Predicting Software Faults at Commit-Time Using Machine Learning

Wahab Hamou-Lhadj

ECE, Concordia University

wahab.hamou-lhadj@concordia.ca

UQAM, Montréal, QC
April 10, 2019

Acknowledgment

- This work is done with my PhD student Mathieu Nayrolles
- Mathieu received this Master’s degree from UQAM!
- Most of these techniques are published in his PhD thesis and various research papers

Motivations

- Maintenance of software systems can reach up **to 70% of the overall cost.**
- Up to **50%** of the overall maintenance cost can be spent **on identifying and correcting defects.**
- Defects in software cost the U.S. economy **$56 billion annually**.

NIST

U.S Department of Commerce
Technology Administration

Source: Health, Social and Research, E. 2002. The Economic Impacts of Inadequate Infrastructure for Software Testing

SW Development Challenges

- Increased complexity
- High cost
- Heavy reliance on people
- Lack of automated tools
- Time to market pressure
- Maintaining quality

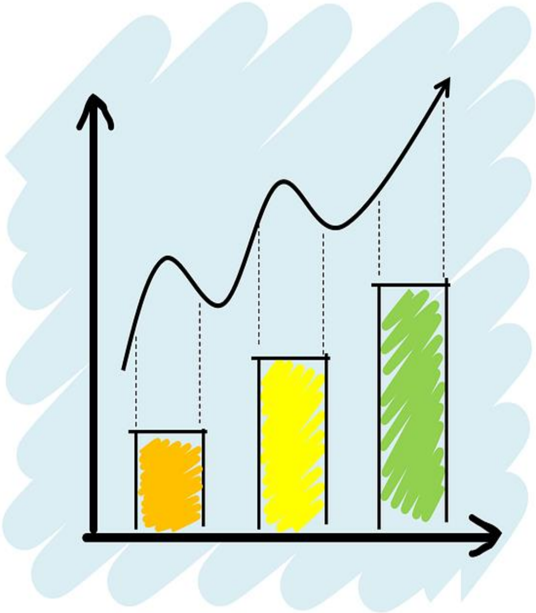
Fault Prediction Research

- Fault Prediction
  - Code or process metrics
  - Statistical analysis and call-graph analysis
  - Analysis of code changes
  - Leverage of historical data

- Automated Patch Generation
  - Development of fixing patterns
  - Reuse of human written patches
  - Directed patches towards specific bug types

Problems with existing techniques

- Offline processing (after the code is built)
- Presence of the entire source code
- Extensive setup and high learning curve
- Lack of clear actions to developers
- High rate of false positives



Our Solution

- Detection of potential bugs at commit-time
  - Before code is submitted to the central repository
- No external tools or setup required
  - Integration with developers' workflow
- Leverage of historical bugs and fixes
  - Learning from other people's mistakes
- Usefulness, usability, and scalability
  - High technology readiness level

A research roadmap

bumper

Find Out How Others Fixed It

Bug Metarepository Search Engine for Developers and Reseachers

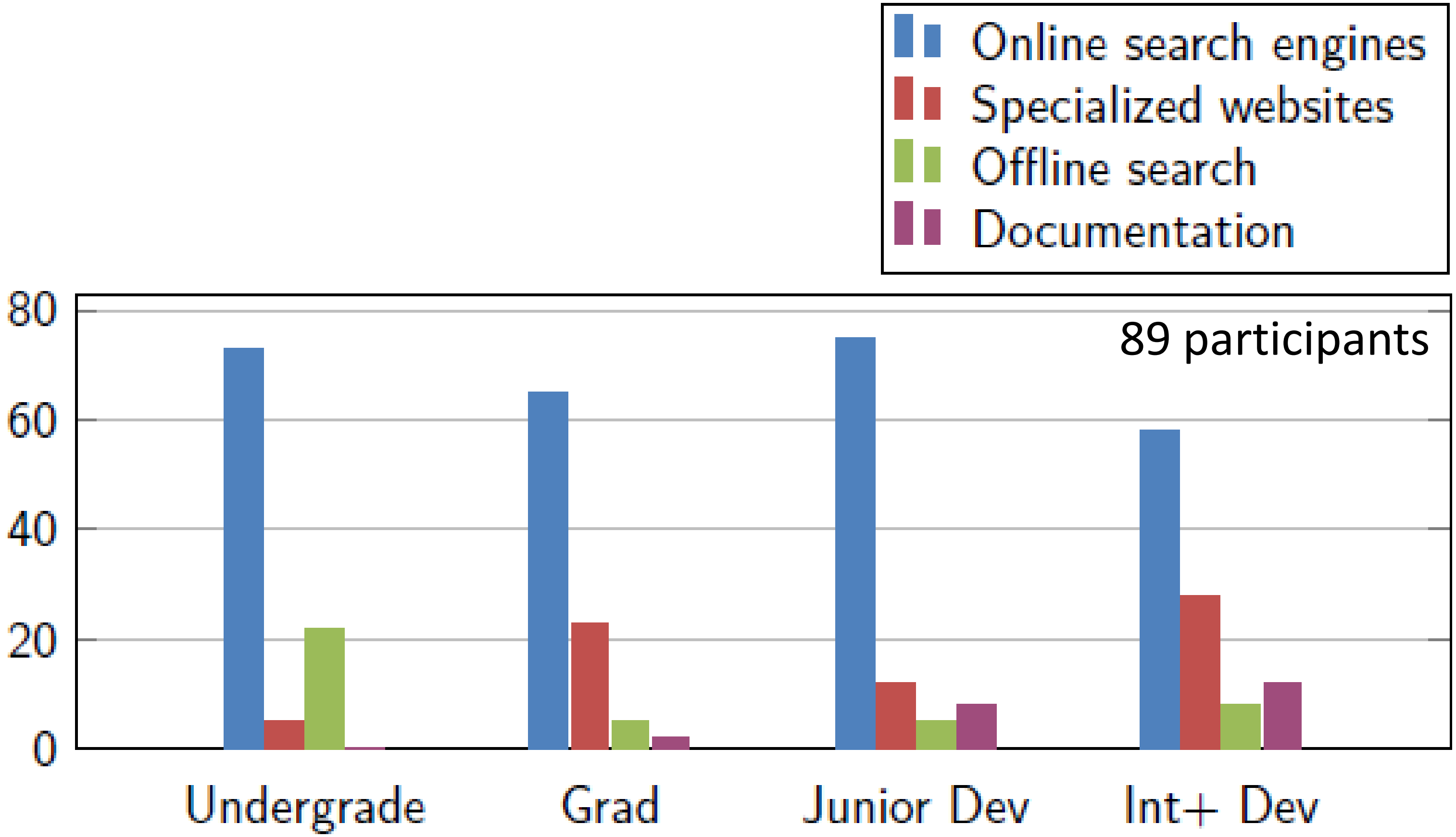
BIANCA

Preventing Bug Insertion at Commit-Time Using Clone Detection

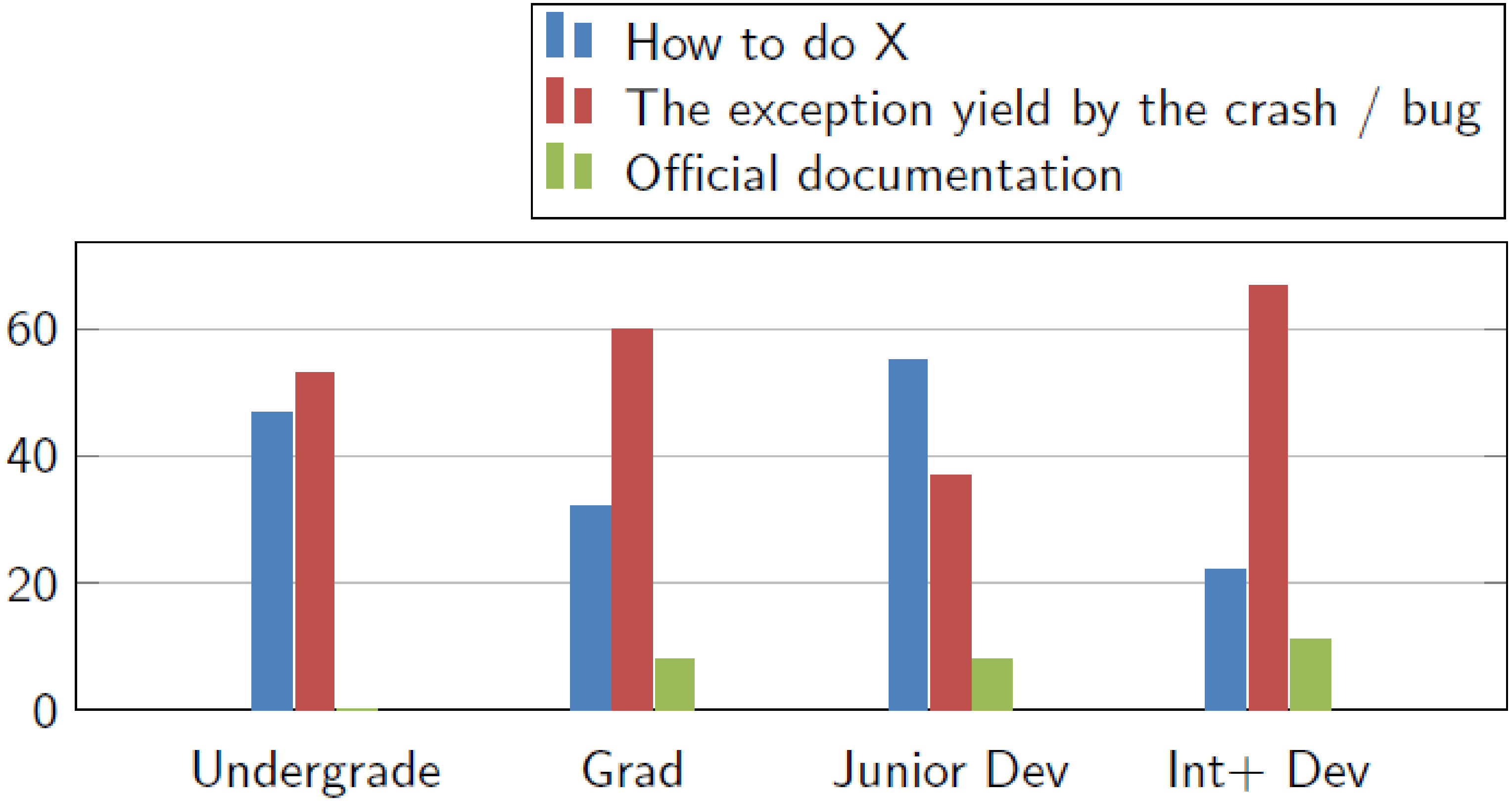
CLEVER

Combining Levels of Bug Prevention and Resolution Techniques

Where do developers look for information when facing an unknown bug/crash?



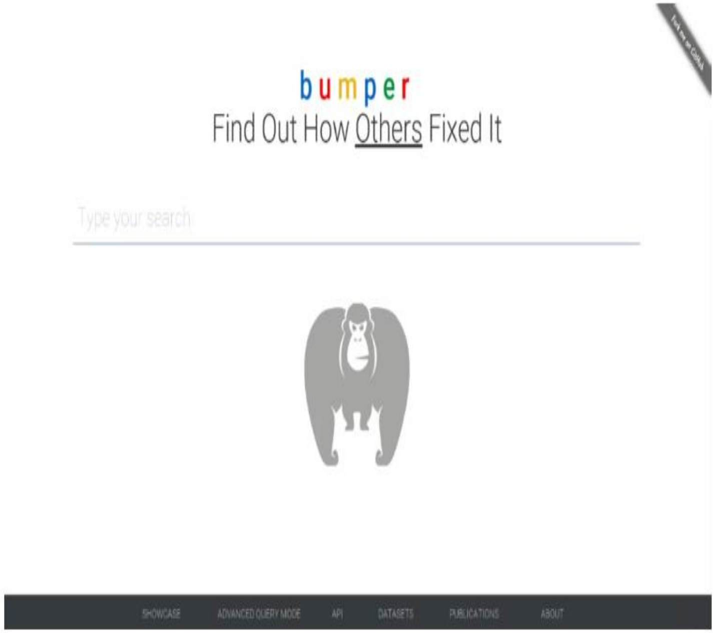
What do developers search for when facing an unknown bug/crash?



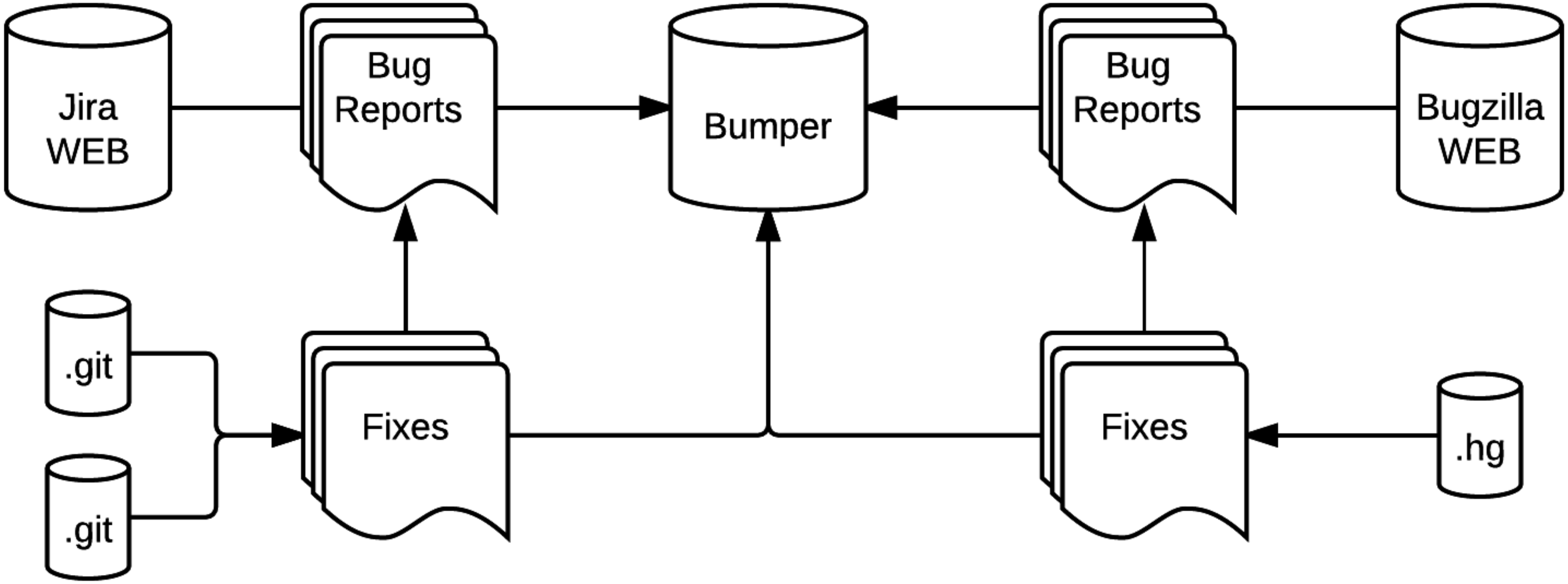
89 participants

BUMPER: Bug Metarepository Search Engine for Developers and Researchers

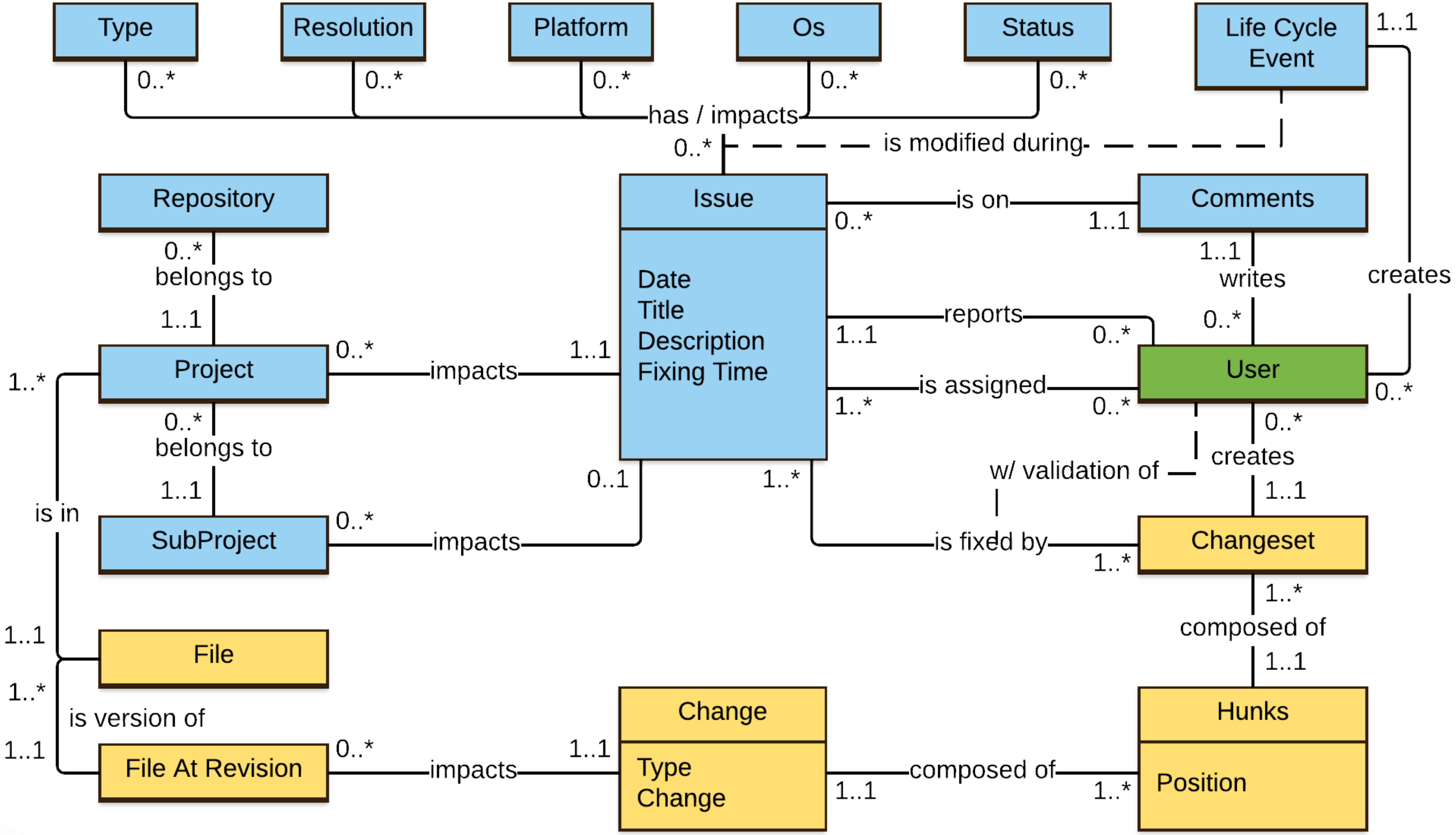
- Aggregates information from many bug report and code versioning systems
- Offers an online search engine to millions of bug reports and fixes from open-source repositories
- Uses a query system for developers and advanced API for researchers
- Leverages the concept of collective coding → collective intelligence

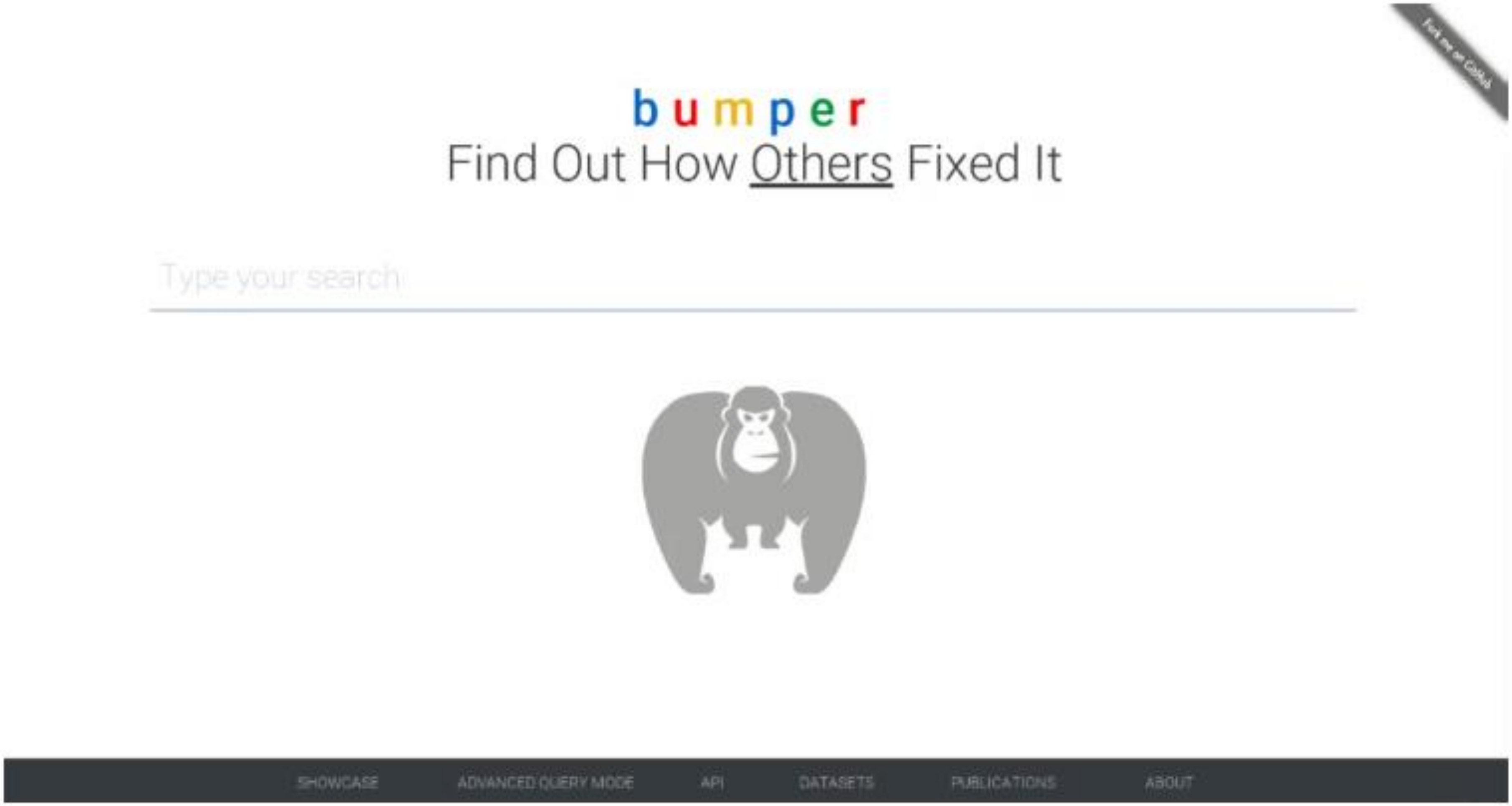


BUMPER Architecture



BUMPER Metamodel





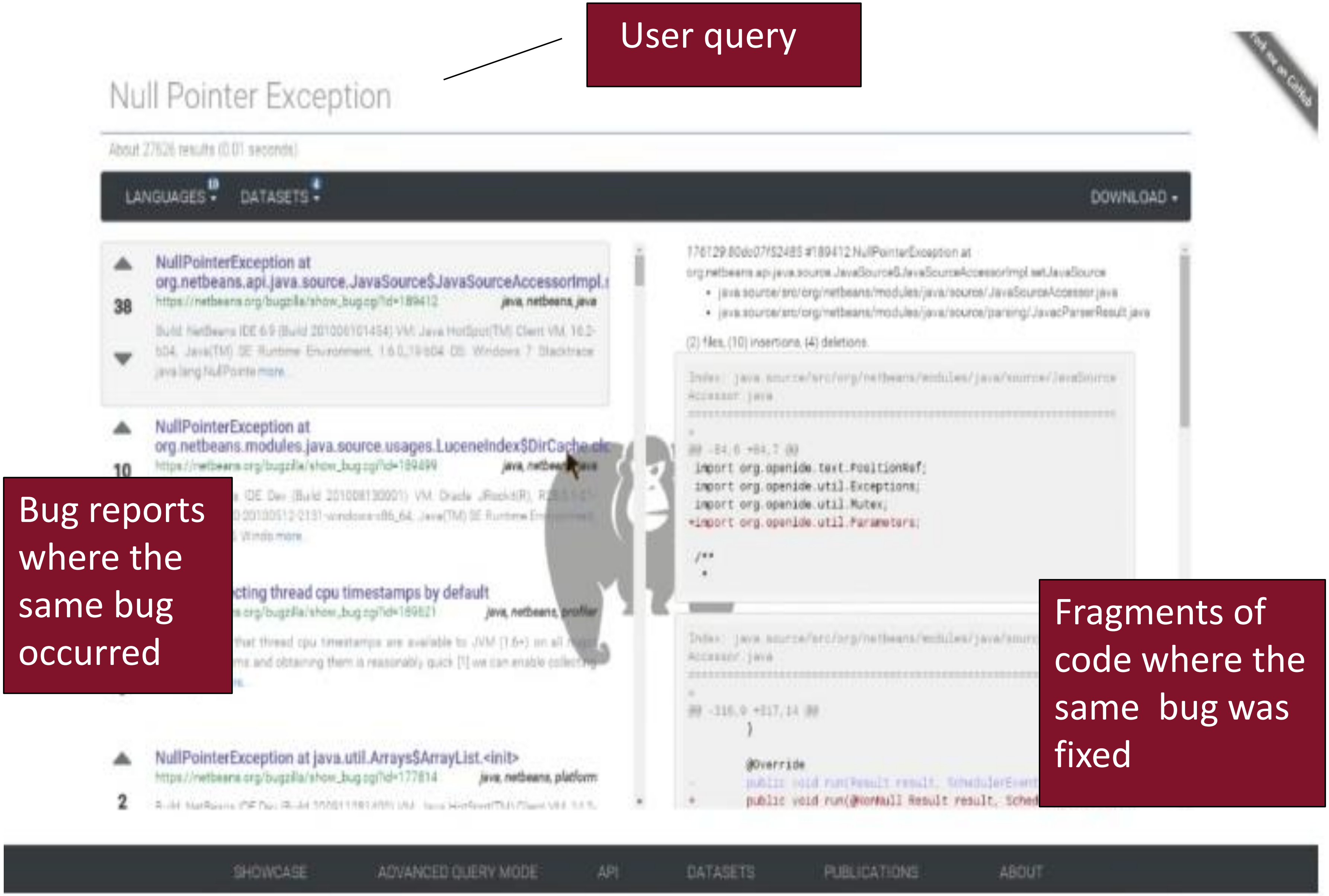
bumper

Find Out How Others Fixed It

Type your search

Developers can search millions of lines of code and bug reports for a bug or crash they encountered.

User query



Null Pointer Exception

About 27626 results (0.01 seconds)

38 **NullPointerException at org.netbeans.api.java.source.JavaSource$JavaSourceAccessorImpl.**
https://netbeans.org/bugzilla/show_bug.cgi?id=189412 — java, netbeans, java
Build: NetBeans IDE 6.9 (Build 201006101454) VM: Java HotSpot(TM) Client VM, 16.2-b04, Java(TM) SE Runtime Environment, 1.6.0_18-b04 OS: Windows 7 Stacktrace: java.lang.NullPointe more

10 **NullPointerException at org.netbeans.modules.java.source.usages.LuceneIndex$DirCache.clo**
https://netbeans.org/bugzilla/show_bug.cgi?id=189499 — java, netbeans, java
IDE Dev (Build 201008130001) VM: Oracle JRockit(R), R28.0.1-21 ... 20100512-2131-windows-x86_64, Java(TM) SE Runtime Environment ... Windo more

...cting thread cpu timestamps by default
...s.org/bugzilla/show_bug.cgi?id=189821 — java, netbeans, profiler
... that thread cpu timestamps are available to JVM (1.6+) on all major ... ms and obtaining them is reasonably quick [1] we can enable collecting ...

2 **NullPointerException at java.util.Arrays$ArrayList.<init>**
https://netbeans.org/bugzilla/show_bug.cgi?id=177814 — java, netbeans, platform
Build: NetBeans IDE Dev (Build 200911181400) VM: Java HotSpot(TM) Client VM, 14.3-

176129.80dc07f52485 #189412 NullPointerException at org.netbeans.api.java.source.JavaSource$JavaSourceAccessorImpl.setJavaSource

- java.source/src/org/netbeans/modules/java/source/JavaSourceAccessor.java
- java.source/src/org/netbeans/modules/java/source/parsing/JavacParserResult.java

(2) files, (10) insertions, (4) deletions.

```
Index: java.source/src/org/netbeans/modules/java/source/JavaSource
Accessor.java
===================================================================
-
@@ -84,6 +84,7 @@
 import org.openide.text.PositionRef;
 import org.openide.util.Exceptions;
 import org.openide.util.Mutex;
+import org.openide.util.Parameters;

 /**
  *
```

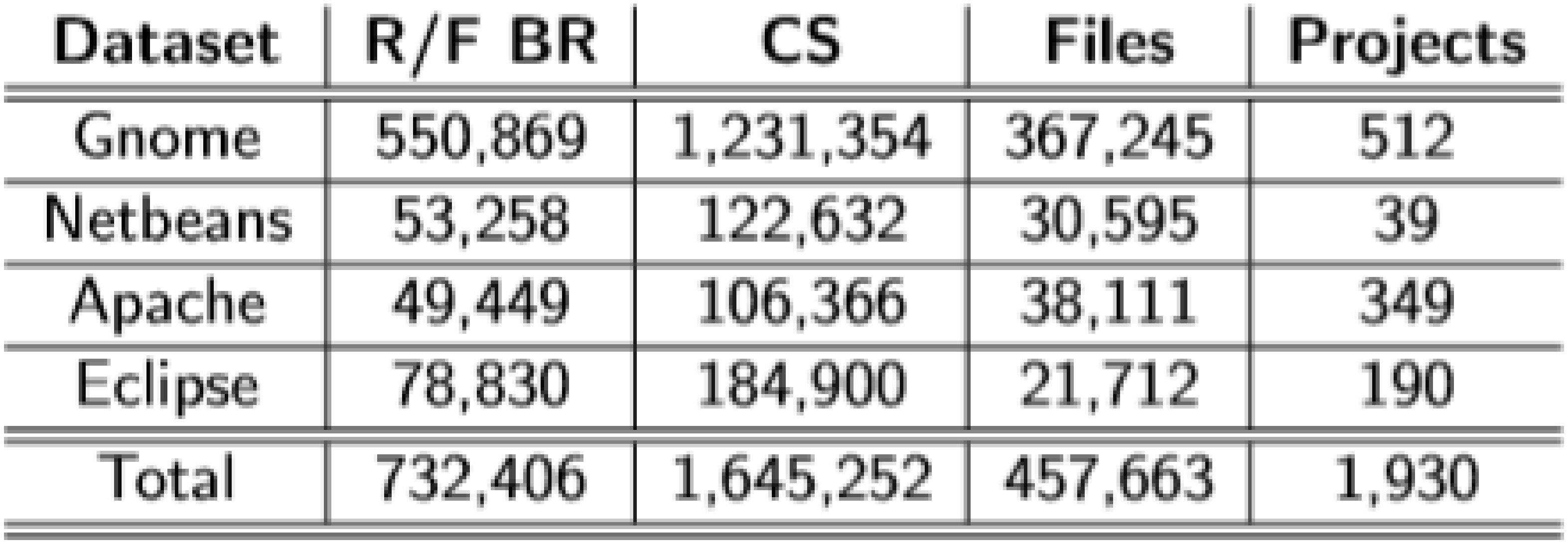
```
Index: java.source/src/org/netbeans/modules/java/sourc
Accessor.java
=========================================================
-
@@ -116,9 +117,14 @@
         }

         @Override
-        public void run(Result result, SchedulerEvent
+        public void run(@NonNull Result result, Sched
```

Bug reports where the same bug occurred

Fragments of code where the same bug was fixed

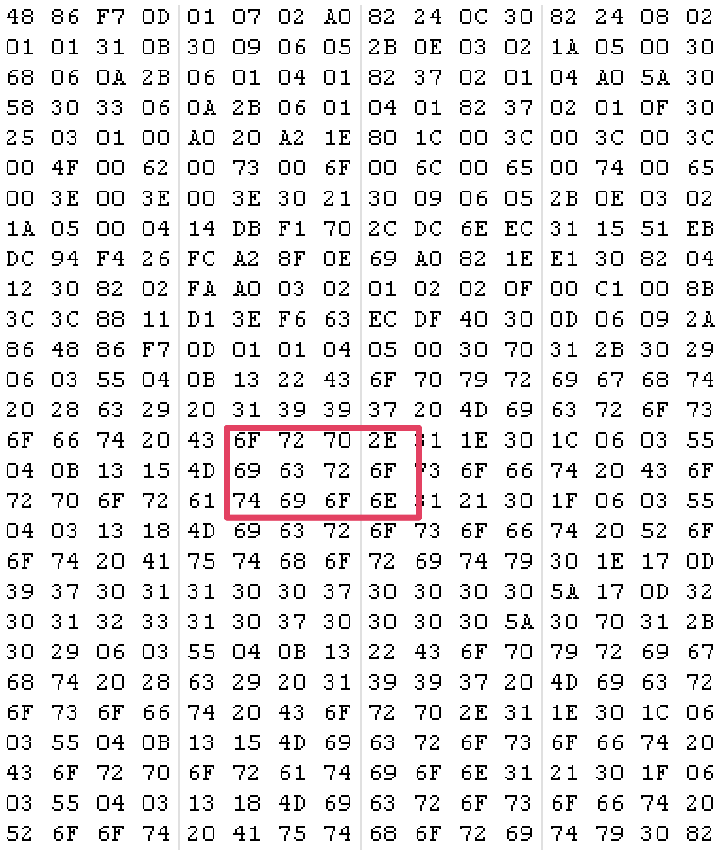
Supported projects



Dataset	R/F BR	CS	Files	Projects
Gnome	550,869	1,231,354	367,245	512
Netbeans	53,258	122,632	30,595	39
Apache	49,449	106,366	38,111	349
Eclipse	78,830	184,900	21,712	190
Total	732,406	1,645,252	457,663	1,930

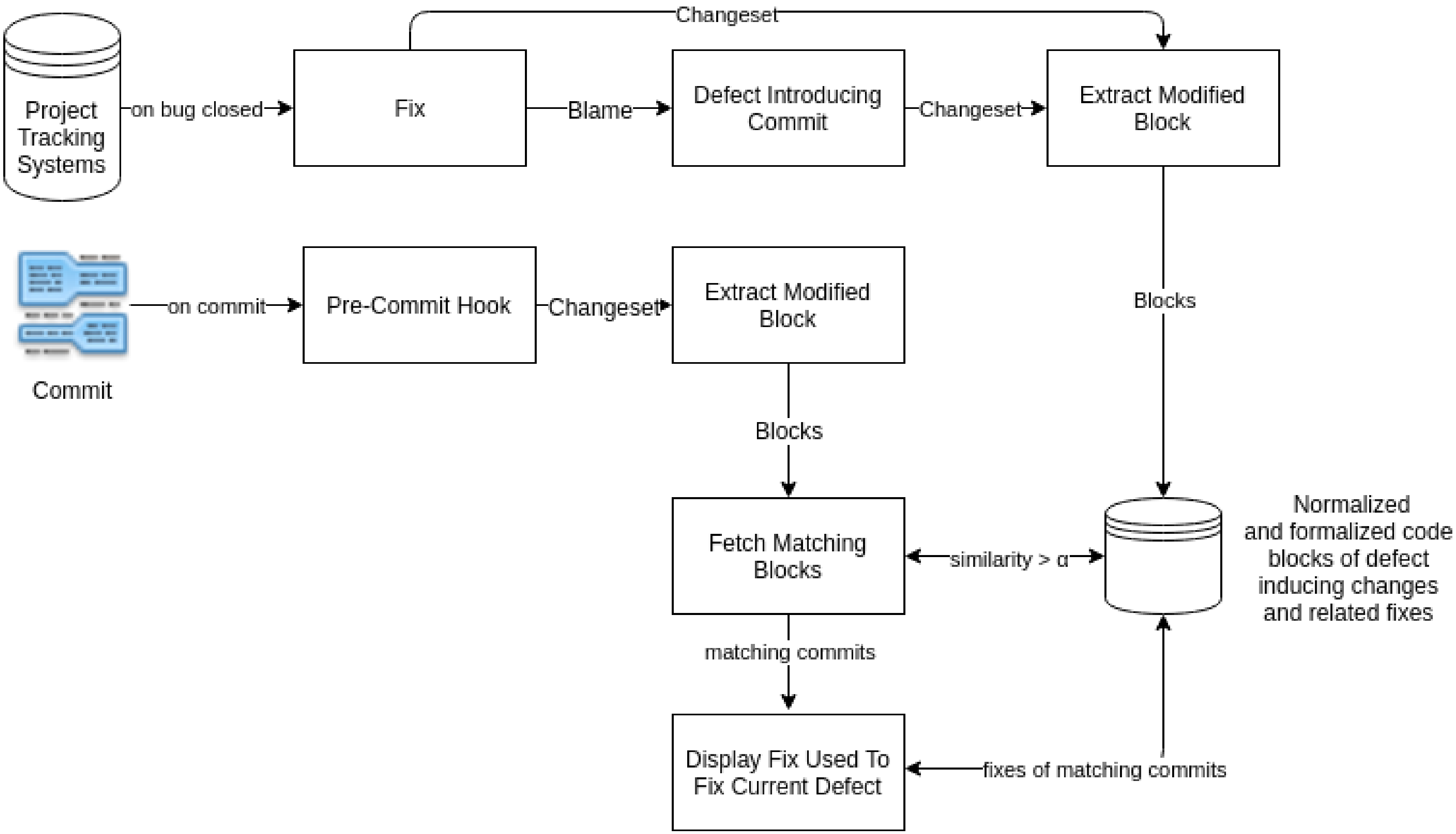
BIANCA: Preventing Bug Insertion at Commit-Time Using Clone Detection

- Learns known defects by mining BUMPER-indexed systems
- Builds a model of defects and their corresponding fixes
- Intercepts developer’s code and compares it to known defect signatures
- If a match exists, a flag is raised and a fix is proposed

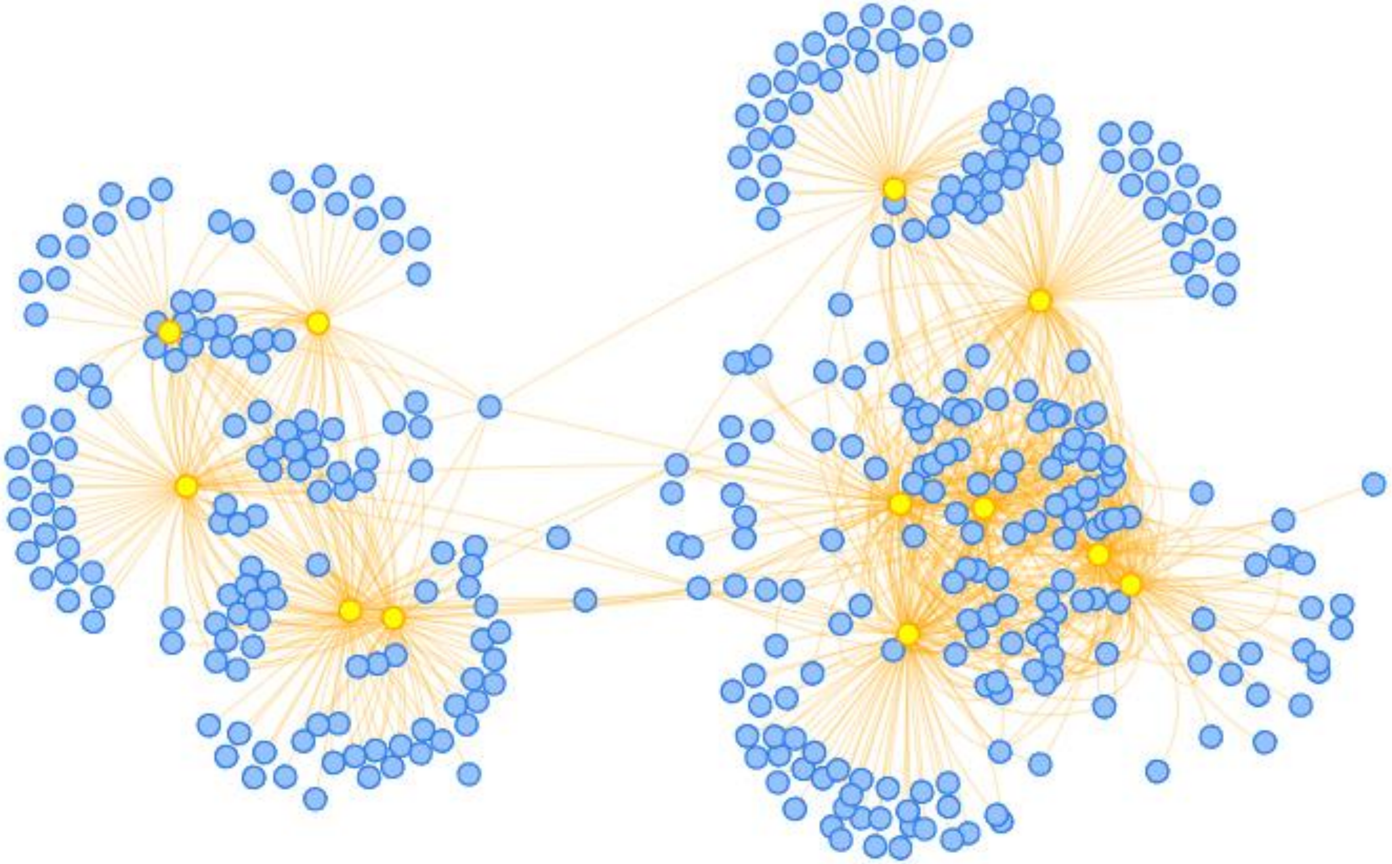


```
48 86 F7 0D | 01 07 02 A0 | 82 24 0C 30 | 82 24 08 02
01 01 31 0B | 30 09 06 05 | 2B 0E 03 02 | 1A 05 00 30
68 06 0A 2B | 06 01 04 01 | 82 37 02 01 | 04 A0 5A 30
58 30 33 06 | 0A 2B 06 01 | 04 01 82 37 | 02 01 0F 30
25 03 01 00 | A0 20 A2 1E | 80 1C 00 3C | 00 3C 00 3C
00 4F 00 62 | 00 73 00 6F | 00 6C 00 65 | 00 74 00 65
00 3E 00 3E | 00 3E 30 21 | 30 09 06 05 | 2B 0E 03 02
1A 05 00 04 | 14 DB F1 70 | 2C DC 6E EC | 31 15 51 EB
DC 94 F4 26 | FC A2 8F 0E | 69 A0 82 1E | E1 30 82 04
12 30 82 02 | FA A0 03 02 | 01 02 02 0F | 00 C1 00 8B
3C 3C 88 11 | D1 3E F6 63 | EC DF 40 30 | 0D 06 09 2A
86 48 86 F7 | 0D 01 01 04 | 05 00 30 70 | 31 2B 30 29
06 03 55 04 | 0B 13 22 43 | 6F 70 79 72 | 69 67 68 74
20 28 63 29 | 20 31 39 39 | 37 20 4D 69 | 63 72 6F 73
6F 66 74 20 | 43 6F 72 70 | 2E 31 1E 30 | 1C 06 03 55
04 0B 13 15 | 4D 69 63 72 | 6F 73 6F 66 | 74 20 43 6F
72 70 6F 72 | 61 74 69 6F | 6E 31 21 30 | 1F 06 03 55
04 03 13 18 | 4D 69 63 72 | 6F 73 6F 66 | 74 20 52 6F
6F 74 20 41 | 75 74 68 6F | 72 69 74 79 | 30 1E 17 0D
39 37 30 31 | 31 30 30 37 | 30 30 30 30 | 5A 17 0D 32
30 31 32 33 | 31 30 37 30 | 30 30 30 5A | 30 70 31 2B
30 29 06 03 | 55 04 0B 13 | 22 43 6F 70 | 79 72 69 67
68 74 20 28 | 63 29 20 31 | 39 39 37 20 | 4D 69 63 72
6F 73 6F 66 | 74 20 43 6F | 72 70 2E 31 | 1E 30 1C 06
03 55 04 0B | 13 15 4D 69 | 63 72 6F 73 | 6F 66 74 20
43 6F 72 70 | 6F 72 61 74 | 69 6F 6E 31 | 21 30 1F 06
03 55 04 03 | 13 18 4D 69 | 63 72 6F 73 | 6F 66 74 20
52 6F 6F 74 | 20 41 75 74 | 68 6F 72 69 | 74 79 30 82
```

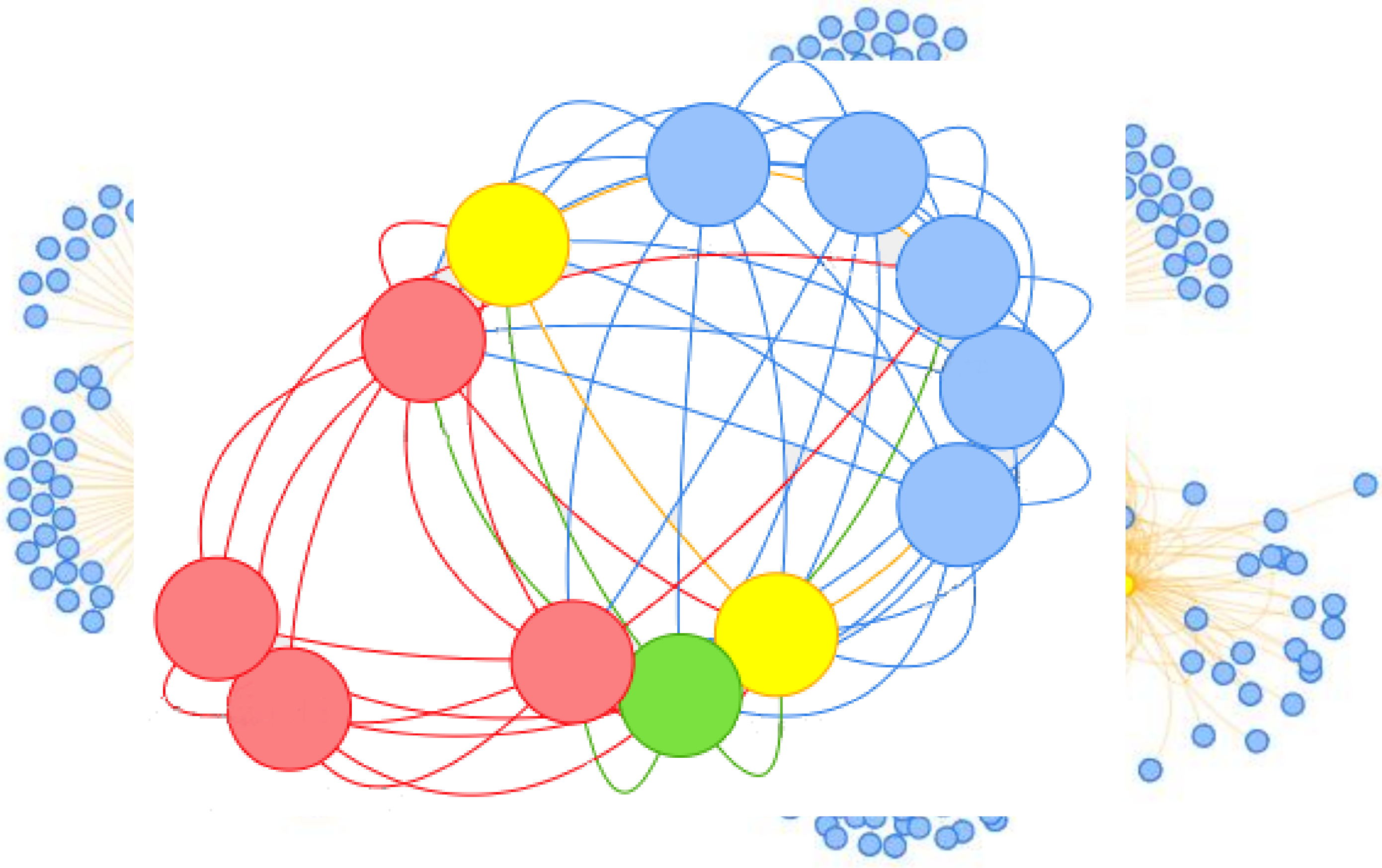
Approach


Changeset
Project Tracking Systems
on bug closed
Fix
Blame
Defect Introducing Commit
Changeset
Extract Modified Block
Commit
on commit
Pre-Commit Hook
Changeset
Extract Modified Block
Blocks
Blocks
Fetch Matching Blocks
similarity > α
Normalized and formalized code blocks of defect inducing changes and related fixes
matching commits
Display Fix Used To Fix Current Defect
fixes of matching commits

BIANCA works across projects and uses clustering techniques to create groups of related projects



BIANCA works across projects and uses clustering techniques to create groups of related projects



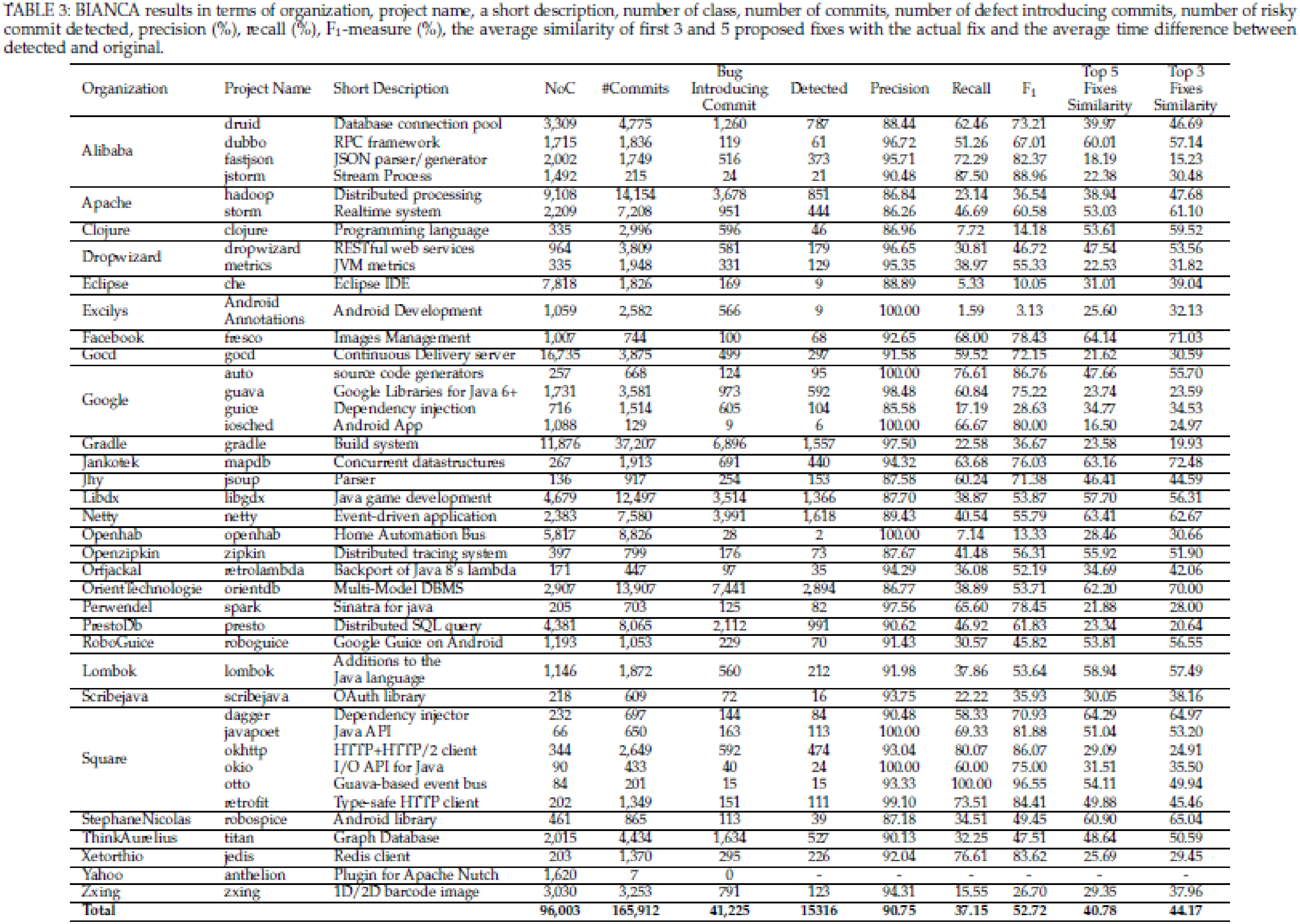


TABLE 3: BIANCA results in terms of organization, project name, a short description, number of class, number of commits, number of defect introducing commits, number of risky commit detected, precision (%), recall (%), F_1-measure (%), the average similarity of first 3 and 5 proposed fixes with the actual fix and the average time difference between detected and original.

Organization	Project Name	Short Description	NoC	#Commits	Bug Introducing Commit	Detected	Precision	Recall	F_1	Top 5 Fixes Similarity	Top 3 Fixes Similarity
Alibaba	druid	Database connection pool	3,309	4,775	1,260	787	88.44	62.46	73.21	39.97	46.69
	dubbo	RPC framework	1,715	1,836	119	61	96.72	51.26	67.01	60.01	57.14
	fastjson	JSON parser/ generator	2,002	1,749	516	373	95.71	72.29	82.37	18.19	15.23
	jstorm	Stream Process	1,492	215	24	21	90.48	87.50	88.96	22.38	30.48
Apache	hadoop	Distributed processing	9,108	14,154	3,678	851	86.84	23.14	36.54	38.94	47.68
	storm	Realtime system	2,209	7,208	951	444	86.26	46.69	60.58	53.03	61.10
Clojure	clojure	Programming language	335	2,996	596	46	86.96	7.72	14.18	53.61	59.52
Dropwizard	dropwizard	RESTful web services	964	3,809	581	179	96.65	30.81	46.72	47.54	53.56
	metrics	JVM metrics	335	1,948	331	129	95.35	38.97	55.33	22.53	31.82
Eclipse	che	Eclipse IDE	7,818	1,826	169	9	88.89	5.33	10.05	31.01	39.04
Excilys	Android Annotations	Android Development	1,059	2,582	566	9	100.00	1.59	3.13	25.60	32.13
Facebook	fresco	Images Management	1,007	744	100	68	92.65	68.00	78.43	64.14	71.03
Gocd	gocd	Continuous Delivery server	16,735	3,875	499	297	91.58	59.52	72.15	21.62	30.59
Google	auto	source code generators	257	668	124	95	100.00	76.61	86.76	47.66	55.70
	guava	Google Libraries for Java 6+	1,731	3,581	973	592	98.48	60.84	75.22	23.74	23.59
	guice	Dependency injection	716	1,514	605	104	85.58	17.19	28.63	34.77	34.53
	iosched	Android App	1,088	129	9	6	100.00	66.67	80.00	16.50	24.97
Gradle	gradle	Build system	11,876	37,207	6,896	1,557	97.50	22.58	36.67	23.58	19.93
Jankotek	mapdb	Concurrent datastructures	267	1,913	691	440	94.32	63.68	76.03	63.16	72.48
Jhy	jsoup	Parser	136	917	254	153	87.58	60.24	71.38	46.41	44.59
Libdx	libgdx	Java game development	4,679	12,497	3,514	1,366	87.70	38.87	53.87	57.70	56.31
Netty	netty	Event-driven application	2,383	7,580	3,991	1,618	89.43	40.54	55.79	63.41	62.67
Openhab	openhab	Home Automation Bus	5,817	8,826	28	2	100.00	7.14	13.33	28.46	30.66
Openzipkin	zipkin	Distributed tracing system	397	799	176	73	87.67	41.48	56.31	55.92	51.90
Orfjackal	retrolambda	Backport of Java 8's lambda	171	447	97	35	94.29	36.08	52.19	34.69	42.06
OrientTechnologie	orientdb	Multi-Model DBMS	2,907	13,907	7,441	2,894	86.77	38.89	53.71	62.20	70.00
Perwendel	spark	Sinatra for java	205	703	125	82	97.56	65.60	78.45	21.88	28.00
PrestoDb	presto	Distributed SQL query	4,381	8,065	2,112	991	90.62	46.92	61.83	23.34	20.64
RoboGuice	roboguice	Google Guice on Android	1,193	1,053	229	70	91.43	30.57	45.82	53.81	56.55
Lombok	lombok	Additions to the Java language	1,146	1,872	560	212	91.98	37.86	53.64	58.94	57.49
Scribejava	scribejava	OAuth library	218	609	72	16	93.75	22.22	35.93	30.05	38.16
Square	dagger	Dependency injector	232	697	144	84	90.48	58.33	70.93	64.29	64.97
	javapoet	Java API	66	650	163	113	100.00	69.33	81.88	51.04	53.20
	okhttp	HTTP+HTTP/2 client	344	2,649	592	474	93.04	80.07	86.07	29.09	24.91
	okio	I/O API for Java	90	433	40	24	100.00	60.00	75.00	31.51	35.50
	otto	Guava-based event bus	84	201	15	15	93.33	100.00	96.55	54.11	49.94
	retrofit	Type-safe HTTP client	202	1,349	151	111	99.10	73.51	84.41	49.88	45.46
StephaneNicolas	robospice	Android library	461	865	113	39	87.18	34.51	49.45	60.90	65.04
ThinkAurelius	titan	Graph Database	2,015	4,434	1,634	527	90.13	32.25	47.51	48.64	50.59
Xetorthio	jedis	Redis client	203	1,370	295	226	92.04	76.61	83.62	25.69	29.45
Yahoo	anthelion	Plugin for Apache Nutch	1,620	7	0	-	-	-	-	-	-
Zxing	zxing	1D/2D barcode image	3,030	3,253	791	123	94.31	15.55	26.70	29.35	37.96
Total			**96,003**	**165,912**	**41,225**	**15316**	**90.75**	**37.15**	**52.72**	**40.78**	**44.17**

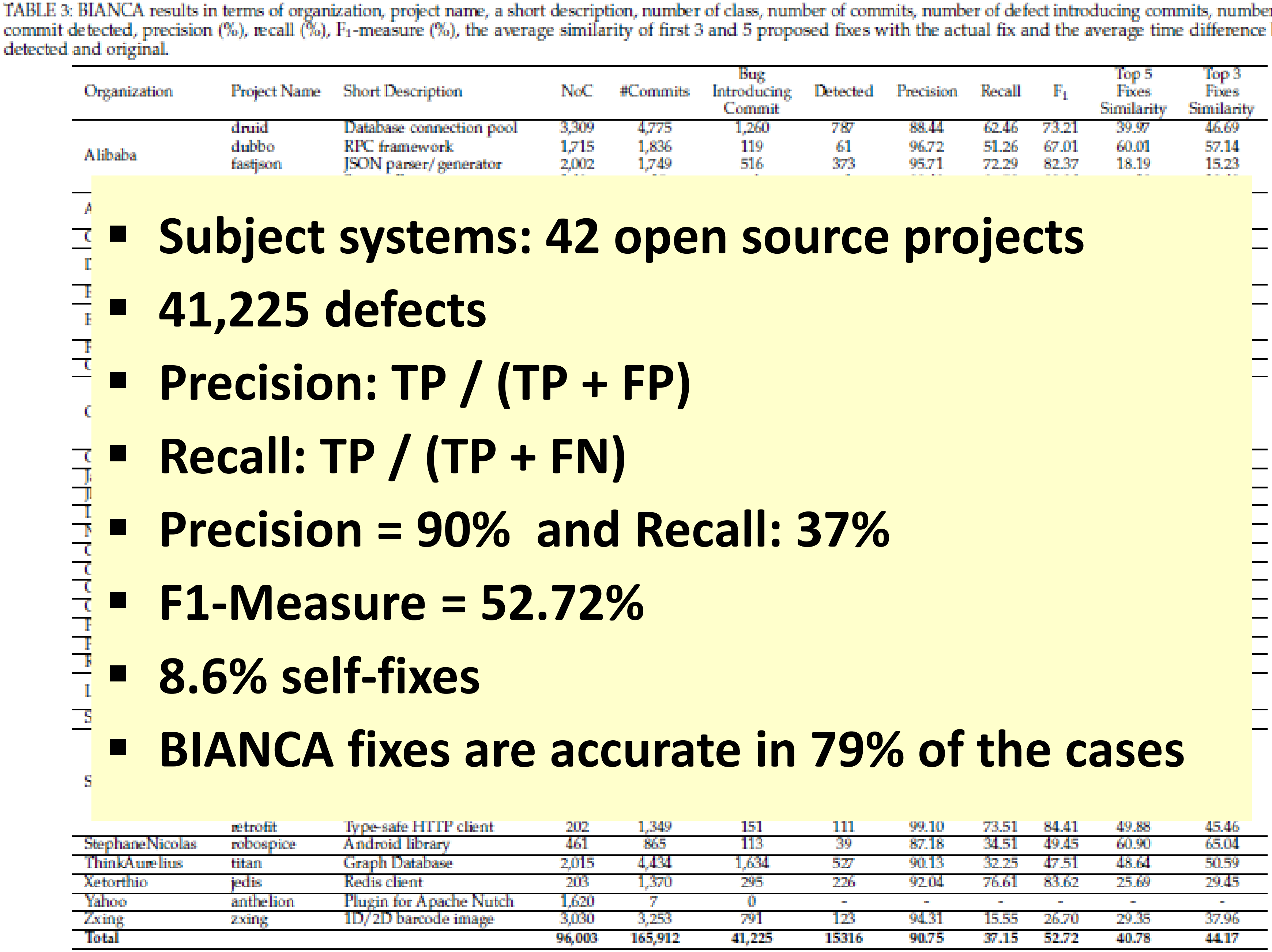


TABLE 3: BIANCA results in terms of organization, project name, a short description, number of class, number of commits, number of defect introducing commits, number of risky commit detected, precision (%), recall (%), F_1-measure (%), the average similarity of first 3 and 5 proposed fixes with the actual fix and the average time difference between detected and original.

Organization	Project Name	Short Description	NoC	#Commits	Bug Introducing Commit	Detected	Precision	Recall	F_1	Top 5 Fixes Similarity	Top 3 Fixes Similarity
Alibaba	druid	Database connection pool	3,309	4,775	1,260	787	88.44	62.46	73.21	39.97	46.69
	dubbo	RPC framework	1,715	1,836	119	61	96.72	51.26	67.01	60.01	57.14
	fastjson	JSON parser/ generator	2,002	1,749	516	373	95.71	72.29	82.37	18.19	15.23
	retrofit	Type-safe HTTP client	202	1,349	151	111	99.10	73.51	84.41	49.88	45.46
StephaneNicolas	robospice	Android library	461	865	113	39	87.18	34.51	49.45	60.90	65.04
ThinkAurelius	titan	Graph Database	2,015	4,434	1,634	527	90.13	32.25	47.51	48.64	50.59
Xetorthio	jedis	Redis client	203	1,370	295	226	92.04	76.61	83.62	25.69	29.45
Yahoo	anthelion	Plugin for Apache Nutch	1,620	7	0	-	-	-	-	-	-
Zxing	zxing	1D/2D barcode image	3,030	3,253	791	123	94.31	15.55	26.70	29.35	37.96
Total			96,003	165,912	41,225	15316	90.75	37.15	52.72	40.78	44.17

- Subject systems: 42 open source projects
- 41,225 defects
- Precision: TP / (TP + FP)
- Recall: TP / (TP + FN)
- Precision = 90% and Recall: 37%
- F1-Measure = 52.72%
- 8.6% self-fixes
- BIANCA fixes are accurate in 79% of the cases

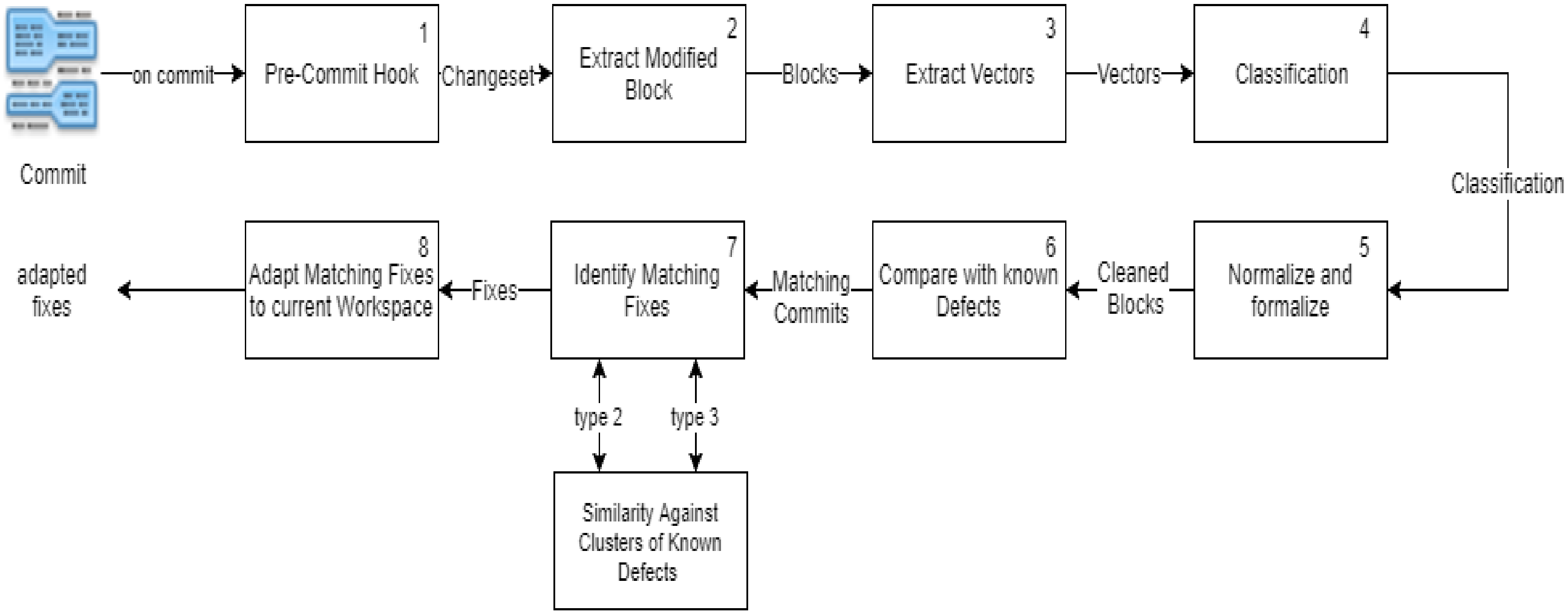
CLEVER: Combining Levels of Bug Prevention and Resolution Techniques

- Developed in the context on an NSERC project in collaboration with Ubisoft.
- Goal: To empower SW developers with an intelligent tool that detects defects as developers write code, and proposes fixes.

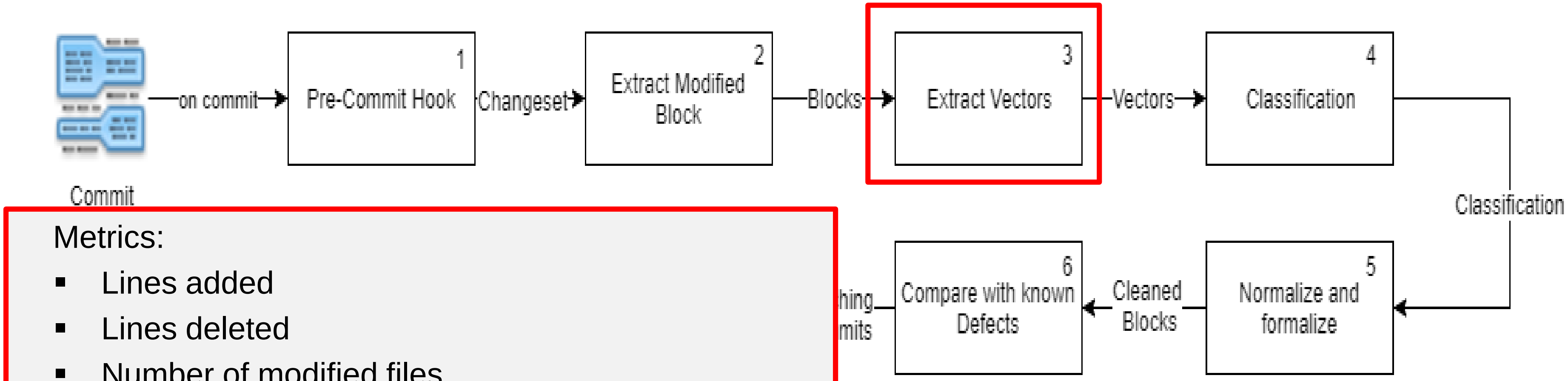
How does it work?

- Combines **code and process metrics** and clone detection to determine defect signatures
- Uses **various classification algorithms** (moving towards deep learning)
- Uses **domain expertise** to create clusters of projects for improved accuracy
- Uses **better code matching** techniques
- Is evaluated on industrial systems (12 Ubisoft systems)

Approach


Commit
on commit
1
Pre-Commit Hook
Changeset
2
Extract Modified Block
Blocks
3
Extract Vectors
Vectors
4
Classification
Classification
5
Normalize and formalize
Cleaned Blocks
6
Compare with known Defects
Matching Commits
7
Identify Matching Fixes
type 2
type 3
Similarity Against Clusters of Known Defects
Fixes
8
Adapt Matching Fixes to current Workspace
adapted fixes

Approach



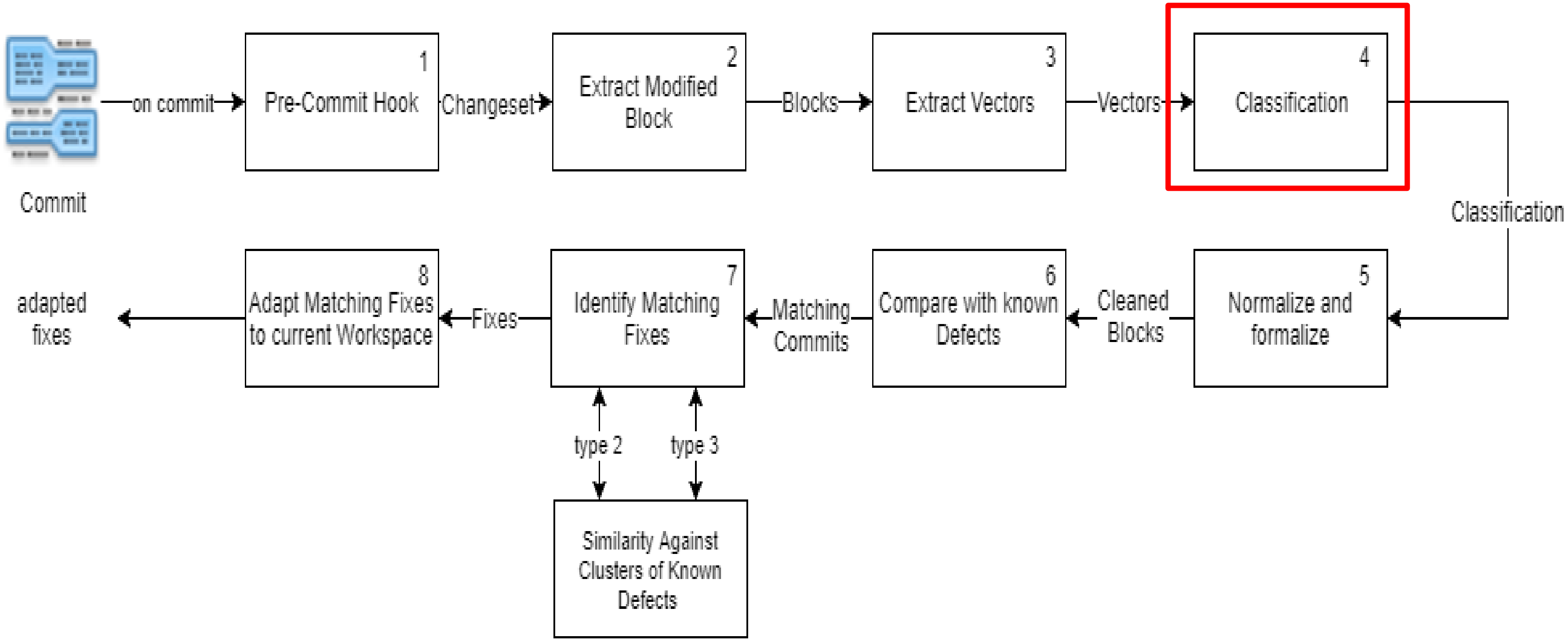
Metrics:

- Lines added
- Lines deleted
- Number of modified files
- Number of modified subsystems
- Number of modified directories
- Distribution of modified code across each file
- Number of developers that modified the files in a commit
- Total number of modified LOC across all files
- Etc.

Approach

Classification Algorithms

- Linear Regression
- Random Forest
- KNN
- Others



Evaluation of CLEVER at Ubisoft

- Subject systems: 12 Ubisoft systems
- Precision = 79%
- Recall = 65%
- Approved fixes: 67%

Impact

- Commit-Assistant (prototype implementation of CLEVER) is designed to integrate well with the workflow of Ubisoft developers

- Ubisoft announced in a press release that Commit-Assistant can cut the bug fixing time by 20%

- Mozilla announced that it is working with Ubisoft to contribute to Commit-Assistant and use it in the development of Firefox

Conclusion

- We proposed approaches to predict software faults at commit-time and propose fixes to developers
- These approaches rely of classification and code matching techniques
- We showed that these approaches can be used successfully in practice

Future Direction

- Experiment with more systems
- Add more machine learning techniques
- Reduce commit space for scalability
- Improve the recommendation of fixes
- Work on adopting the tool

CONCORDIA.CA