Tracking Patches that Fix Bugs found by Static Bug Finders

Dongsun Kim
Kyungpook National University

11th February 2022

KYUNGPOOK NATIONAL UNIVERSITY

~~Tracking Patches that Fix Bugs found by Static Bug Finders~~

Dongsun Kim
Kyungpook National University

11th February 2022

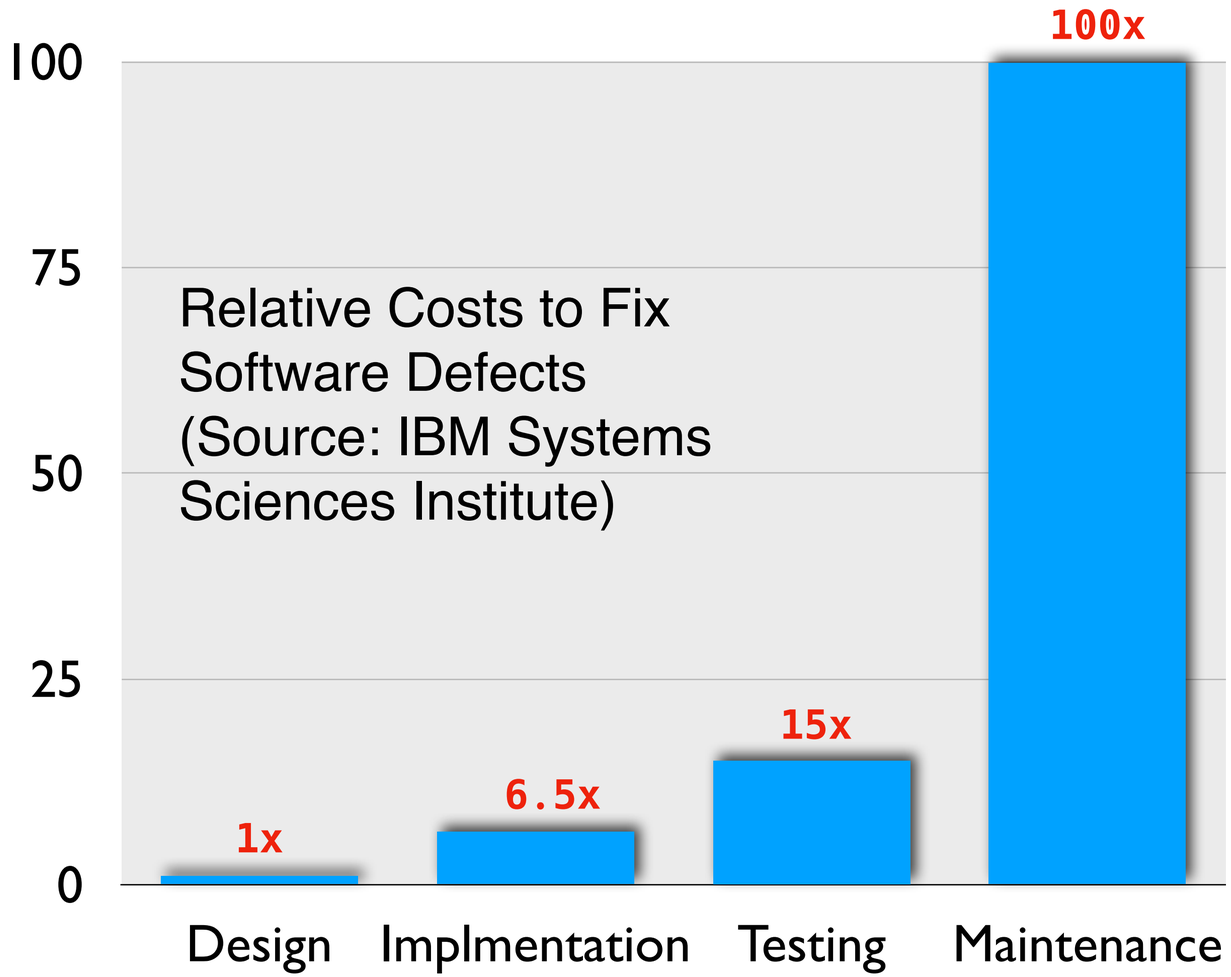
Automated Program Repair

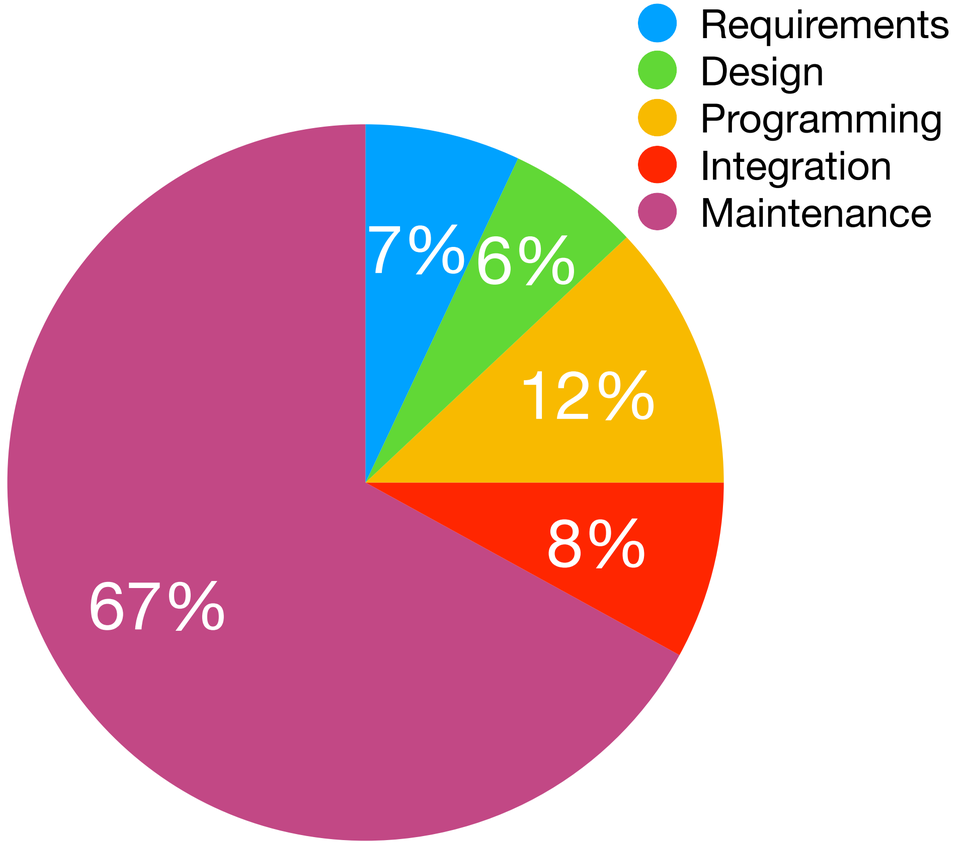
Dongsun Kim
Kyungpook National University

11th February 2022

Debugging

Debugging is expensive

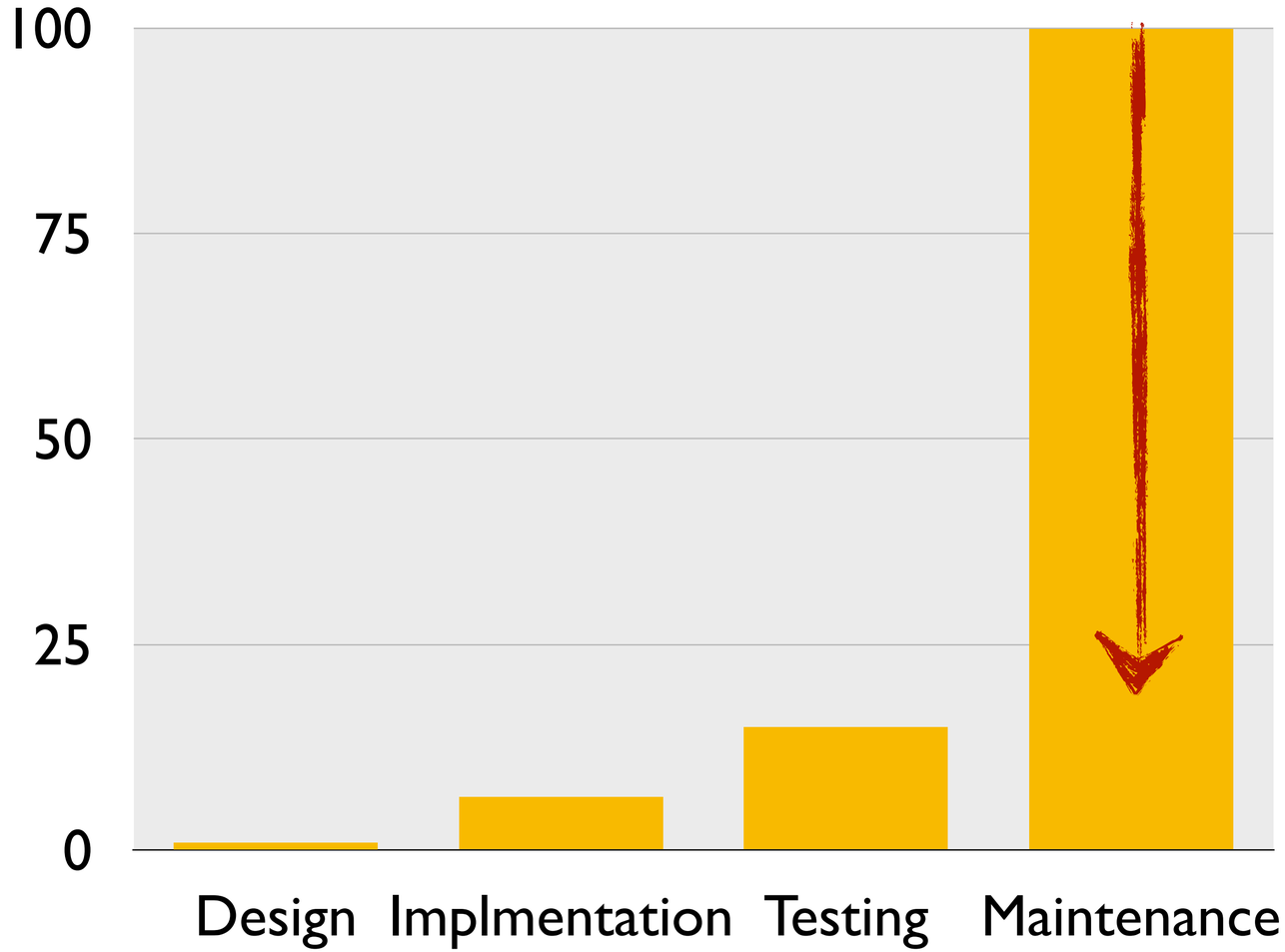




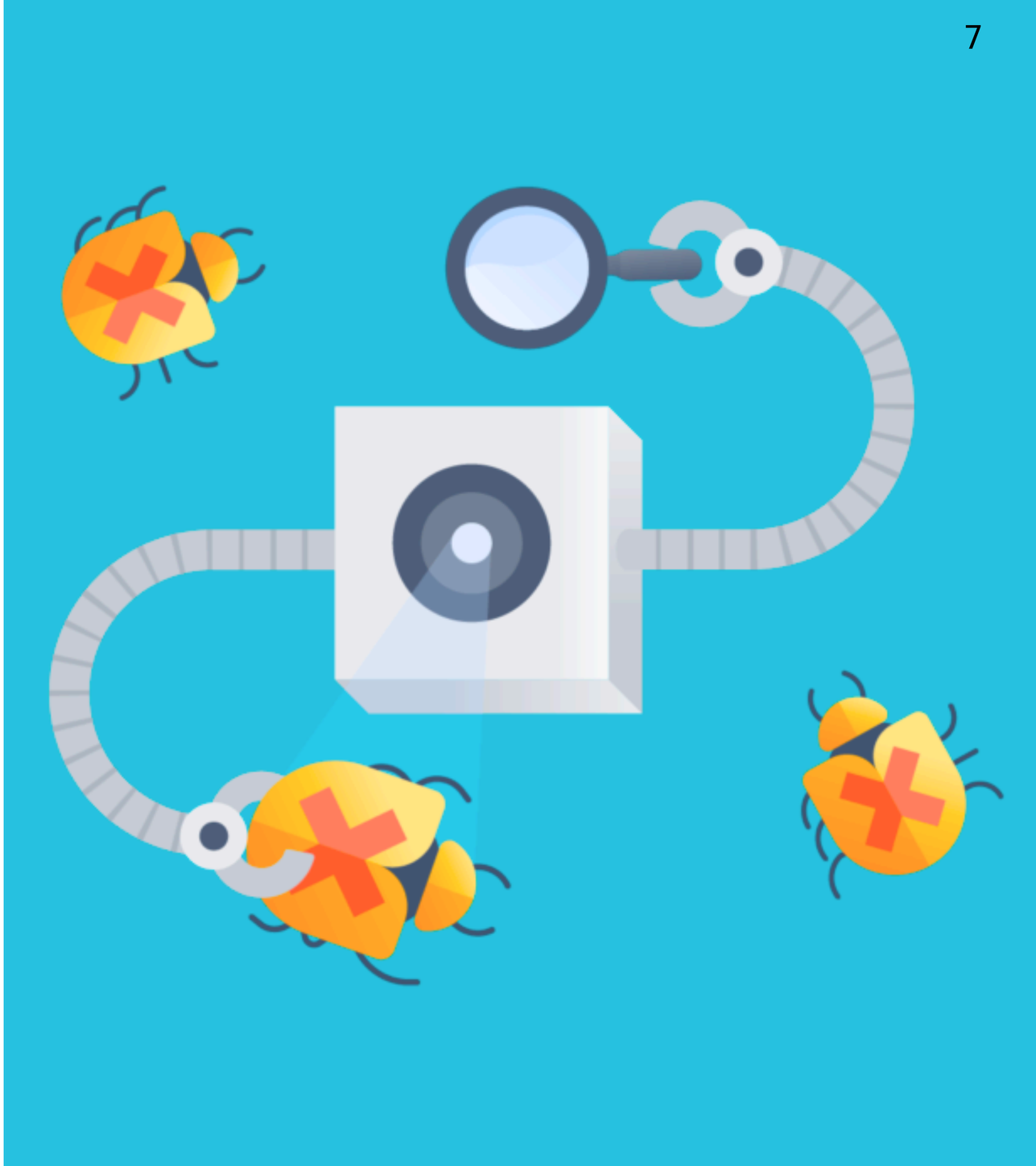
Approximate relative costs of the phases of the software life cycle

Schach, R. (1999), Software Engineering, Fourth Edition, McGraw-Hill, Boston, MA, pp. 11.

Goal of My Research

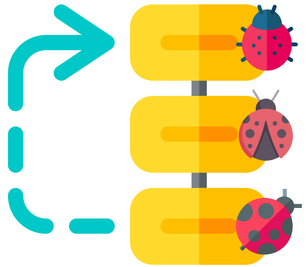


Automated Debugging

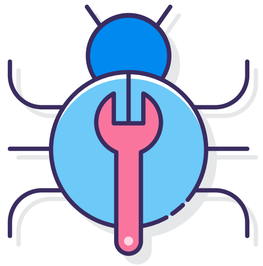


Localization

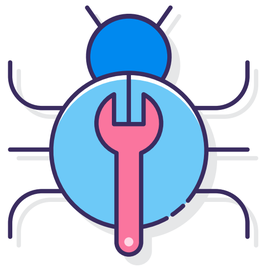
Prioritization



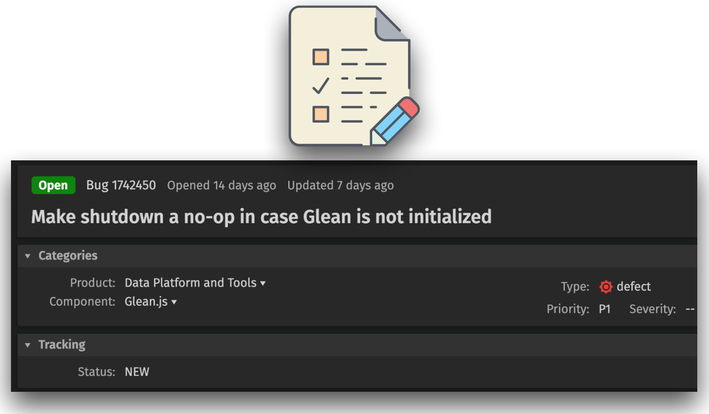
Repair



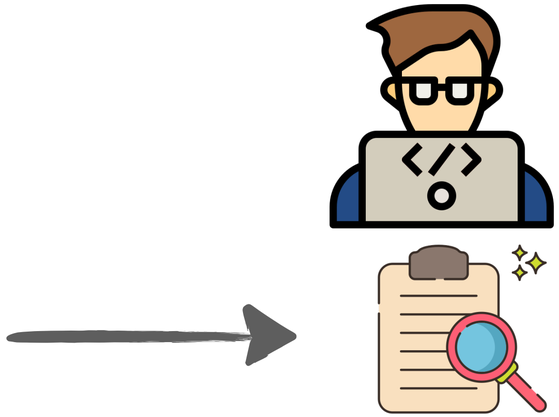
Repair



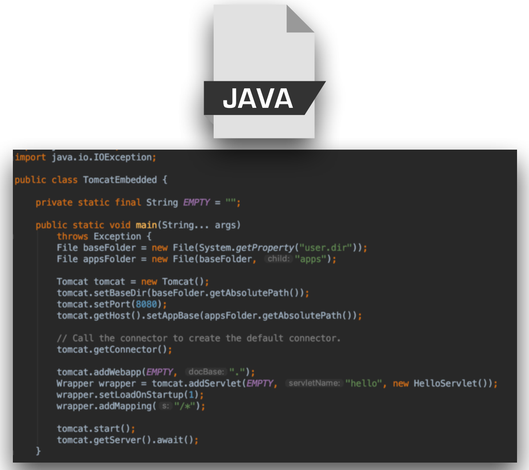
Fixing *one* bug



Bug Reports

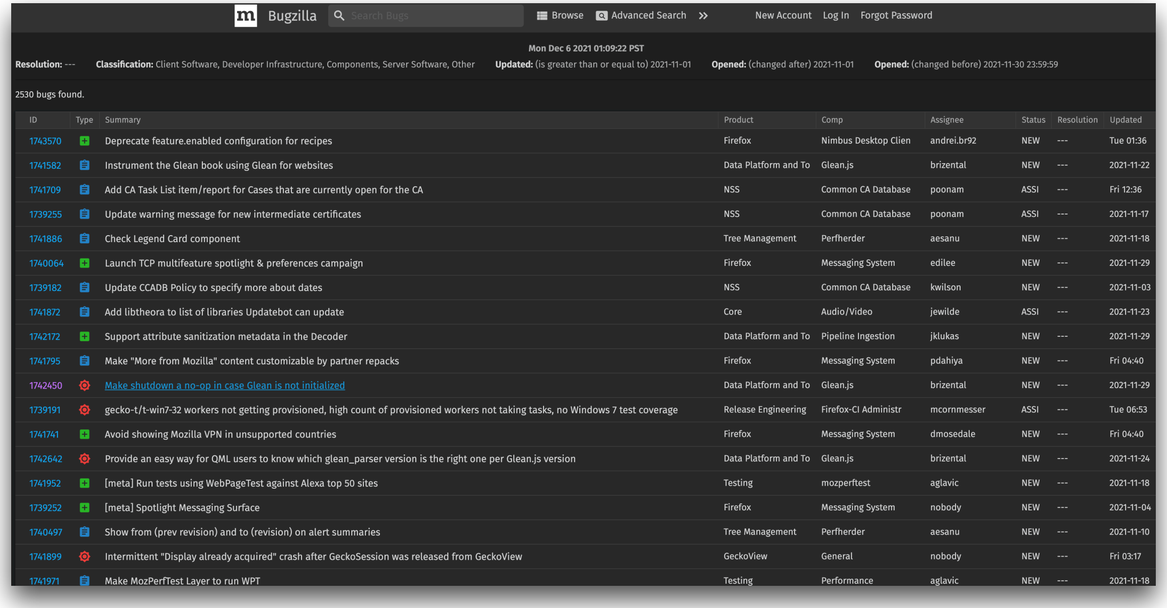


Inspection



Source code files

Fixing *thousands* of bugs?



Bugzilla | Search Bugs | Browse | Advanced Search | New Account | Log In | Forgot Password

Mon Dec 6 2021 01:09:22 PST

Resolution: --- **Classification:** Client Software, Developer Infrastructure, Components, Server Software, Other **Updated:** (is greater than or equal to) 2021-11-01 **Opened:** (changed after) 2021-11-01 **Opened:** (changed before) 2021-11-30 23:59:59

2530 bugs found.

ID	Type	Summary	Product	Comp	Assignee	Status	Resolution	Updated
1743570		Deprecate feature.enabled configuration for recipes	Firefox	Nimbus Desktop Clien	andrei.br92	NEW	---	Tue 01:36
1741582		Instrument the Glean book using Glean for websites	Data Platform and To	Glean.js	brizental	NEW	---	2021-11-22
1741709		Add CA Task List item/report for Cases that are currently open for the CA	NSS	Common CA Database	poonam	ASSI	---	Fri 12:36
1739255		Update warning message for new intermediate certificates	NSS	Common CA Database	poonam	ASSI	---	2021-11-17
1741886		Check Legend Card component	Tree Management	Perfherder	aesanu	NEW	---	2021-11-18
1740064		Launch TCP multifeature spotlight & preferences campaign	Firefox	Messaging System	edilee	NEW	---	2021-11-29
1739182		Update CCADB Policy to specify more about dates	NSS	Common CA Database	kwilson	NEW	---	2021-11-03
1741872		Add libtheora to list of libraries Updatebot can update	Core	Audio/Video	jewilde	ASSI	---	2021-11-23
1742172		Support attribute sanitization metadata in the Decoder	Data Platform and To	Pipeline Ingestion	jklukas	NEW	---	2021-11-29
1741795		Make "More from Mozilla" content customizable by partner repacks	Firefox	Messaging System	pdahiya	NEW	---	Fri 04:40
1742450		Make shutdown a no-op in case Glean is not initialized	Data Platform and To	Glean.js	brizental	NEW	---	2021-11-29
1739191		gecko-t/t-win7-32 workers not getting provisioned, high count of provisioned workers not taking tasks, no Windows 7 test coverage	Release Engineering	Firefox-CI Administr	mcornmesser	ASSI	---	Tue 06:53
1741741		Avoid showing Mozilla VPN in unsupported countries	Firefox	Messaging System	dmosedale	NEW	---	Fri 04:40
1742642		Provide an easy way for QML users to know which glean_parser version is the right one per Glean.js version	Data Platform and To	Glean.js	brizental	NEW	---	2021-11-24
1741952		[meta] Run tests using WebPageTest against Alexa top 50 sites	Testing	mozperftest	aglavic	NEW	---	2021-11-18
1739252		[meta] Spotlight Messaging Surface	Firefox	Messaging System	nobody	NEW	---	2021-11-04
1740497		Show from (prev revision) and to (revision) on alert summaries	Tree Management	Perfherder	aesanu	NEW	---	2021-11-10
1741899		Intermittent "Display already acquired" crash after GeckoSession was released from GeckoView	GeckoView	General	nobody	NEW	---	Fri 03:17
1741971		Make MozPerfTest Layer to run WPT	Testing	Performance	aglavic	NEW	---	2021-11-18

2,530 bugs are reported to the Mozilla projects within November 2021.

First Genuine Approach (GenProg)

Automatically Finding Patches Using Genetic Programming *

Westley Weimer
University of Virginia
weimer@virginia.edu

ThanhVu Nguyen
University of New Mexico
tnguyen@cs.unm.edu

Claire Le Goues
University of Virginia
legoues@virginia.edu

Stephanie Forrest
University of New Mexico
forrest@cs.unm.edu

Abstract

Automatic program repair has been a longstanding goal in software engineering, yet debugging remains a largely manual process. We introduce a fully automated method for locating and repairing bugs in software. The approach works on off-the-shelf legacy applications and does not require formal specifications, program annotations or special coding practices. Once a program fault is discovered, an extended form of genetic programming is used to evolve program variants until one is found that both retains required functionality and also avoids the defect in question. Standard test cases are used to exercise the fault and to encode program requirements. After a successful repair has been discovered, it is minimized using structural differencing al-

To alleviate this burden, we propose an automatic technique for repairing program defects. Our approach does not require difficult formal specifications, program annotations or special coding practices. Instead, it works on off-the-shelf legacy applications and readily-available testcases. We use genetic programming to evolve program variants until one is found that both retains required functionality and also avoids the defect in question. Our technique takes as input a program, a set of successful positive testcases that encode required program behavior, and a failing negative testcase that demonstrates a defect.

Genetic programming (GP) is a computational method inspired by biological evolution, which discovers computer programs tailored to a particular task [19]. GP maintains a population of individual programs. Computational analogs

W. Weimer, T. Nguyen, C. Le Goues, and S. Forrest, “Automatically finding patches using genetic programming,” in Proceedings of the 31st International Conference on Software Engineering, 2009, pp. 364–374. doi: http://dx.doi.org/10.1109/ICSE.2009.5070536.

generate-and-validate

A Buggy Program

```
public Field findsField(String name) {
  for (Field field : fieldsList) {
    if (field.getName().equals(name)) {
      return field;
    }
  }
  return null;
}
```

$$P = \{s_1, s_2, \ldots, s_n\}$$

generate-and-validate

A Buggy Program

```
public Field findsField(String name) {
  for (Field field : fieldsList) {
    if (field.getName().equals(name)) {
      return field;
    }
  }
  return null;
}
```

$$P = \{s_1, s_2, \ldots, s_n\}$$

Operators

e,g., Insert, Replace, Remove statements

generate-and-validate

A Buggy Program

```
public Field findsField(String name) {
  for (Field field : fieldsList) {
    if (field.getName().equals(name)) {
      return field;
    }
  }
  return null;
}
```

Operators

$P = \{s_1, s_2, \ldots, s_n\}$

e,g., Insert, Replace, Remove statements

generate-and-validate

A Buggy Program

```
public Field findsField(String name) {
  for (Field field : fieldsList) {
    if (field.getName().equals(name)) {
      return field;
    }
  }
  return null;
}
```

Operators

$P = \{s_1, s_2, \ldots, s_n\}$

e,g., Insert, Replace, Remove statements

Patch Candidates

$$P_1' = \{s'_1, s'_2, \ldots, s'_k\}$$
$$P_2' = \{\ldots\}$$
$$\vdots$$
$$P_m' = \{\ldots\}$$

generate-and-validate

Patch Candidates

$$P_1' = \{s'_1, s'_2, \ldots, s'_k\}$$
$$P_2' = \{\ldots\}$$
$$\vdots$$
$$P_m' = \{\ldots\}$$

Fitness Function

$$fitness(P') = |\{t \in T \mid P' \ \textbf{passes} \ t\}|$$

generate-and-validate

Operators

e,g., Insert, Replace, Remove statements

Patch Candidates

$$P_1' = \{s'_1, s'_2, \ldots, s'_k\}$$
$$P_2' = \{\ldots\}$$
$$\vdots$$
$$P_m' = \{\ldots\}$$

Fitness Function

$$fitness(P') = |\{t \in T \mid P' \textbf{ passes } t\}|$$

generate-and-validate

Operators

e,g., Insert, Replace, Remove statements

Patch Candidates

$$P_1' = \{s'_1, s'_2, \ldots, s'_k\}$$
$$P_2' = \{\ldots\}$$
$$\vdots$$
$$P_m' = \{\ldots\}$$

Fitness Function

$$fitness(P') = |\{t \in T \mid P' \textbf{ passes } t\}|$$

Iterate

Until

$$fitness(P_x) = |T|$$

generate-and-validate

Operators

e,g., Insert, Replace, Remove statements

Patch Candidates

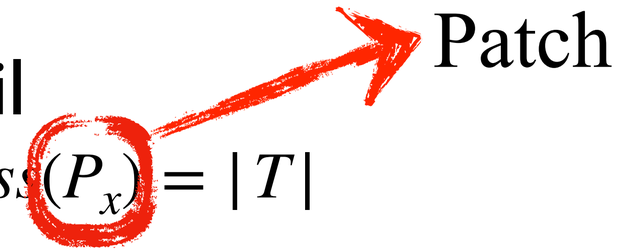
$$P_1' = \{s'_1, s'_2, \ldots, s'_k\}$$
$$P_2' = \{\ldots\}$$
$$\vdots$$
$$P_m' = \{\ldots\}$$

Fitness Function

$$fitness(P') = |\{t \in T \mid P' \textbf{ passes } t\}|$$

Iterate

Until $fitness(P_x) = |T|$



generate-and-validate

Semantic-based techniques

A Decade-long Effort

How many bugs are correctly (plausibly) fixed?

APR Tool	C	Cl	L	M	Mc	T	Total	CR(%)
jGenProg	0 (5)	0 (2)	0 (2)	3 (11)	0 (0)	0 (0)	3 (20)	15
GenProg-A	0 (5)	2 (15)	0 (1)	0 (9)	0 (0)	0 (0)	2 (30)	6.7
jMutRepair	1 (4)	2 (5)	0 (2)	2 (11)	0 (0)	0 (0)	5 (22)	22.7
kPAR	3 (13)	2 (10)	1 (18)	4 (22)	0 (0)	0 (1)	10 (63)	15.9
RSRepair-A	0 (4)	2 (22)	0 (3)	0 (12)	0 (0)	0 (0)	2 (41)	4.9
jKali	0 (4)	1 (8)	1 (4)	2 (9)	0 (0)	0 (0)	4 (25)	16
Kali-A	0 (6)	2 (48)	0 (0)	1 (10)	0 (1)	0 (0)	3 (**65**)	4.6
DynaMoth	0 (6)	N/A	0 (2)	1 (13)	0 (0)	0 (1)	1 (22)	4.5
Nopol	0 (6)	N/A	1 (6)	0 (18)	0 (0)	0 (1)	1 (31)	3.2
ACS	2 (2)	0 (0)	3 (3)	10 (16)	0 (0)	1 (1)	16 (22)	**72.7**
Cardumen	1 (4)	0 (2)	0 (0)	1 (6)	0 (0)	0 (0)	2 (12)	16.7
ARJA	1 (10)	2 (29)	0 (3)	4 (15)	0 (1)	0 (0)	7 (58)	12.1
SimFix	3 (8)	7 (19)	5 (16)	10 (25)	0 (0)	0 (0)	**25** (**68**)	**36.8**
FixMiner	5 (14)	0 (2)	0 (2)	7 (15)	0 (0)	0 (0)	12 (33)	**36.4**
AVATAR	5 (12)	7 (15)	4 (13)	3 (17)	0 (0)	0 (0)	**19** (57)	33.3
TBar	7 (16)	3 (12)	6 (21)	8 (23)	0 (0)	0 (0)	**24** (**72**)	30.8

(out of 395 bugs in Defects4J)

A Decade-long Effort

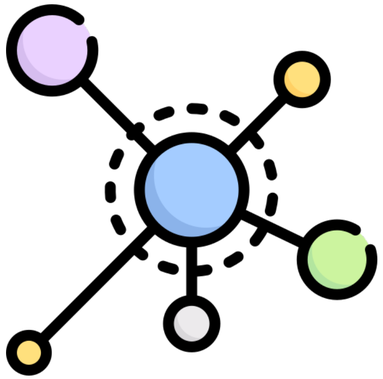
How many bugs are correctly (plausibly) fixed?

APR Tool	C	Cl	L	M	Mc	T	Total	CR(%)
jGenProg	0 (5)	0 (2)	0 (2)	3 (11)	0 (0)	0 (0)	3 (20)	15
GenProg-A	0 (5)	2 (15)	0 (1)	0 (9)	0 (0)	0 (0)	2 (30)	6.7
jMutRepair	1 (4)	2 (5)	0 (2)	2 (11)	0 (0)	0 (0)	5 (22)	22.7
kPAR	3 (13)	2 (10)	1 (18)	4 (22)	0 (0)	0 (1)	10 (63)	15.9
RSRepair-A	0 (4)	2 (22)	0 (3)	0 (12)	0 (0)	0 (0)	2 (41)	4.9
jKali	0 (4)	1 (8)	1 (4)	2 (9)	0 (0)	0 (0)	4 (25)	16
Kali-A	0 (6)	2 (48)	0 (0)	1 (10)	0 (1)	0 (0)	3 (**65**)	4.6
DynaMoth	0 (6)	N/A	0 (2)	1 (13)	0 (0)	0 (1)	1 (22)	4.5
Nopol	0 (6)	N/A	1 (6)	0 (18)	0 (0)	0 (1)	1 (31)	3.2
ACS	2 (2)	0 (0)	3 (3)	10 (16)	0 (0)	1 (1)	16 (22)	**72.7**
Cardumen	1 (4)	0 (2)	0 (0)	1 (6)	0 (0)	0 (0)	2 (12)	16.7
ARJA	1 (10)	2 (29)	0 (3)	4 (15)	0 (1)	0 (0)	7 (58)	12.1
SimFix	3 (8)	7 (19)	5 (16)	10 (25)	0 (0)	0 (0)	**25 (68)**	**36.8**
FixMiner	5 (14)	0 (2)	0 (2)	7 (15)	0 (0)	0 (0)	12 (33)	**36.4**
AVATAR	5 (12)	7 (15)	4 (13)	3 (17)	0 (0)	0 (0)	**19** (57)	33.3
TBar	7 (16)	3 (12)	6 (21)	8 (23)	0 (0)	0 (0)	**24 (72)**	30.8

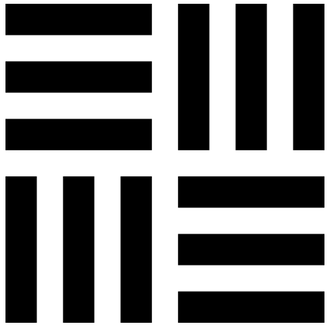
(out of 395 bugs in Defects4J)

Challenges

Defect Classes



Fault Localization



Repair Patterns

Correctness

Defect Classes

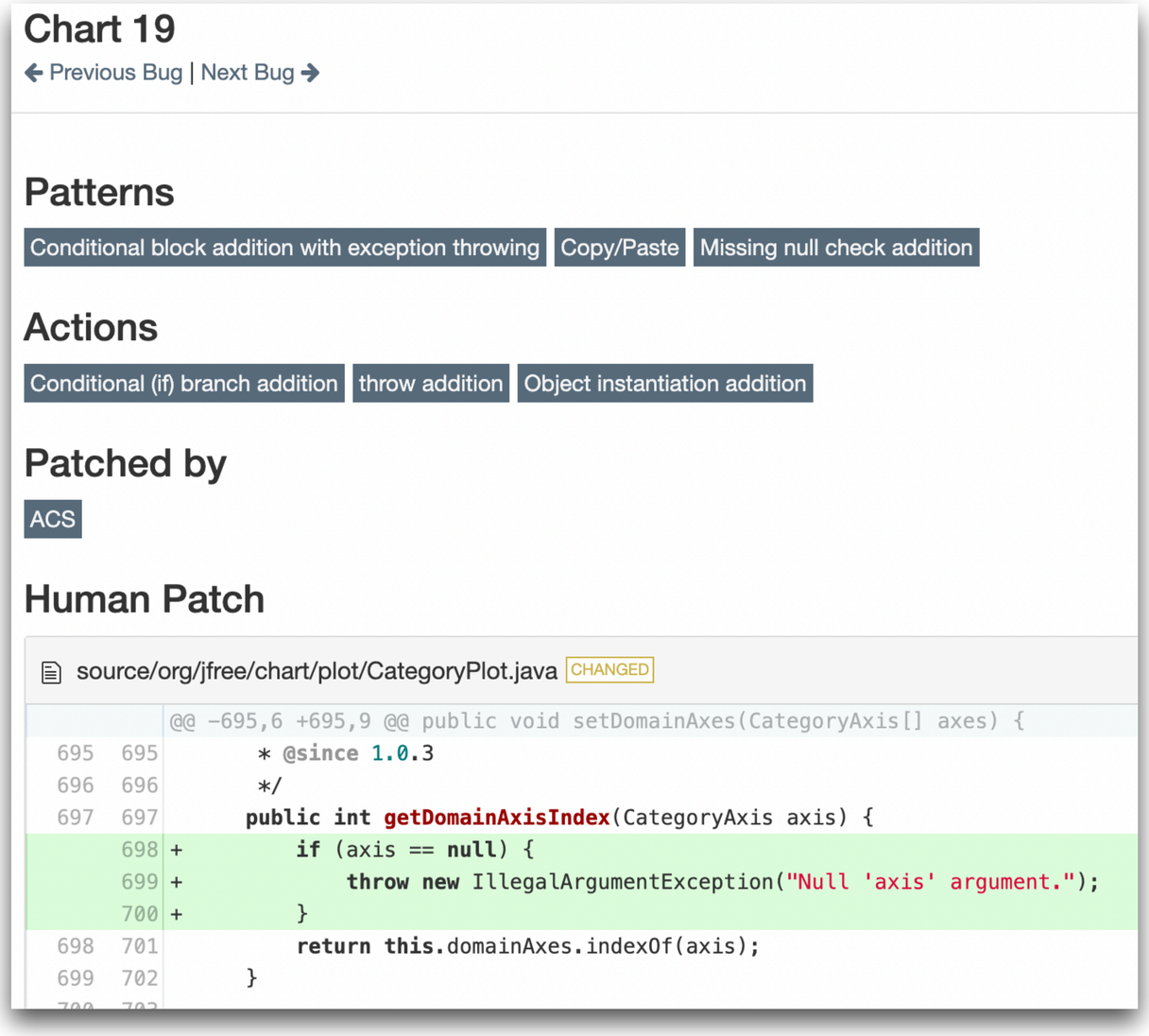


Chart 19

← Previous Bug | Next Bug →

Patterns

Conditional block addition with exception throwing | Copy/Paste | Missing null check addition

Actions

Conditional (if) branch addition | throw addition | Object instantiation addition

Patched by

ACS

Human Patch

source/org/jfree/chart/plot/CategoryPlot.java CHANGED

```
@@ -695,6 +695,9 @@ public void setDomainAxes(CategoryAxis[] axes) {
695 695         * @since 1.0.3
696 696         */
697 697        public int getDomainAxisIndex(CategoryAxis axis) {
    698 +          if (axis == null) {
    699 +              throw new IllegalArgumentException("Null 'axis' argument.");
    700 +          }
698 701            return this.domainAxes.indexOf(axis);
699 702        }
```

No comprehensive work on defect classes vs. program repair.

http://program-repair.org/defects4j-dissection/#!/

Fault Localization

Subjects	RP_{pn}	RP	$UbRP'$	$UbRP$	Sen
jGenProg	9	20	2	6	39.2%
jMutRepair	12	22	0	5	27.3%
jKali	13	25	1	2	51%
Nopol	29	31	1	2	71.8%
ACS	2	22	0	16	**4.5%**
ARJA	46	58	6	11	66.9%
kPAR	29	63	19	33	51.8%
SimFix	26	68	6	29	29.5%
FixMiner	16	33	22	34	56.6%
AVATAR	21	57	11	30	36.8%
TBar	27	72	29	54	45.6%

Liu *et al.*, "A critical review on the evaluation of automated program repair systems," *Journal of Systems and Software*, vol. 171, p. 110817, Jan. 2021.

Fault localization precision has a high impact on the program repair performance.

Patch Correctness

	Ground Truth		PATCH-SIM		BERT + LR	
Project	**Incorrect**	**Correct**	**Incorrect excluded (%)**	**Correct excluded**	**Incorrect excluded (%)**	**Correct excluded**
Chart	23	3	14(60.9%)	0	16(69.6%)	0
Lang	10	5	6(54.5%)	0	1(10%)	0
Math	63	20	33(52.4%)	0	23(36.5%)	0
Time	13	2	9(69.2%)	0	3(23.1%)	0
Total	109	30	62(56.3%)	0	43(39.4%)	0

H. Tian *et al.*, "Evaluating Representation Learning of Code Changes for Predicting Patch Correctness in Program Repair," in *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Sep. 2020, pp. 981–992.

Still too many **incorrect** patches are generated by program repair techqniues.

~~Tracking Patches that Fix Bugs found by Static Bug Finders~~

Dongsun Kim
Kyungpook National University

11th February 2022

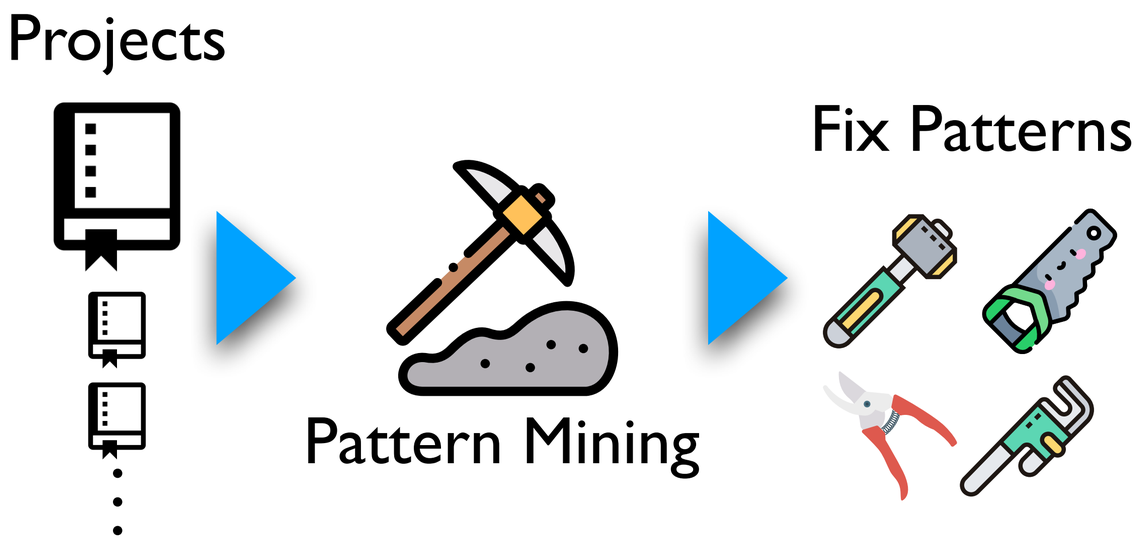
Tracking Patches that Fix Bugs found by Static Bug Finders

K. Liu, D. Kim, T. F. Bissyandé, S. Yoo, and Y. L. Traon, “Mining Fix Patterns for FindBugs Violations,” *IEEE Transactions on Software Engineering*, vol. 47, no. 1, pp. 165–188, Jan. 2021, doi: 10.1109/TSE.2018.2884955.

Dongsun Kim
Kyungpook National University

11th February 2022

Pattern-based Program Repair



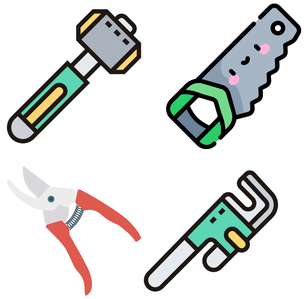
Pattern-based Program Repair

Buggy Program

Projects

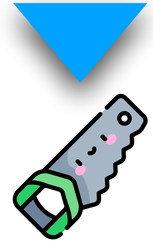
Pattern Mining

Fix Patterns



Pattern-based Program Repair

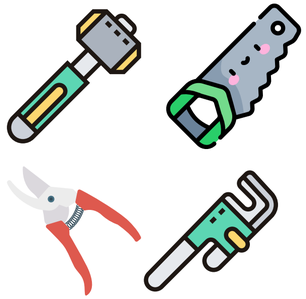
Buggy Program



Projects

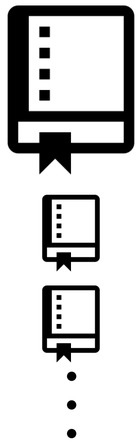
Pattern Mining

Fix Patterns



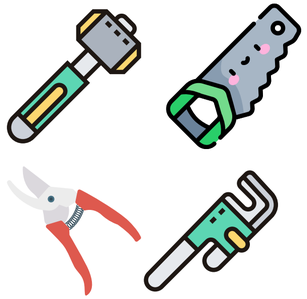
Pattern-based Program Repair

Projects

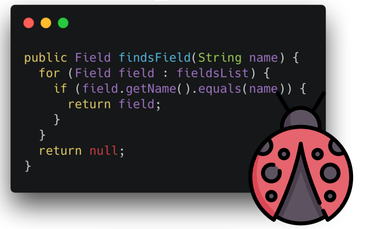


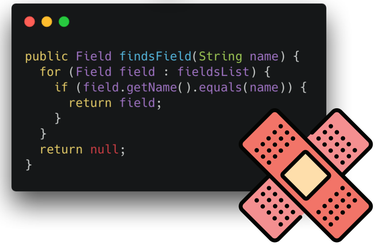
Pattern Mining

Fix Patterns



Buggy Program

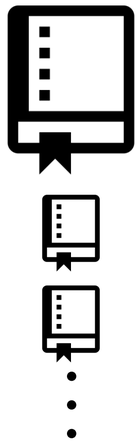




Fixed Program

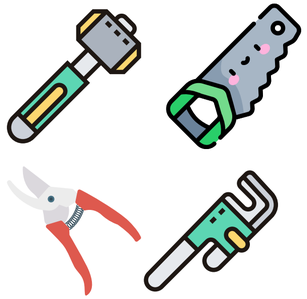
Pattern-based Program Repair

Projects



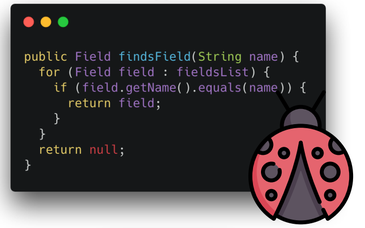
Pattern Mining

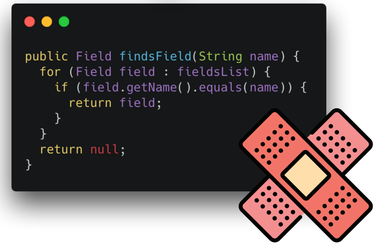
Fix Patterns



Automated

Buggy Program

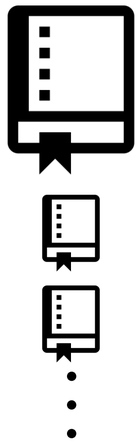




Fixed Program

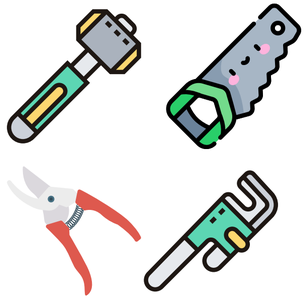
Pattern-based Program Repair

Projects



Pattern Mining

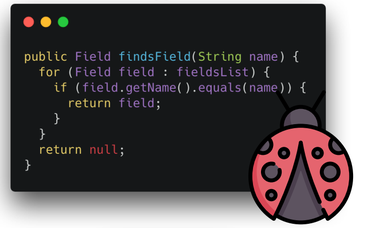
Fix Patterns

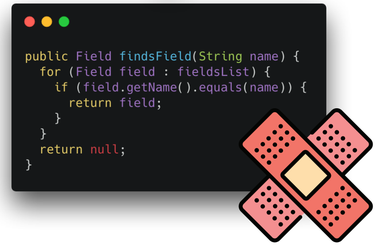


Still by Manual!

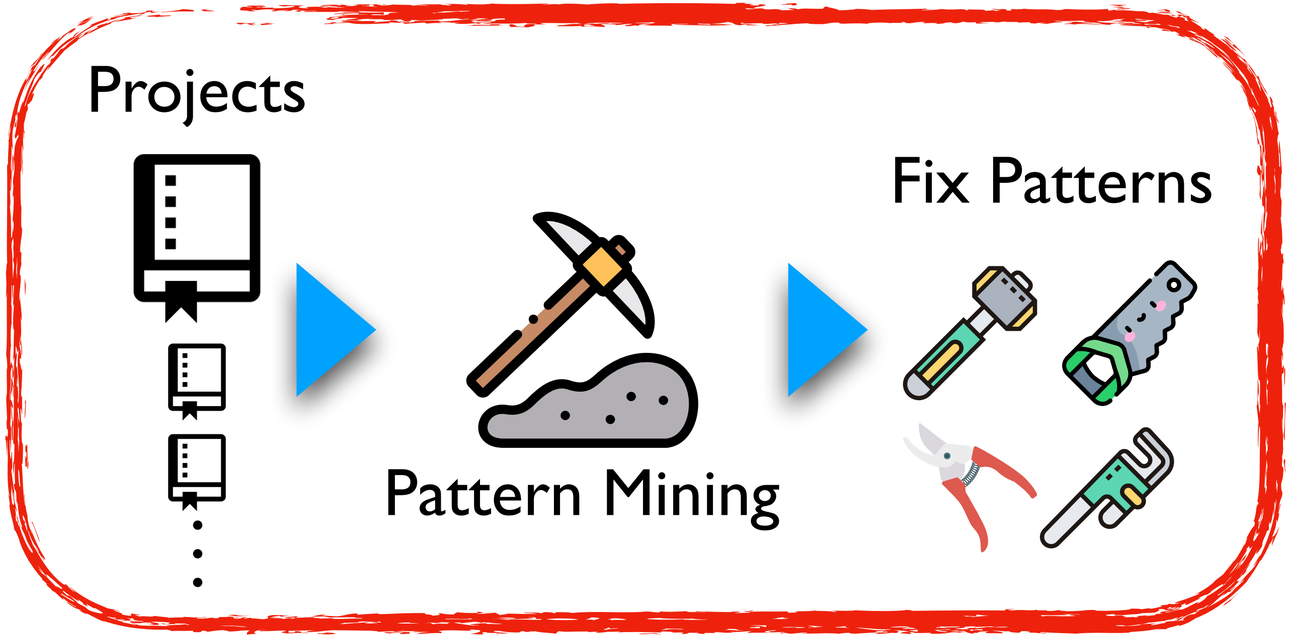
Automated

Buggy Program





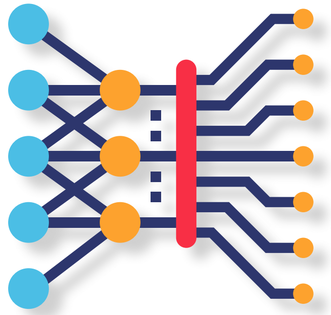
Fixed Program



Still by Manual!

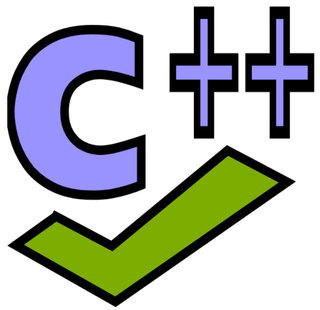
Program Repair

meets



Deep Learning

Static Analysis Tools



Useful to detect common bugs/defects.

Violations from Static Analysis Tools

Static analysis tools such as FindBugs detect violations

Example

```
public boolean equals(Object obj) {
    // Violation Type:
    // BC_EQUALS_METHOD_SHOULD_WORK_FOR_ALL_OBJECTS
    return getModule().equals(
         ((ModuleWrapper) obj).getModule());
}
```

PopulateRepositoryMojo.java **file at revision** bdf3fe **in project** nbm-maven-plugin.

Developers may (or may not) change source code to fix the violations.

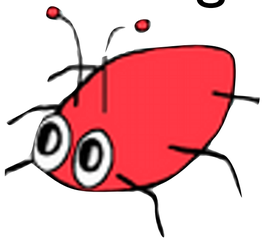
Example

```
   public boolean equals(Object obj) {
-      return getModule().equals(
-           ((ModuleWrapper) obj).getModule());
+      return obj instanceof ModuleWrapper &&
+           getModule().equals(
+           ((ModuleWrapper) obj).getModule());
   }
```

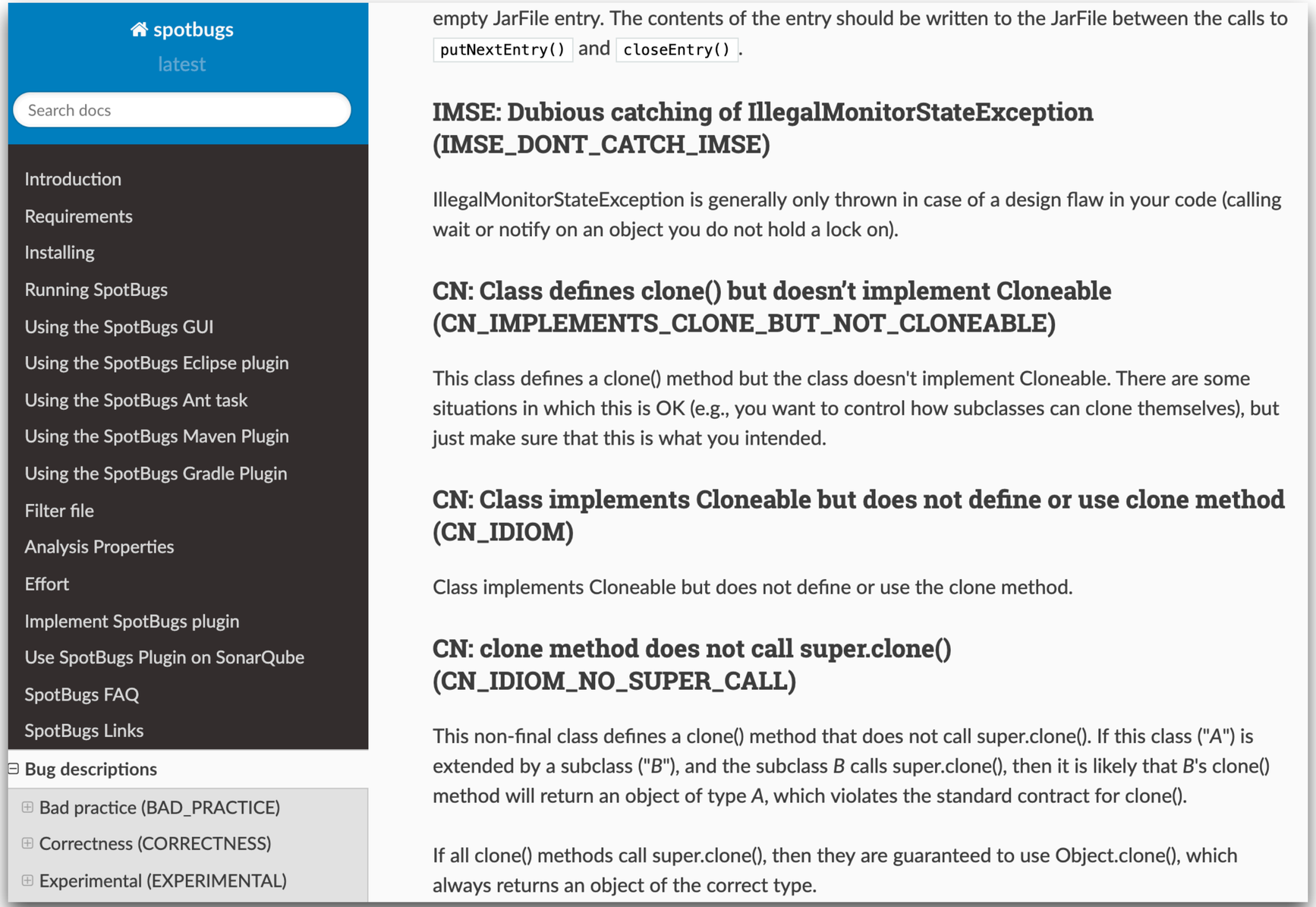
Commit 0fd11c **of project** nbm-maven-plugin

How to fix them?

FindBugs



Fixing based on bug description?



Fixing based on bug description?

BC: Equals method should not assume anything about the type of its argument (BC_EQUALS_METHOD_SHOULD_WORK_FOR_ALL_OBJECTS)

The equals(Object o) method shouldn't make any assumptions about the type of o. It should simply return false if o is not the same type as this.

BIT: Check for sign of bitwise operation (BIT_SIGNED_CHECK)

This method compares an expression such as ((event.detail & SWT.SELECTED) > 0). Using bit arithmetic and then comparing with the greater than operator can lead to unexpected results (of course depending on the value of SWT.SELECTED). If SWT.SELECTED is a negative number, this is a candidate for a bug. Even when SWT.SELECTED is not negative, it seems good practice to use '!= 0' instead of '> 0'.

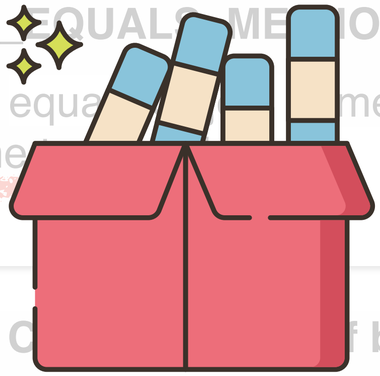
CN: Class implements Cloneable but does not define or use clone method (CN_IDIOM)

Class implements Cloneable but does not define or use the clone method.

Fixing based on bug description?

BC: Equals method should not assume anything about the type of its argument (BC_EQUALS_METHOD_SHOULD_WORK_FOR_ALL_OBJECTS)

The equals method shouldn't make any assumptions about the type of o. It should simply return false if o is not the same



Requires strong background knowledge.

BIT: Check for sign of bitwise operation (BIT_SIGNED_CHECK)

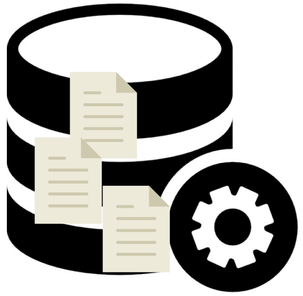
This method compares an expression such as ((event.detail & SWT.SELECTED) > 0). Using bit arithmetic and then comparing with the greater than operator can lead to unexpected results (of course depending on the value of SWT.SELECTED). If SWT.SELECTED is a negative number, this is a candidate for a bug. Even when SWT.SELECTED is not negative, it seems to be good practice to use '!= 0' instead of '> 0'.

Provides not enough details.

CN: Class implements Cloneable but does not define or use clone method (CN_IDIOM)

Class implements Cloneable but does not define or use the clone method.

Collecting violation-fixing changes



Revision
History

Collecting violation-fixing changes

Revision History

Program before changes

FindBugs

Revision History

Program before changes

Revision History

Program before changes

Revision History

Program before changes

Patches

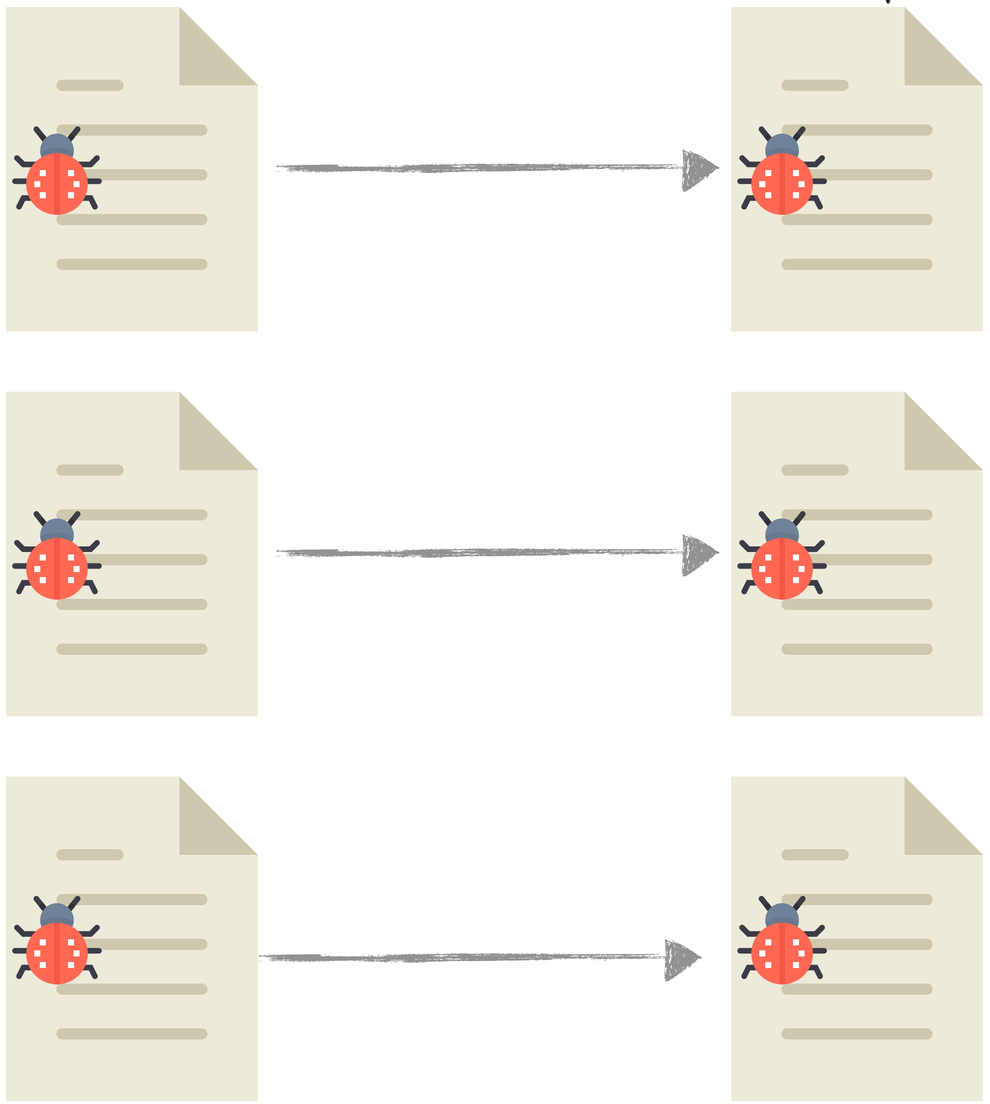
Revision History

Program before changes

FindBugs

Revision History

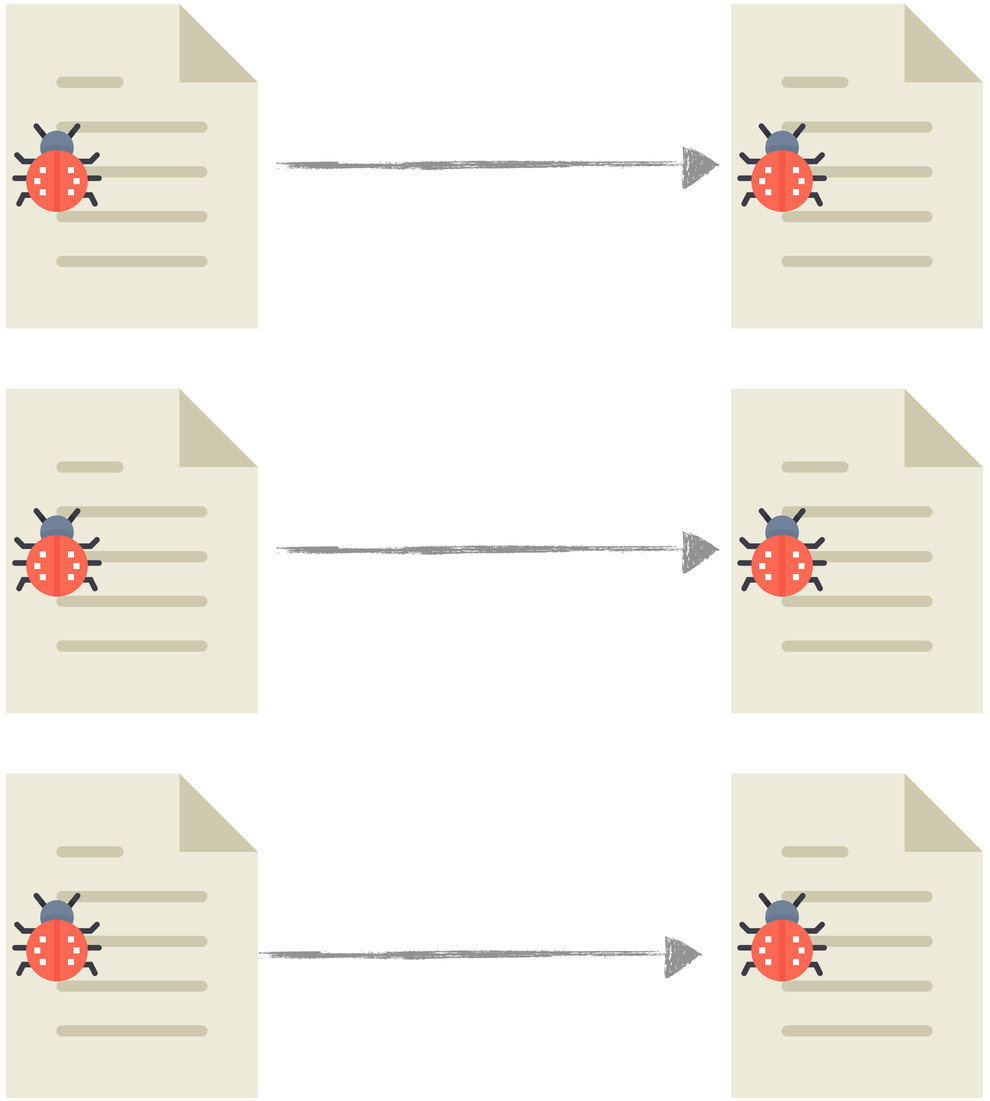
Program before changes



Program after changes

Revision History

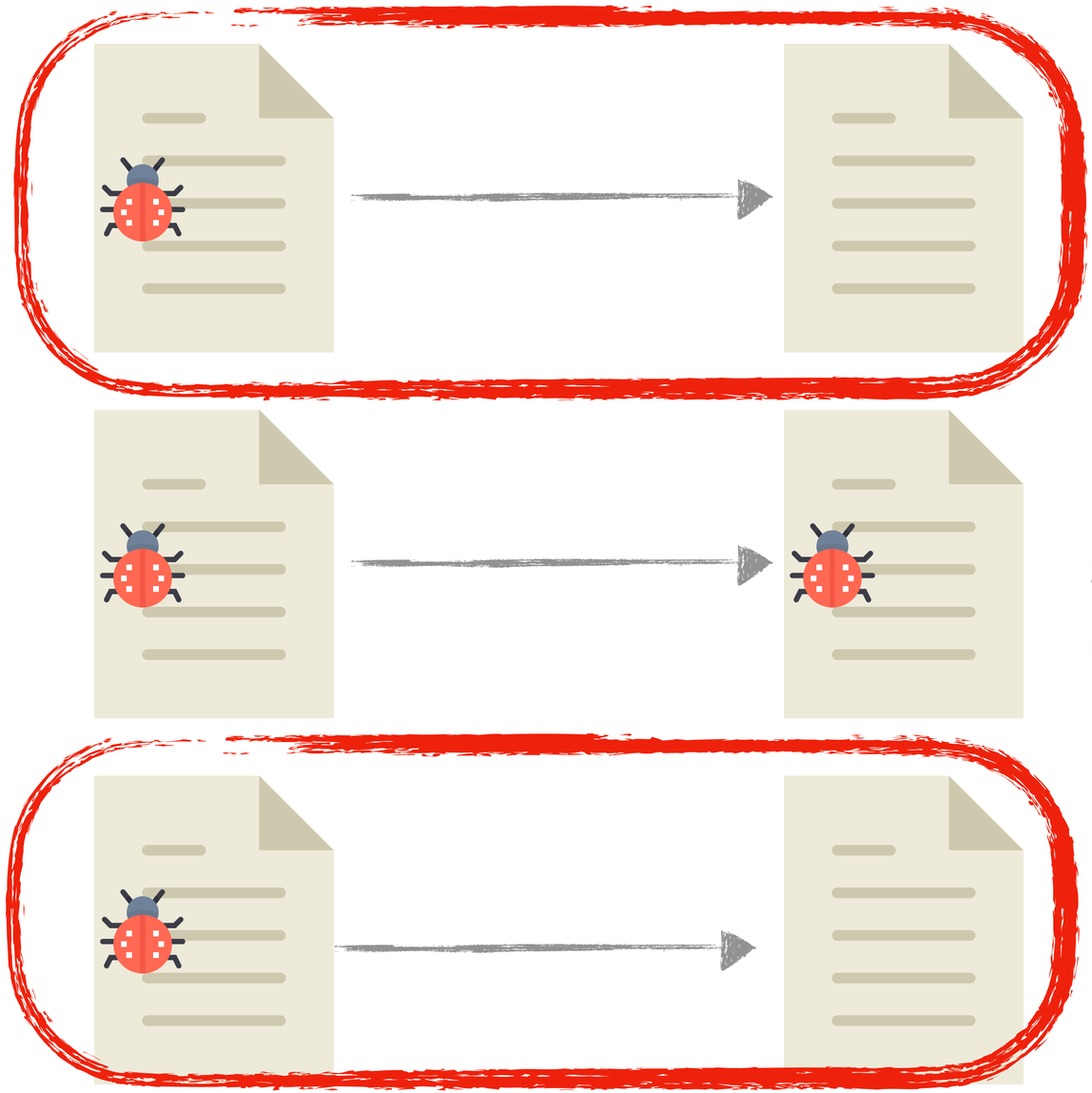
Program before changes



Program after changes

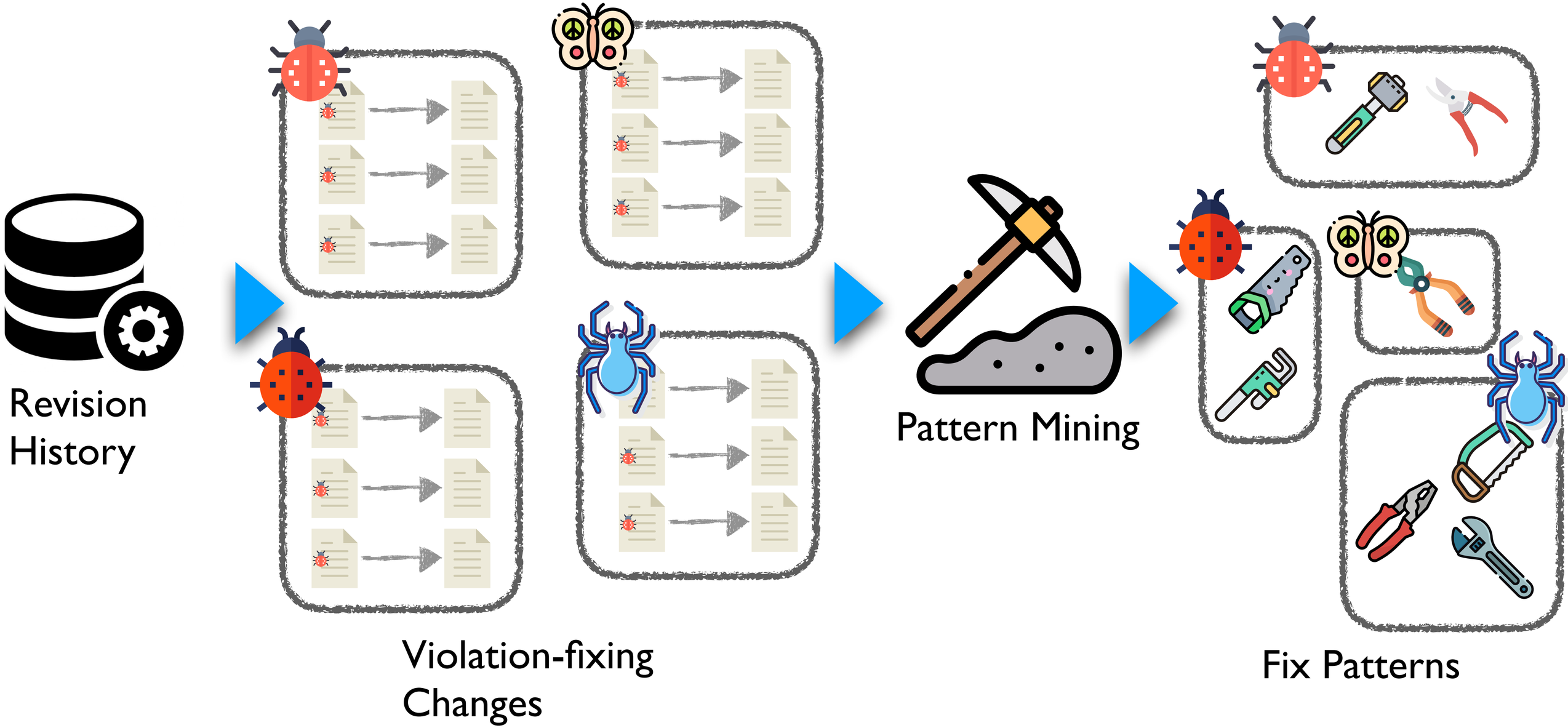
Revision History

Program before changes



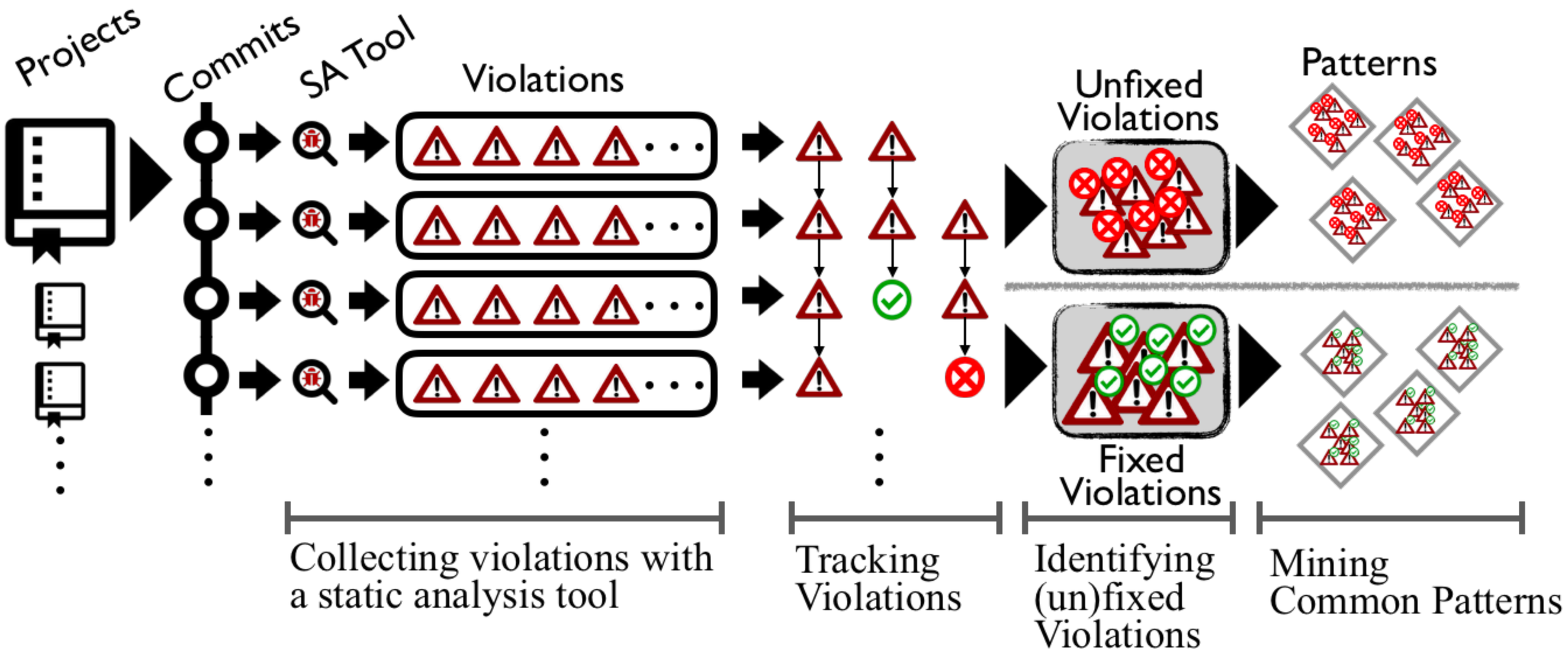
Program after changes

Idea: Mining violation-fixing changes patterns

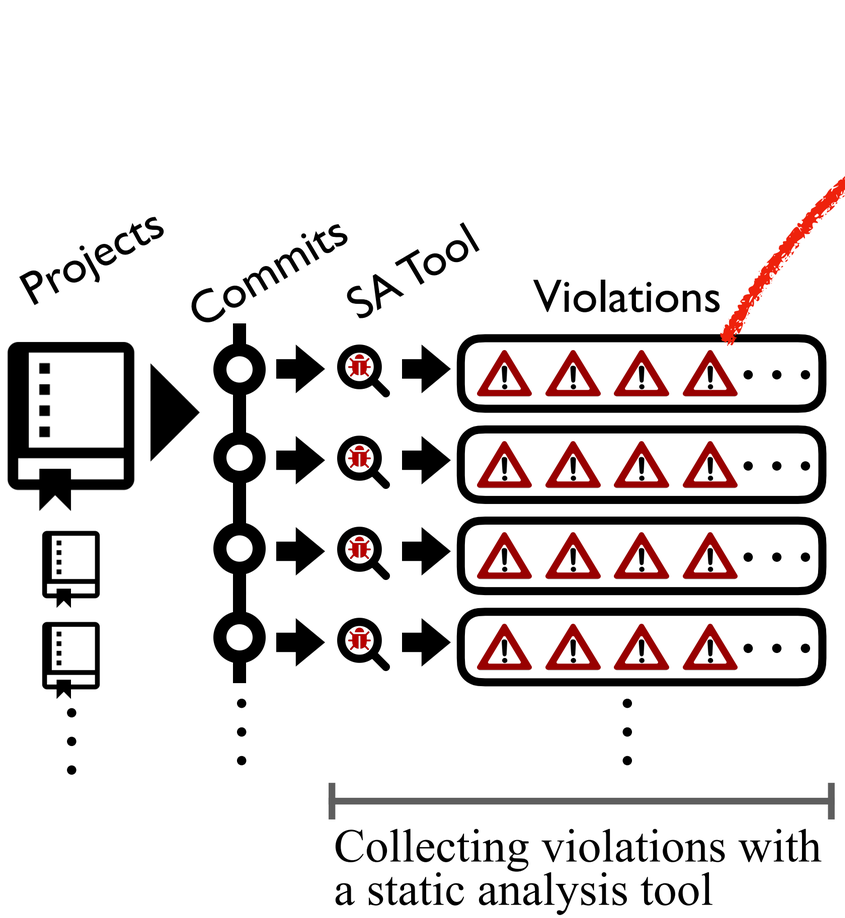


Approach

Overview

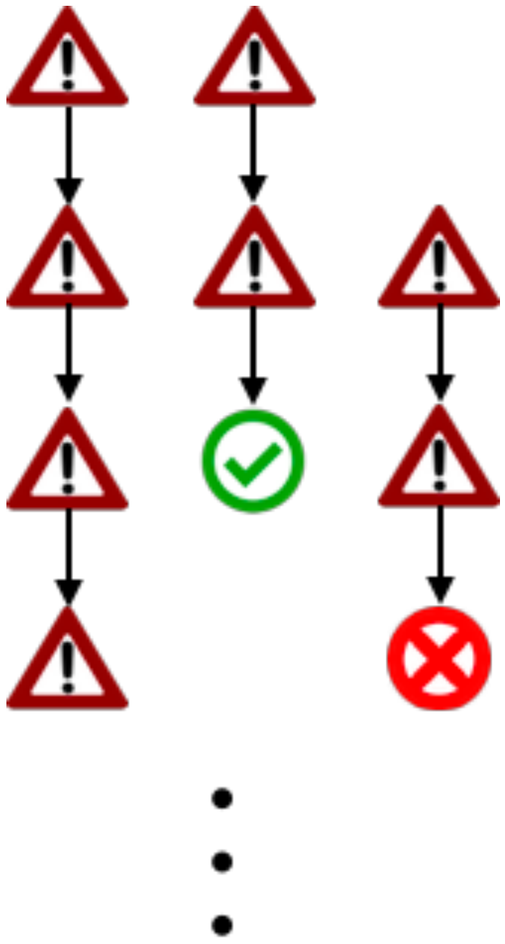

Projects
Commits
SA Tool
Violations
Unfixed Violations
Patterns
Fixed Violations
Collecting violations with a static analysis tool
Tracking Violations
Identifying (un)fixed Violations
Mining Common Patterns

Collecting violations




```
<ViolationInstance>
  <ViolationType>
    BC_EQUALS_METHOD_SHOULD_WORK_FOR_ALL_OBJECTS
  </ViolationType>
  <ProjectName> nbm-maven-plugin</ProjectName>
  <CommitVersionID>bdf3fe</CommitVersionID>
  <FilePath>nb-repository-plugin/src/main/java/org/
      codehaus/mojo/nbm/repository/PopulateRepository
      Mojo.java</FilePath>
  <StartLineNumber>1195</StartLineNumber>
  <EndLineNumber>1195</EndLineNumber>
</ViolationInstance>
```

Tracking violations



Identify identical violations between revisions*.

Detect whether a violation is fixed, or just removed.

[*] P. Avgustinov, A. I. Baars, A. S. Henriksen, G. Lavender, G. Menzel, O. de Moor, M. Schfer, and J. Tibble, “Tracking Static Analysis Violations over Time to Capture Developer Characteristics,” in Proceedings of the 37th International Conference on Software Engineering, 2015, pp. 437–447.

Parsing changes (i.e., patches)

```
  public boolean equals(Object obj) {
-     return getModule().equals(
-             ((ModuleWrapper) obj).getModule());
+     return obj instanceof ModuleWrapper &&
+             getModule().equals(
+             ((ModuleWrapper) obj).getModule());
  }
```

We used GumTree* to identify AST-level changes.

```
UPD ReturnStatement@@”return getModule().equals(((ModuleWrapper) obj).getModule());”
---INS InfixExpression@@”obj instanceof ModuleWrapper…” to ReturnStatement
------INS InstanceofExpression@@”obj instanceof ModuleWrapper ” to InfixExpression
---------INS Variable@@”obj” to InstanceofExpression
---------INS Operator@@”instanceof” to InstanceofExpression
---------INS SimpleType@@”ModuleWrapper” to InstanceofExpression
------MOV MethodInvocation@@” getModule().equals(…)” to InfixExpression
```

[*] J.-R. Falleri, F. Morandat, X. Blanc, M. Martinez, and M. Monperrus, “Fine-grained and accurate source code differencing,” in *ACM/IEEE International Conference on Automated Software Engineering*. Vasteras, Sweden - September 15 - 19: ACM, 2014, pp. 313–324.

Tokenizing change information

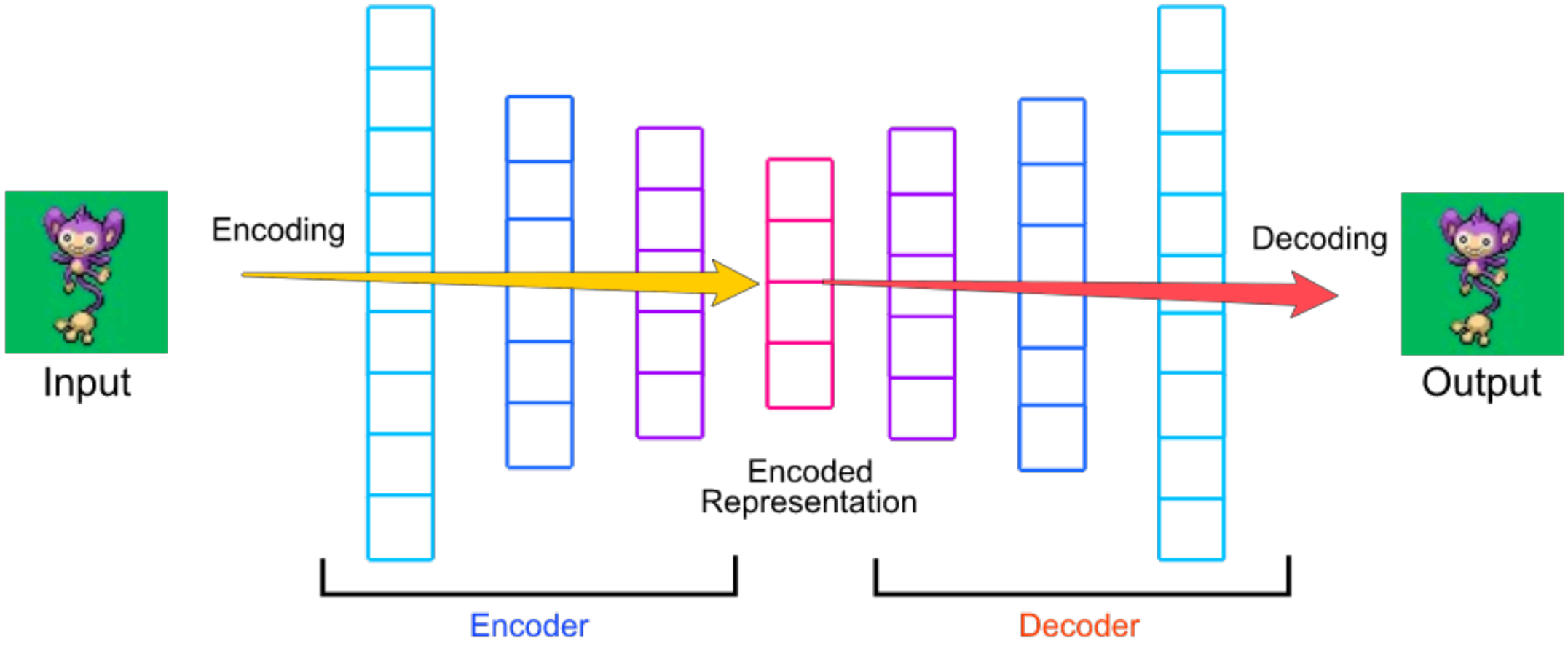
UPD ReturnStatement@@”return getModule().equals(((ModuleWrapper) obj).getModule());”
---INS InfixExpression@@”obj instanceof ModuleWrapper…” to ReturnStatement
------INS InstanceofExpression@@”obj instanceof ModuleWrapper ” to InfixExpression
---------INS Variable@@”obj” to InstanceofExpression
---------INS Operator@@”instanceof” to InstanceofExpression
---------INS SimpleType@@”ModuleWrapper” to InstanceofExpression
------MOV MethodInvocation@@” getModule().equals(…)” to InfixExpression

[UPD ReturnStatement, INS InfixExpression, INS InstanceofExpression, INS Variable, INS Operator, INS SimpleType, MOV MethodInvocation]

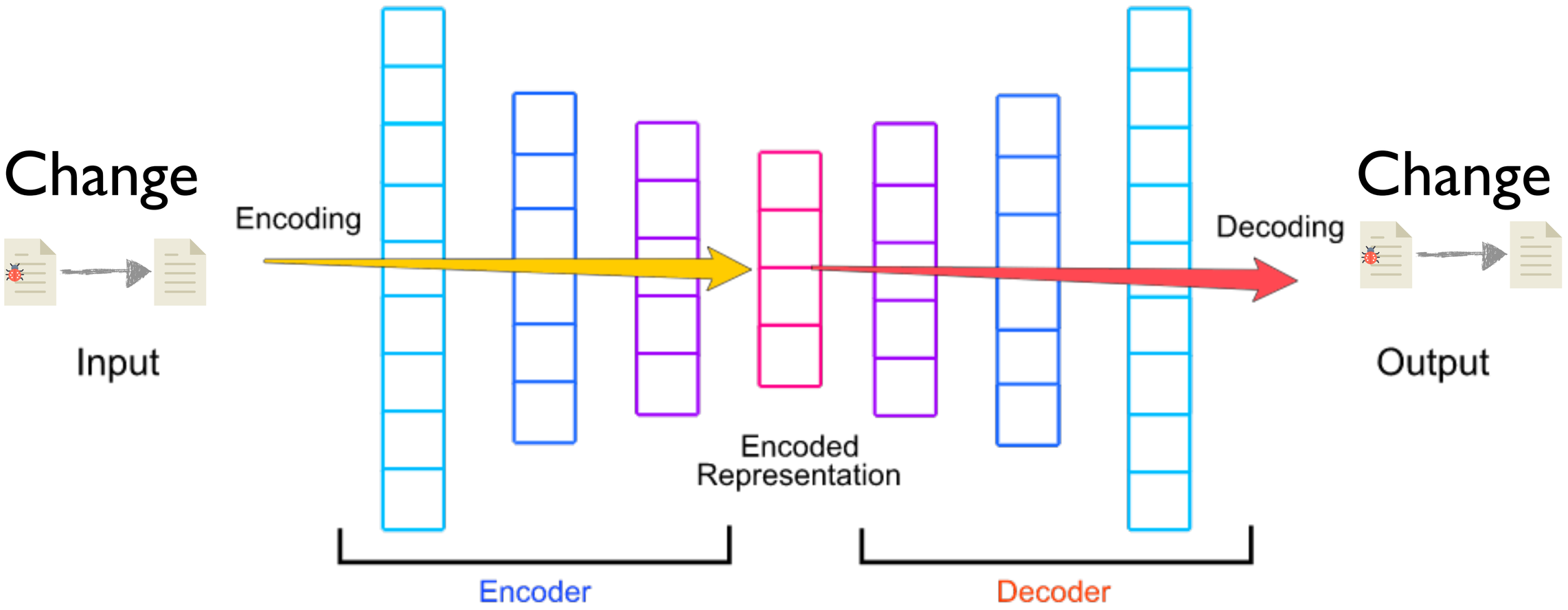
Token Embedding (Word2Vec)

<2, 6, 9, …> <8, 4, 1, …> <9, 0, 7, …> <2, 3, 0, …> … <7, 1, 2, …> …

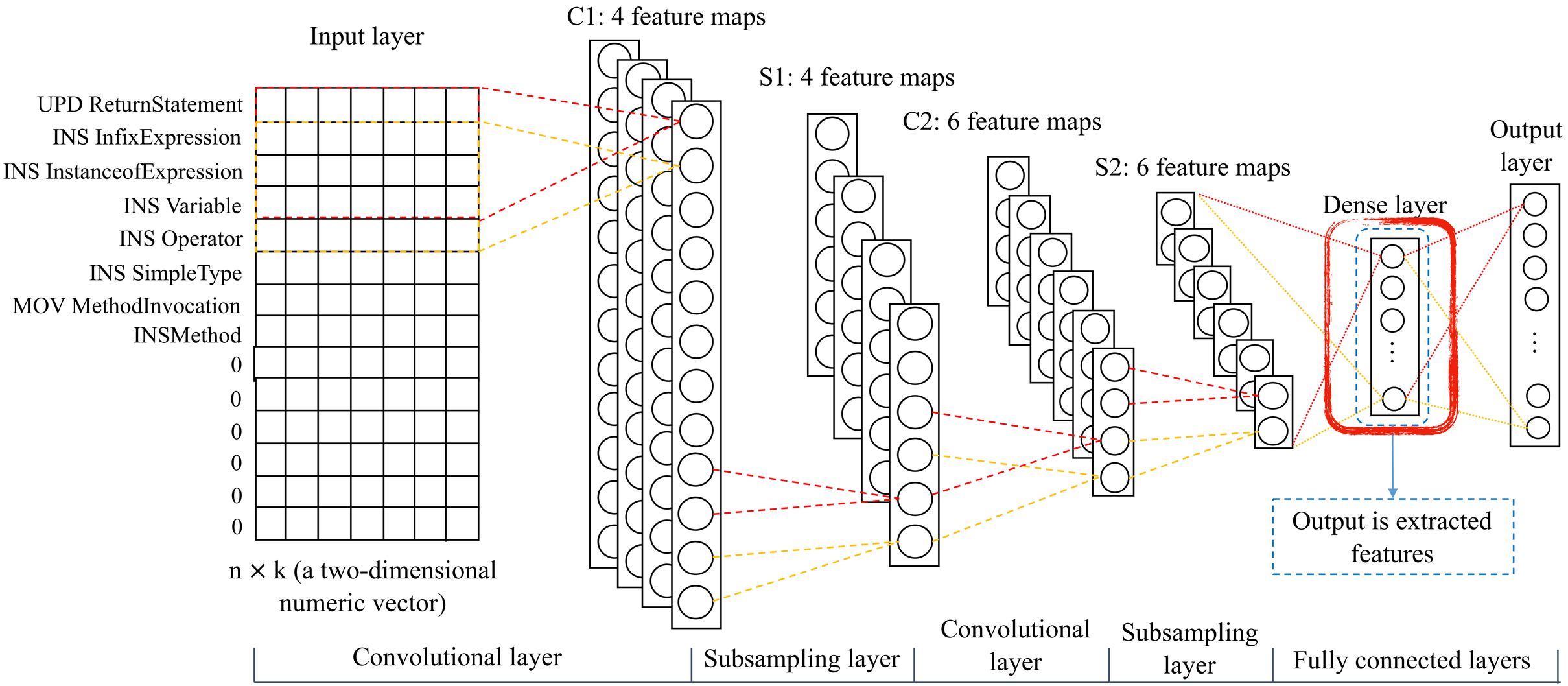
Autoencoder



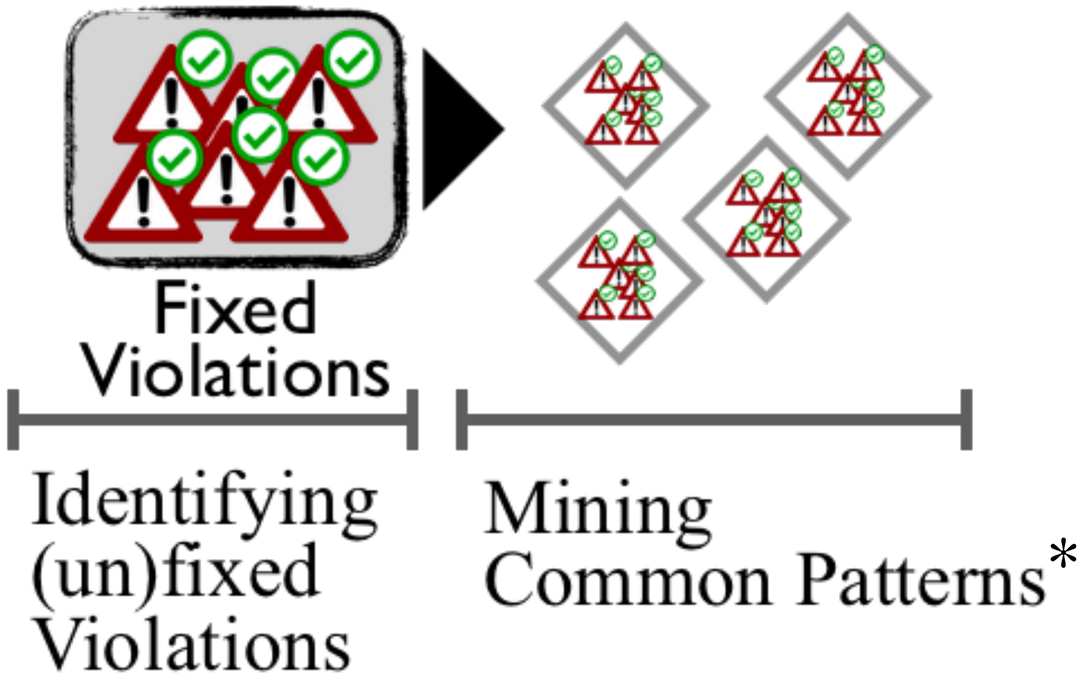
Autoencoder



Embedding change information



Clustering Patches and Identifying Fix Patterns




```
Violation Type:
BC_EQUALS_METHOD_SHOULD_WORK_FOR_ALL_OBJECTS

Patch Example:
-    return exp1().equals(((T)obj).exp2());
+    return obj instanceof T && exp1().
                  equals(((T)obj).exp2());

Fix Pattern###:
UPD ReturnStatement
---INS InfixExpression
------MOV MethodInvocation
------INS InstanceofExpression
---------INS Variable
---------INS Instanceof
---------INS SimpleType
------INS Operator
```

[*] D. Pelleg, A. W. Moore et al., “X-means: Extending k-means with efficient estimation of the number of clusters.” in ICML, vol. 1, 2000, pp. 727–734.

Evaluation

Subjects

# Projects	730
# Commits	291,615
# Violations (detected)	250,387,734
# Distinct violations	16,918,530
# Violations types	400

Collected from GitHub.com.

With at least one violation fixing commits.

Fix Patterns Identified

We have identified
174 fix patterns for
111 violation types.

Example

```
Violation Type:
BC_EQUALS_METHOD_SHOULD_WORK_FOR_ALL_OBJECTS

Patch Example:
-    return exp1().equals(((T)obj).exp2());
+    return obj instanceof T && exp1().equals(((T)obj).exp2());

Fix Pattern###:
UPD ReturnStatement
---INS InfixExpression
------MOV MethodInvocation
------INS InstanceofExpression
---------INS Variable
---------INS Instanceof
---------INS SimpleType
------INS Operator
```

SA_LOCAL_SELF_ASSIGNMENT_INSTEAD_OF_FIELD	Mined Fix Patterns.	7 months ago
SA_LOCAL_SELF_COMPARISON	Mined Fix Patterns.	7 months ago
SBSC_USE_STRINGBUFFER_CONCATENATION	Mined Fix Patterns.	7 months ago
SC_START_IN_CTOR	Mined Fix Patterns.	7 months ago
SE_BAD_FIELD_INNER_CLASS	Mined Fix Patterns.	7 months ago
SE_NO_SERIALVERSIONID	Mined Fix Patterns.	7 months ago
SIC_INNER_SHOULD_BE_STATIC	Mined Fix Patterns.	7 months ago
SIC_INNER_SHOULD_BE_STATIC_ANON	Mined Fix Patterns.	7 months ago
SIO_SUPERFLUOUS_INSTANCEOF	Mined Fix Patterns.	7 months ago
SS_SHOULD_BE_STATIC	Mined Fix Patterns.	7 months ago
UCF_USELESS_CONTROL_FLOW	Mined Fix Patterns.	7 months ago
UC_USELESS_CONDITION	Mined Fix Patterns.	7 months ago
UC_USELESS_OBJECT	Mined Fix Patterns.	7 months ago
UPM_UNCALLED_PRIVATE_METHOD	Mined Fix Patterns.	7 months ago
URF_UNREAD_FIELD	Mined Fix Patterns.	7 months ago
URF_UNREAD_PUBLIC_OR_PROTECTED_FIELD	Mined Fix Patterns.	7 months ago

Comparison (Defects4J)

Note that our fix patterns are extracted only from violation fixing patterns.

	Chart	Closure	Lang	Math	Mokito	Time	Total
AVATAR*	**5**	**8**	5	6	**2**	1	27
CapGen	4	0	5	12	0	0	21
Nopol	1	0	3	1	0	0	5
ACS	2	0	3	12	0	1	18
SimFix	4	6	**9**	**14**	0	1	**34**

[*] K. Liu, A. Koyuncu, D. Kim, and T. F. Bissyandè, “AVATAR: Fixing Semantic Bugs with Fix Patterns of Static Analysis Violations,” in 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER), 2019, pp. 1–12.

Live Study

Subject	# Pull Requests				
	Submitted	Merged	Improved	Rejected	Ignored
json-simple	2				2
commons-io	2			2	
commons-lang	7		1	1	5
commons-math	6				6
ant	16	9	1	4	2
cassandra	9				9
mahout	3				3
aries	5				5
poi	44	44			
camel	22	14		8	
Total	116	67	2	15	32

Plan


FindBugs
TM


SpotBugs


Pmd
DON'T SHOOT THE MESSENGER

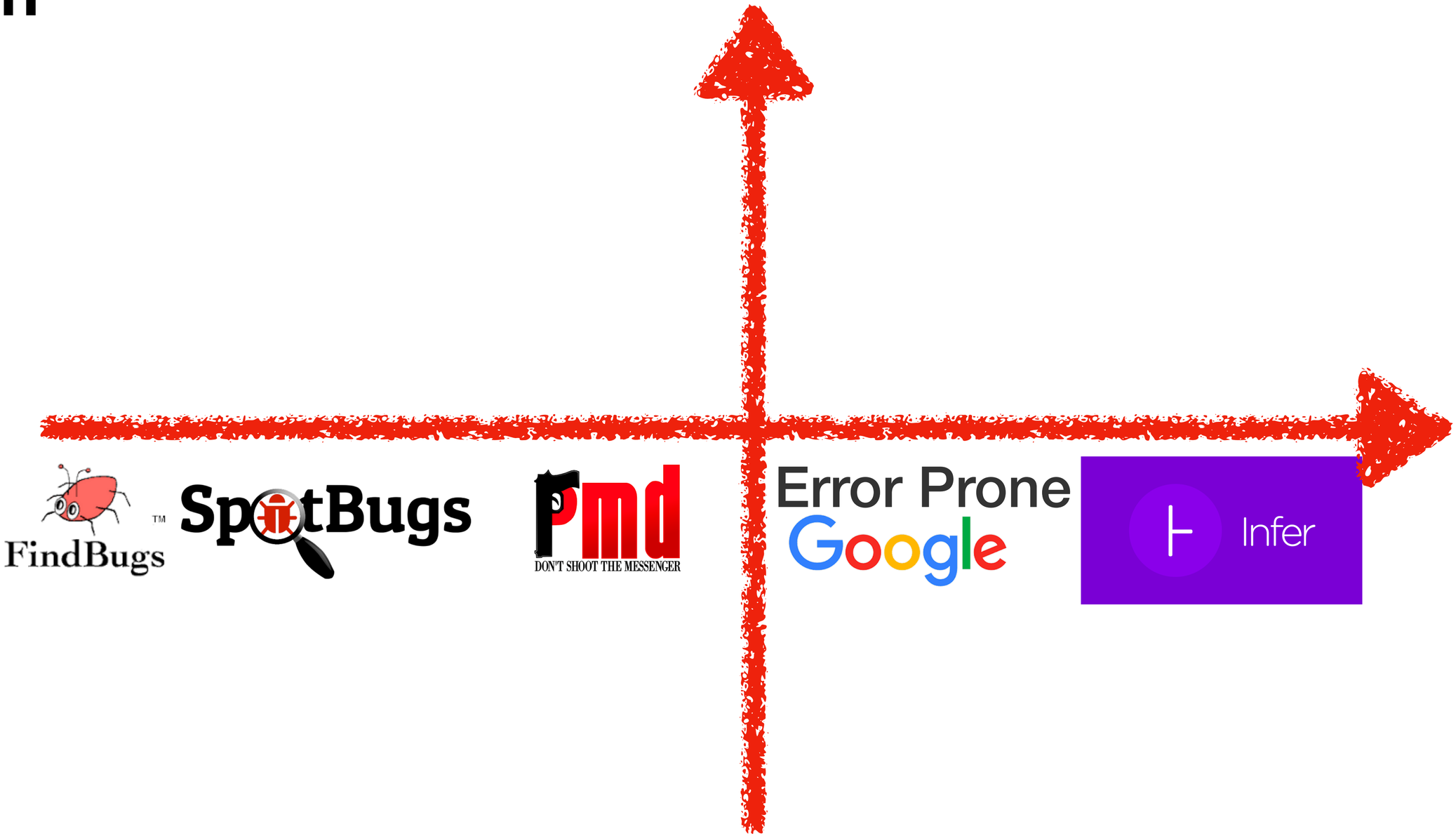

Error Prone
Google


Infer

Plan

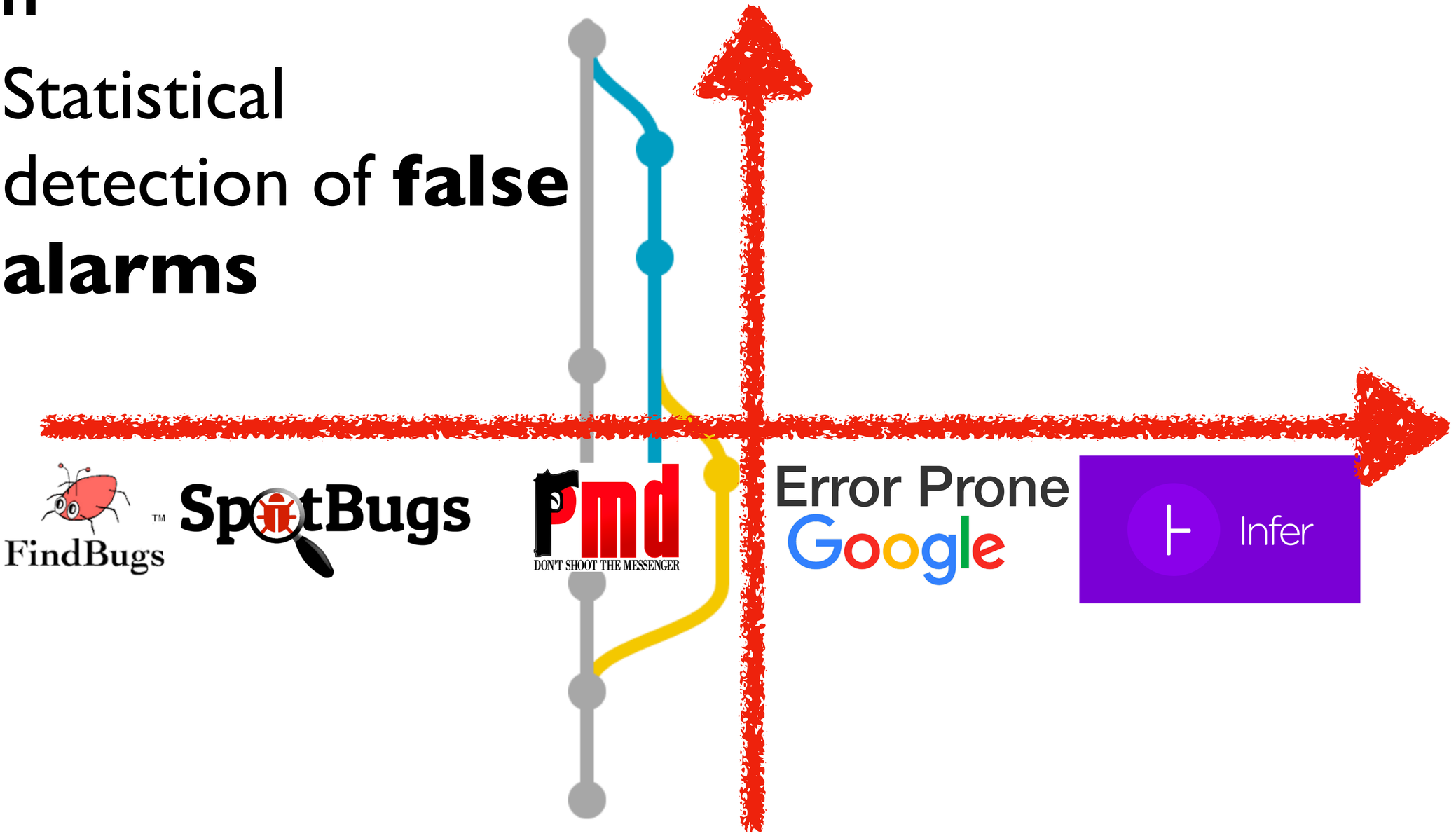
Statistical detection of **bugs** in bug finders

Plan


FindBugs™
SpotBugs
Pmd
DON'T SHOOT THE MESSENGER
Error Prone
Google
Infer

Plan

Statistical detection of **false alarms**



Summary

Fixing based on bug description?

BC: Equals method should not assume anything about the type of its argument (BC_EQUALS_METHOD_SHOULD_WORK_FOR_ALL_OBJECTS)

The equals(Object o) method shouldn't make any assumptions about the type of o. It should simply return false if o is not the same type as this.

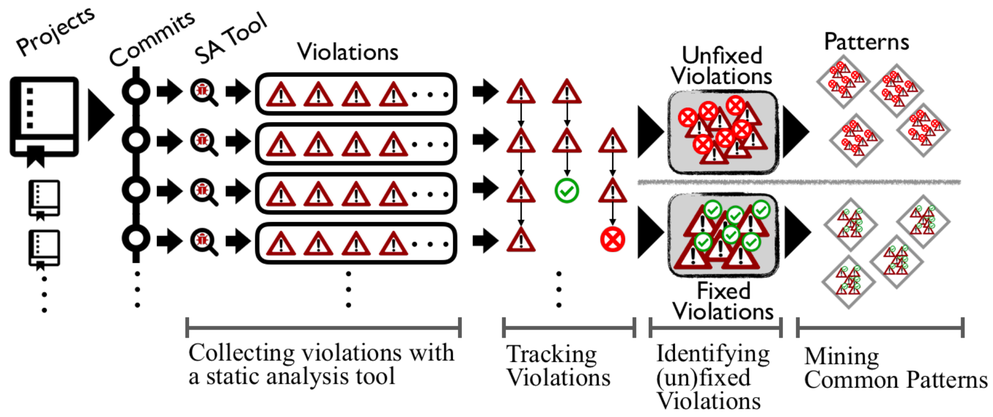
BIT: Check for sign of bitwise operation (BIT_SIGNED_CHECK)

This method compares an expression such as ((event.detail & SWT.SELECTED) > 0). Using bit arithmetic and then comparing with the greater than operator can lead to unexpected results (of course depending on the value of SWT.SELECTED). If SWT.SELECTED is a negative number, this is a candidate for a bug. Even when SWT.SELECTED is not negative, it seems good practice to use '!= 0' instead of '> 0'.

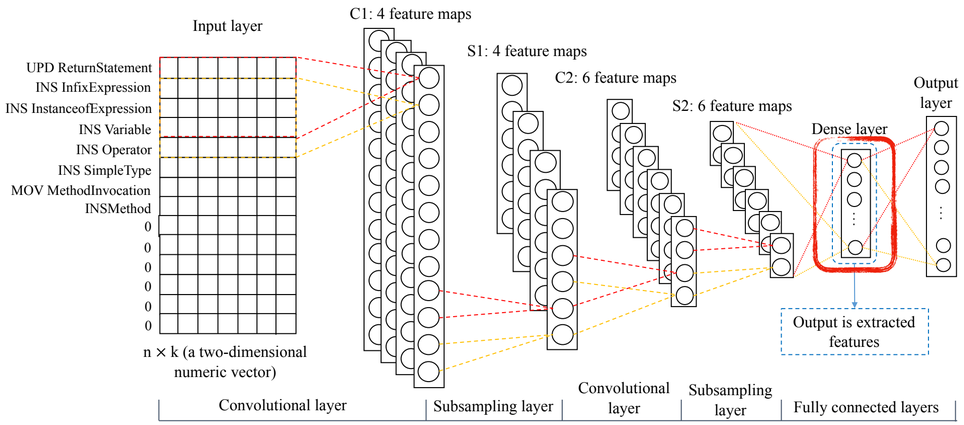
CN: Class implements Cloneable but does not define or use clone method (CN_IDIOM)

Class implements Cloneable but does not define or use the clone method.

Overview



Embedding change information



Live Study

Subject	# Pull Requests				
	Submitted	Merged	Improved	Rejected	Ignored
json-simple	2				2
commons-io	2			2	
commons-lang	7		1	1	5
commons-math	6				6
ant	16	9	1	4	2
cassandra	9				9
mahout	3				3
aries	5				5
poi	44	44			
camel	22	14		8	
Total	116	67	2	15	32