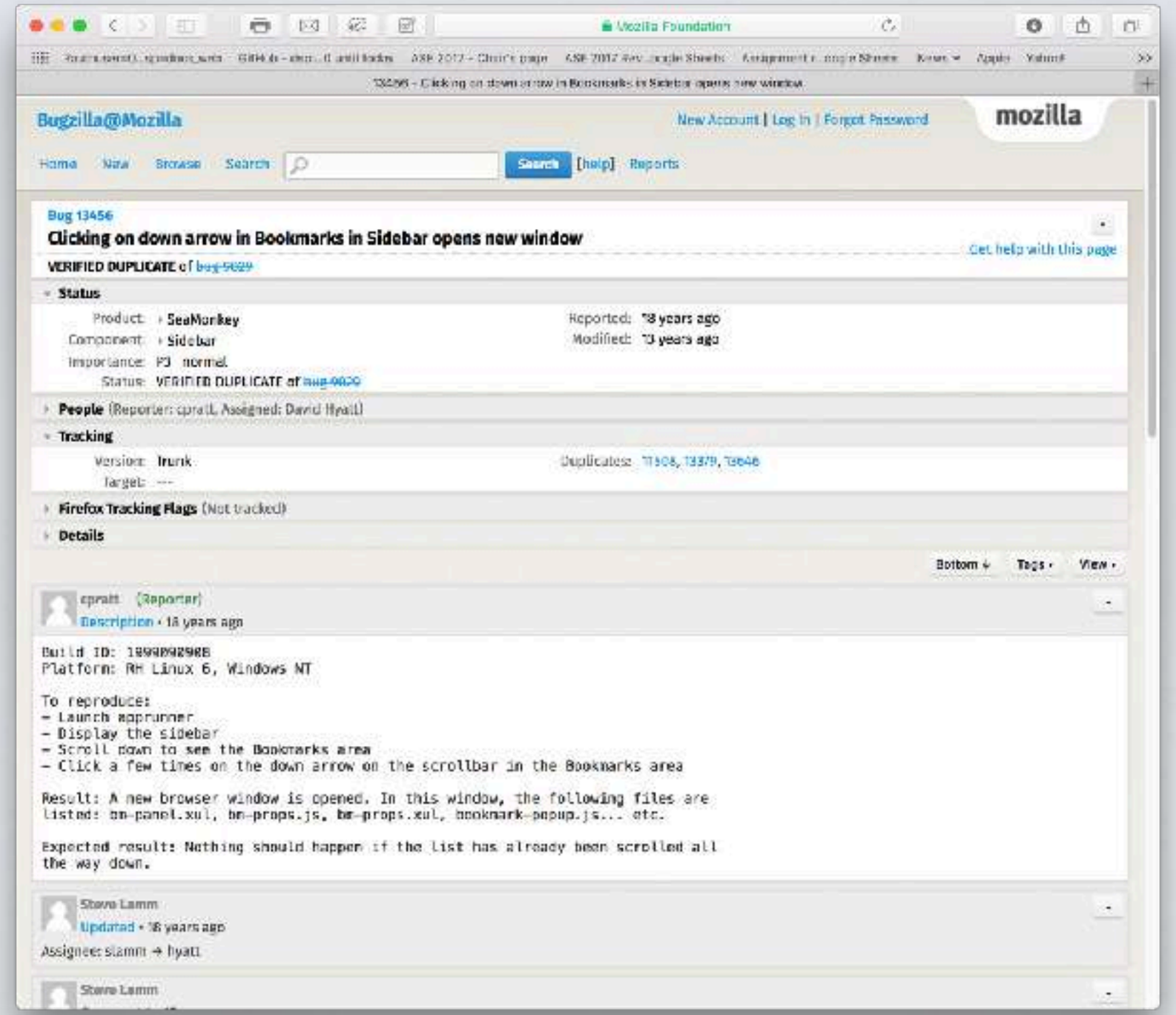
Open source development projects typically support an open bug repository to which both developers and users can report bugs. The reports that appear in this repository must be triaged to determine if the report is one which requires attention and if it is, which developer will be assigned the responsibility of resolving the report. Large open source developments are burdened by the rate at which new bug reports appear in the bug repository. In this paper, we present a semi-automated approach intended to ease one part of this process, the assignment of reports to a developer. Our approach applies a machine learning algorithm to the open bug repository to learn the kinds of reports each developer resolves. When a new report arrives, the classifier produced by the machine learning technique suggests a small number of developers suitable to resolve the report. With this approach, we have reached precision levels of 57% and 64% on the Eclipse and Firefox development projects respectively. We have also applied our approach to the gcc open source development with less positive results. We describe the conditions under which the approach is applicable and also report on the lessons we learned about applying machine learning to repositories used in open source development.

However, this potential advantage also comes with a significant cost. Each bug that is reported must be *triaged* to determine if it describes a meaningful new problem or enhancement, and if it does, it must be assigned to an appropriate developer for further handling [13]. Consider the case of the Eclipse open source project[1] over a four month period (January 1, 2005 to April 30, 2005) when 3426 reports were filed, averaging 29 reports per day. Assuming that a triager takes approximately five minutes to read and handle each report, two person-hours per day is being spent on this activity. If all of these reports led to improvements in the code, this might be an acceptable cost to the project. However, since many of the reports are duplicates of existing reports or are not valid reports, much of this work does not improve the product. For instance, of the 3426 reports for Eclipse, 1190 (36%) were marked either as invalid, a duplicate, a bug that could not be replicated, or one that will not be fixed.

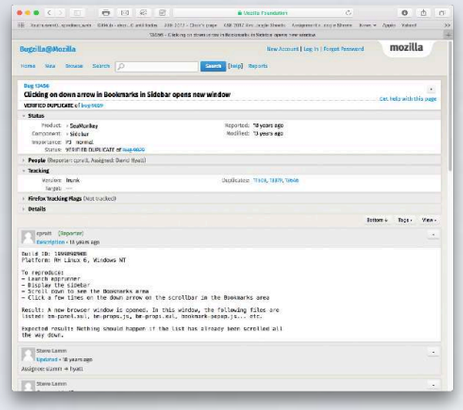
As a means of reducing the time spent triaging, we present an approach for semi-automating one part of the process, the assignment of a developer to a newly received report. Our approach uses a machine learning algorithm to recommend to a triager a set of developers who may be appropriate for resolving the bug. This information can help the triage

John Anvik, Lyndon Hiew, Gail C. Murphy: Who should fix this bug? ICSE 2006: 361-370

BUG TRIAGING



BUG TRIAGING



MAIN IDEA

Assign a bug to available developers who previously fixed similar bugs

AUTOMATED GENERATION OF RELEASE NOTES

ARENA: An Approach for the Automated Generation of Release Notes

Laura Moreno, *Member, IEEE,* Gabriele Bavota, *Member, IEEE,* Massimiliano Di Penta, *Member, IEEE,* Rocco Oliveto, *Member, IEEE,* Andrian Marcus, *Member, IEEE,* Gerardo Canfora

Abstract—Release notes document corrections, enhancements, and, in general, changes that were implemented in a new release of a software project. They are usually created manually and may include hundreds of different items, such as descriptions of new features, bug fixes, structural changes, new or deprecated APIs, and changes to software licenses. Thus, producing them can be a time-consuming and daunting task. This paper describes ARENA (**A**utomatic **RE**lease **N**otes gener**A**tor), an approach for the automatic generation of release notes. ARENA extracts changes from the source code, summarizes them, and integrates them with information from versioning systems and issue trackers. ARENA was designed based on the manual analysis of 990 existing release notes. In order to evaluate the quality of the release notes automatically generated by ARENA, we performed four empirical studies involving a total of 56 participants (48 professional developers and 8 students). The obtained results indicate that the generated release notes are very good approximations of the ones manually produced by developers and often include important information that is missing in the manually created release notes.

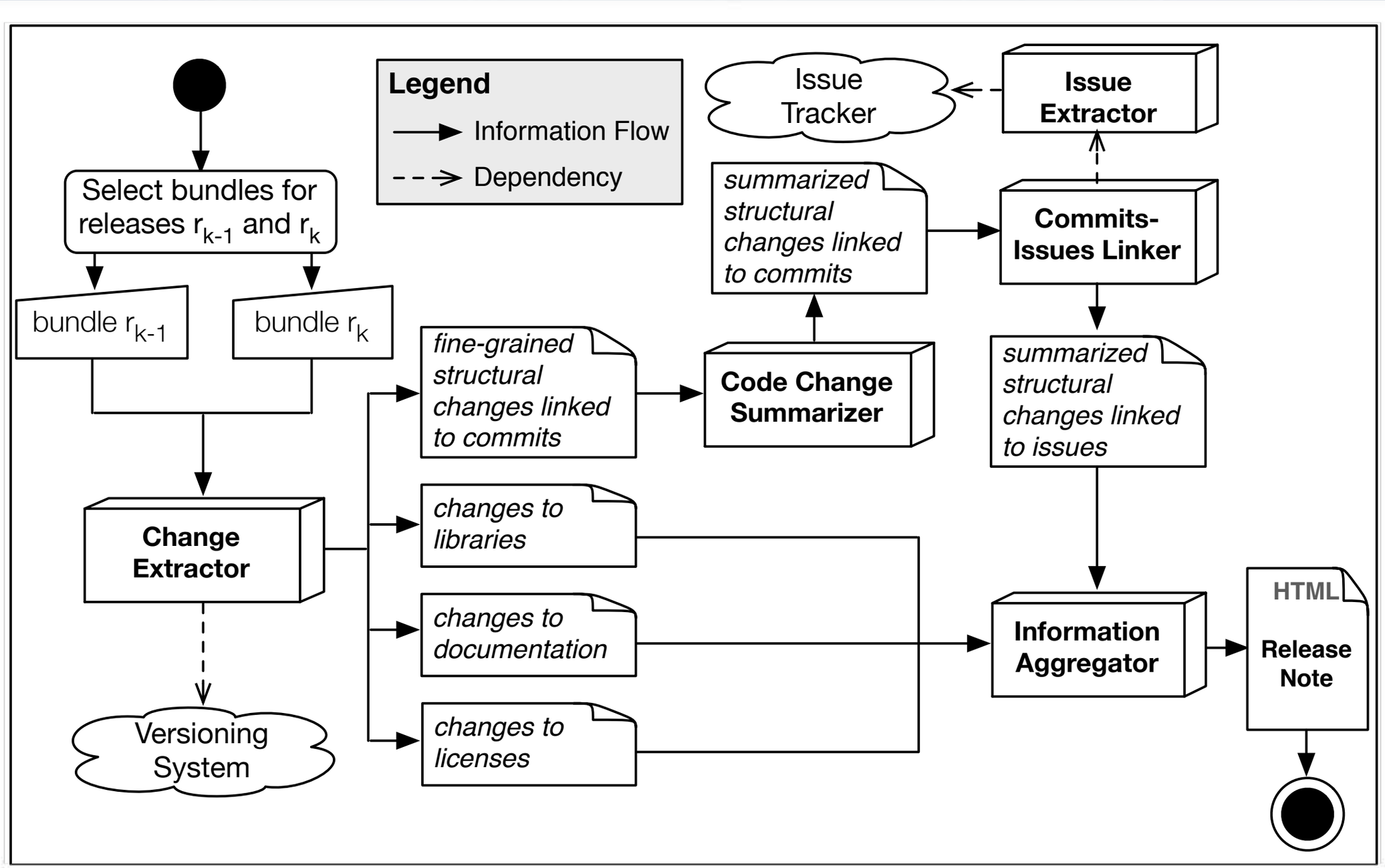
Index Terms—Release notes, Software documentation, Software evolution

✦

Laura Moreno, Gabriele Bavota, Massimiliano Di Penta, Rocco Oliveto, Andrian Marcus, Gerardo Canfora: Automatic generation of release notes. SIGSOFT FSE 2014: 484-495

Laura Moreno, Gabriele Bavota, Massimiliano Di Penta, Rocco Oliveto, Andrian Marcus, Gerardo Canfora: ARENA: An Approach for the Automated Generation of Release Notes. IEEE Transactions on Software Engineering

ARENA (AUTOMATIC RELEASE NOTE GENERATOR)



EXAMPLE OF GENERATED RELEASE NOTE

ARENA

LUCENE 4.0.0

New Features (1)

1. LUCENE-3842: Analyzing Suggester [more info]

Bug Fixes (30)

1. LUCENE-4459: TestWeakIdentityMap.testConcurrentHashMap fails periodically in jenkins [more info]

[...]

12. LUCENE-4364: MMapDirectory makes too many maps for CFS [more info]

    - New abstract class ByteBufferIndexInput extending IndexInput. This entity class includes accessor and mutator methods, and some business logic. It provides access to short, byte buffer index input long, length, file pointer. It allows managing bytes, and byte. It also allows closing byte buffer index input, seeking byte buffer index input, cloning byte buffer index input, and slicing byte buffer index input.
    - [...]

Improvements (17)

1. LUCENE-4448: speedups for AnalyzingSuggester
2. LUCENE-4440: FilterCodec should take a delegate Codec in its ctor [more info]

    - Modified methods forName(String), and availablePostingsFormats() in PostingsFormat.

[...]

RECOMMENDING CODE EXAMPLES

2015 IEEE/ACM 37th IEEE International Conference on Software Engineering

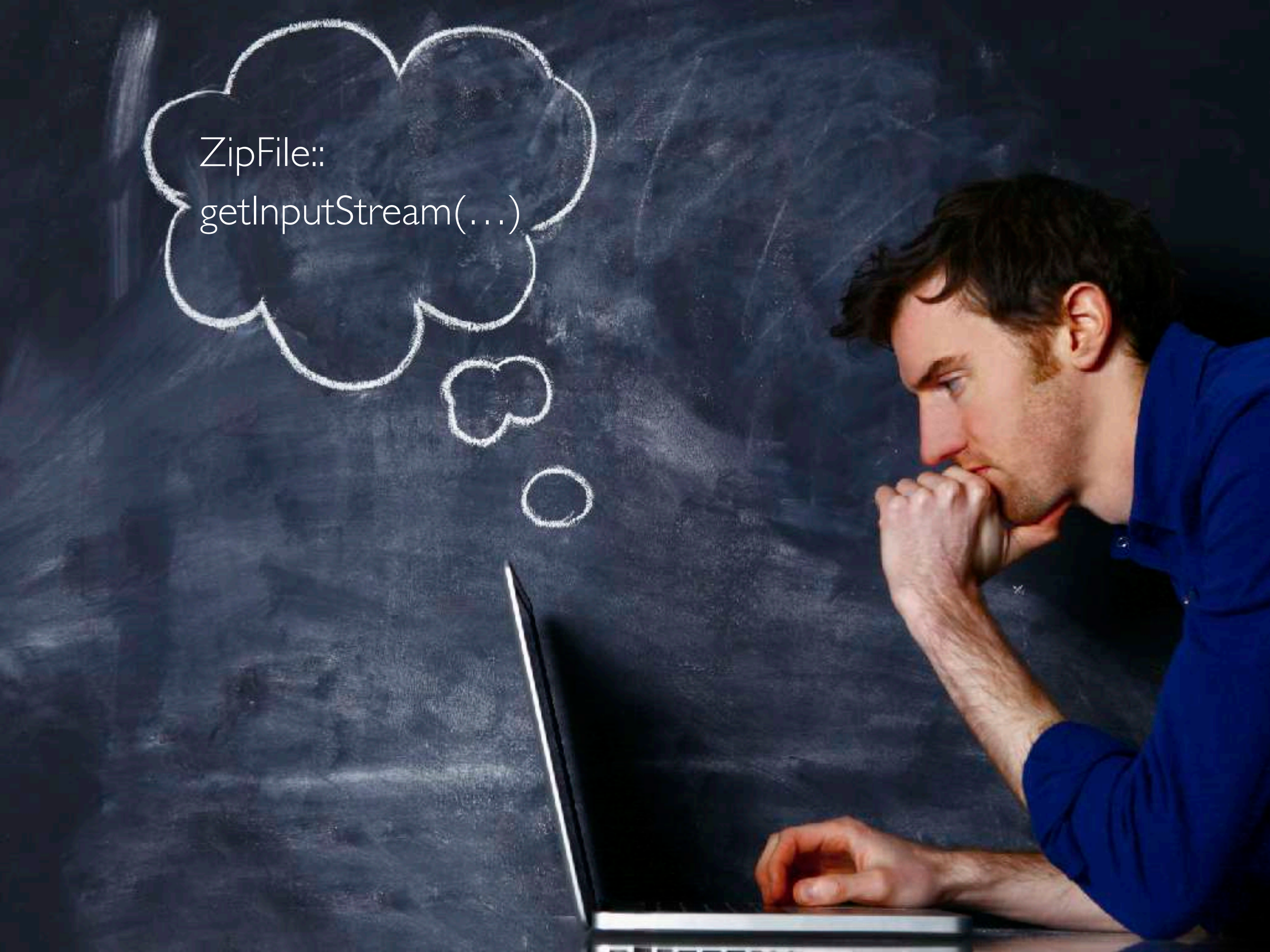
How Can I Use This Method?

Laura Moreno*, Gabriele Bavota†, Massimiliano Di Penta‡, Rocco Oliveto§ and Andrian Marcus*
*The University of Texas at Dallas, USA; †Free University of Bozen-Bolzano, Italy;
‡University of Sannio, Italy; §University of Molise, Italy

***Abstract*—Code examples are small source code fragments whose purpose is to illustrate how a programming language construct, an API, or a specific function/method works. Since code examples are not always available in the software documentation, researchers have proposed techniques to automatically extract them from existing software or to mine them from developer discussions. In this paper we propose MUSE (Method USage Examples), an approach for mining and ranking actual code examples that show how to use a specific method. MUSE combines static slicing (to simplify examples) with clone detection (to group similar examples), and uses heuristics to select and rank the best examples in terms of reusability, understandability, and popularity. MUSE has been empirically evaluated using examples mined from six libraries, by performing three studies involving a total of 140 developers to: (i) evaluate the selection and ranking heuristics, (ii) provide their perception on the usefulness of the selected examples, and (iii) perform specific programming tasks using the MUSE examples. The results indicate that MUSE selects and ranks examples close to how humans do, most of the code examples (82%) are perceived as useful, and they actually help when performing programming tasks.**

concrete method usages would augment abstract code examples and result in better understanding of the method usage. For this reason, we focus on the still open problem of mining relevant concrete code examples for a given method. Specifically, we aim at answering the following question: "*Given a specific method needed to perform a task, what are the necessary steps to use it?*" For instance, once a developer has understood the purpose of an API and has gained an idea of what the various methods do (*e.g.,* through a reference manual), she wants to know what are the typical invocation scenarios for a given method, say `copyInputStreamToFile`. To this aim, she needs to find one or more examples that have the necessary steps to invoke this method, such as, invoking other methods of the API or manipulating the method's parameters. Such a method usage example (see Fig. 1) shows that in order to use the desired method (line 12) two arguments are required (*e.g.,* `zip.getInputStream(entry)` and `file`). The inline comments (lines 8-11) provide information about each

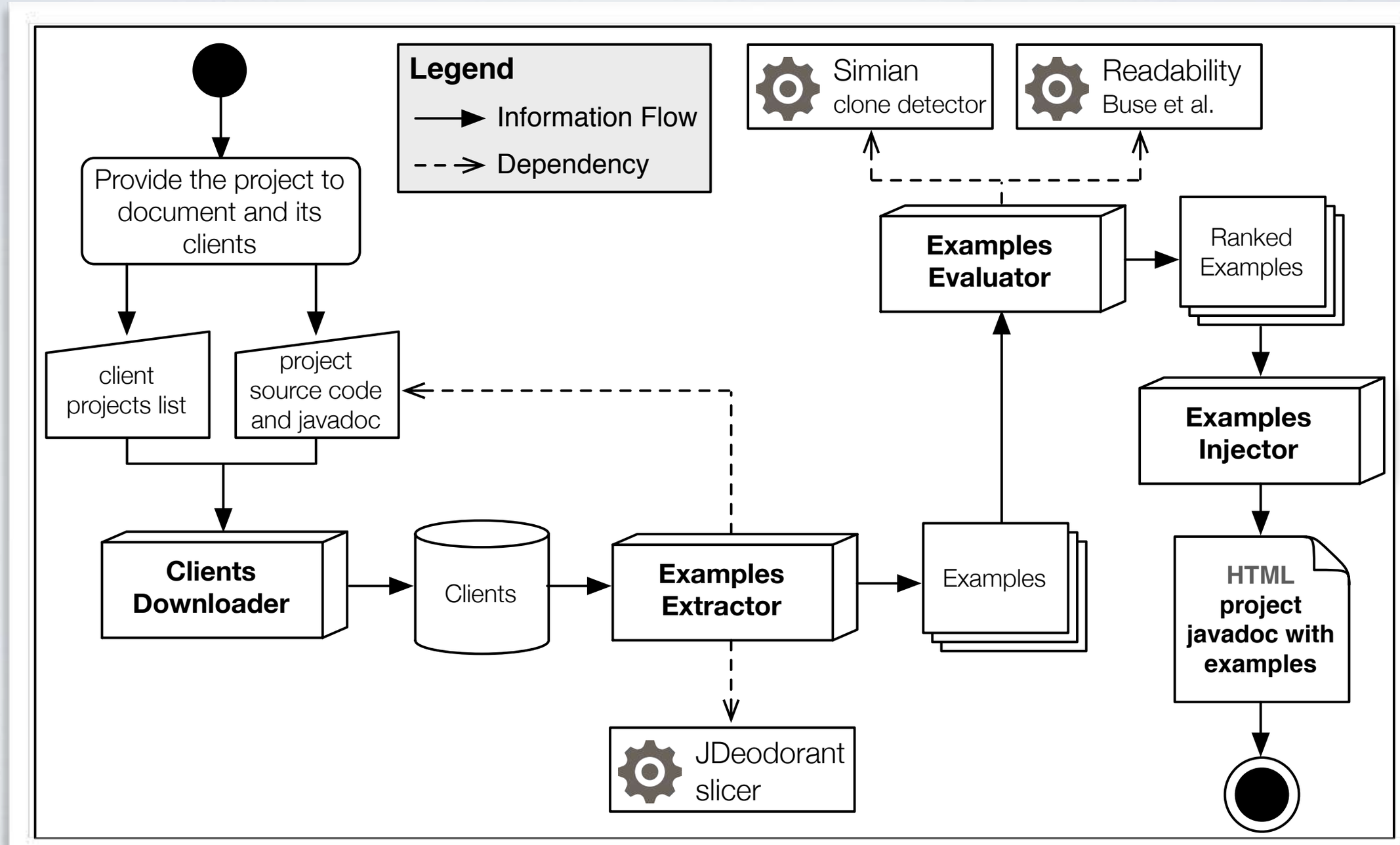
Laura Moreno, Gabriele Bavota, Massimiliano Di Penta, Rocco Oliveto, Andrian Marcus: How Can I Use This Method? ICSE (1) 2015: 880-890


ZipFile::
getInputStream(…)

EXAMPLE

```
01 File source;
02 File target;
03 ZipFile zip=new ZipFile(source);
04 Enumeration<? extends ZipEntry> entries=zip.entries();
05 while(entries.hasMoreElements()) {
06  ZipEntry entry=entries.nextElement();
07  File file=new File(target,entry.getName());
08  //zip.getInputStream(entry)->the InputStream to copy bytes from,
09  //must not be null
10  //file->the non-directory File to write bytes to (possibly
11  //overwriting), must not be null
12  FileUtils.copyInputStreamToFile(zip.getInputStream(entry),file);
13 }
```

MUSE (METHOD USAGE EXAMPLES)



RECOMMENDING RELEVANT STACKOVERFLOW DISCUSSIONS

Mining StackOverflow to Turn the IDE into a Self-Confident Programming Prompter

Luca Ponzanelli[1], Gabriele Bavota[2], Massimiliano Di Penta[2], Rocco Oliveto[3], Michele Lanza[1]
1: REVEAL @ Faculty of Informatics – University of Lugano, Switzerland
2: University of Sannio, Benevento, Italy 3: University of Molise, Pesche (IS), Italy

ABSTRACT

Developers often require knowledge beyond the one they possess, which often boils down to consulting sources of information like Application Programming Interfaces (API) documentation, forums, Q&A websites, *etc.* Knowing what to search for and how is non-trivial, and developers spend time and energy to formulate their problems as queries and to peruse and process the results.

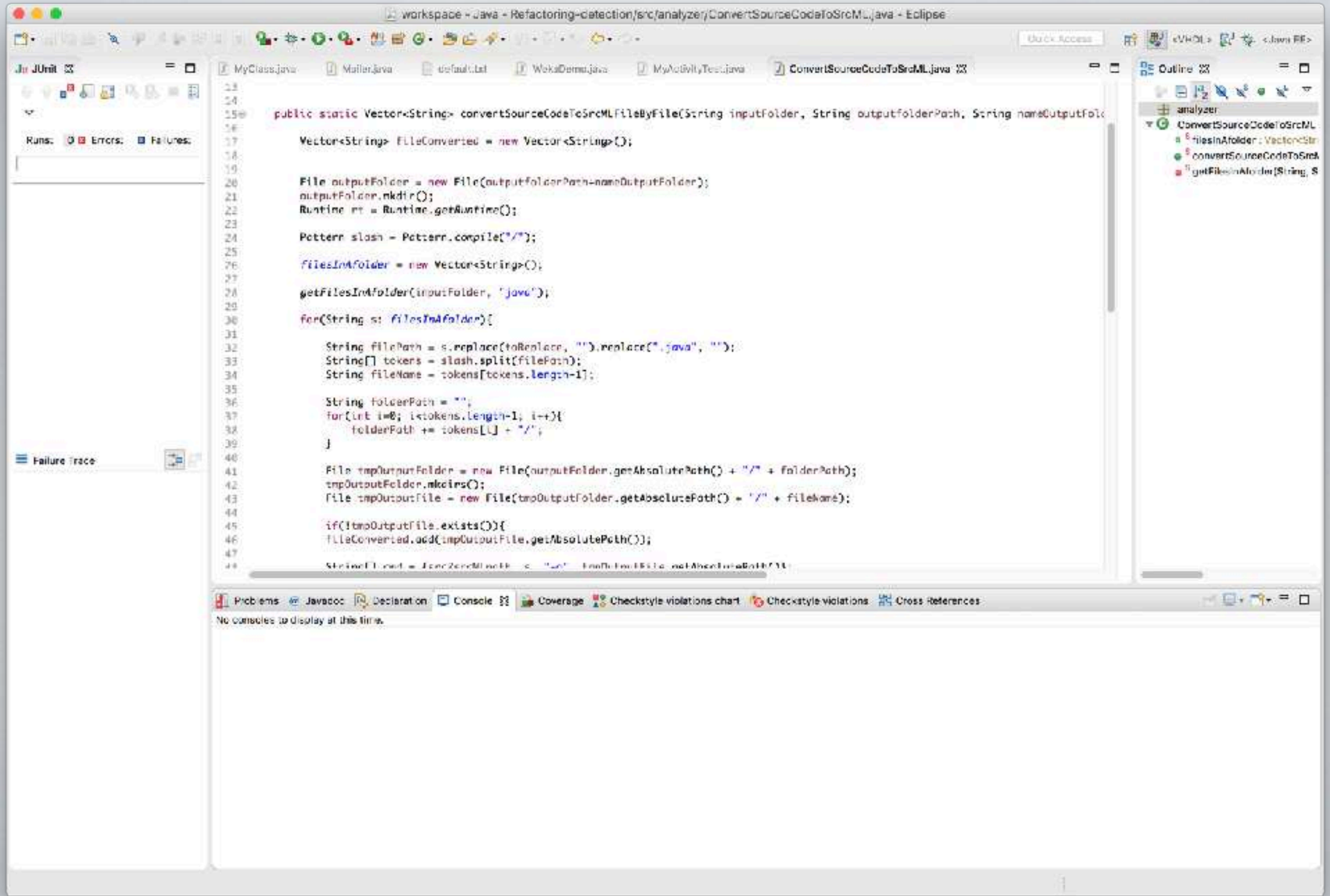
We propose a novel approach that, given a context in the IDE, automatically retrieves pertinent discussions from Stack Overflow, evaluates their relevance, and, if a given confidence threshold is surpassed, notifies the developer about the available help. We have implemented our approach in PROMPTER, an Eclipse plug-in. PROMPTER has been evaluated through two studies. The first was aimed at evaluating the devised ranking model, while the second was conducted to evaluate the usefulness of PROMPTER

problems, the main one being the absence of automation: Every time developers need to look for information, they interrupt their work flow, leave the IDE, and use a Web browser to perform and refine searches, and assess the results. Finally, they transfer the obtained knowledge to the problem context in the IDE. The information is retrieved from different sources, such as forums, mailing lists [2], blogs, Q&A websites, bug trackers [1], *etc.* A prominent example is Stack Overflow, popular among developers as a venue for sharing programming knowledge. Stack Overflow is vast: In 2010 it already had 300k users, and millions of questions, answers, and comments [23]. This makes finding the right piece of information cumbersome and challenging.

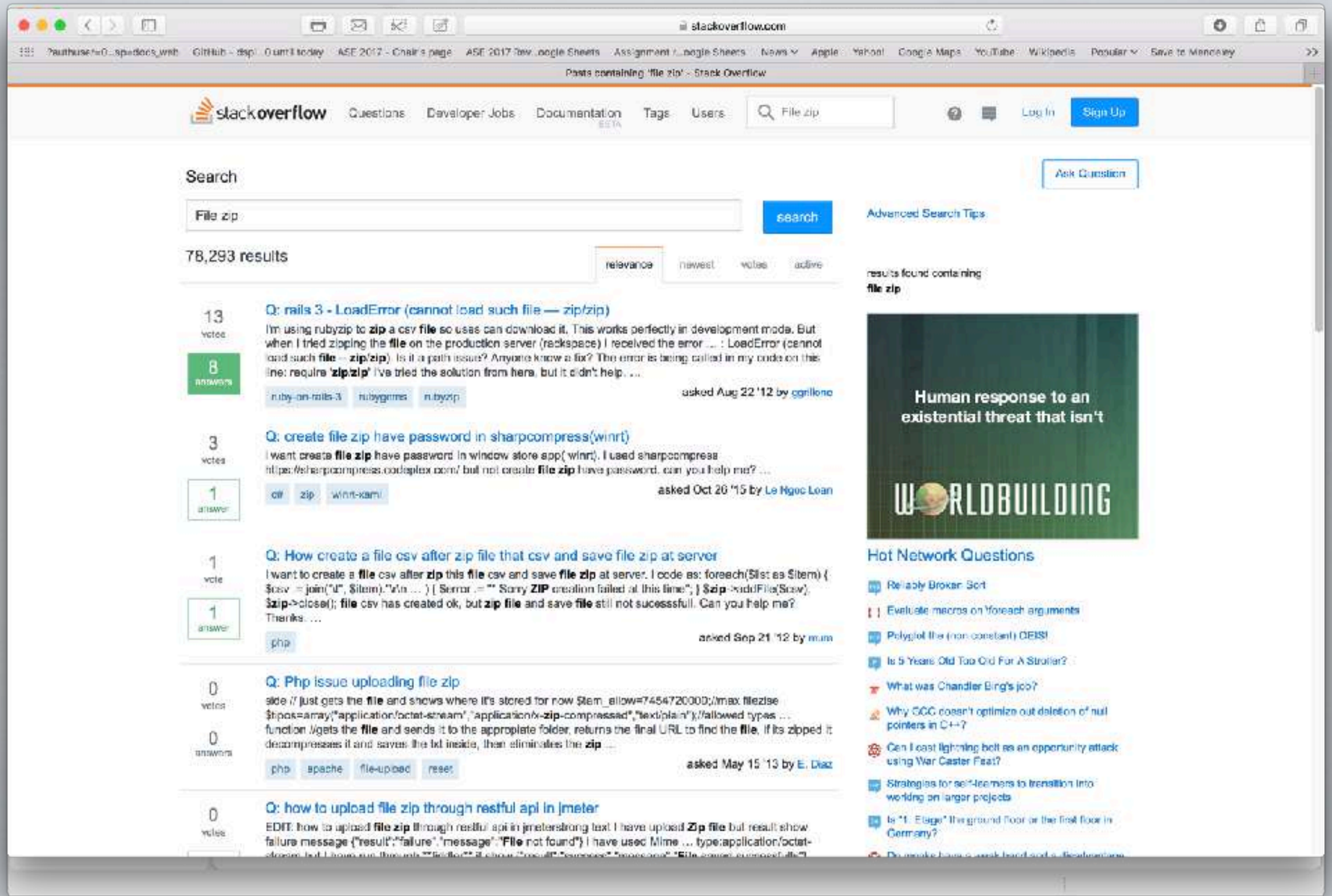
Recommender systems [33] represent a possible solution to this problem. A recommender system gathers and analyzes data, identifies useful artifacts, and suggests them to the developer. Seminal

Luca Ponzanelli, Gabriele Bavota, Massimiliano Di Penta, Rocco Oliveto, Michele Lanza: Prompter - Turning the IDE into a self-confident programming assistant. Empirical Software Engineering 21(5): 2190-2231 (2016)
Luca Ponzanelli, Gabriele Bavota, Massimiliano Di Penta, Rocco Oliveto, Michele Lanza: Mining StackOverflow to turn the IDE into a self-confident programming prompter. MSR 2014: 102-111

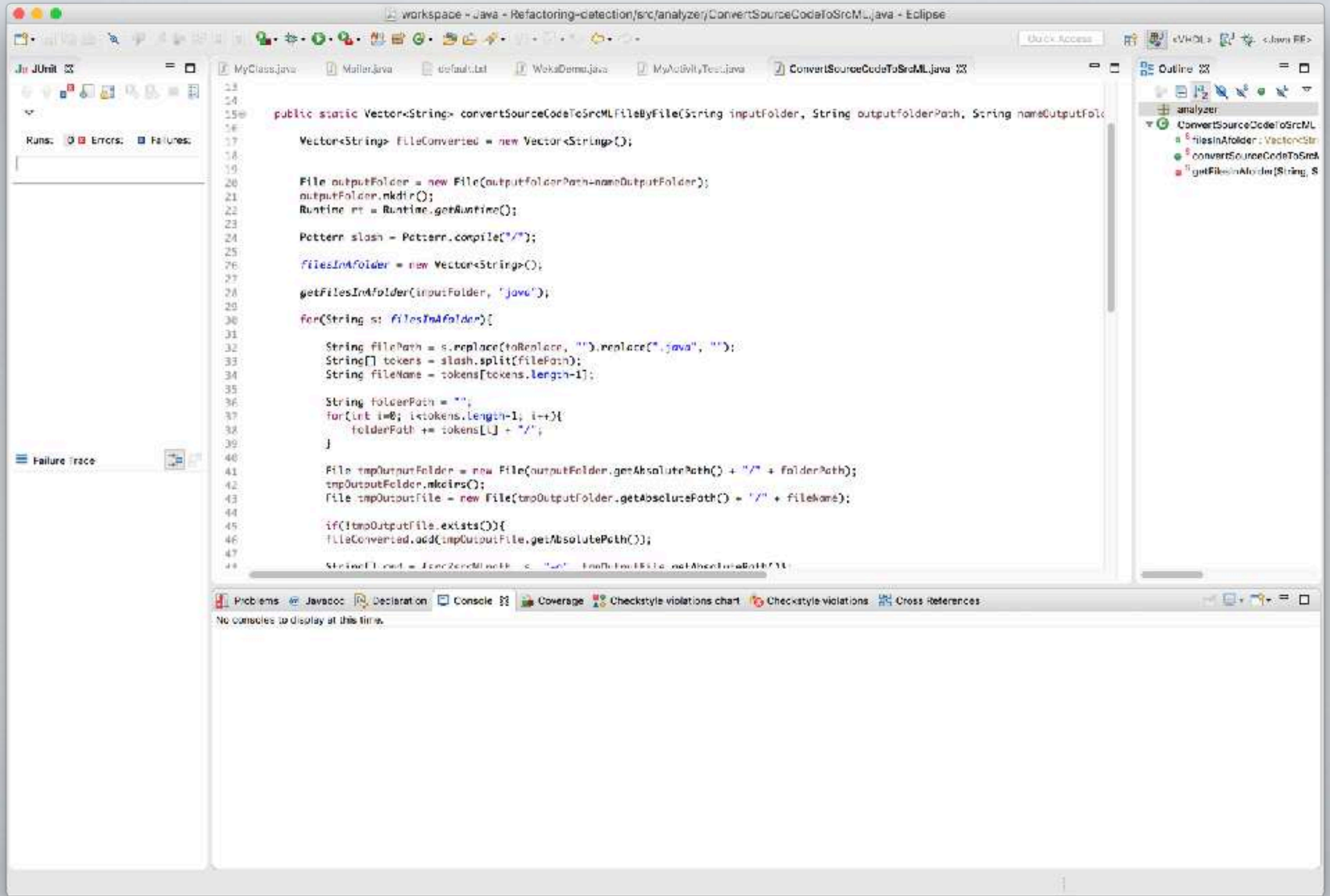
MOTIVATIONS



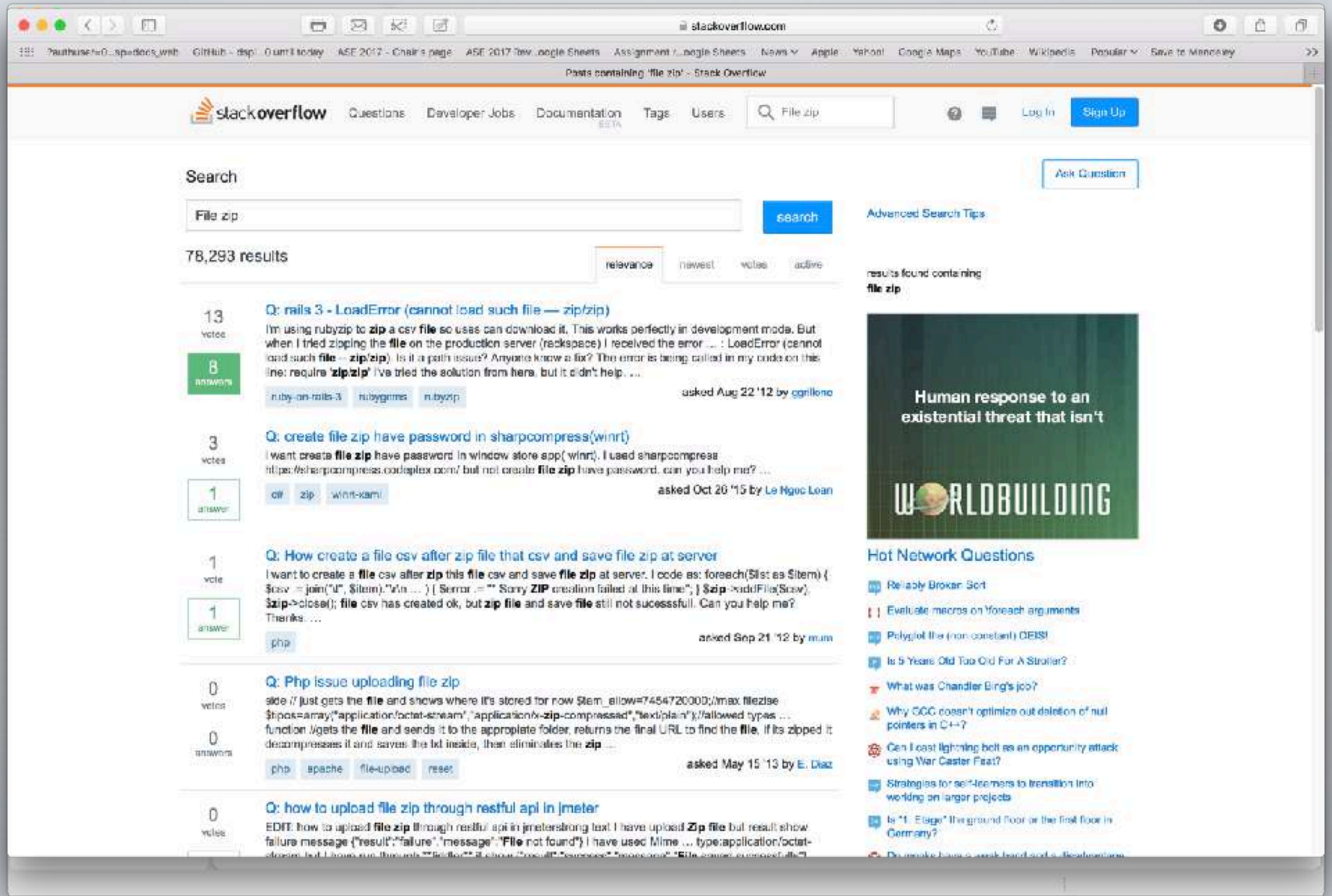
MOTIVATIONS



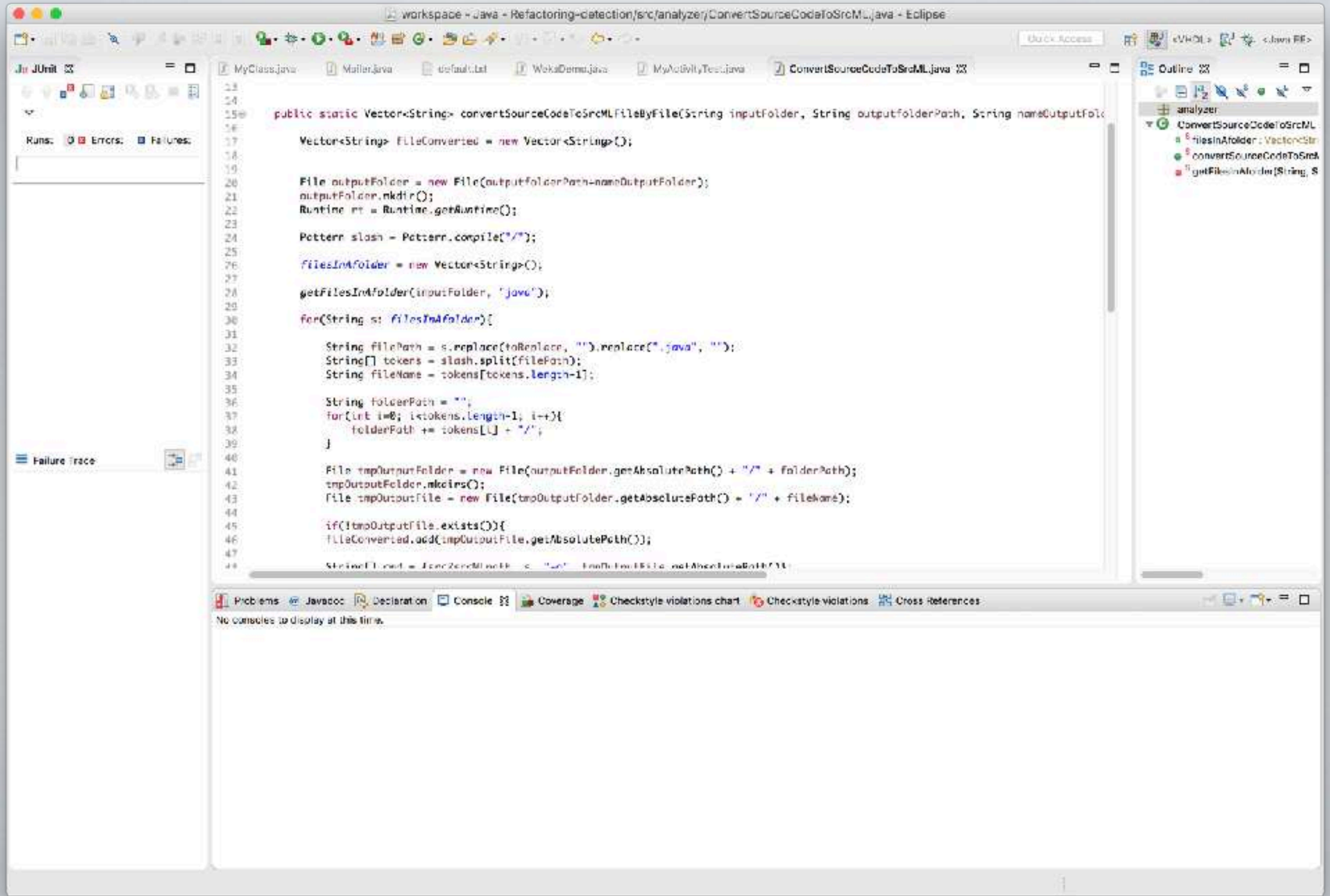
MOTIVATIONS



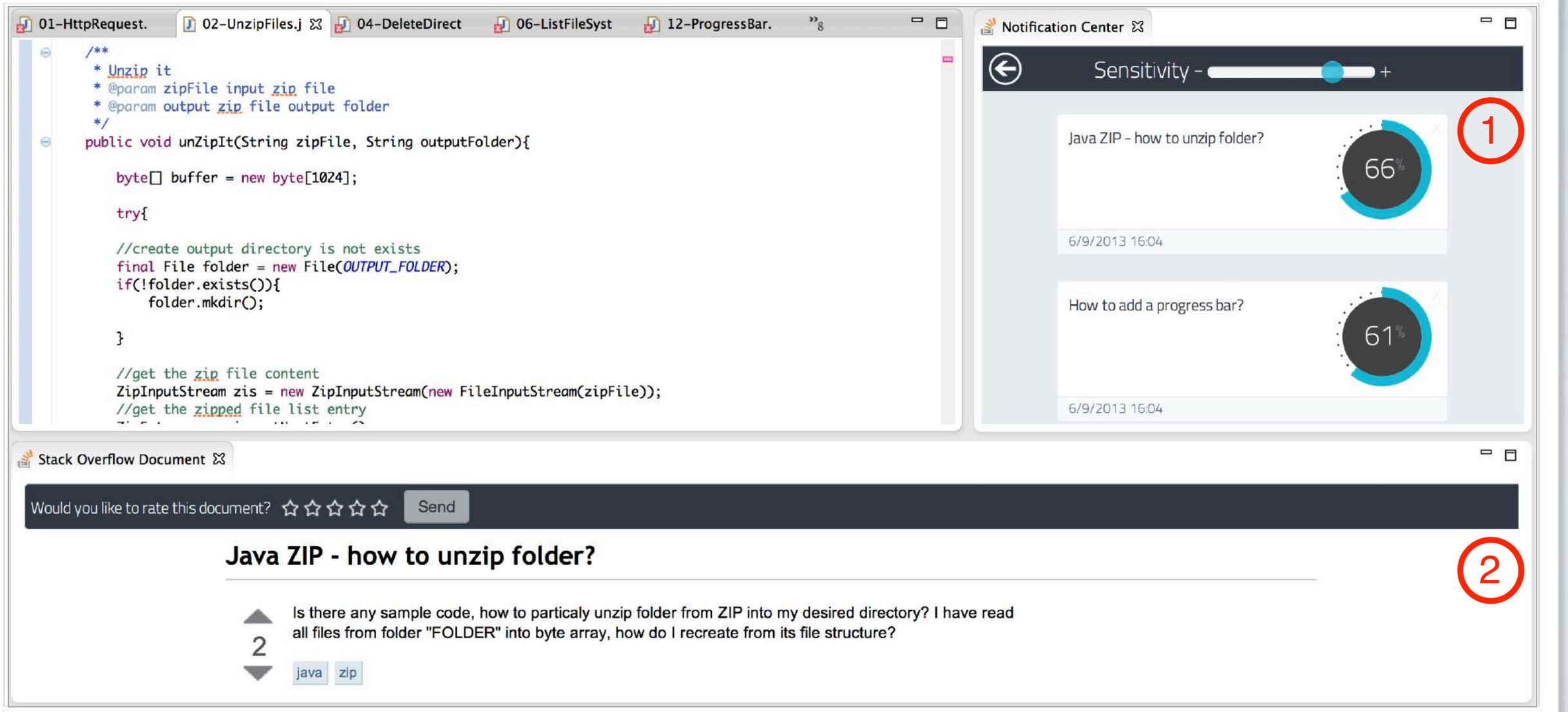
MOTIVATIONS



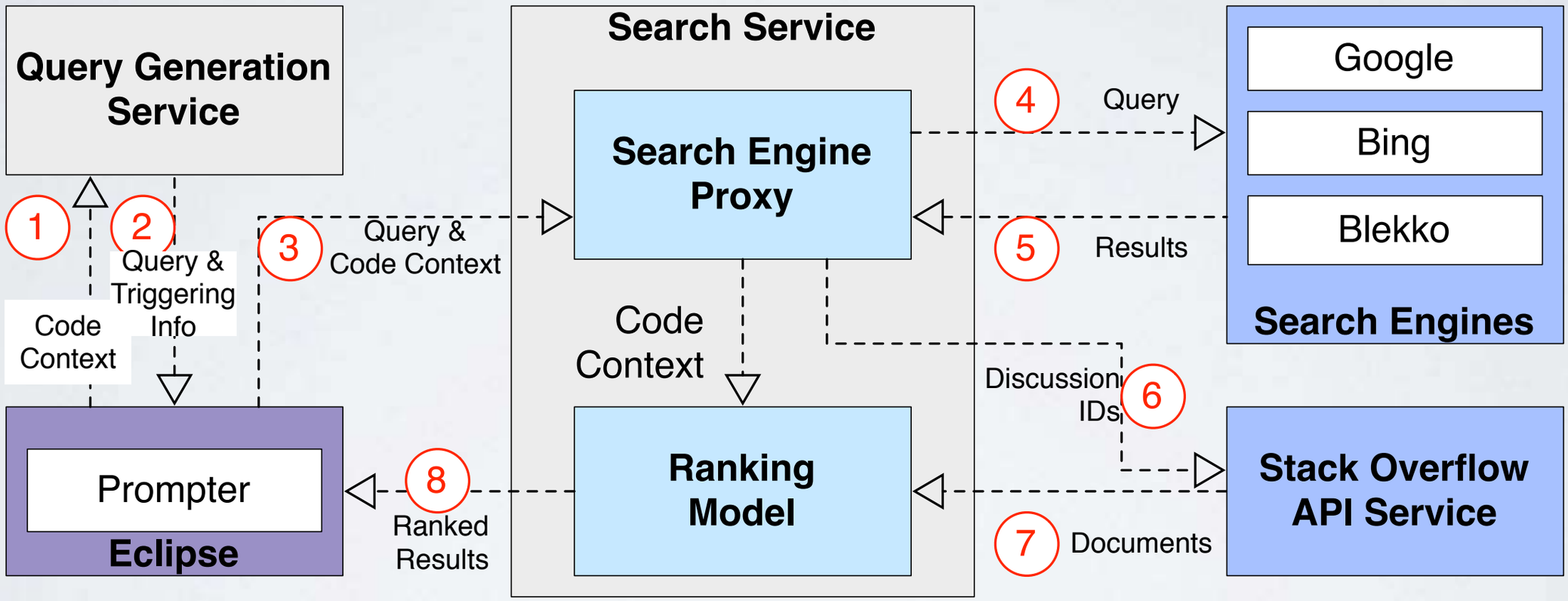
MOTIVATIONS



TOOL (PROMPTER)



APPROACH


Query Generation Service
Search Service
Search Engine Proxy
Ranking Model
Code Context
Google
Bing
Blekko
Search Engines
Stack Overflow API Service
Prompter
Eclipse
1
2
Code Context
Query & Triggering Info
3
Query & Code Context
4
Query
5
Results
6
Discussion IDs
7
Documents
8
Ranked Results

THESE SOUND LIKE VERY PROMISING DIRECTIONS…

…BUT TRAPS ARE AROUND THE CORNER!

CHALLENGE 1

?

BACK TO THE DEFINITION…

> "A software application that provides information items estimated to be valuable for a software engineering task in a given context"

Martin P. Robillard, Robert J. Walker, Thomas Zimmermann:
Recommendation Systems for Software Engineering. IEEE Software 27(4): 80-86 (2010)

BACK TO THE DEFINITION…

"A software application that provides **information items** estimated to be valuable for a software engineering task in a given context"

Martin P. Robillard, Robert J. Walker, Thomas Zimmermann:
Recommendation Systems for Software Engineering. IEEE Software 27(4): 80-86 (2010)

CAN WE FULLY RELY ON SOFTWARE REPOSITORY DATA?

CHALLENGE I: NOISY AND INCOMPLETE DATA

THE PROBLEM

Many pieces of information filled by humans → imprecise and incomplete

MISSING LINKS

Quieten level 0 debug when probing for modules. We shouldn't display so loud an error when a smb_probe_module() fails. Also tidy up debugs a bit. **Bug 375**.

nmbd_incomingdgrams.c: Fix bug with Syntax 5.1 servers reported by SGI where they do host announcements to LOCAL_MASTER_BROWSER_NAME<00 rather than WORKGROUP<1d

Adrian Bachmann, Christian Bird, Foyzur Rahman, Premkumar T. Devanbu, Abraham Bernstein: The missing links: bugs and bug-fix commits. SIGSOFT FSE 2010: 97-106

SECRET LIFE OF BUGS

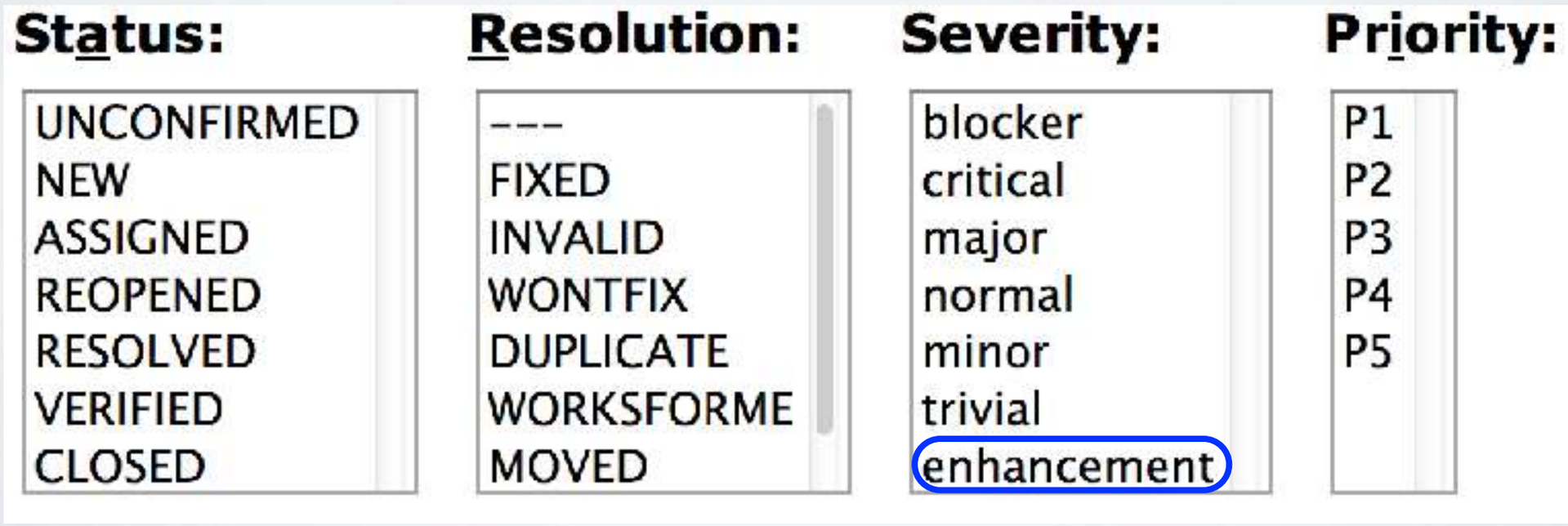
Software repositories do not capture everything of a software project

Not all discussions, not all decisions, and after all also not all changes

Jorge Aranda, Gina Venolia: The secret life of bugs: Going past the errors and omissions in software repositories. ICSE 2009: 298-308

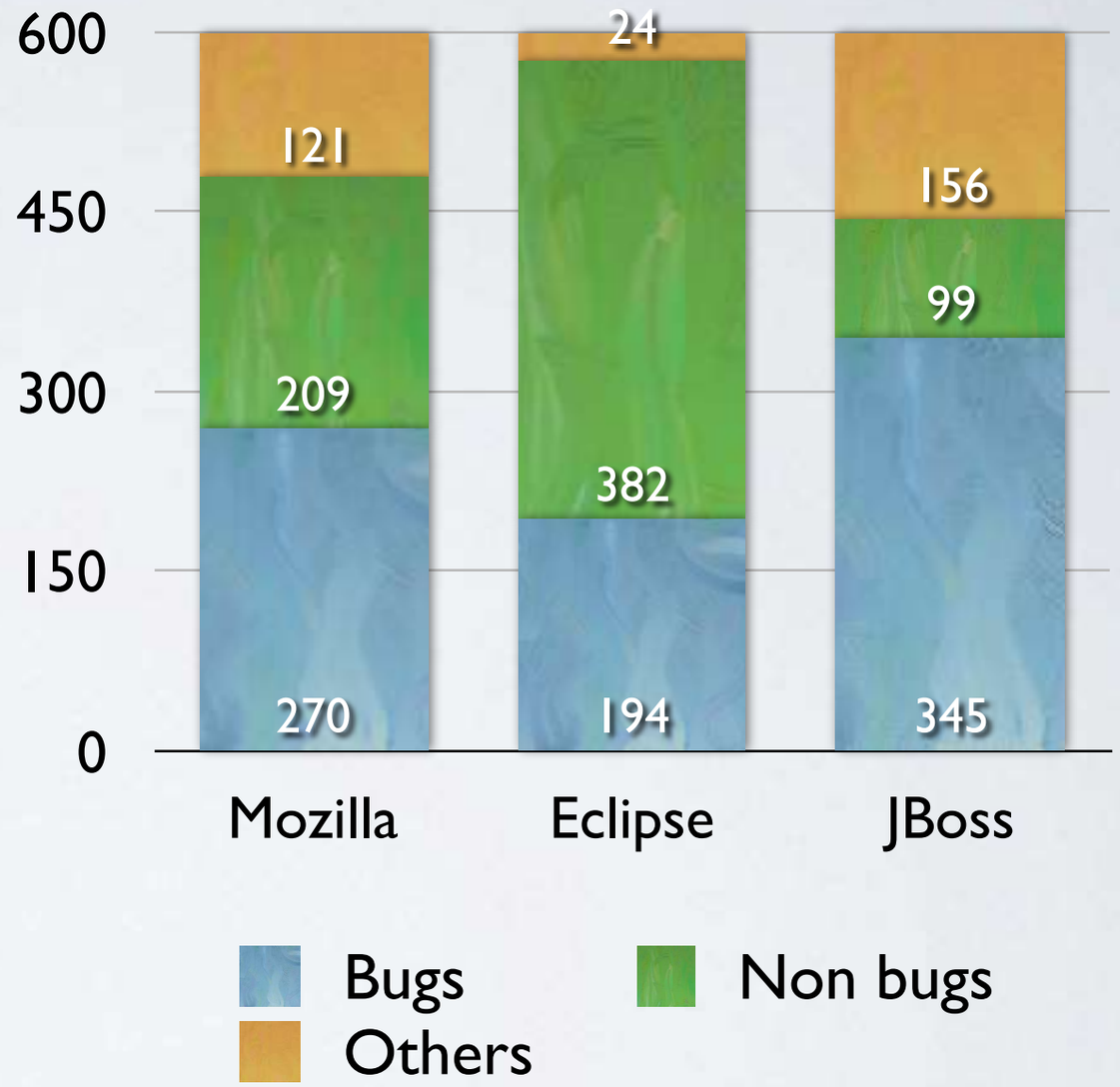
INCORRECT CLASSIFICATION

- Issue tracking systems contain various kinds of changes
- Classified using inadequate fields, or just poorly and subjectively classified



RESULTS OF A MANUAL CLASSIFICATION

We manually classified 1,800 randomly selected bugs from Mozilla, Eclipse, JBoss



Giuliano Antoniol, Kamel Ayari, Massimiliano Di Penta, Foutse Khomh, Yann-Gaël Guéhéneuc: Is it a bug or an enhancement?: a text-based approach to classify change requests. CASCON 2008: 23

It's not a Bug, it's a Feature: How Misclassification Impacts Bug Prediction

Kim Herzig
Saarland University
Saarbrücken, Germany
herzig@cs.uni-saarland.de

Sascha Just
Saarland University
Saarbrücken, Germany
just@st.cs.uni-saarland.de

Andreas Zeller
Saarland University
Saarbrücken, Germany
zeller@cs.uni-saarland.de

***Abstract*—In a manual examination of more than 7,000 issue reports from the bug databases of five open-source projects, we found 33.8% of all bug reports to be *misclassified*—that is, rather than referring to a code fix, they resulted in a new feature, an update to documentation, or an internal refactoring. This misclassification introduces *bias* in bug prediction models, confusing bugs and features: On average, 39% of files marked as defective actually never had a bug. We estimate the impact of this misclassification on earlier studies and recommend manual data validation for future studies.**

***Index Terms*—mining software repositories; bug reports; data quality; noise; bias**

I. Introduction

In empirical software engineering, it has become commonplace to mine data from change and bug databases to detect where bugs have occurred in the past, or to predict where they

TABLE I
PROJECT DETAILS.

	Maintainer	Tracker type	# reports
HTTPClient	APACHE	Jira	746
Jackrabbit	APACHE	Jira	2,402
Lucene-Java	APACHE	Jira	2,443
Rhino	MOZILLA	Bugzilla	1,226
Tomcat5	APACHE	Bugzilla	584

These are the questions we address in this paper. From five open source projects (Section II), we manually classified more than 7,000 issue reports into a fixed set of issue report categories clearly distinguishing the kind of maintenance work required to resolve the task (Section III). Our findings indicate substantial data quality issues:

Kim Herzig, Sascha Just, Andreas Zeller: It's not a bug, it's a feature: how misclassification impacts bug prediction. ICSE 2013: 392-401

IT'S NOT A BUG…

Confirmed our results: 1/3 of bug reports are not about bugs

When predicting the top 10% defect-prone files, 16% to 40% do not belong to that category

HINTS

Do not trust data from software repositories

Use heuristics to prevent problems

Validate, validate, validate, validate!

CHALLENGE II

?

No question, when you develop a recommender, you need to evaluate it

TYPICAL QUESTIONS YOU ASK

HOW ACCURATE IS IT?

HOW FAST IS IT?

IS IT ANY BETTER THAN COMPETITOR'S TOOL?

WHAT ARE WE MISSING HERE?

BACK TO THE DEFINITION…

> "A software application that provides information items estimated to be valuable for a software engineering task in a given context"

Martin P. Robillard, Robert J. Walker, Thomas Zimmermann:
Recommendation Systems for Software Engineering. IEEE Software 27(4): 80-86 (2010)

BACK TO THE DEFINITION…

"A software application that provides information items estimated **to be valuable for a software engineering task** in a given context"

Martin P. Robillard, Robert J. Walker, Thomas Zimmermann:
Recommendation Systems for Software Engineering. IEEE Software 27(4): 80-86 (2010)

IS THE TOOL GOING TO HELP A DEVELOPER FOR A GIVEN TASK?

CHALLENGE II:
EVALUATION

DIFFERENT KINDS OF EVALUATIONS

- Surveys
- Controlled Experiments
- Case Studies

SURVEY

Retrospective (post mortem), e.g. about a technology/tool being adopted for a period of time

CONTROLLED EXPERIMENT

Study performed in a laboratory setting, with a high degree of control

CASE STUDY

Aims at monitoring a (real) project in a realistic environment

SCALE VS. RISK

[Linkerman and Rombach, 2000]

QUANTITATIVE VS QUALITATIVE STUDIES

Quantitative: to get numerical relations among variables

Qualitative: to interpret a phenomenon just observing it in its context

EXAMPLES

ARENA

ARENA

LUCENE 4.0.0

New Features (1)

1. LUCENE-3842: Analyzing Suggester [more info]

Bug Fixes (30)

1. LUCENE-4459: TestWeakIdentityMap.testConcurrentHashMap fails periodically in jenkins [more info]

[...]

12. LUCENE-4364: MMapDirectory makes too many maps for CFS [more info]

    - New abstract class ByteBufferIndexInput extending IndexInput. This entity class includes accessor and mutator methods, and some business logic. It provides access to short, byte buffer index input long, length, file pointer. It allows managing bytes, and byte. It also allows closing byte buffer index input, seeking byte buffer index input, cloning byte buffer index input, and slicing byte buffer index input.
    - [...]

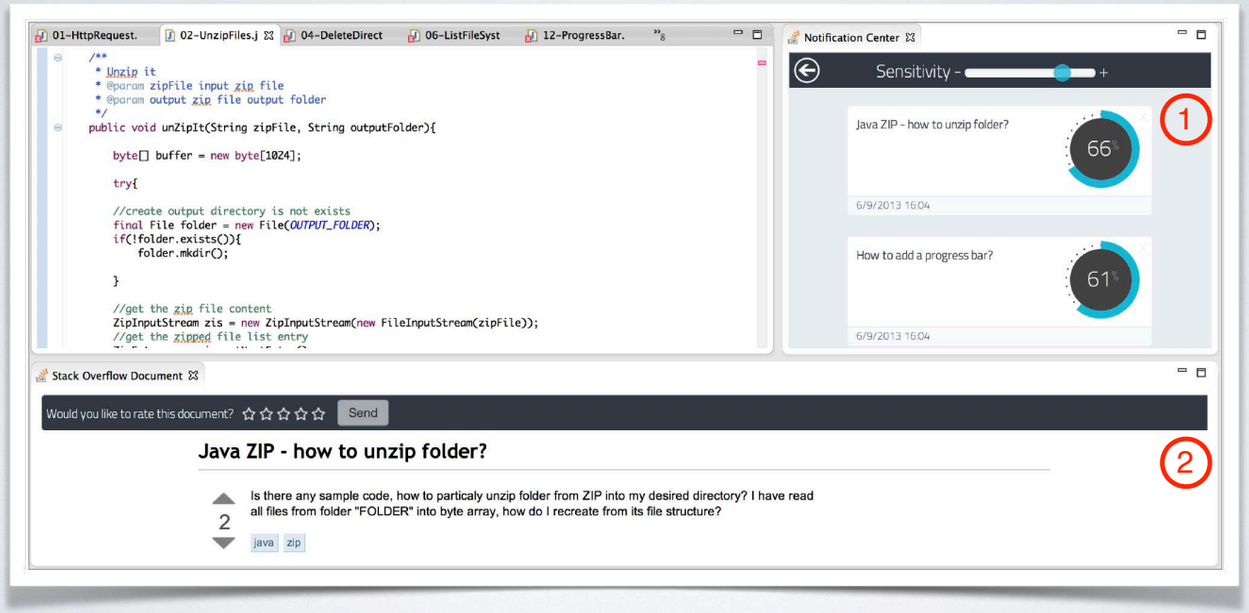
Improvements (17)

1. LUCENE-4448: speedups for AnalyzingSuggester
2. LUCENE-4440: FilterCodec should take a delegate Codec in its ctor [more info]

    - Modified methods forName(String), and availablePostingsFormats() in PostingsFormat.

[...]

PROMPTER



MUSE

```
01 File source;
02 File target;
03 ZipFile zip=new ZipFile(source);
04 Enumeration<? extends ZipEntry> entries=zip.entries();
05 while(entries.hasMoreElements()) {
06  ZipEntry entry=entries.nextElement();
07  File file=new File(target,entry.getName());
08  //zip.getInputStream(entry)->the InputStream to copy bytes from,
09  //must not be null
10  //file->the non-directory File to write bytes to (possibly
11  //overwriting), must not be null
12  FileUtils.copyInputStreamToFile(zip.getInputStream(entry),file);
13 }
```

ARENA

ARENA

LUCENE 4.0.0

New Features (1)

1. LUCENE-3842: Analyzing Suggester [more info]

Bug Fixes (30)

1. LUCENE-4459: TestWeakIdentityMap.testConcurrentHashMap fails periodically in jenkins [more info]

[...]

12. LUCENE-4364: MMapDirectory makes too many maps for CFS [more info]

    - New abstract class ByteBufferIndexInput extending IndexInput. This entity class includes accessor and mutator methods, and some business logic. It provides access to short, byte buffer index input long, length, file pointer. It allows managing bytes, and byte. It also allows closing byte buffer index input, seeking byte buffer index input, cloning byte buffer index input, and slicing byte buffer index input.
    - [...]

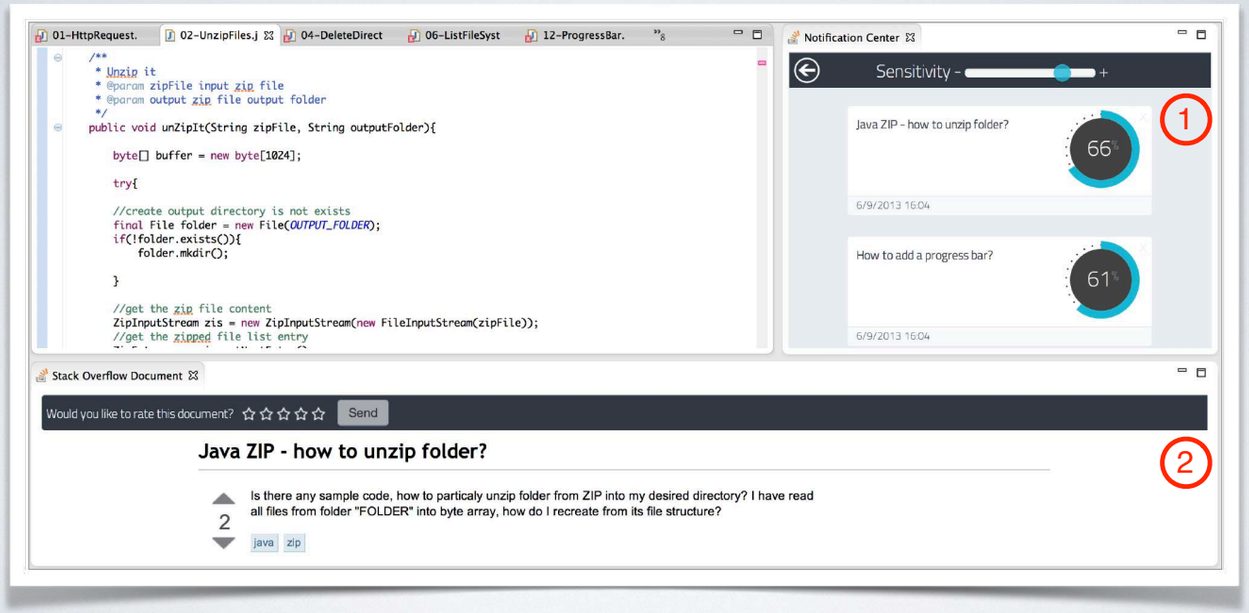
Improvements (17)

1. LUCENE-4448: speedups for AnalyzingSuggester
2. LUCENE-4440: FilterCodec should take a delegate Codec in its ctor [more info]

    - Modified methods forName(String), and availablePostingsFormats() in PostingsFormat.

[...]

PROMPTER



MUSE

```
01 File source;
02 File target;
03 ZipFile zip=new ZipFile(source);
04 Enumeration<? extends ZipEntry> entries=zip.entries();
05 while(entries.hasMoreElements()) {
06  ZipEntry entry=entries.nextElement();
07  File file=new File(target,entry.getName());
08  //zip.getInputStream(entry)->the InputStream to copy bytes from,
09  //must not be null
10  //file->the non-directory File to write bytes to (possibly
11  //overwriting), must not be null
12  FileUtils.copyInputStreamToFile(zip.getInputStream(entry),file);
13 }
```

ARENA

ARENA

LUCENE 4.0.0

New Features (1)

1. LUCENE-3842: Analyzing Suggester [more info]

Bug Fixes (30)

1. LUCENE-4459: TestWeakIdentityMap.testConcurrentHashMap fails periodically in jenkins [more info]

[...]

12. LUCENE-4364: MMapDirectory makes too many maps for CFS [more info]

    - New abstract class ByteBufferIndexInput extending IndexInput. This entity class includes accessor and mutator methods, and some business logic. It provides access to short, byte buffer index input long, length, file pointer. It allows managing bytes, and byte. It also allows closing byte buffer index input, seeking byte buffer index input, cloning byte buffer index input, and slicing byte buffer index input.
    - [...]

Improvements (17)

1. LUCENE-4448: speedups for AnalyzingSuggester
2. LUCENE-4440: FilterCodec should take a delegate Codec in its ctor [more info]

    - Modified methods forName(String), and availablePostingsFormats() in PostingsFormat.

[...]

ARENA

Study I: Completeness of release notes

Study II: Importance of different pieces

Study III: Comparison with human generated release notes

Study IV: Field Study

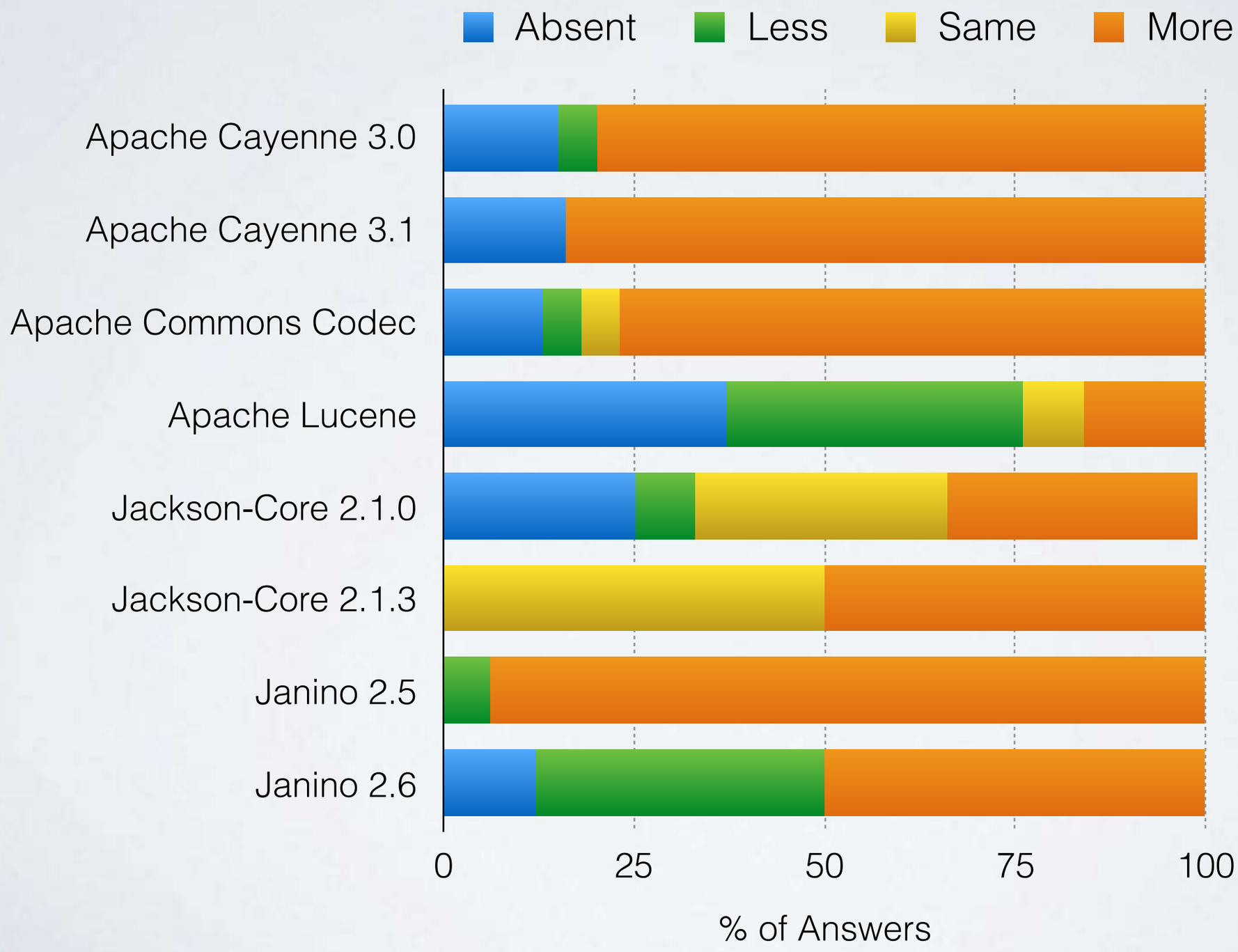
EVALUATION I: COMPLETENESS

Design: We asked study participants to compare the generated release note with the original one

Study participants: 10 among students, faculties, industrial developers

Objects: 10 releases of open source projects

EVALUATION I: RESULTS



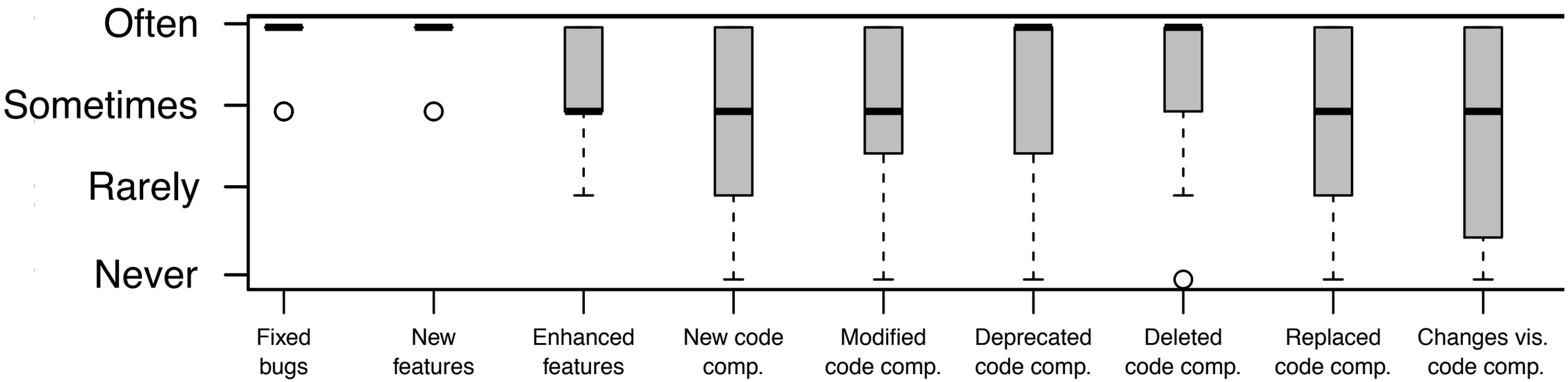
EVALUATION II: IMPORTANCE

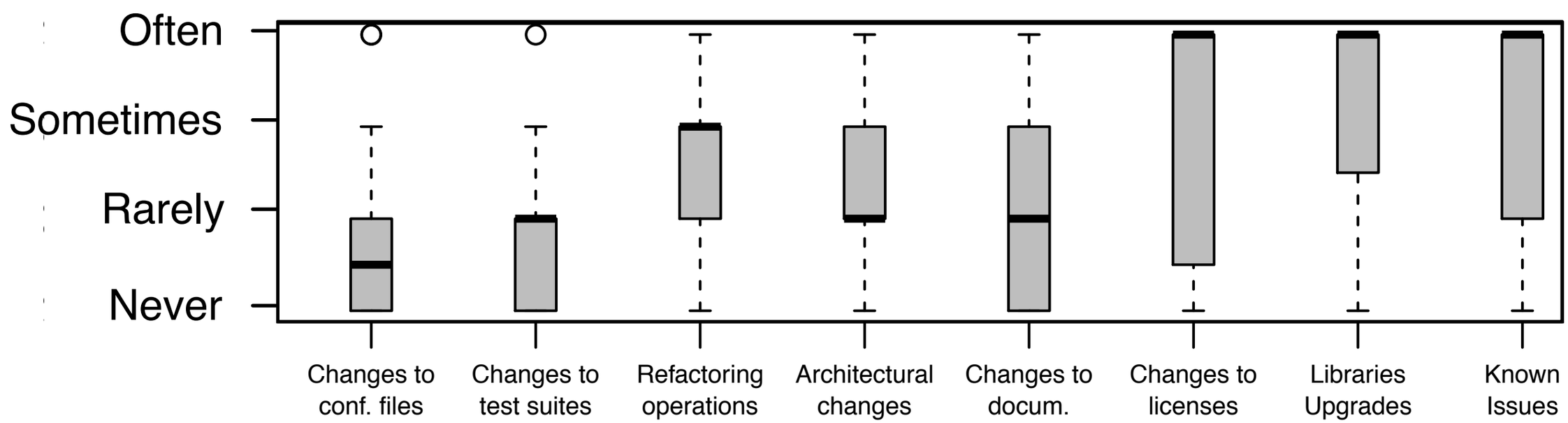
Online survey, in which we showed to participant release notes (they didn't know whether these were generated or not)

We asked participants of a online survey to evaluate:

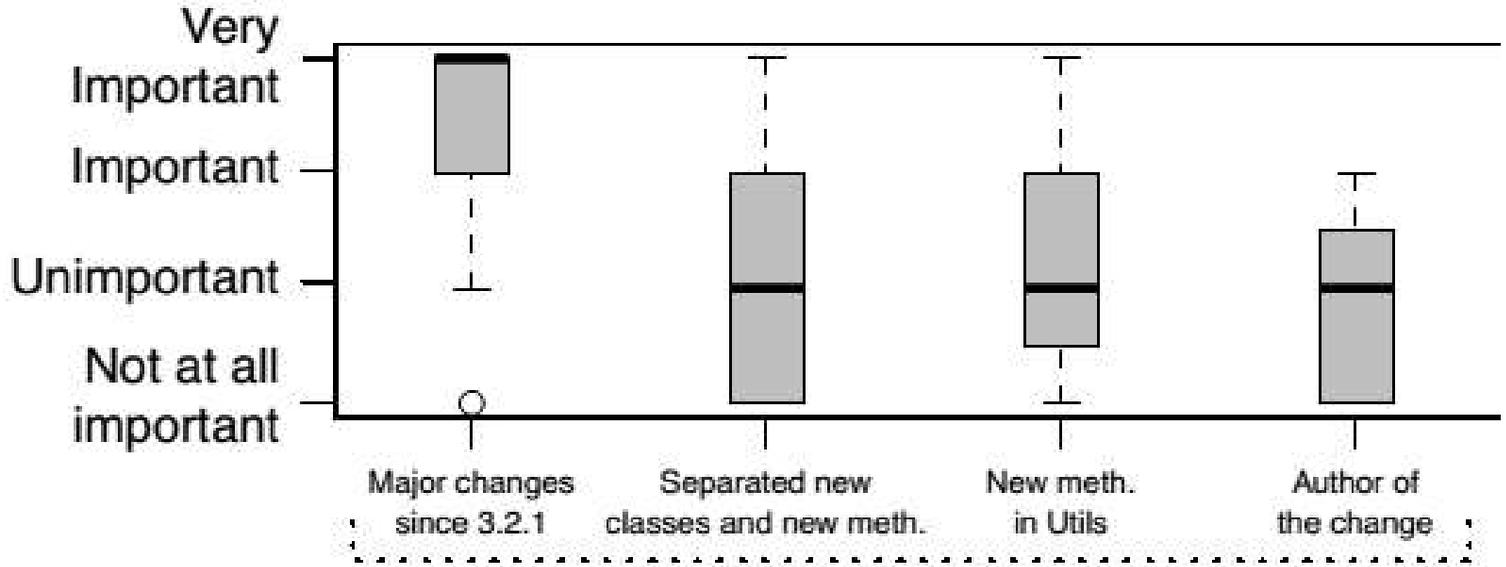
- What they consider important in a release note and often
- What they considered important in both ARENA release notes and actual release notes

WHAT DO DEVELOPERS INCLUDE IN RELEASE NOTES?

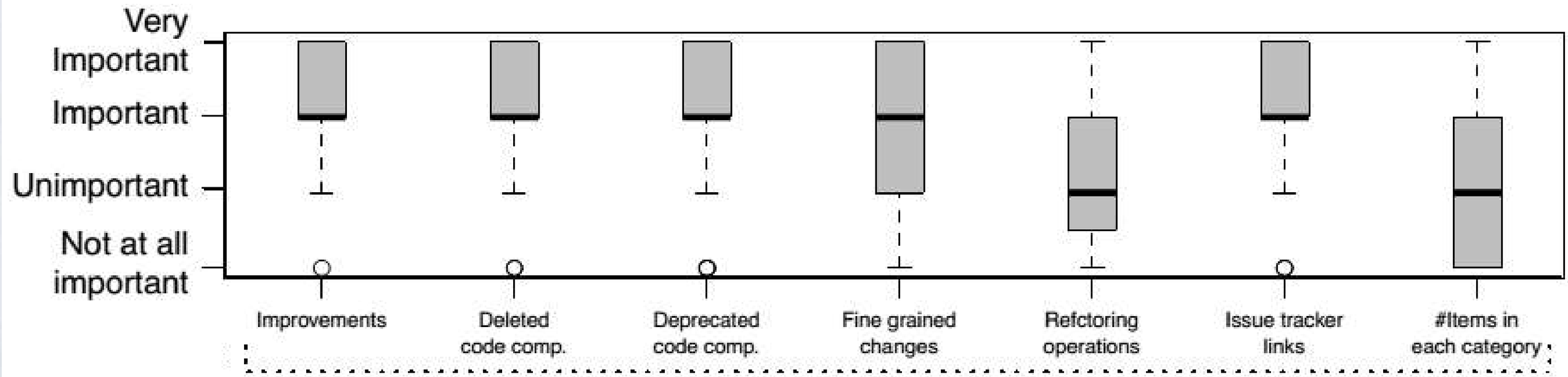




IMPORTANCE OF RELEASE NOTE CONTENT



ORIGINAL RELEASE NOTE



ARENA RELEASE NOTE

ARENA

ARENA

LUCENE 4.0.0

New Features (1)

1. LUCENE-3842: Analyzing Suggester [more info]

Bug Fixes (30)

1. LUCENE-4459: TestWeakIdentityMap.testConcurrentHashMap fails periodically in jenkins [more info]

[...]

12. LUCENE-4364: MMapDirectory makes too many maps for CFS [more info]

    - New abstract class ByteBufferIndexInput extending IndexInput. This entity class includes accessor and mutator methods, and some business logic. It provides access to short, byte buffer index input long, length, file pointer. It allows managing bytes, and byte. It also allows closing byte buffer index input, seeking byte buffer index input, cloning byte buffer index input, and slicing byte buffer index input.
    - [...]

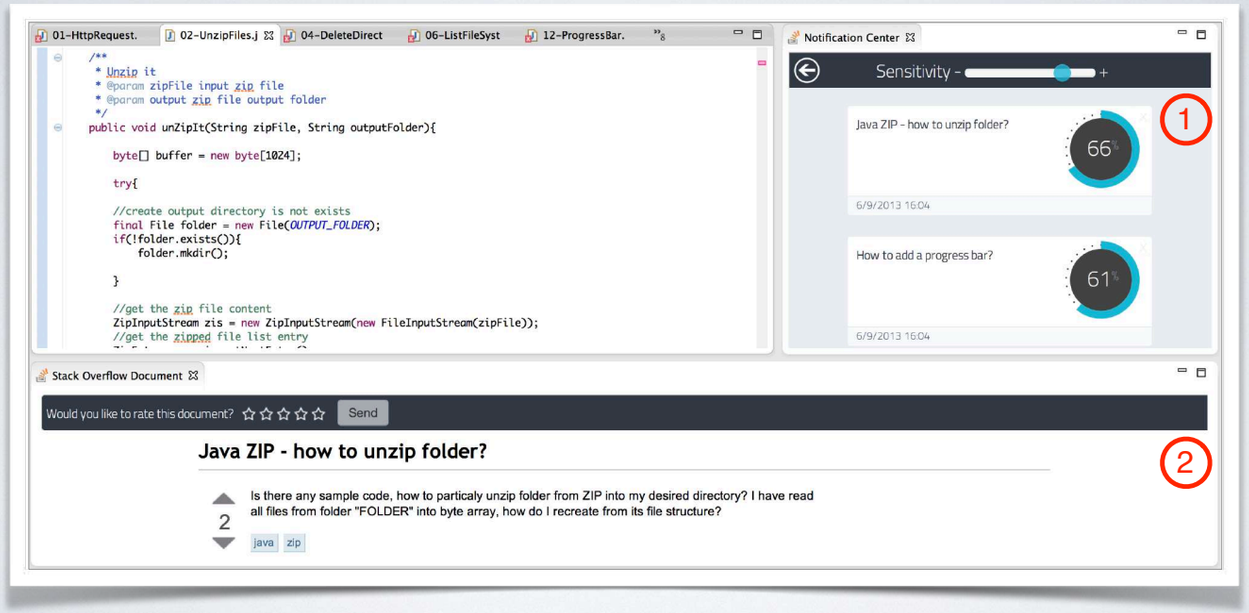
Improvements (17)

1. LUCENE-4448: speedups for AnalyzingSuggester
2. LUCENE-4440: FilterCodec should take a delegate Codec in its ctor [more info]

    - Modified methods forName(String), and availablePostingsFormats() in PostingsFormat.

[...]

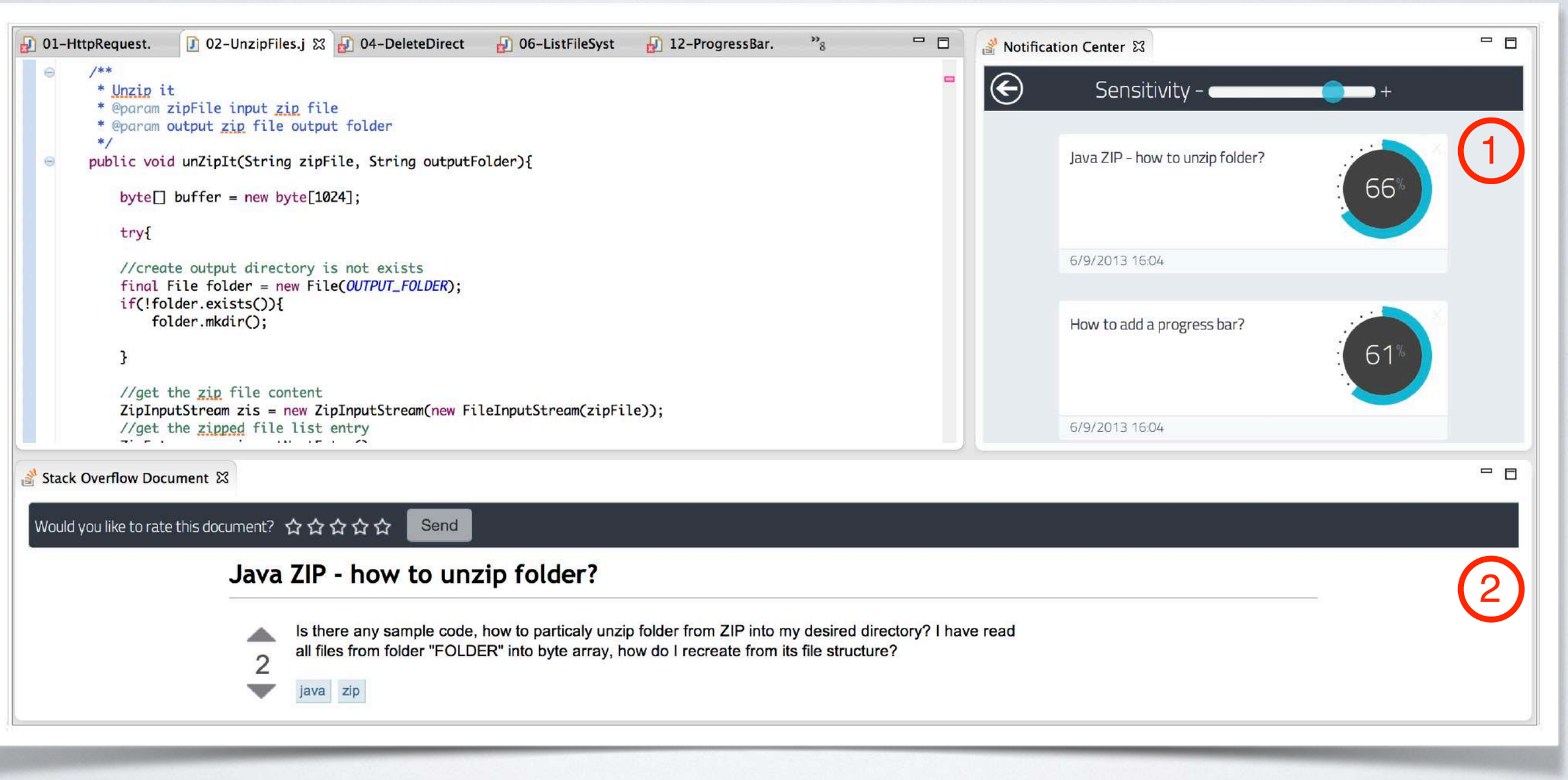
PROMPTER



MUSE

```
01 File source;
02 File target;
03 ZipFile zip=new ZipFile(source);
04 Enumeration<? extends ZipEntry> entries=zip.entries();
05 while(entries.hasMoreElements()) {
06  ZipEntry entry=entries.nextElement();
07  File file=new File(target,entry.getName());
08  //zip.getInputStream(entry)->the InputStream to copy bytes from,
09  //must not be null
10  //file->the non-directory File to write bytes to (possibly
11  //overwriting), must not be null
12  FileUtils.copyInputStreamToFile(zip.getInputStream(entry),file);
13 }
```

PROMPTER

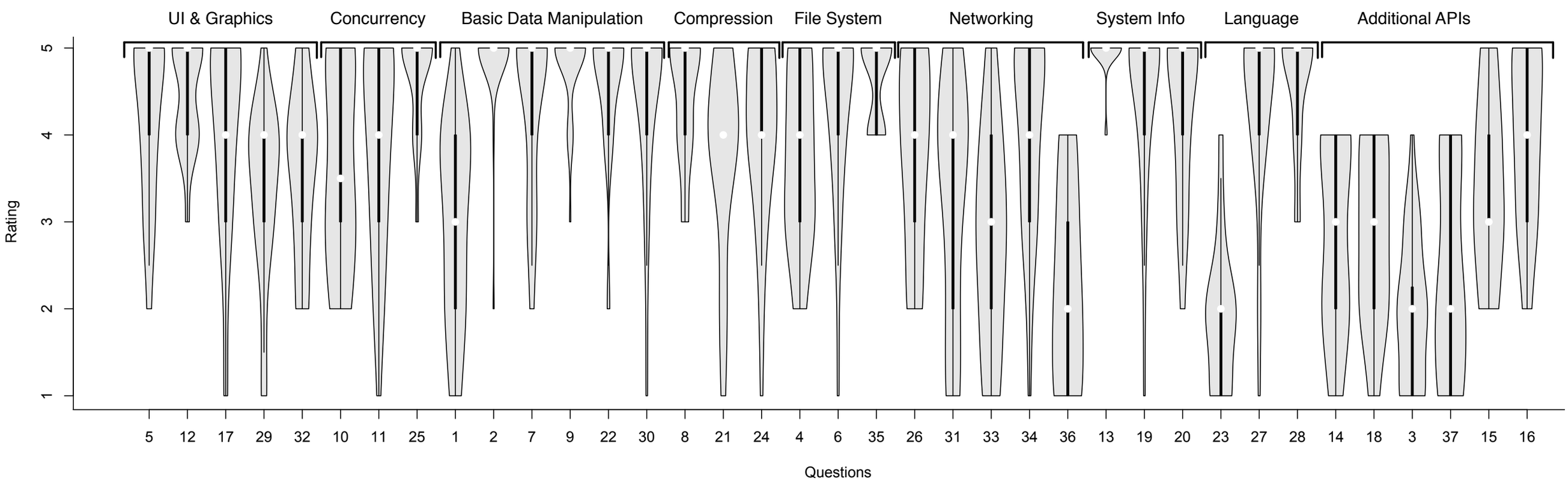

01-HttpRequest.
02-UnzipFiles.j
04-DeleteDirect
06-ListFileSyst
12-ProgressBar.
/**
* Unzip it
* @param zipFile input zip file
* @param output zip file output folder
*/
public void unZipIt(String zipFile, String outputFolder){
byte[] buffer = new byte[1024];
try{
//create output directory is not exists
final File folder = new File(OUTPUT_FOLDER);
if(!folder.exists()){
folder.mkdir();
}
//get the zip file content
ZipInputStream zis = new ZipInputStream(new FileInputStream(zipFile));
//get the zipped file list entry
Notification Center
Sensitivity -
+
Java ZIP - how to unzip folder?
66%
6/9/2013 16:04
How to add a progress bar?
61%
6/9/2013 16:04
1
Stack Overflow Document
Would you like to rate this document?
Send
Java ZIP - how to unzip folder?
2
Is there any sample code, how to particaly unzip folder from ZIP into my desired directory? I have read all files from folder "FOLDER" into byte array, how do I recreate from its file structure?
2
java
zip
2

PROMPTER

First evaluation: assess the correctness of the provided recommendations

Second evaluation: ask developers to perform a task with and without Prompter

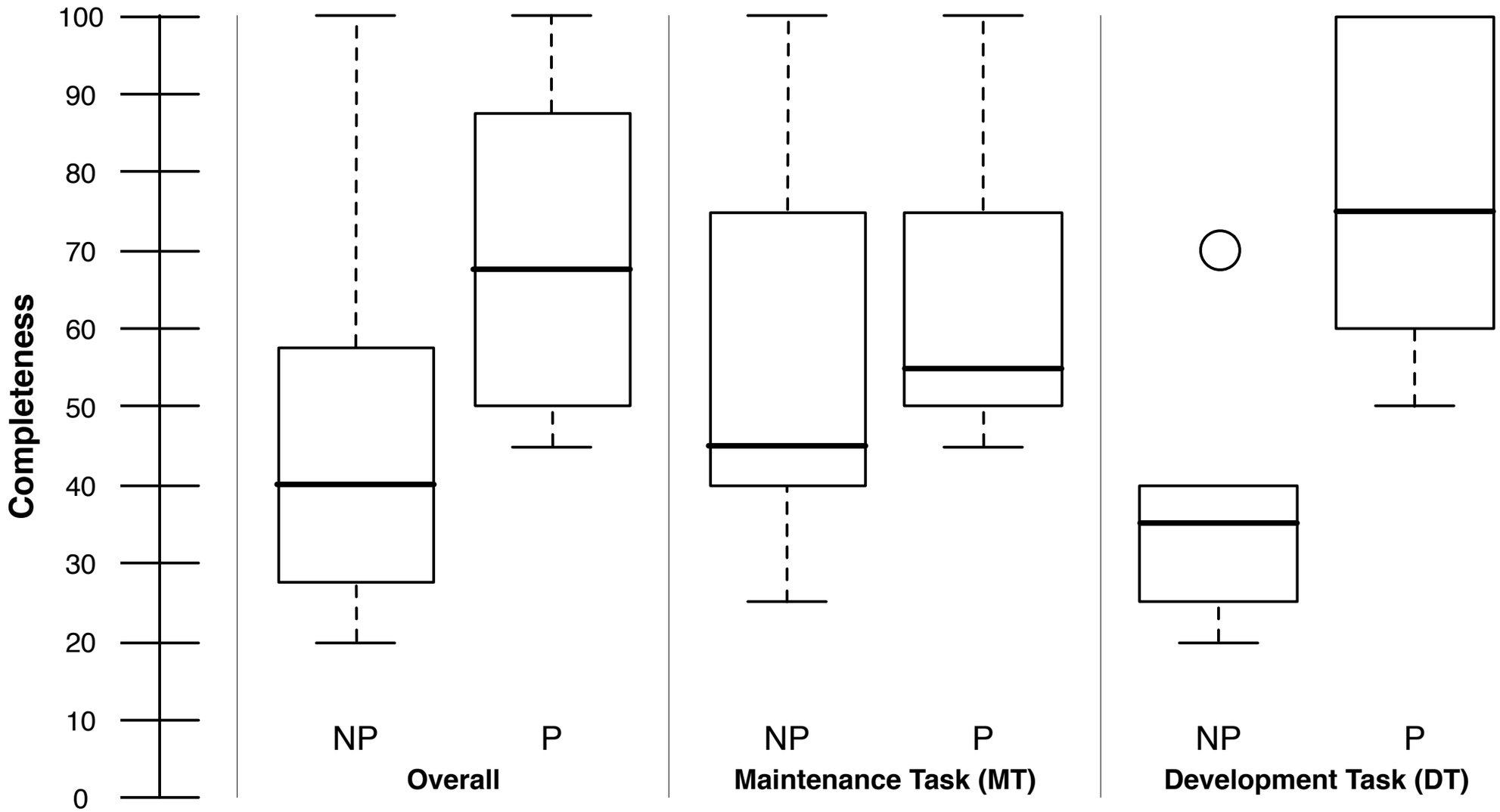
ARE RECOMMENDATIONS RELEVANT?



EXPERIMENT DESIGN

	Group I	Group 2	Group 3	Group 4
Lab I	Task I Prompter	Task I Just Web	Task II Just Web	Task II Prompter
Lab II	Task II Just Web	Task II Prompter	Task I Prompter	Task I Just Web

DOES IT HELP?


Completeness
100
90
80
70
60
50
40
30
20
10
0
NP
P
Overall
NP
P
Maintenance Task (MT)
NP
P
Development Task (DT)

ARENA

ARENA

LUCENE 4.0.0

New Features (1)

1. LUCENE-3842: Analyzing Suggester [more info]

Bug Fixes (30)

1. LUCENE-4459: TestWeakIdentityMap.testConcurrentHashMap fails periodically in jenkins [more info]

[...]

12. LUCENE-4364: MMapDirectory makes too many maps for CFS [more info]

    - New abstract class ByteBufferIndexInput extending IndexInput. This entity class includes accessor and mutator methods, and some business logic. It provides access to short, byte buffer index input long, length, file pointer. It allows managing bytes, and byte. It also allows closing byte buffer index input, seeking byte buffer index input, cloning byte buffer index input, and slicing byte buffer index input.
    - [...]

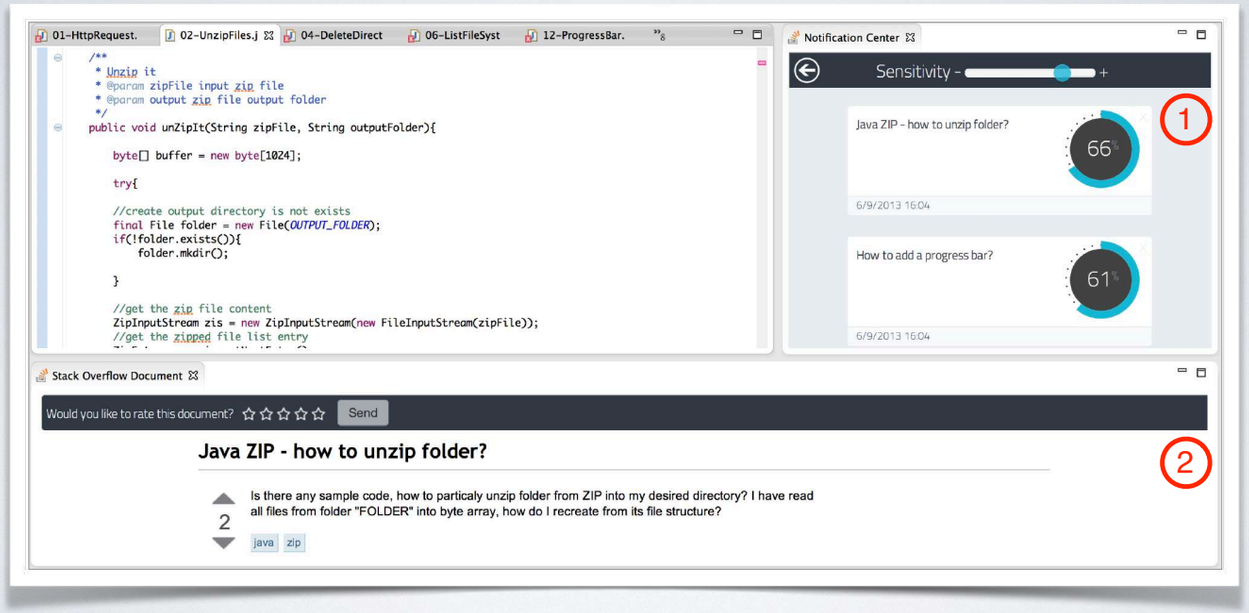
Improvements (17)

1. LUCENE-4448: speedups for AnalyzingSuggester
2. LUCENE-4440: FilterCodec should take a delegate Codec in its ctor [more info]

    - Modified methods forName(String), and availablePostingsFormats() in PostingsFormat.

[...]

PROMPTER



MUSE

```
01 File source;
02 File target;
03 ZipFile zip=new ZipFile(source);
04 Enumeration<? extends ZipEntry> entries=zip.entries();
05 while(entries.hasMoreElements()) {
06  ZipEntry entry=entries.nextElement();
07  File file=new File(target,entry.getName());
08  //zip.getInputStream(entry)->the InputStream to copy bytes from,
09  //must not be null
10  //file->the non-directory File to write bytes to (possibly
11  //overwriting), must not be null
12  FileUtils.copyInputStreamToFile(zip.getInputStream(entry),file);
13 }
```

MUSE

```
01 File source;
02 File target;
03 ZipFile zip=new ZipFile(source);
04 Enumeration<? extends ZipEntry> entries=zip.entries();
05 while(entries.hasMoreElements()) {
06  ZipEntry entry=entries.nextElement();
07  File file=new File(target,entry.getName());
08  //zip.getInputStream(entry)->the InputStream to copy bytes from,
09  //must not be null
10  //file->the non-directory File to write bytes to (possibly
11  //overwriting), must not be null
12  FileUtils.copyInputStreamToFile(zip.getInputStream(entry),file);
13 }
```

MUSE

Study I: (intrinsic) manual evaluation of produced examples

Study II: (extrinsic) controlled experiment

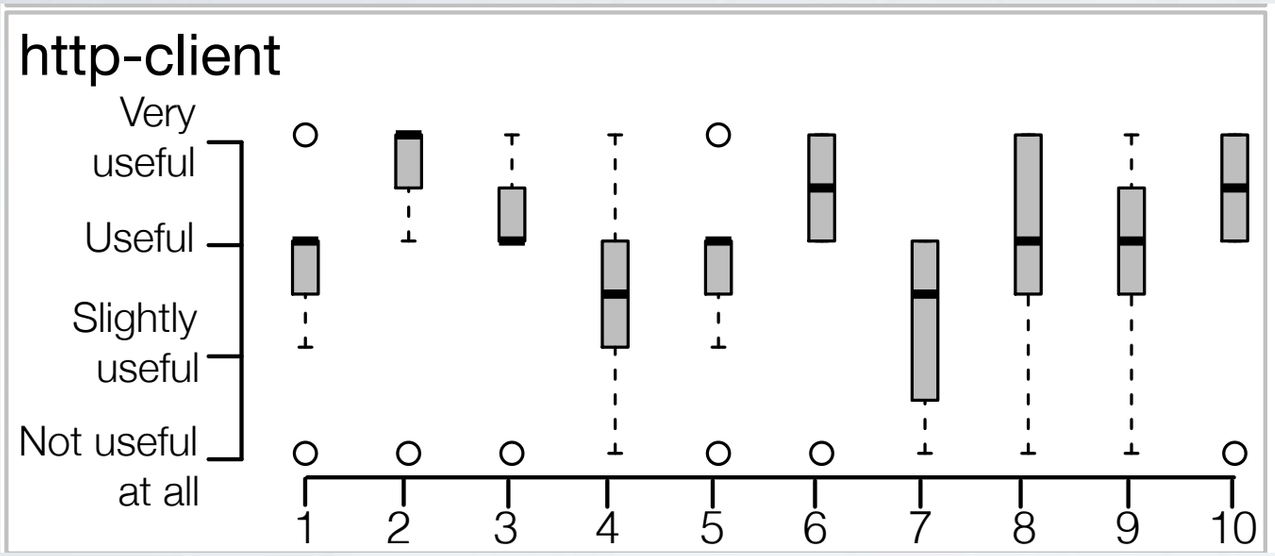
INTRINSIC EVALUATION

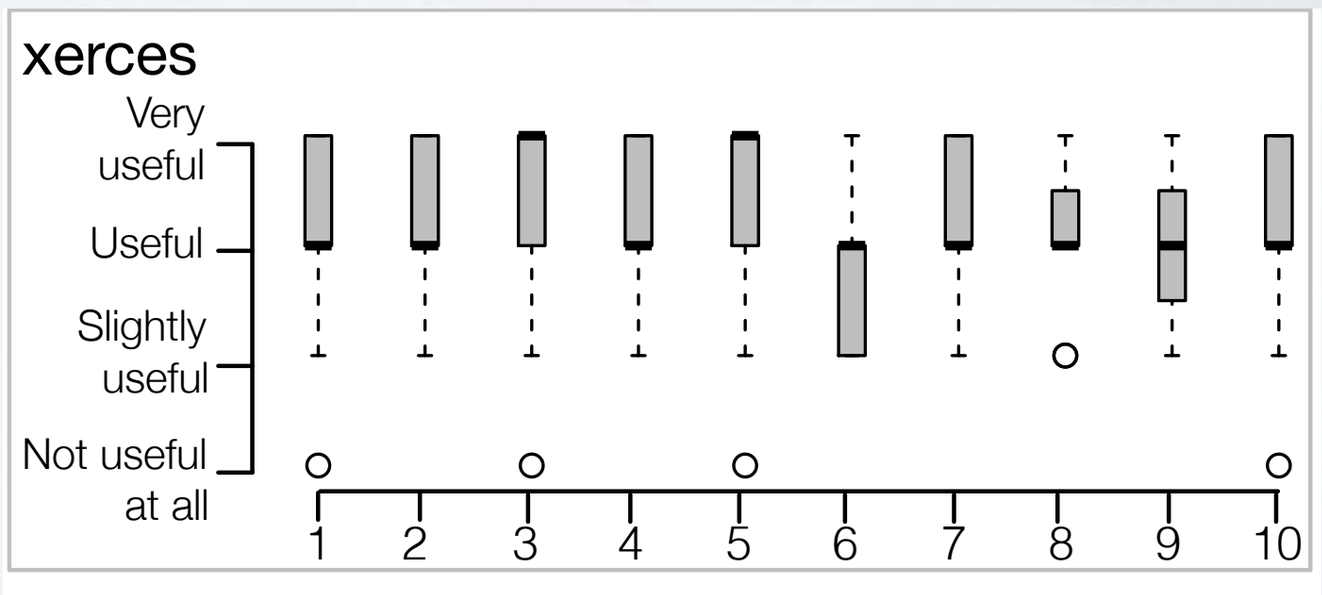
Research question: Are MUSE's usage examples considered useful by developers?

Context:

- 60 code examples from 6 Apache libraries
- 119 developers recruited among those who developed the libraries and from open source projects using such libraries

EXAMPLES OF RESULTS





EXTRINSIC EVALUATION

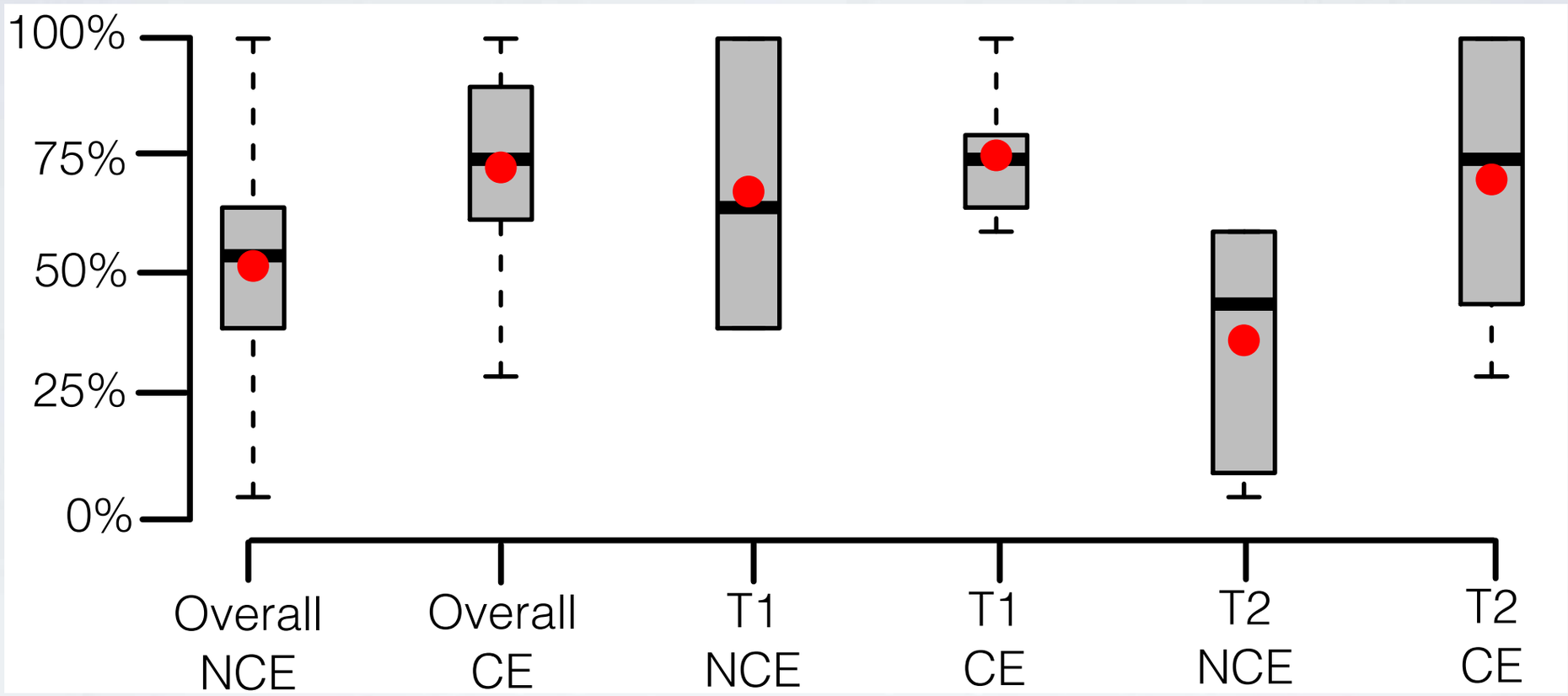
Research Question: Do MUSE's examples help developers to complete their programming tasks?

Participants: 12 Industrial developers

Study design: similar to Prompter

	Group I	Group 2	Group 3	Group 4
Lab I	Task I Without Examples	Task I With Examples	Task II Without Examples	Task II With Examples
Lab II	Task II With Examples	Task II Without Examples	Task I With Examples	Task I Without Examples

RESULTS



Difference between CE and NCE statistically significant (p- value=0.03) with a medium effect size (d=0.472)

CHALLENGE III

?

SO FAR…

SO FAR…

Our approach is very accurate

SO FAR…

- Our approach is very accurate
- It is very fast

SO FAR…

Our approach is very accurate

It is very fast

It is better than other tools

SO FAR…

- Our approach is very accurate
- It is very fast
- It is better than other tools
- It HELPS developers

WHAT ARE WE MISSING?

BACK TO THE DEFINITION…

> "A software application that provides information items estimated to be valuable for a software engineering task in a given context"

Martin P. Robillard, Robert J. Walker, Thomas Zimmermann:
Recommendation Systems for Software Engineering. IEEE Software 27(4): 80-86 (2010)

BACK TO THE DEFINITION…

"A software application that provides information items estimated to be valuable for a software engineering task **in a given context**"

Martin P. Robillard, Robert J. Walker, Thomas Zimmermann:
Recommendation Systems for Software Engineering. IEEE Software 27(4): 80-86 (2010)

CHALLENGE III:
PRACTICAL APPLICABILITY

INFORMED INTERVIEWS

Easy solution

1. Let developers play with the tool

2. Then, interview them…

IN-FIELD CASE STUDY

1. Let developers use your tool in a real task

2. Observe…

3. Interview…

ARENA: 6-MONTHS IN-FIELD STUDY

We let developers of a E-health project to use ARENA for 6 months

FEEDBACK

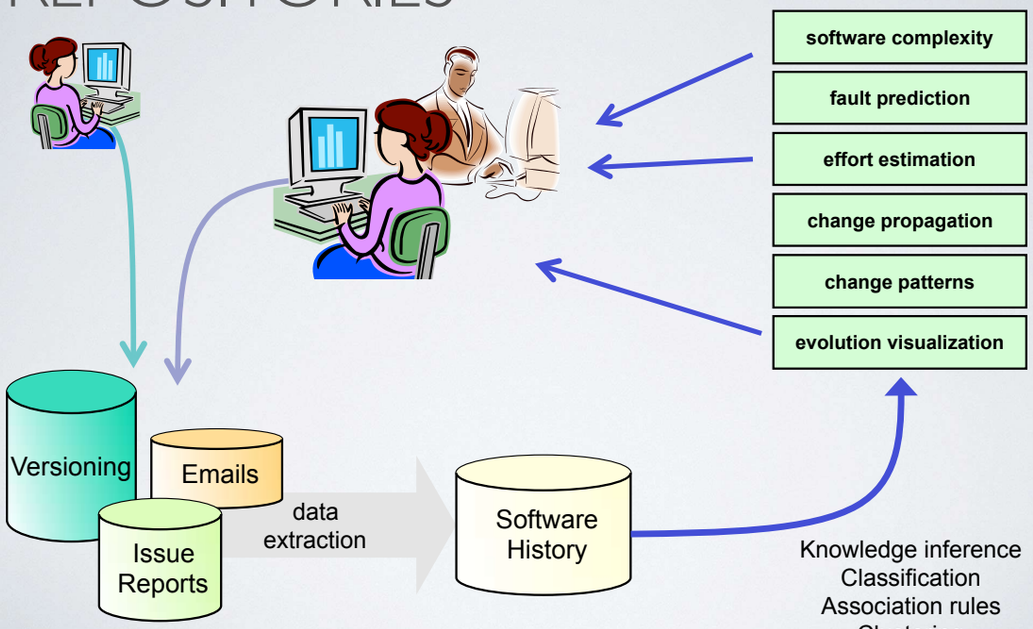
> "The possibility to expand and retract the details about any of the different items allows the reader of the release note to control in some way the level of redundancy she wants."

FEEDBACK

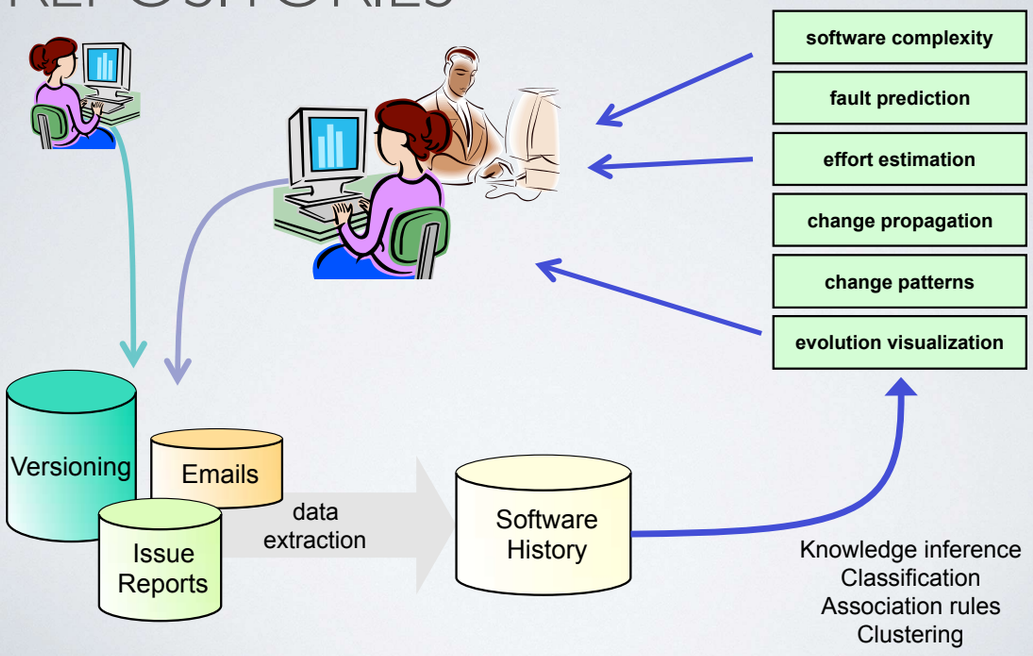
"ARENA helps to save time, especially when you need to create release notes and you cannot really allocate too much time on such a task."

SUMMARY

MINING SOFTWARE REPOSITORIES

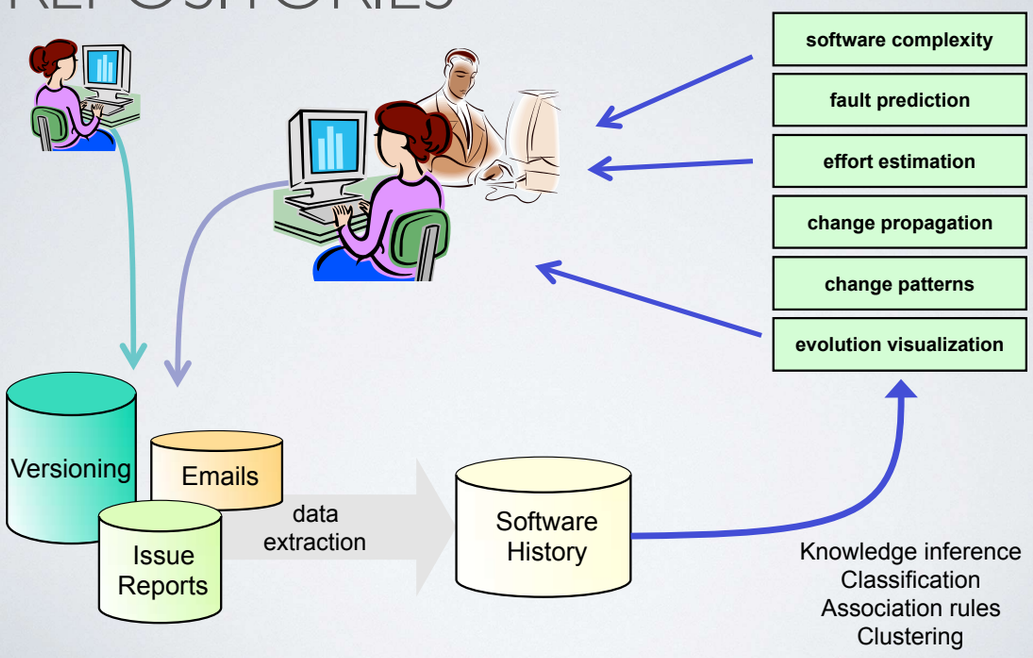


MINING SOFTWARE REPOSITORIES



CHALLENGE I: NOISY AND INCOMPLETE DATA

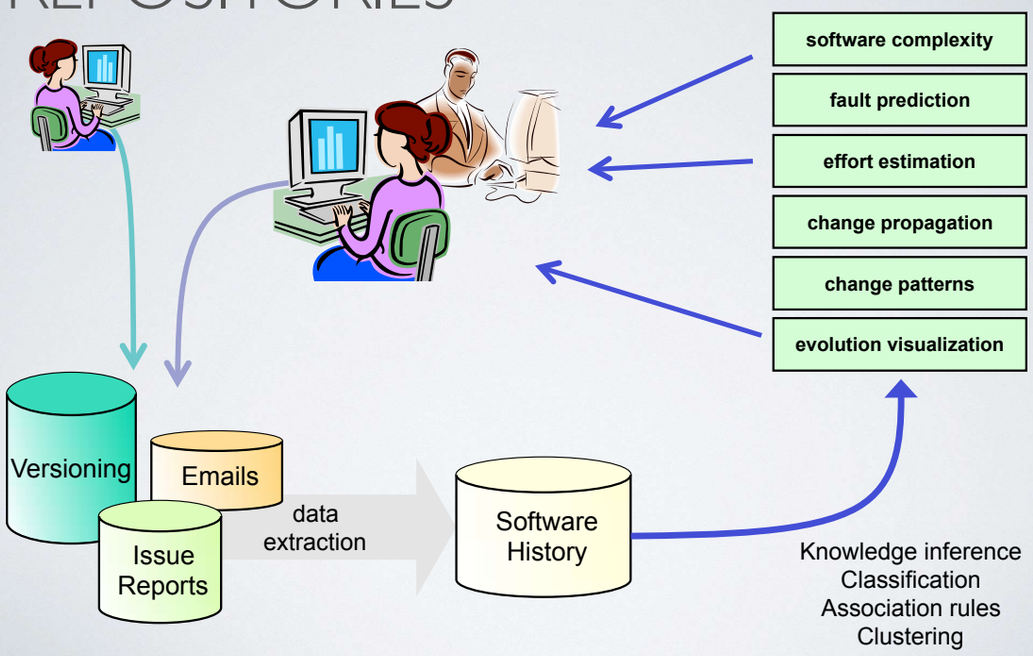
MINING SOFTWARE REPOSITORIES



CHALLENGE I: NOISY AND INCOMPLETE DATA

CHALLENGE II: EVALUATION

MINING SOFTWARE REPOSITORIES



CHALLENGE I: NOISY AND INCOMPLETE DATA

CHALLENGE II: EVALUATION

CHALLENGE III: PRACTICAL APPLICABILITY

TAKEAWAYS

Don’t trust data you’re using

Don’t trust data you’re using

Never get tired about manual validation

Don’t trust data you’re using

Never get tired about manual validation

One study is not enough

→ multiple studies give you different perspectives

Don’t trust data you’re using

Never get tired about manual validation

One study is not enough

→ multiple studies give you different perspectives

Humans need to be involved in tool evaluation

Don't trust data you're using

Never get tired about manual validation

One study is not enough

→ multiple studies give you different perspectives

Humans need to be involved in tool evaluation

Ask questions: dipenta@unisannio.it @mdipenta