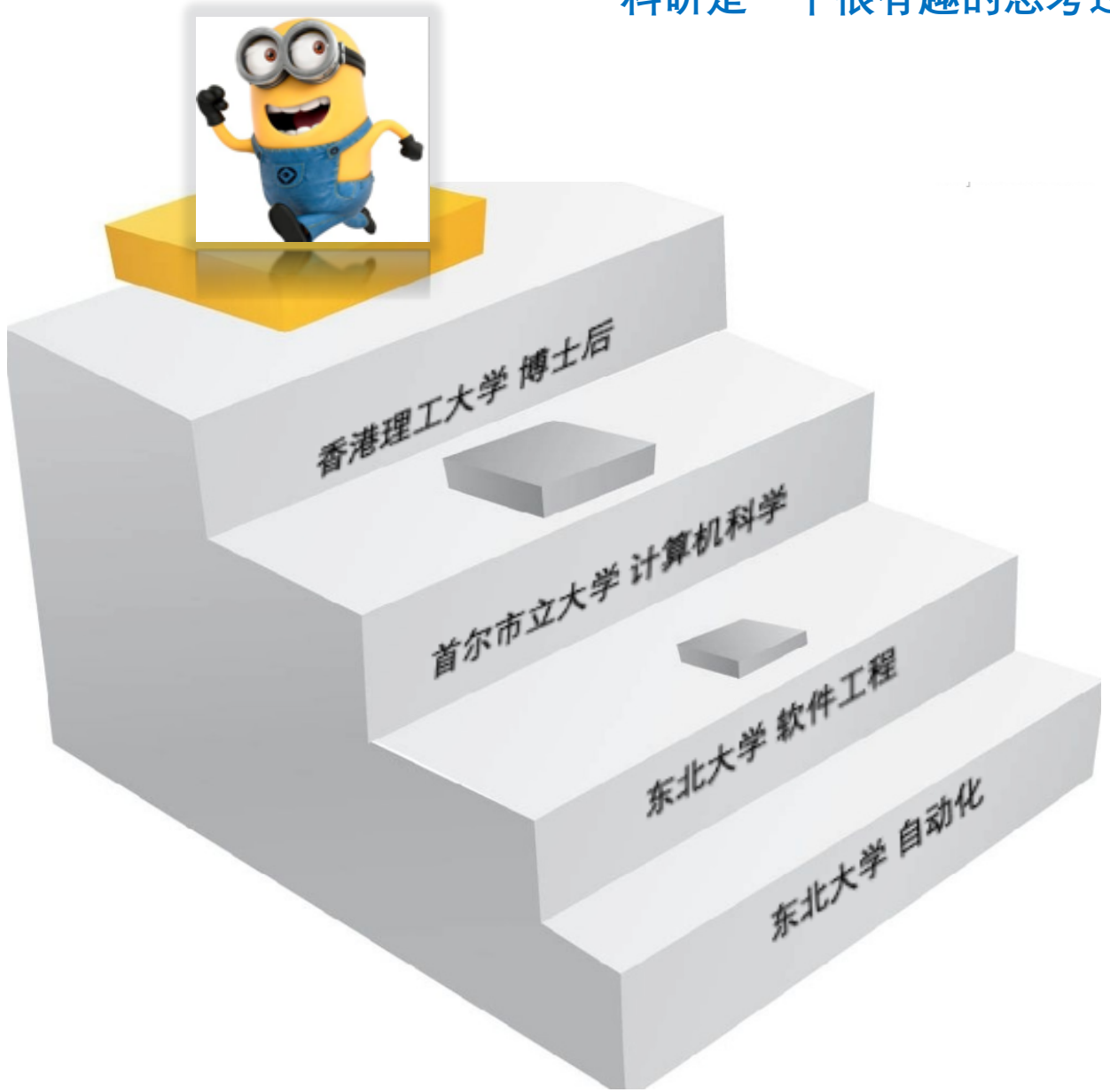
浅谈软件仓库挖掘在软件维护中的简单应用

张　涛

哈尔滨工程大学

科研菜鸟成长记录

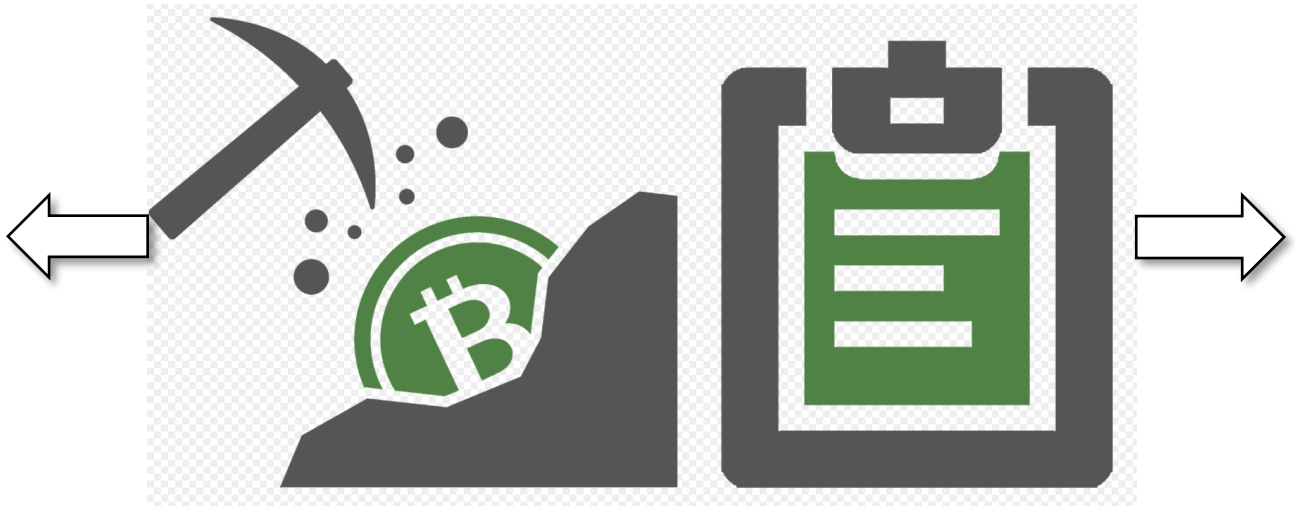
科研是一个很有趣的思考过程。



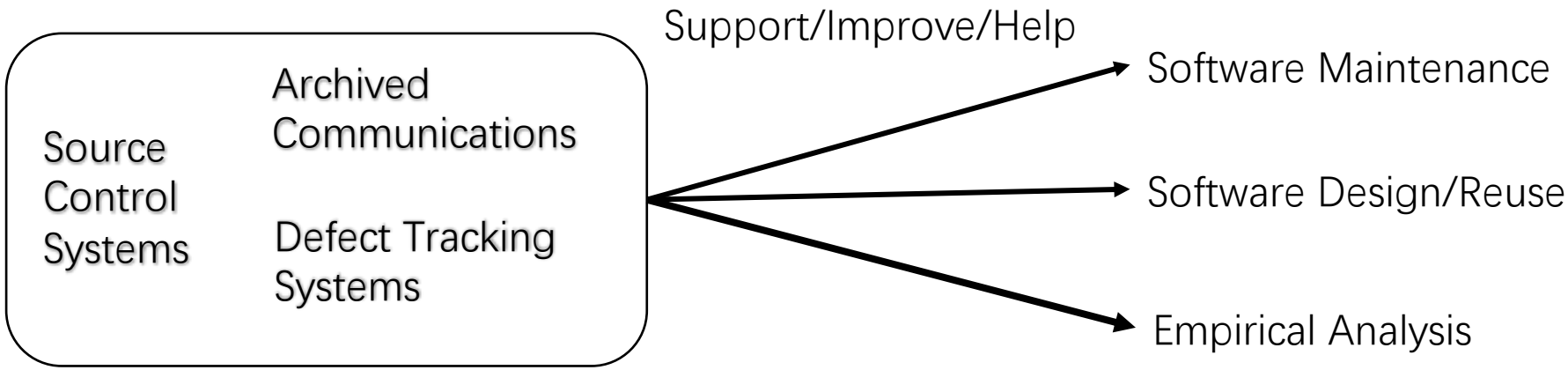
Introduction

- What is mining software repositories?

软件仓库挖掘是一个新兴的软件工程领域，通过数据挖掘技术分析软件仓库中海量的数据，来提高软件的质量和生产效率。
《软件仓库挖掘领域：贡献者和研究热点》



The Mining Software Repositories (MSR) field analyzes the rich data available in software repositories to uncover interesting and actionable information about software systems and projects.
《MSR web site》



Introduction

- How can we do by mining software repositories?

Bug triage

- Given a new bug report, how to find a developer to fix this new bug?

Reopened bug prediction

- How to predict which closed bug reports will be opened again in the future ?

Severity identification

- Given a new bug report, how to rank its severity ?

Duplicate bug detection

- How to find duplicate bug reports in bug repositories?

Bug summarization

- How to extract an abstract for a given bug report?

Bug localization

- Given a bug report, how to localize its corresponding source code?

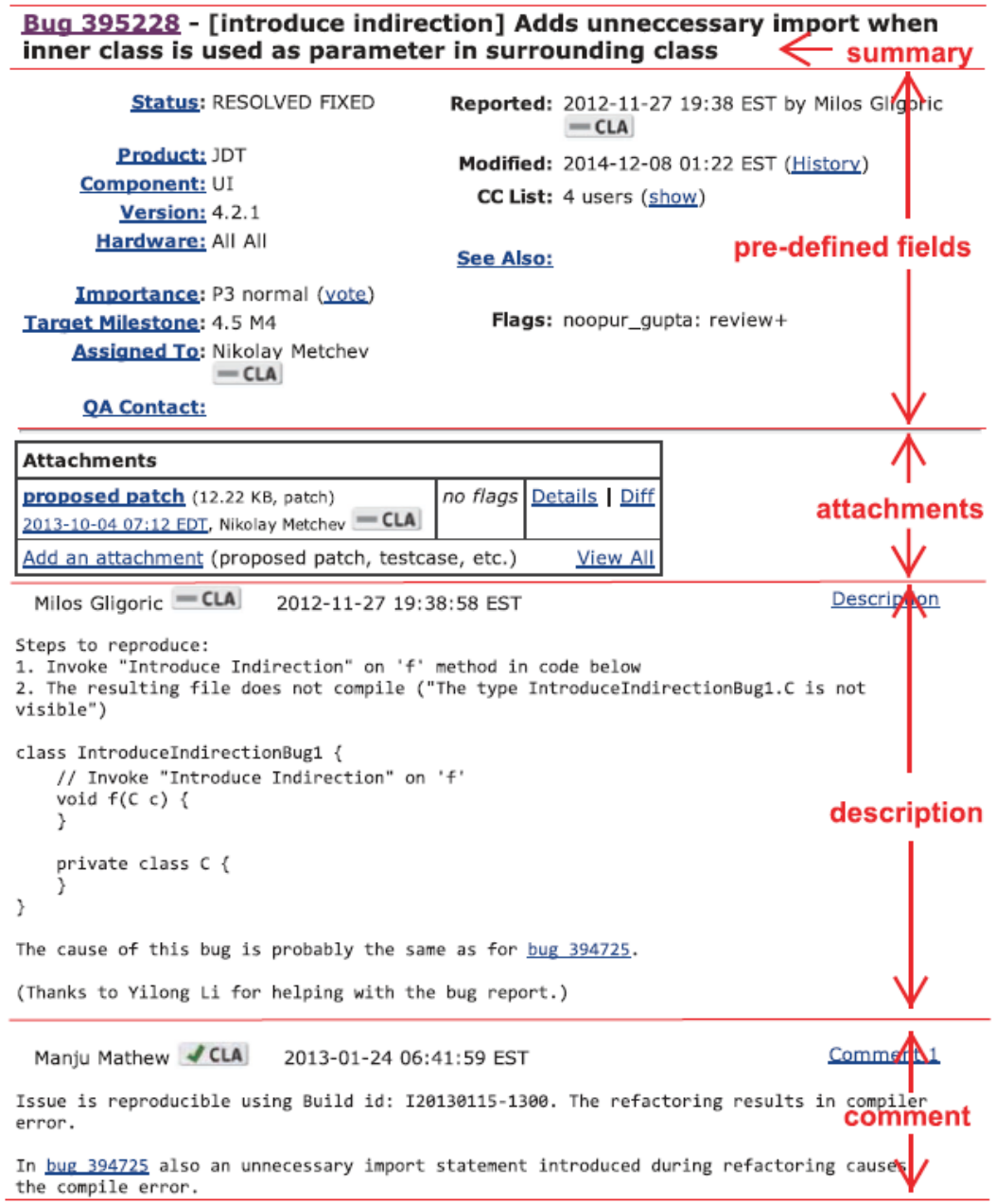
过 去

Overview

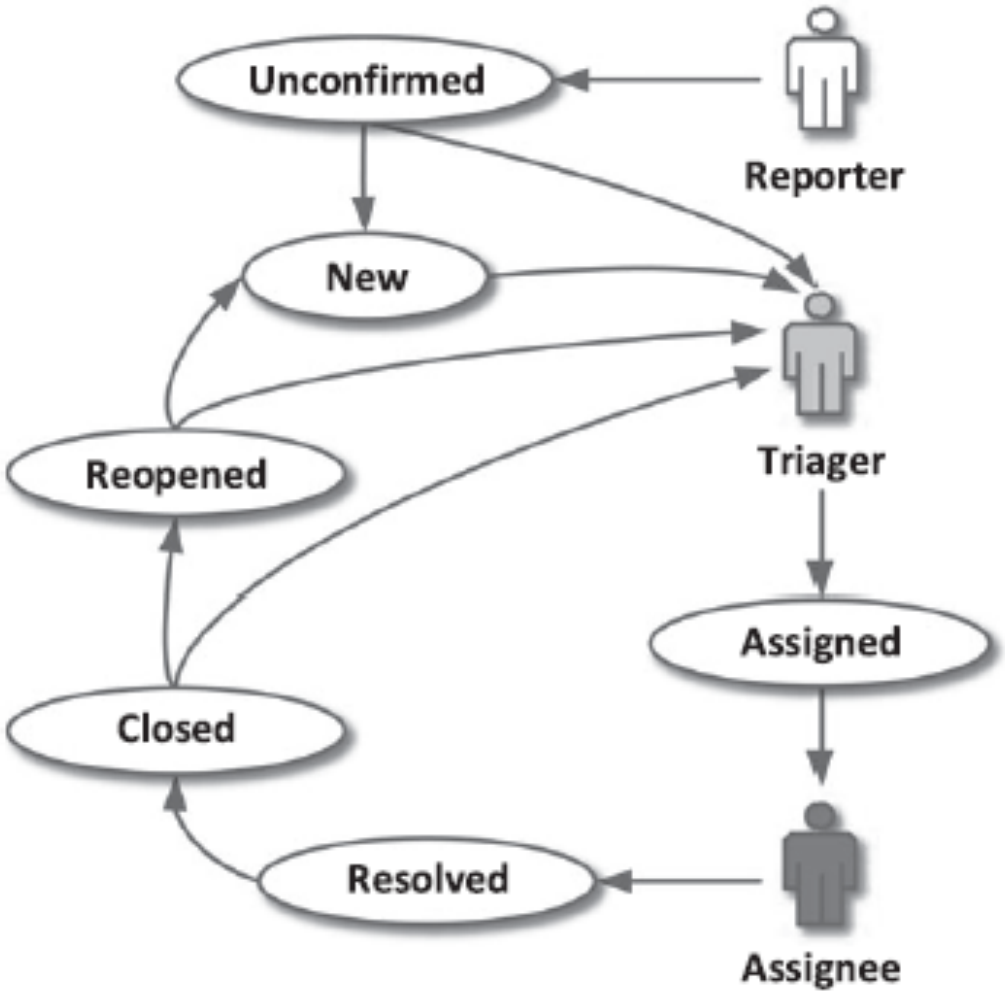
- **Bug Resolution using Mining Software Repositories**

Tao Zhang, He Jiang, Xiapu Luo, Alvin T.S. Chan, "A Literature Review of Research in Bug Resolution: Tasks, Challenges and Future Directions," *The Computer Journal*, 59(5), pp.741-773, 2016.

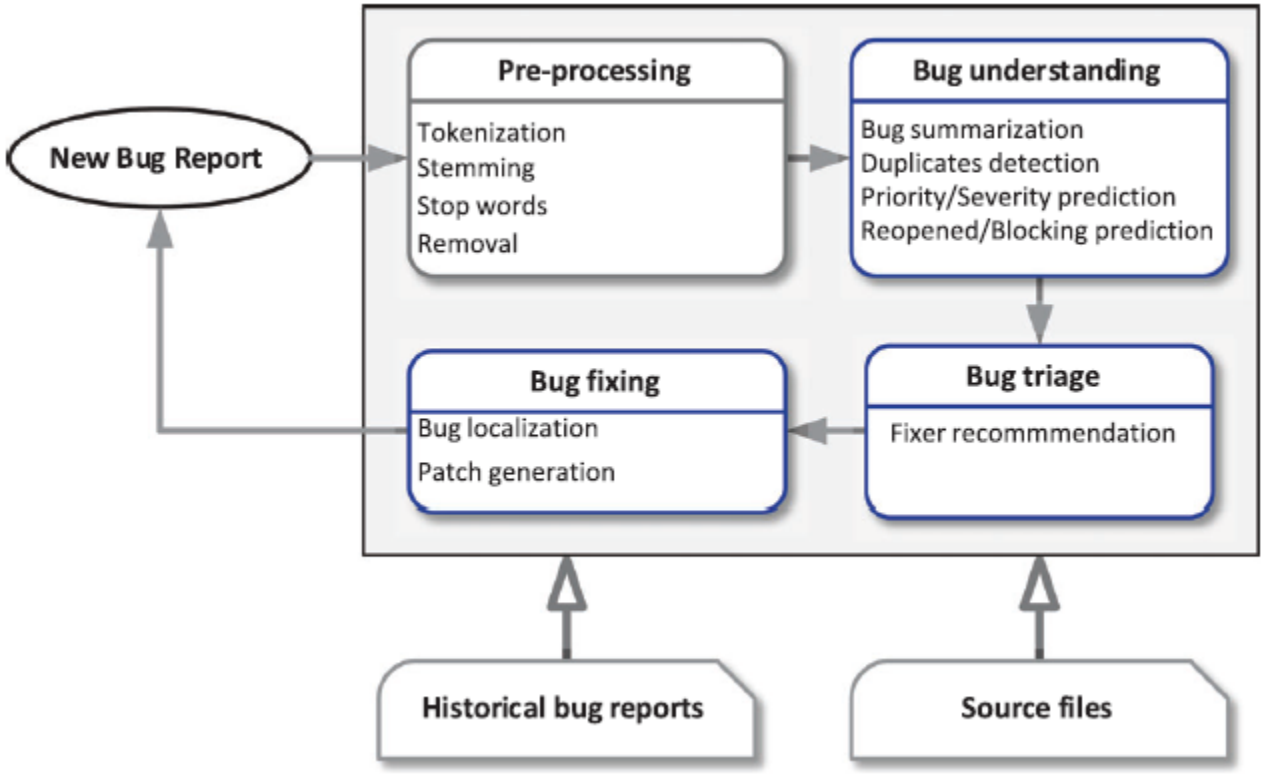
Bug reports


Bug 395228 - [introduce indirection] Adds unneccessary import when inner class is used as parameter in surrounding class
summary
Status: RESOLVED FIXED
Reported: 2012-11-27 19:38 EST by Milos Gligoric
CLA
Product: JDT
Component: UI
Version: 4.2.1
Hardware: All All
Modified: 2014-12-08 01:22 EST (History)
CC List: 4 users (show)
See Also:
pre-defined fields
Importance: P3 normal (vote)
Target Milestone: 4.5 M4
Assigned To: Nikolay Metchev
CLA
Flags: noopur_gupta: review+
QA Contact:
Attachments
proposed patch (12.22 KB, patch)
2013-10-04 07:12 EDT, Nikolay Metchev
CLA
no flags
Details | Diff
Add an attachment (proposed patch, testcase, etc.)
View All
attachments
Milos Gligoric
CLA
2012-11-27 19:38:58 EST
Description
Steps to reproduce:
1. Invoke "Introduce Indirection" on 'f' method in code below
2. The resulting file does not compile ("The type IntroduceIndirectionBug1.C is not visible")
class IntroduceIndirectionBug1 {
// Invoke "Introduce Indirection" on 'f'
void f(C c) {
}
private class C {
}
}
The cause of this bug is probably the same as for bug 394725.
(Thanks to Yilong Li for helping with the bug report.)
description
Manju Mathew
CLA
2013-01-24 06:41:59 EST
Comment 1
Issue is reproducible using Build id: I20130115-1300. The refactoring results in compiler error.
In bug 394725 also an unnecessary import statement introduced during refactoring causes the compile error.
comment

Life cycle


Unconfirmed
Reporter
New
Triager
Reopened
Assigned
Closed
Resolved
Assignee

Bug resolution

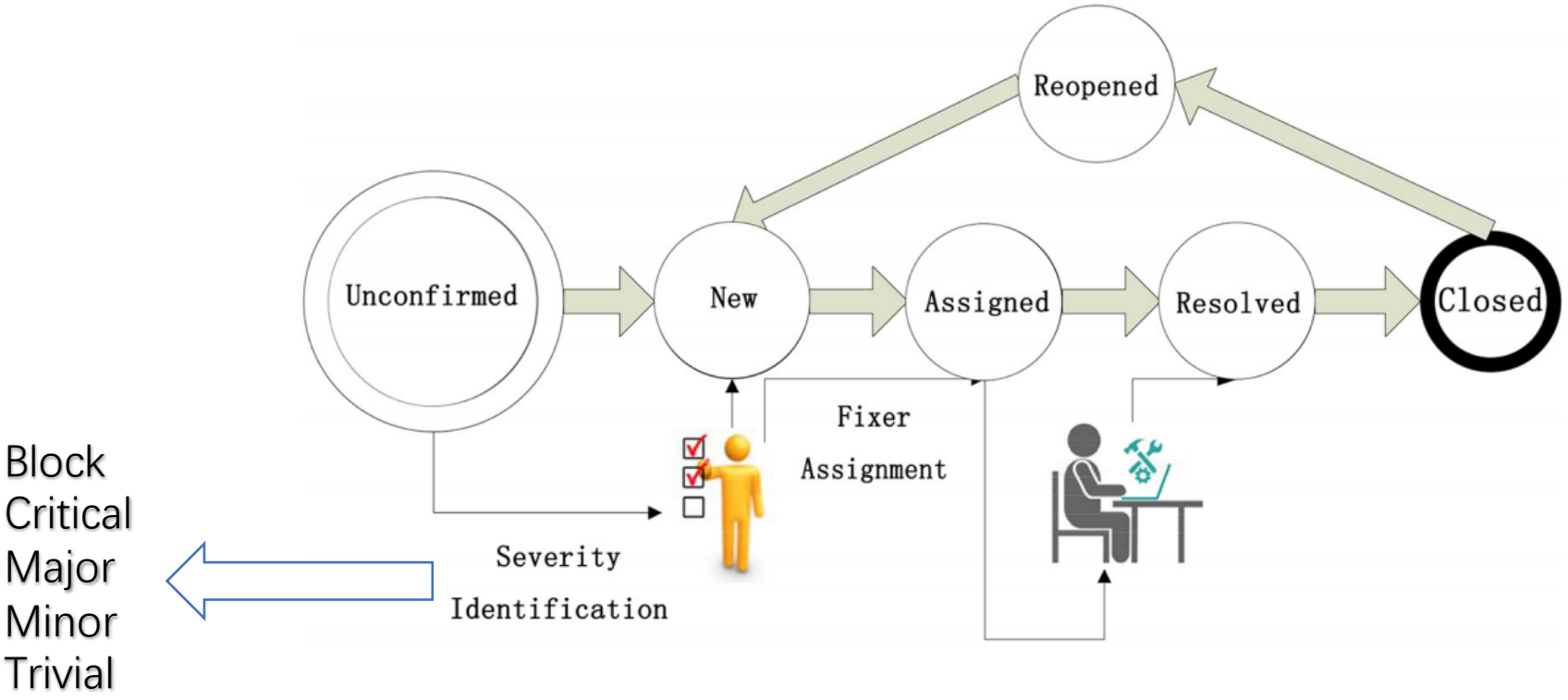


Implementation

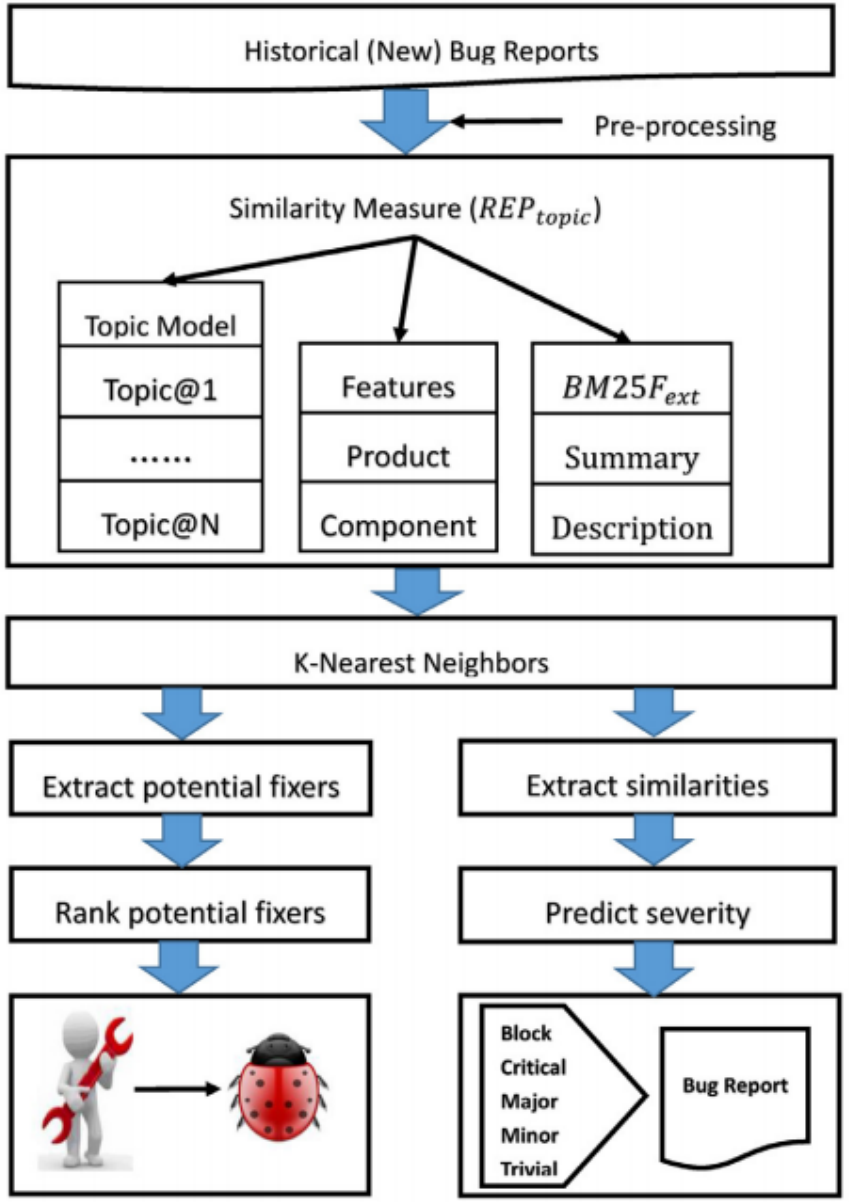
- Severity prediction and fixer recommendation

Tao Zhang, Jiachi Chen, Geunseok Yang, Byungjeong Lee, Xiapu Luo, “Towards more accurate severity prediction and fixer recommendation,” *Journal of Systems and Software*, Vol.117, pp.166-184.

What is severity prediction and fixer recommendation



Method: Framework



Method: Severity prediction

Number of the neighbors whose severity level is l.

Probability of severity level being l

$$P(br_{new}|l) = \gamma_1 \frac{k_l}{K} + \gamma_2 \frac{\sum_{i=1}^{k_l} sim(br_{new}, br_i)}{\sum_{i=1}^{K} sim(br_{new}, br_i)}$$

Similarity between bug report and its neighbor.

Total number of the neighbors.

Method: Fixer recommendation

$$FRScore(d) = \delta_1 \times SocialScore(d) + \delta_2 \times ExperienceScore(d)$$

$$SocialScore(d) = \frac{nc \times od}{MAX_{1 \le i \le M}(nc_i \times od_i)}$$

Number of comments that the developer *d* posts

Number of comments that the other developers post to the bug reports assigned to *d*

$$ExperienceScore(d) = \frac{n_{fix}/n_{reopen}}{MAX_{1 \le i \le M}(n_{fix_i}/n_{reopen_i})}$$

Number of bugs fixed by the developer *d* successfully

Number of bugs which have the reopened records among all historical bugs assigned to *d*

Results: severity prediction

Performance comparison among severity prediction algorithms when K=5 on P(Precision), R(Recall), and F(F-measure).

Project	Severity	Our approach			INSpect			NB Multinomial		
		P (%)	R (%)	F (%)	P (%)	R (%)	F (%)	P (%)	R (%)	F (%)
Eclipse	Blocker	30.63	24.80	**27.41**	1.84	33.37	3.50	23.76	17.10	19.89
	Critical	28.35	30.10	**29.19**	27.45	26.79	27.12	11.62	22.31	15.28
	Major	59.51	47.97	**53.12**	62.71	45.95	53.04	46.16	42.90	44.47
	Minor	43.23	44.88	**44.04**	46.28	39.64	42.71	32.81	23.21	27.19
	Trivial	18.93	41.87	**26.08**	4.27	47.51	7.83	28.42	17.03	21.30
Mozilla	Blocker	48.15	64.19	**55.02**	46.67	66.55	54.86	39.01	23.91	29.65
	Critical	47.27	52.84	**49.90**	45.82	44.72	45.26	24.09	31.52	27.31
	Major	46.51	40.21	**43.13**	41.90	37.86	39.77	58.57	31.43	40.91
	Minor	50.16	44.20	**46.99**	54.84	41.01	46.92	27.10	26.04	26.56
	Trivial	27.25	41.70	**32.96**	14.00	49.93	21.87	3.25	15.27	5.36
GCC	Blocker	8.00	54.67	13.96	10.41	38.45	**16.39**	12.67	13.07	12.87
	Critical	83.72	77.07	**80.25**	86.22	67.94	76.00	61.04	67.56	64.14
	Major	42.08	25.95	**32.10**	37.87	15.20	21.70	5.75	5.80	5.78
	Minor	29.77	47.53	**36.61**	11.72	50.98	19.06	30.11	17.31	21.98
	Trivial	25.71	16.53	**20.12**	0.56	10.00	1.05	1.25	0.28	0.45

Results: fixer recommendation

Eclipse: performance comparison among semi-automatic fixer recommendation algorithms when K=5 on Precision, Recall, and F-measure.

Evaluation metrics	# Recommended fixers	Semi-automatic fixer recommendation methods			
		Our approach (%)	DRETOM (%)	DREX (%)	DevRec (%)
Precision	5	**25.63**	16.04	10.09	25.21
	10	**23.74**	12.58	12.89	15.73
Recall	5	**55.21**	26.62	19.87	45.8
	10	**69.71**	39.44	39.37	56.67
F-measure	5	**35.01**	20.02	13.38	32.51
	10	**35.42**	19.07	19.42	24.62

NetBeans: performance comparison among semi-automatic fixer recommendation algorithms when K=5 on Precision, Recall, and F-measure.

Evaluation metrics	# Recommended fixers	Semi-automatic fixer recommendation methods			
		Our approach (%)	DRETOM (%)	DREX (%)	DevRec (%)
Precision	5	25.21	22.57	9.51	**27.97**
	10	**22.76**	15.55	12.83	17.03
Recall	5	**54.54**	32.77	16.6	41.71
	10	**65.47**	45.42	37.43	50.24
F-measure	5	**34.48**	26.73	12.09	33.49
	10	**33.78**	23.16	19.11	25.43

Mozilla: Performance comparison among semi-automatic fixer recommendation algorithms when K=5 on Precision, Recall, and F-measure

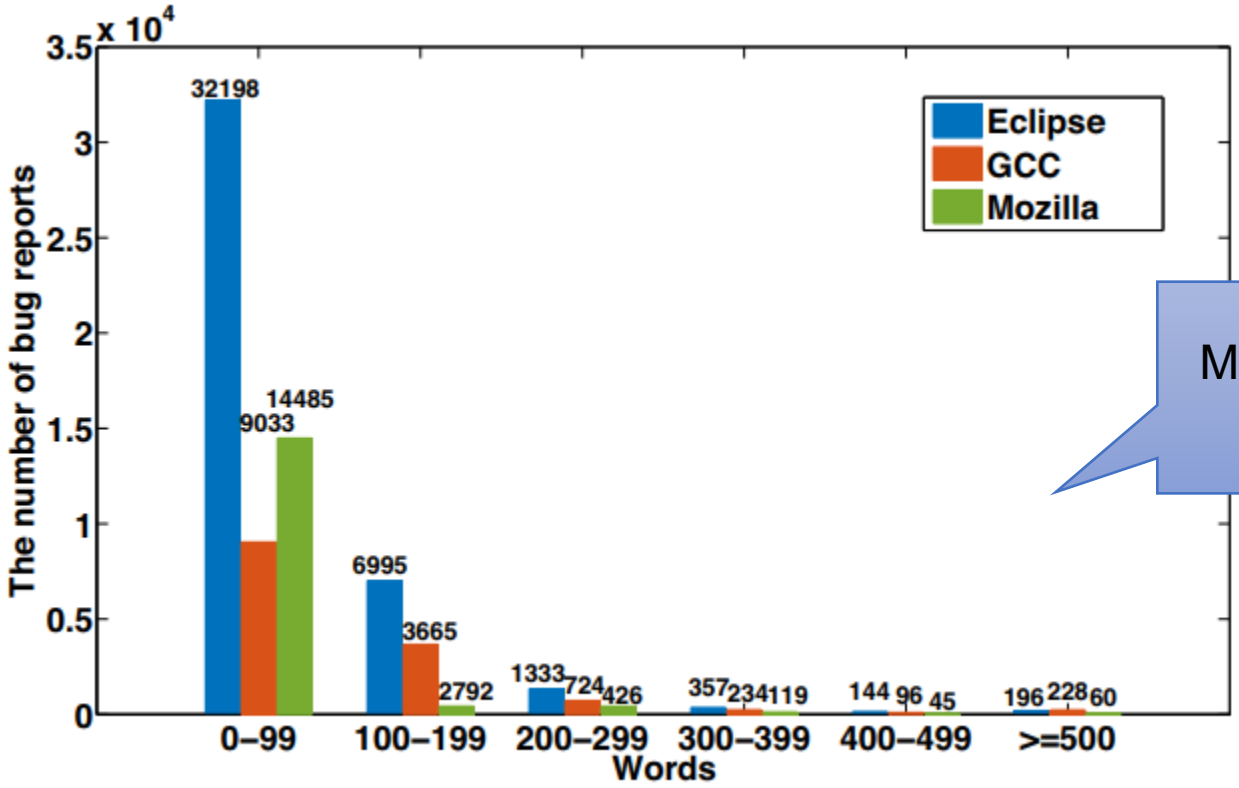
Evaluation metrics	# Recommended fixers	Semi-automatic fixer recommendation methods			
		Our approach (%)	DRETOM (%)	DREX (%)	DevRec (%)
Precision	5	26.42	16.05	9.18	**27.8**
	10	**25.02**	12.58	10.20	18.23
Recall	5	**57.58**	26.62	12.84	40.71
	10	**70.46**	39.43	22.26	39.43
F-measure	5	**36.22**	20.02	10.71	33.03
	10	**36.93**	19.07	13.99	21.18

Implementation

- Bug report enrichment

Tao Zhang, Jiachi Chen, He Jiang, Xiapu Luo, Xin Xia, "Bug Report Enrichment with Application of Automated Fixer Recommendation," *IEEE 25th International Conference on Program Comprehension (ICPC)*, pp.230-240, 2017.

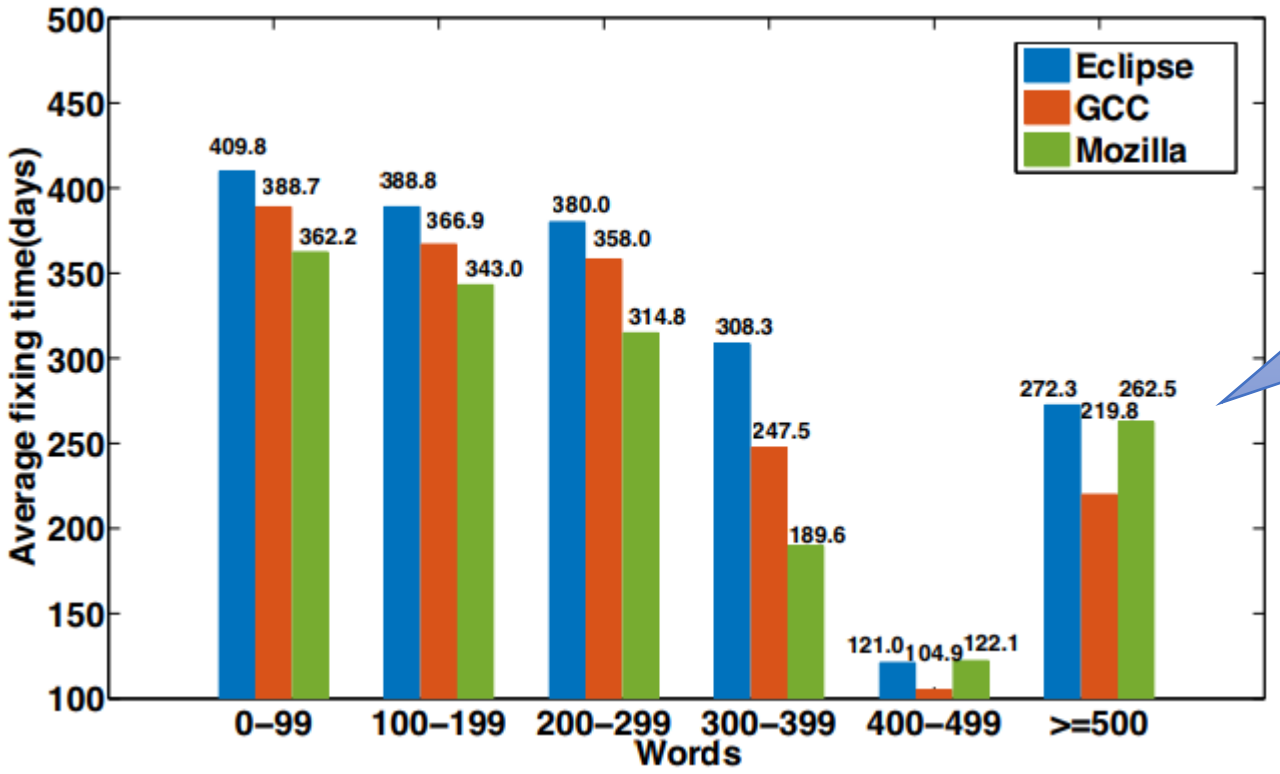
Motivation



Most of bug reports are short.

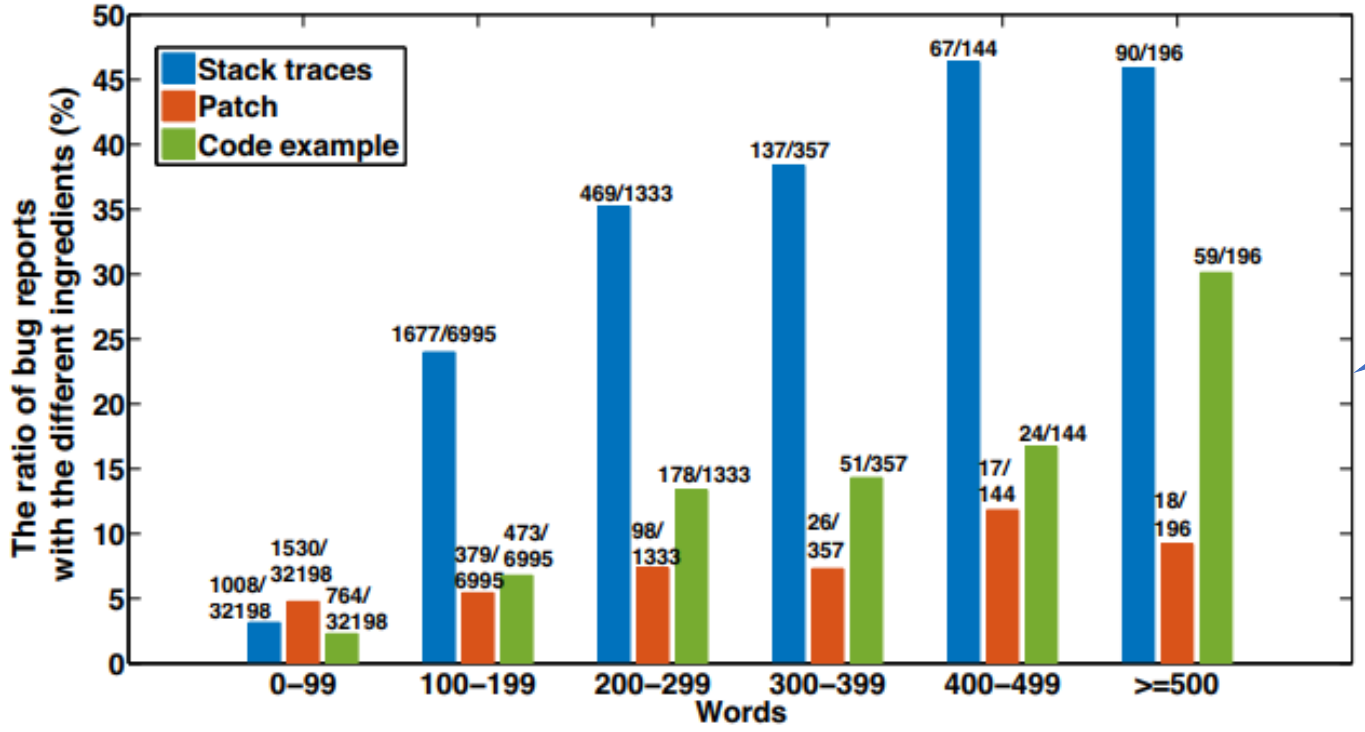
Fig. 2: The distribution of bug reports with their lengths (words)

Motivation



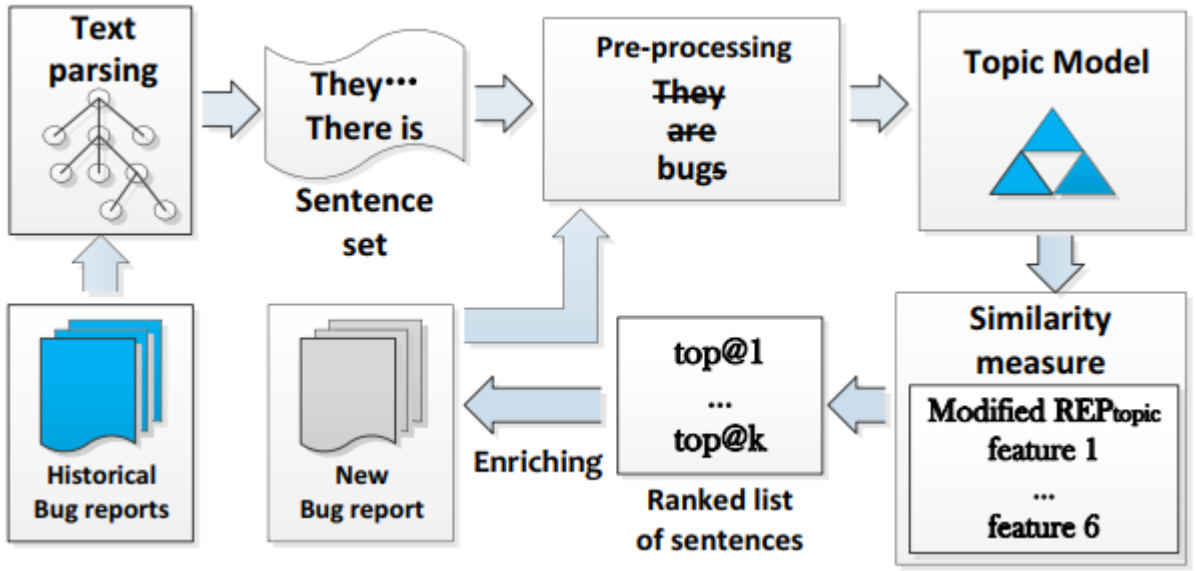
Short bug reports delay the average fixing time.

Motivation



Short bug reports include lower proportion of ingredients.

Method: overview



Method: similarity measure

$$BM25(q,s) = \sum_{t \in q \cap s} IDF(t) \times \frac{TF(t,s)(k_1+1)}{TF(t,s) + k_1(1 - b + b \times \frac{l_s}{l})} \times \frac{(k_3+1)TF(t,q)}{k_3 + TF(t,q)} \quad (2)$$

$$log(\frac{N - n_t + 0.5}{n_t + 0.5})$$

$$REP^{\#}_{topic}(q,s) = \sum_{i=1}^{6} \omega_i \times feature_i$$

$$feature_1(q,s) = BM25(q,s) \; //of \; unigrams$$

$$feature_2(q,s) = BM25(q,s) \; //of \; bigrams$$

$$feature_3(q,s) = \begin{cases} 1 & \text{if q.product=s.product} \\ 0 & \text{otherwise} \end{cases}$$

$$feature_4(q,s) = \begin{cases} 1 & \text{if q.component=s.component} \\ 0 & \text{otherwise} \end{cases}$$

$$feature_5(q,s) = \begin{cases} \frac{q.n_{topic}}{q.N} & \text{if q.topic=s.topic} \\ 0 & \text{otherwise} \end{cases}$$

$$feature_6(q,s) = \begin{cases} 1 & \text{if s}\in S_{stack\,trace} \text{ or } S_{code} \text{ or } S_{patch} \\ 0 & \text{otherwise} \end{cases}$$

Case

Bug 41321 - Ada runtime not initializing fpu (finit)

Status: RESOLVED FIXED
Alias: None
Product: gcc
Component: ada (show other bugs)
Version: 4.4.1 **Importance:** P3 normal
Target Milestone: 4.5.0

Reported: 2009-09-09 16:39 UTC by Aran Clauson
Modified: 2010-10-01 17:47 UTC (History)
CC List: 2 users (show)
Host: x86_64-unknown-netbsd5.99.16
Target: x86_64-unknown-netbsd5.99.16
Build: x86_64-unknown-netbsd5.99.16

Aran Clauson **2009-09-09 16:39:21 UTC** Description

The function __gnat_init_float doesn't call asm("finit") when compiled on NetBSD for x86_64. It looks like Win32, Interix, emx, Lynx, FreeBSD, and OpenBDS have the same problem, but this is unconfirmed.

<bug_id>41321<bug_id>

TOP-1:

Ada runtime missing floating point and error handler initialization on OpenBSD

TOP-2:

"Ada runtime (gcc trunk, r131220) is missing floating point and error handler initialization on OpenBSD."

TOP-3:

../../gcc/gcc/ada/init.c: In function '__gnat_error_handler':
../../gcc/gcc/ada/init.c:444:56: error: unused parameter 'ucontext'
make[3]: *** [ada/init.o] Error 1

TOP-4:

However, gcc occasionally
evaluates hexadecimal (and possibly decimal) floating-point constants
incorrectly starting with version 3.4:

```
das@VARK: cat bar.c
#include stdio.h
int main(int argc, char *argv[]) {
    volatile long double d = 0x3.243f6a8885a31p0L;
    printf("%La\n", d);
    d = 2.0 * d / 2.0;
    printf("%La\n", d);
    return (0);
}
```

TOP-5:

soft float patch
--- gcc-3.3-branch/gcc/config/arm/elf.h 2004-03-30 12:43:45.000000000 -0800
+++ gcc-3.3.3-fg/gcc/config/arm/elf.h 2004-04-07 21:42:05.000000000 -0700

••• •••

TOP-30:

[Ada] Compiler assertion in iterator

Results

RQ1-Human evaluation: Can the enriched bug reports provide further help to fixer assignment?

- Evaluation objects: randomly selected 300 pairs of bug reports from Eclipse, Mozilla, and GCC. Each pair includes a bug report and its enriched version with the Top@30 additional sentences.
- Sent each pair of bug reports to the corresponding real triager by the emails.
- Each triager should answer the following questions and give 1-5 scores.

  - *Q1: Are the additional sentences related to the original bug report?*
  - *Q2: Can the additional sentences help you improve the efficiency of fixer assignment?*

- Results:

Answer to RQ1: The additional sentences ranked ahead in the enriched bug reports are related to the original bug reports. They may provide more help for fixer assignment.

Question	al sentences T26-30
Q1	2.46(0.92)
Q2	2.52(0.90)

Results

RQ2-Effectiveness evaluation: Can the enriched bug reports improve the performance of the automated fixer recommendation in bug repositories?

Project	Top@K	F-measure for different approaches (%) DREX	DRETOM	DevRec
Eclipse	Original	20.75	21.19	28.86
	Top@1	24.42(3.67)	26.84(5.65)	29.52(0.66)
	Top@5	25.57(4.82)	30.92(9.73)	29.93(1.07)
	Top@10	26.21(5.46)	33.42(12.23)	32.46(3.60)
	Top@15	26.49(5.74)	33.79(12.60)	34.25(5.39)
	Top@20	32.57(9.98)	31.12(8.59)	**39.15(5.97)**
	Top@25	32.55(9.96)	30.66(8.13)	39.53(5.35)
	Top@30	32.38(9.79)	29.79(7.26)	39.31(6.13)
GCC	Original	21.53	32.02	28.31
	Top@1	24.63(3.10)	36.60(4.58)	30.67(2.36)
	Top@5	**27.44(5.91)**	**39.62(7.60)**	**36.31(8.00)**
	Top@10	25.02(3.49)	37.51(5.49)	35.03(6.72)
	Top@15	24.97(3.44)	37.05(5.03)	34.75(6.44)
	Top@20	24.61(3.08)	36.68(4.66)	34.23(5.92)
	Top@25	24.50(2.97)	36.42(4.40)	34.06(5.75)
	Top@30	24.12(2.59)	36.28(4.26)	33.93(5.62)

Answer to RQ2: The proposed enrichment approach for bug reports can be used to improve the performance of automated fixer recommendation in bug repositories. However, adding the low-ranked sentences may influence the performance.

Results

RQ3-Feature influence: Which features in the similarity measure have the greatest impact on bug report enrichment for improving automated fixer recommendation?

Variants	F-measure (%) of automated fixer recommendation approaches with the different variants of $REP^{\#}_{topic}$								
	Eclipse			Mozilla			GCC		
	DREX	DRETOM	DevRec	DREX	DRETOM	DevRec	DREX	DRETOM	DevRec
OBR	20.75	21.19	28.86	22.59	22.53	33.18	21.53	32.02	28.31
EBR+$REP^{\#}_{topic}$	26.72	34.56	36.31	32.59	31.76	39.53	27.44	39.62	36.31
EBR+$REP^{\#}_{topic}(\omega_1 = 0)$	26.09	31.44	33.47	30.00	28.62	**34.19**	25.24	**30.94**	**32.29**
EBR+$REP^{\#}_{topic}(\omega_2 = 0)$	25.55	31.56	33.52	30.89	30.02	35.54	25.52	34.62	32.42
EBR+$REP^{\#}_{topic}(\omega_3 = 0)$	**22.33**	**28.95**	**33.12**	**28.00**	28.58	39.22	**21.82**	32.25	32.49
EBR+$REP^{\#}_{topic}(\omega_4 = 0)$	24.74	30.99	33.45	30.41	**28.46**	38.15	25.82	36.13	34.22
EBR+$REP^{\#}_{topic}(\omega_5 = 0)$	26.12	32.85	35.40	30.79	31.53	39.46	25.61	37.19	34.34
EBR+$REP^{\#}_{topic}(\omega_6 = 0)$	26.03	31.79	33.39	30.51	30.92	36.10	25.92	36.99	32.82
EBR+REP_{topic}	26.48	29.02	33.96	29.74	29.49	34.62	23.03	37.07	32.65

Answer to RQ3: In Eclipse, $feature_3$ in $REP^{\#}_{topic}$ has the greatest impact to enrich bug reports for implementing the three automated fixer recommendation approaches such as DREX, DRETOM, and DevRec. In Mozilla, $feature_3$, $feature_4$ (*i.e.*, `Component`), and $feature_1$ (*i.e.*, `Textual similarity`) have the greatest impact when we implement DREX, DRETOM, and DevRec, respectively. In GCC, $feature_1$ has the greatest impact to perform DRETOM and DevRec, and $feature_3$ has the greatest impact to implement DREX.

Results

RQ3-Feature influence: Which features in the similarity measure have the greatest impact on bug report enrichment for improving automated fixer recommendation?

Variants	F-measure (%) of automated fixer recommendation approaches with the different variants of $REP^{\#}_{topic}$								
	Eclipse			Mozilla			GCC		
	DREX	DRETOM	DevRec	DREX	DRETOM	DevRec	DREX	DRETOM	DevRec
OBR	20.75	21.19	28.86	22.59	22.53	33.18	21.53	32.02	28.31
EBR+$REP^{\#}_{topic}$	26.72	34.56	36.31	32.59	31.76	39.53	27.44	39.62	36.31
EBR+$REP^{\#}_{topic}(\omega_1 = 0)$	26.09	31.44	33.47	30.00	28.62	**34.19**	25.24	**30.94**	**32.29**
EBR+$REP^{\#}_{topic}(\omega_2 = 0)$	25.55	31.56	33.52	30.89	30.02	35.54	25.52	34.62	32.42
EBR+$REP^{\#}_{topic}(\omega_3 = 0)$	**22.33**	**28.95**	**33.12**	**28.00**	28.58	39.22	**21.82**	32.25	32.49
EBR+$REP^{\#}_{topic}(\omega_4 = 0)$	24.74	30.99	33.45	30.41	**28.46**	38.15	25.82	36.13	34.22
EBR+$REP^{\#}_{topic}(\omega_5 = 0)$	26.12	32.85	35.40	30.79	31.53	39.46	25.61	37.19	34.34
EBR+$REP^{\#}_{topic}(\omega_6 = 0)$	26.03	31.79	33.39	30.51	30.92	36.10	25.92	36.99	32.82
EBR+REP_{topic}	26.48	29.02	33.96	29.74	29.49	34.62	23.03	37.07	32.65

Answer to RQ4: The performance of automated fixer recommendation approaches using $REP^{\#}_{topic}$ to enrich bug reports is better than using REP_{topic}.

Future

- Mining repositories of mobile apps

Tao Zhang, Jiachi Chen, Xiapu Luo, Tao Li, "Bug Reports for Desktop Software and Mobile Apps in GitHub: What is the difference?" *IEEE Software*, online, 2017. (**Invited to Journal First Session of ICSME 2017**)

Motivation

- Selection of bug tracking systems: GitHub

- Desktop software VS. Mobile apps

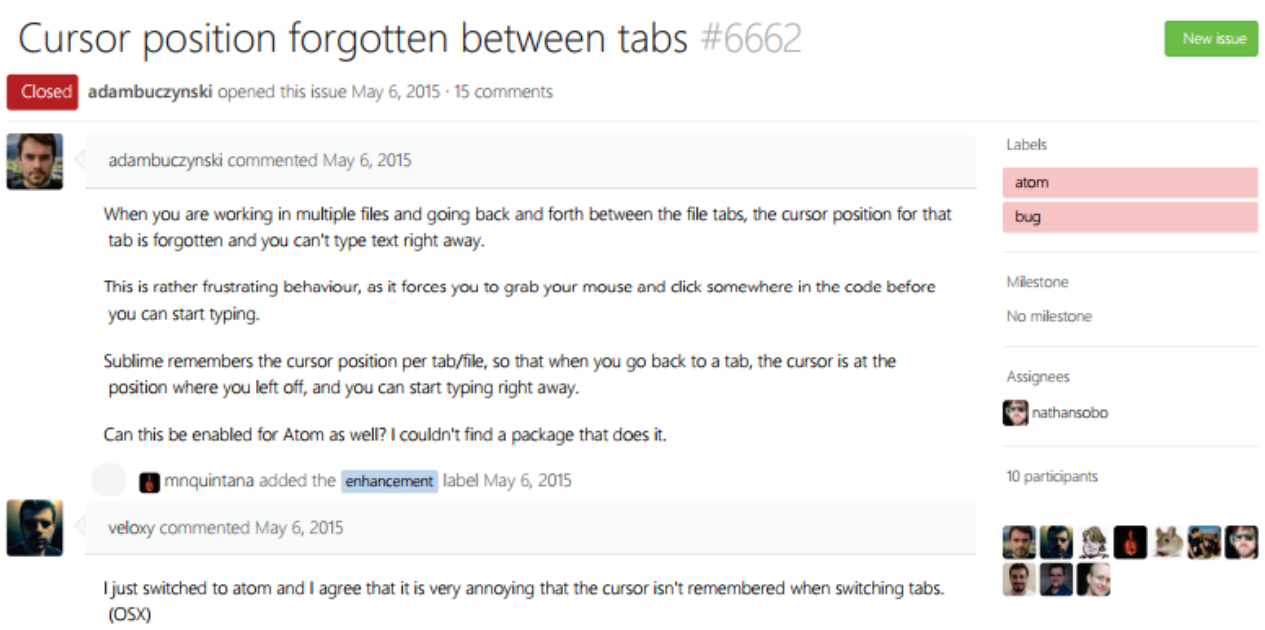
Data Scale

Date sets overview				
Software Type	# Project	# BRs	# Contributors	Average Period/Project
Desktop software	100	117,814	125,941	1033 (days)
Mobile apps	100	19,774	30,077	851 (days)

Top-10 popular projects in our data sets				
Software Type	Project	# BRs	# Contributors	Period
Desktop software	Atom	8,329	329	02/02/2012-11/11/2016
	Brackets	6,221	312	13/12/2011-09/11/2016
	Chosen	1,956	101	18/07/2011-10/11/2016
	Docker	11,421	171	20/01/2013-11/11/2016
	Gitlab	6,668	1,127	13/10/2011-29/10/2016
	Homebrew	17,041	5,641	18/11/2009-01/11/2016
	Lodash	2,135	221	28/08/2012-10/11/2016
	Oh-My-Zsh	3,024	1,032	21/03/2010-11/11/2016
	Select2	3,024	360	24/03/2013-08/11/2016
	Socket.io	1,688	124	01/09/2010-10/11/2016
Mobile apps	ActionBarSherlock	795	53	10/03/2011-20/03/2016
	Android-Universal-Image-Loader	749	35	08/12/2011-28/09/2016
	Butterknife	474	59	27/03/2013-08/11/2016
	Fresco	1,242	63	27/03/2015-08/11/2016
	Glide	1,242	43	08/08/2013-11/11/2016
	Iosched	82	31	10/06/2014-25/08/2016
	MPAndroidChart	1,918	53	28/03/2014-10/11/2016
	Picasso	944	73	14/03/2013-10/11/2016
	ViewPagerIndicator	143	15	04/08/2011-15/06/2016
	Zxing	525	60	18/01/2014-09/11/2016

https: //github.com/PolyUCJC317/IEEE-Software

Textual feature



Cursor position forgotten between tabs #6662

New issue

Closed **adambuczynski** opened this issue May 6, 2015 · 15 comments

adambuczynski commented May 6, 2015

When you are working in multiple files and going back and forth between the file tabs, the cursor position for that tab is forgotten and you can't type text right away.

This is rather frustrating behaviour, as it forces you to grab your mouse and click somewhere in the code before you can start typing.

Sublime remembers the cursor position per tab/file, so that when you go back to a tab, the cursor is at the position where you left off, and you can start typing right away.

Can this be enabled for Atom as well? I couldn't find a package that does it.

mnquintana added the enhancement label May 6, 2015

veloxy commented May 6, 2015

I just switched to atom and I agree that it is very annoying that the cursor isn't remembered when switching tabs. (OSX)

Labels

atom

bug

Milestone

No milestone

Assignees

nathansobo

10 participants

Atom (desktop software)

ImagePipelineFactory doesn't use the PlatformBitmapFactory that is set in ImagePipelineConfig. #665

New issue

Closed **rey5137** opened this issue Oct 19, 2015 · 1 comment

rey5137 commented Oct 19, 2015

You can set a custom `PlatformBitmapFactory` object via `ImagePipelineConfig.Builder.setPlatformBitmapFactory()` method, but `ImagePipelineFactory` seem like not using the config value, and create a new ones in `buildPlatformBitmapFactory(PoolFactory)` method.

michalgr added the bug label Oct 19, 2015

aagnes was assigned by michalgr Oct 19, 2015

aagnes added the needs-details label Oct 19, 2015

aagnes commented Oct 19, 2015

If you want to use your own config, you'll need to pass it when you create your own ImagePipelineFactory. The sequence here really matters. Can you please share your code, or steps how you do?

We expect the usage is

- ImagePipelineConfig config = ImagePipelineConfig.newBuilder()
  `.setPlatformBitmapFactory(platformBitmapFactory)`
  `.build();`
  ImagePipelineFactory factory = new ImagePipelineFactory(config);
  ImagePipeline pipeline = factory.getImagePipeline();

Labels

bug

needs-details

Milestone

No milestone

Assignees

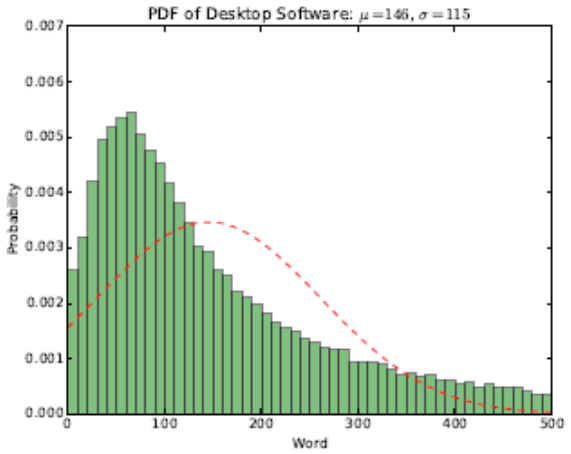
aagnes

4 participants

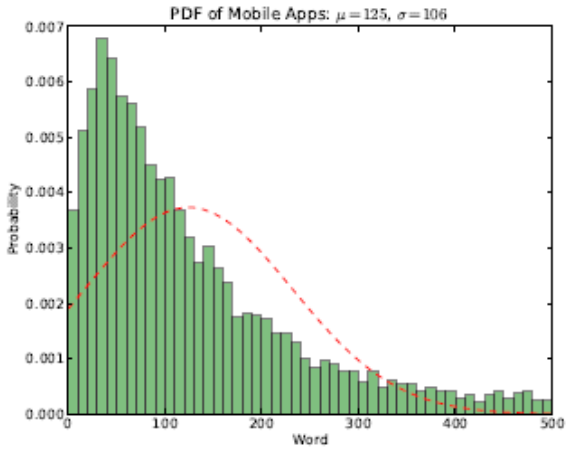
Fresco (mobile apps)

Textual feature

Length comparison: Desktop 251.2 words/BR VS Mobile 181.3 words/BR
Median value: Desktop 129 VS Mobile 101
Probability density function:



(a) PDF of BRs' lengths in desktop software



(b) PDF of BRs' lengths in mobile apps

Compared with BRs in desktop software, the average length per BR in mobile apps is shorter.

Content feature

The quality of a bug report is decided by its Main Elements (MEs).
N. Bettenburg et al., "What makes a good bug report?" FSE 2008.

MEs: Stack trace, Code example, Patch

Stack trace: a stack trace is a part of the active stack frames at a certain point when a program throws the exception.

Code example: a code example can indicate the reason why the bug appears.

Patch: a patch is a piece of software designed to update a program, it is used to fix (or improve) it.

Content feature

How to recognize MEs?

Stack traces: regular expressions (i.e., *at* + *path* + ：*numbers*)

Code examples: island parsing

Patches: regular expressions (i.e., **.**.**.py and **/**/**.py)

Platform	Ratio of BRs including stack traces	Ratio of BRs including patch	Ratio of BRs including code example	All ratio
Desktop software	2.85%	6.12%	11.64%	18.47%
Mobile apps	10.13%	4.41%	23.40%	31.75%

For MEs, their *all ratio* in BRs for mobile apps is larger than that for desktop apps.

Relationship between BRs' features and fixing time

Software Type	Feature	Length					
		0-99	100-199	200-299	300-399	400-499	$\geq$ 500
Desktop software	BR-ME/NME Ratio(%)	5.75/94.25	16.23/83.77	24.37/75.63	29.96/70.04	35.48/64.52	54.21/45.79
	Average Fixing Time(days)	41.94/43.18	58.63/62.04	65.94/69.51	72.26/73.51	76.8/80.21	88.21/95.80
Mobile apps	BR-ME/NME Ratio(%)	13.72/86.28	37.65/62.35	54.15/45.85	63.63/36.37	70.88/29.12	79.18/20.82
	Average Fixing Time(days)	30.62/31.82	33.94/39.99	45.37/49.40	51.26/52.83	51.33/64.46	75.95/80.17
Desktop software	Average Number of MEs	0.06	0.17	0.26	0.33	0.40	0.65
	Average Size (Words) of MEs	16.38	24.26	28.49	33.68	36.27	77.95
	Average Fixing Time(days)	42.91	61.50	68.85	70.33	78.99	92.33
Mobile apps	Average Number of MEs	0.15	0.42	0.66	0.81	0.91	1.11
	Average Size (Words) of MEs	29.33	46.23	71.94	85.51	93.28	174.71
	Average Fixing Time(days)	30.97	37.71	48.79	50.58	55.15	79.29

Some interesting findings

- The ME ratio in mobile apps is higher than that in desktop software at all length fields: $p=0.00153<0.05$ via t-test
- The average fixing time for BRs that include at least one ME is shorter than that for BRs without any ME: $p=0.01547<0.05$ for desktop software; $p=0.03671<0.05$ for mobile apps;
- The average fixing time in mobile apps is shorter than that in desktop software. $P=0.0005744<0.05$
- With the increase of BR's lengths, the average time increases.
  - The small number and size of MEs per BR.
- ME ratio and BR's length: t-test for the values of ME Ratio/Average BR length: $p=0.002071$
- Conclusion: *the high ratio of MEs in BRs of mobile apps may be a major reason why the average fixing time is shorter.*

Online survey: start

Feature influence of bug reports in Github

1. Do you think whether the bug reports with long-length delay the fixing time?

- ◯ Yes
- ◯ No
- ◯ I am not sure

2. Which element(s) can help you shorten the fixing time?

- ☐ Stack trace
- ☐ Code example
- ☐ Patch
- ☐ Description by natural language
- ☐ Noisy data
- ☐ Other (please specify)

3. Which element(s) can delay the fixing process?

- ☐ Stack trace
- ☐ Code example
- ☐ Patch
- ☐ Description by natural language
- ☐ Noisy data
- ☐ Other (please specify)

4. Do you think whether historical commit messages related to reported bugs can help you shorten the fixing time?

- ◯ Yes
- ◯ No
- ◯ I am not sure

5. Which type of software product you are serving?

- ◯ Desktop software
- ◯ Mobile apps

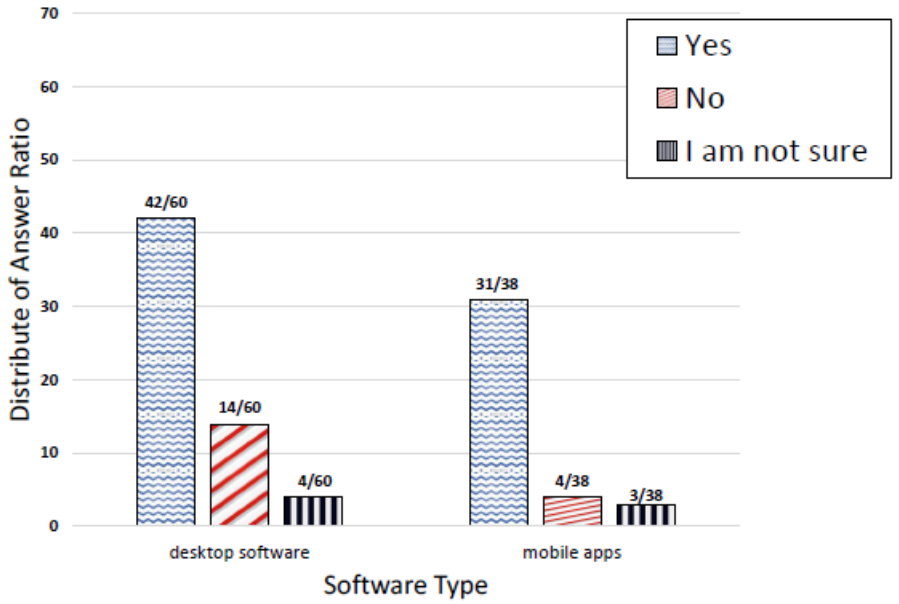
https://www.surveymonkey.com/r/CFZN9FN

Online survey: process

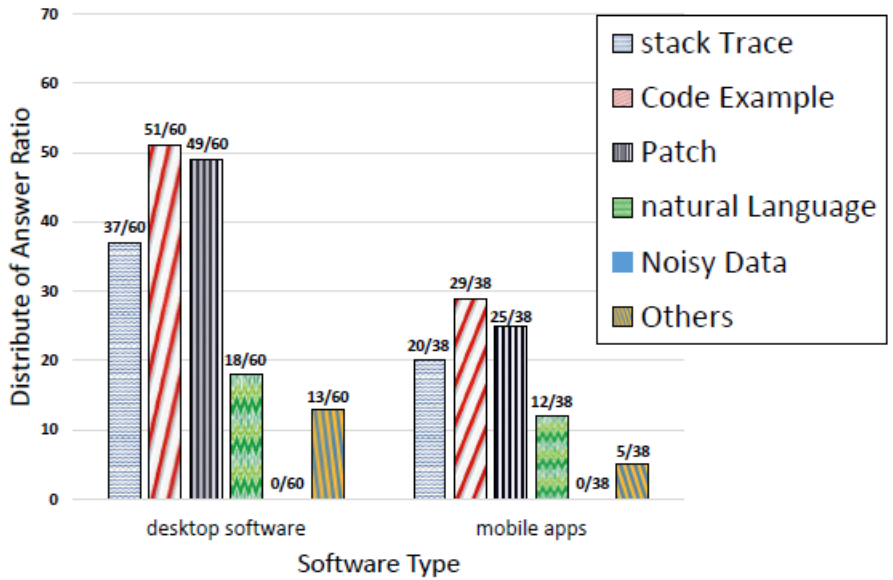
- Send to 100 active contributors who posted the largest number of comments to BRs in desktop software and mobile apps.

- 60 responses (60%) in desktop software, and 38 responses (38%) in mobile apps.

Online survey: result(Q1-Q2)

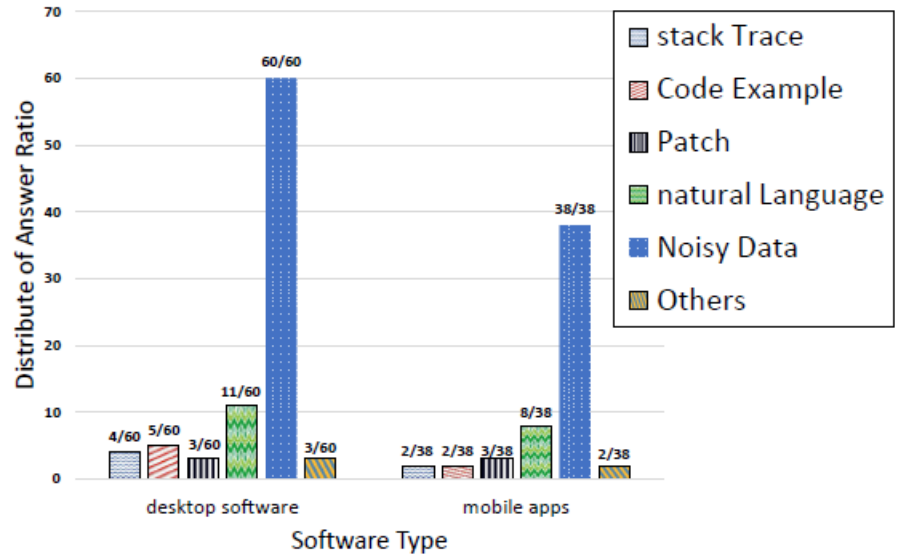


(a) Q1: Do you think whether the bug reports with long-length delay the fixing time?

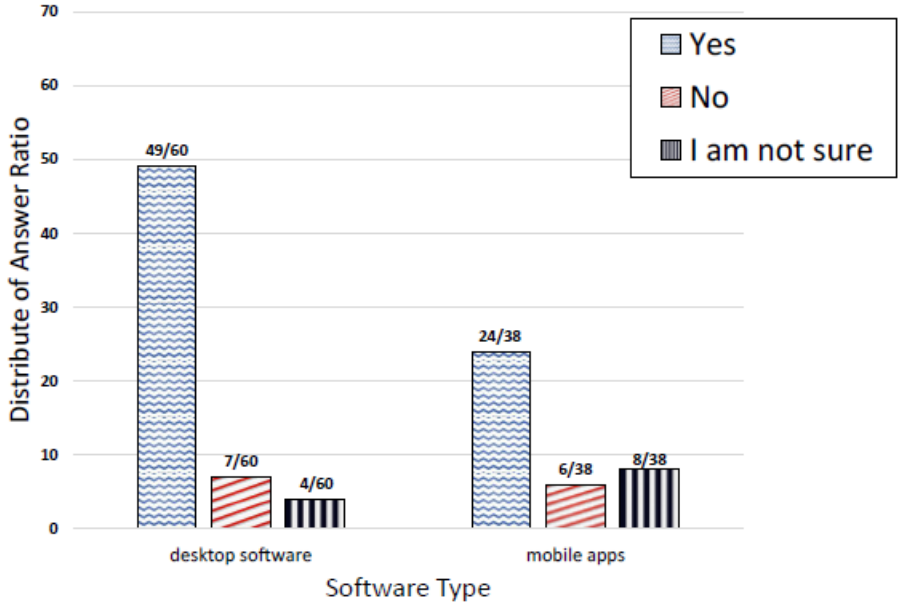


(b) Q2: Which element(s) can help you shorten the fixing time?

Online survey: result(Q3-Q4)



(c) Q3: Which element(s) can delay the fixing process?



(d) Q4: Do you think whether historical commit messages related to reported bugs can help you shorten the fixing time?

Limitation

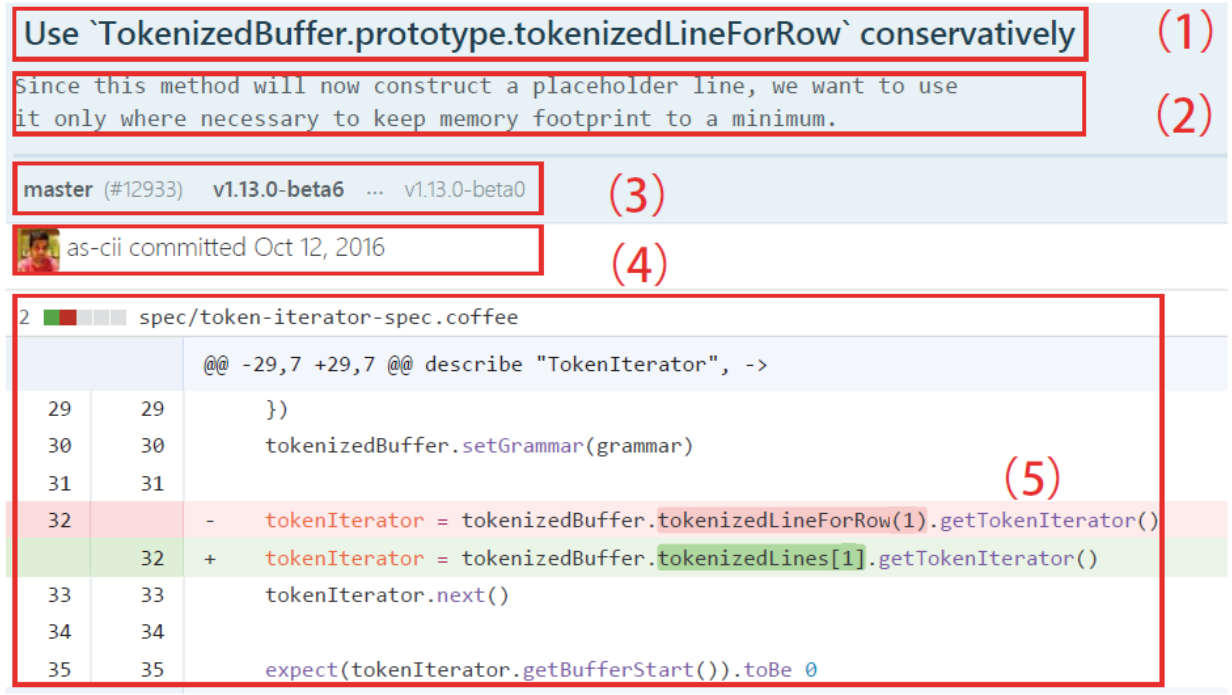
- We did not consider natural language in BRs.

- But it is necessary to deeply analyze NL in BRs.
  - NL： OB (observed behavior)，S2R (steps to reproduce)，and EB (the software’ s expected behavior)

- How can we know the knowledge?
  - Chaparro et al.， “Detecting Missing Information in Bug Descriptions”，FSE 2017.

New resource: commit message

- Enrichment and relevancy
  - Enrichment: high ratio of MEs
  - Relevancy: related to the reported bugs in BRs
- Commit messages



1) title
2) description
3) feature
4) committer
5) changed code

Conclusion

- Show the difference of bug reports in desktop software and mobile apps
- Commit messages can help to enrich bug reports, especially for desktop software
- But this is just a start…

現 在

CLARE: Change Locator for App REviews

- 在移动应用中，真实的定位过程是怎样？

Review Title: Notifications Blocking My View

Comment: Everytime I visit my blog through this app the **notification panel** just occupies the three-fourths of the screen and there is no way to remove it because everytime I tap it what is clicked is the blog underneath.

Review Title: Cannot be trusted

Comment: I used to love this app. Until things started changing for the worst. Doesn't refresh o sync, what I've done online sometimes doesn't show up here and vice versa. Cant even read **notifications**!!!!

Review Title: Latest update

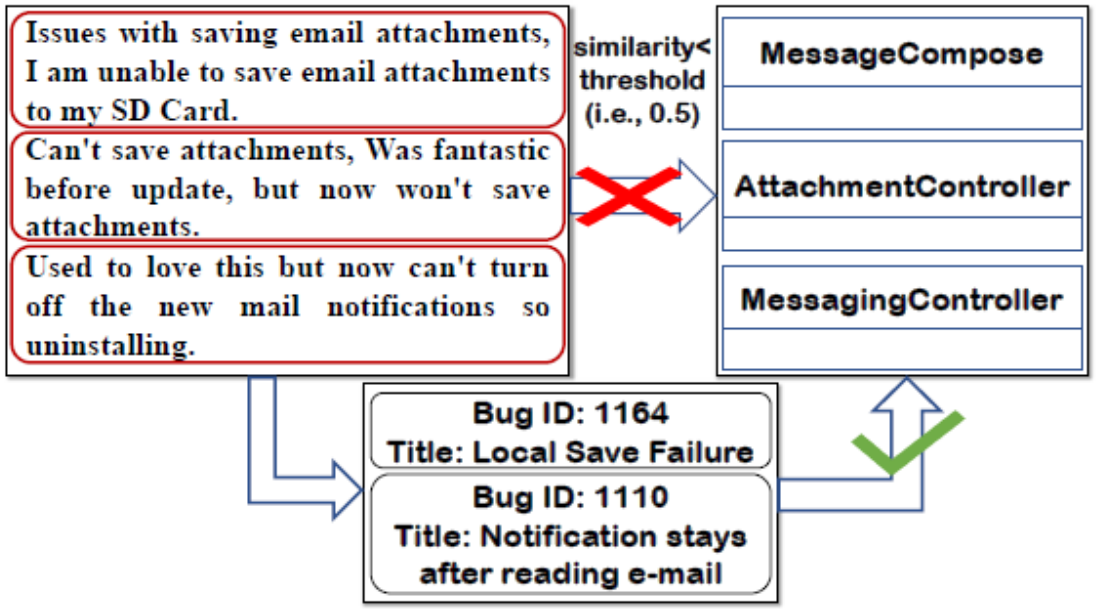
Comment: I'm having problems with the latest update of this app. Once I open it by clicking on a **notification**, I cannot do anything else as it freezes. It's a shame, as the previous version has worked relatively well most of the times.

CLARE: Change Locator for App REviews

- 如果我们只是考虑用户评论信息，结果会是？

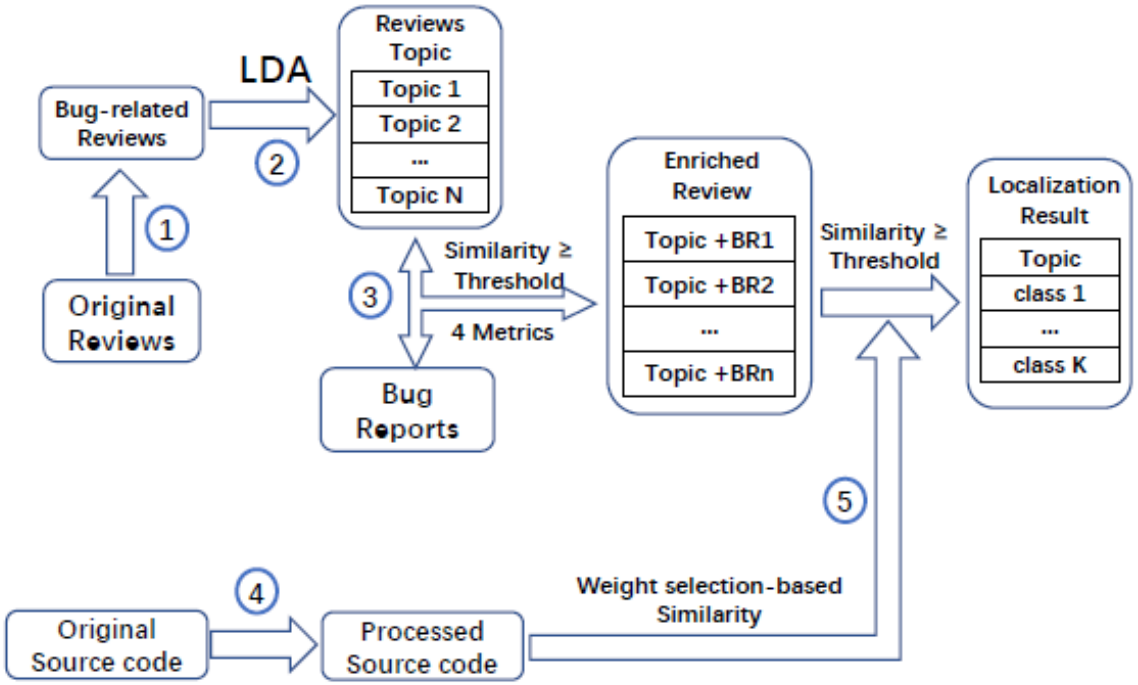
Dice similarity coefficient

$$\mathrm{sim}\left(cluster_j, class_i\right) = \frac{\left|W_{cluster_j} \cap W_{class_i}\right|}{\min\left(\left|W_{cluster_j}\right|, \left|W_{class_i}\right|\right)}$$



CLARE: Change Locator for App REviews

- 我们的做法



很多有趣的工作即将展示

To be continue

First Workshop on Intelligent Bug Fixing (IBF 2019)

co-located with SANER 2019 (Feb. 24-27, 2019, Hangzhou, China)

- Big Data in Bug Fixing Activities
- Bug Analysis
- Software Artifacts Generation
- Bug Summarization/Enrichment
- Duplicates Detection
- Bug Severity/Priority Prediction
- Bug Assignment
- Bug Localization
- Automated Program Repair
- Empirical Studies in Bug Analysis and Fixing
- Intelligent Software Repair
- Intelligent Software Testing
- Knowledge Graph for Bug Fixing
- AI in Intelligent Bug Fixing

Abstract submission deadline: **December 14, 2018 AoE**

Paper submission deadline: **December 21, 2018 AoE**

Notifications: **January 11, 2019**

Camera Ready deadline: ** **January 22, 2019** **

Workshop day: **February 24, 2019**

All submissions must be in English. Full paper submissions should not exceed 10 pages (the last 2 pages can be used for references). Short paper submissions should not exceed 6 pages and poster paper submission should not exceed 2 pages. We invite the authors of selected papers to submit their extended versions to a special issue of IEEE Transactions on Reliability or IEEE Access.

QRS 2019

The 19th IEEE International Conference on
Software Quality, Reliability, and Security
July 22-26, 2019 • Sofia, Bulgaria

Topics of Interest

- Reliability, Security, Availability, and Safety of Software Systems
- Software Testing, Verification, and Validation
- Program Debugging and Comprehension
- Information and Software Quality Assurance
- Fault Tolerance for Software Reliability Improvement
- Modeling, Prediction, Simulation, and Evaluation
- Metrics, Measurements, and Analysis
- Secure and Reliable Storage
- Software Penetration and Protection
- Software Vulnerabilities
- Formal Methods
- Malware Detection and Analysis
- Intrusion Detection and Prevention
- Operating System Security and Reliability
- Mobile and Smartphone Applications
- Internet of Things and Cloud Computing
- Information and Knowledge Management
- Benchmark, Tools, Industrial Applications, and Empirical Studies

Important Dates

Jan 15, 2019
Workshop proposals

Mar 8, 2019
Abstracts

Mar 15, 2019
Regular & Short papers due

May 1, 2019
Workshop papers due

May 1, 2019
Fast Abstract & Industry Track due

May 25, 2019
Author notification

Jun 10, 2019
Camera-ready & author registration

Jul 22 - 26, 2019
Conference dates

Authors of selected papers from QRS 2019 will be invited to submit an extended version to a special issue of Journal of Systems and Software (JSS).