5篇论文

名称	期刊/会议
基于信息检索的软件缺陷定位方法综述	软件学报
Learning to Combine Multiple Ranking Metrics for Fault Localization	ICSME
What makes a good bug report?	TSE
Improving bug localization with word embedding and enhanced convolutional neural networks	IST
Improved bug localization based on code change histories and bug reports	IST

What Makes a Good Bug Report?

汇报人：王昭丹

CONTENTS

INTRODUCTION

- In software development, bug reports provide crucial information to developers.
- Bug reports **vary in their quality of content**; they often provide inadequate or incorrect information.

Examples

- "Sem Web" (APACHE bug COCOON-1254)
- "wqqwqw" (ECLIPSE bug #145133)
- "GUI" with comment "The page is too clumsy" (MOZILLA bug #109242)

CUEZILLA TOOL

Figure 1: Mockup of CUEZILLA's user interface. It recommends improvements to the report (left image). To encourage the user to follow the advice, CUEZILLA provides facts that are mined from history (right image).

SURVEY DESIGN

Selection of Participants

- **experienced developers** — — at least 50 bug reports
- **experienced reporters** — — at least 25 bug reports (=a user) zero bugs (=not a developer)

The Questionnaire

Table 1: Number of invitations sent to and responses by developers and reporters of the APACHE, ECLIPSE, and MOZILLA projects.

	Developers					Reporters				
Project	Contacted	Bounces	Reached	Responses (Rate)	Comments	Contacted	Bounces	Reached	Responses (Rate)	Comments
APACHE	194	5	189	34 (18.0%)	12	165	17	148	37 (25.0%)	10
ECLIPSE	365	29	336	50 (14.9%)	15	378	8	370	50 (13.5%)	20
MOZILLA	313	29	284	72 (25.4%)	21	811	130	681	223 (32.7%)	97
Total	872	63	809	156 (19.3%)	48	1354	155	1199	310 (25.9%)	127

Contents of bug reports.

D1: Which of the following items have you previously used when fixing bugs?
D2: Which three items helped you the most?

R1: Which of the following items have you previously provided when reporting bugs?
R2: Which three items were the most difficult to provide?

R3: In your opinion, which three items are most relevant for developers when fixing bugs?

❑ product	❑ hardware	❑ observed behavior	❑ screenshots
❑ component	❑ operating system	❑ expected behavior	❑ code examples
❑ version	❑ summary	❑ steps to reproduce	❑ error reports
❑ severity	❑ build information	❑ stack traces	❑ test cases

Problems with bug reports.

D3: Which of the following problems have you encountered when fixing bugs?
D4: Which three problems caused you most delay in fixing bugs?

You were given wrong:	There were errors in:	The reporter used:	Others:
❑ product name	❑ code examples	❑ bad grammar	❑ duplicates
❑ component name	❑ steps to reproduce	❑ unstructured text	❑ spam
❑ version number	❑ test cases	❑ prose text	❑ incomplete information
❑ hardware	❑ stack traces	❑ too long text	❑ viruses/worms
❑ operating system		❑ non-technical language	
❑ observed behavior		❑ no spell check	
❑ expected behavior			

Comments.

D5/R4: Please feel free to share any interesting thoughts or experiences.

Figure 2: The questionnaire presented to APACHE, ECLIPSE, and MOZILLA developers (D*x*) and reporters (R*x*).

SURVEY RESULTS

Steps to reproduce

Observed and expected behavior

All consistent responses for the project
Number of times that *item* was selected in D1
Number of times that *item* was selected in D1 and D2
Number of times that *item* was selected in D1 but not D2

Table 2: Results from the survey among developers. (130 consistent responses by APACHE, ECLIPSE, and MOZILLA developers.)

Contents of bug reports (D1/D2). *In parentheses: importance of item.*

- product (5%)
- component (3%)
- version (12%)
- severity (0%)
- hardware (0%)
- operating system (4%)
- summary (13%)
- build information (8%)
- observed behavior (33%)
- expected behavior (22%)
- steps to reproduce (83%)
- stack traces (57%)
- screenshots (26%)
- code examples (14%)
- error reports (12%)
- test cases (51%)

Problems with bug reports (D3/D4). *In parentheses: severeness of problem.*

You were given wrong:
- product name (7%)
- component name (15%)
- version number (22%)
- hardware (8%)
- operating system (20%)
- observed behavior (48%)
- expected behavior (27%)

There were errors in:
- code examples (15%)
- steps to reproduce (79%)
- test cases (38%)
- stack traces (25%)

The reporter used:
- bad grammar (16%)
- unstructured text (34%)
- prose text (18%)
- too long text (26%)
- non-technical language (19%)
- no spell check (0%)

Others:
- duplicates (10%)
- spam (0%)
- incomplete information (74%)
- viruses/worms (0%)

Table 3: Results from the survey among reporters. (215 consistent responses by APACHE, ECLIPSE, and MOZILLA reporters.)

Contents of bug reports (R1/R2). *In parentheses: difficulty of item.*

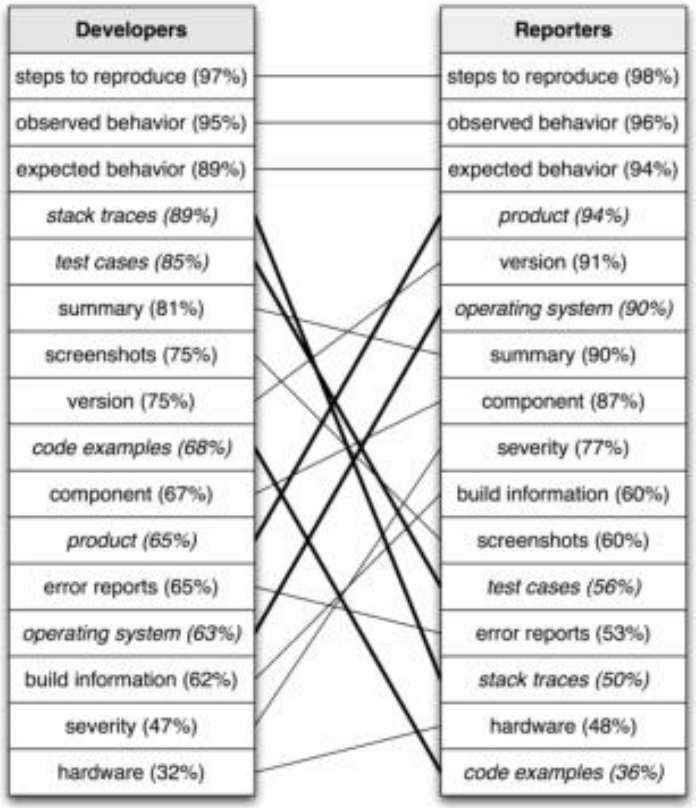
- product (0%)
- component (22%)
- version (1%)
- severity (5%)
- hardware (1%)
- operating system (1%)
- summary (4%)
- build information (3%)
- observed behavior (2%)
- expected behavior (3%)
- steps to reproduce (51%)
- stack traces (24%)
- screenshots (8%)
- code examples (43%)
- error reports (2%)
- test cases (75%)

Contents considered to be relevant for developers (R3). *In parentheses: frequency of item in R3.*

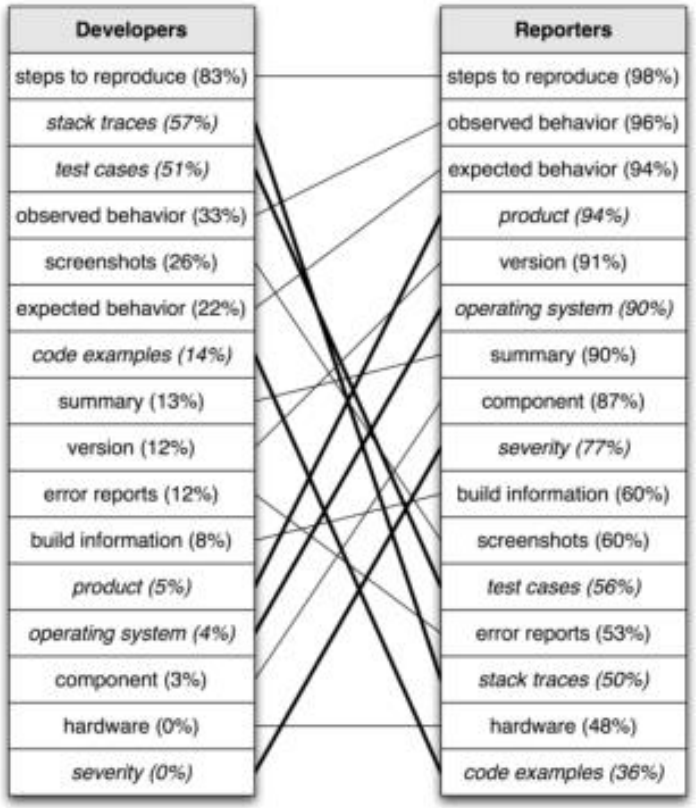
- product (7%)
- component (4%)
- version (12%)
- severity (2%)
- hardware (0%)
- operating system (4%)
- summary (6%)
- build information (8%)
- observed behavior (33%)
- expected behavior (22%)
- steps to reproduce (78%)
- stack traces (33%)
- screenshots (5%)
- code examples (9%)
- error reports (9%)
- test cases (43%)

SURVEY RESULTS

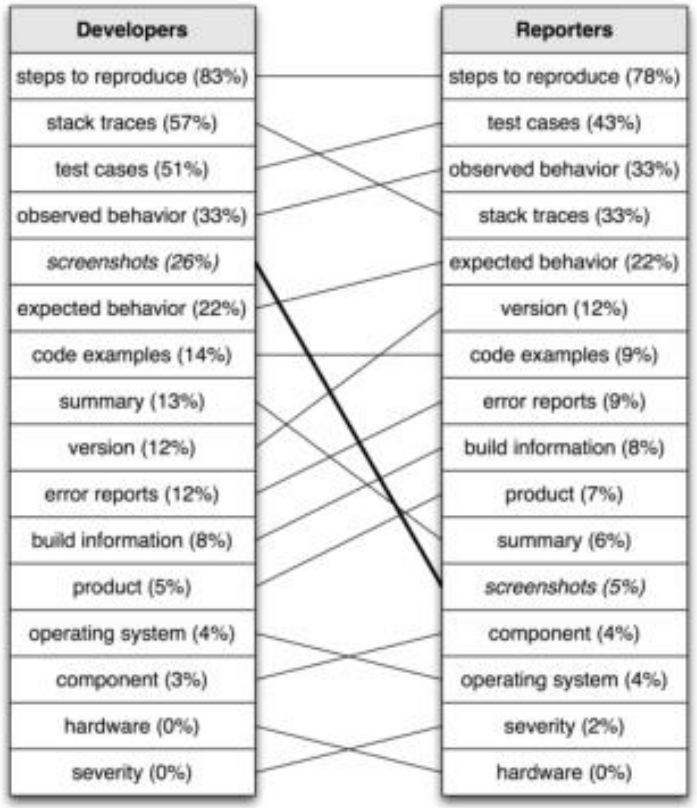
Mismatch between what developers consider most helpful and what users provide !



(a) Information used by developers vs. provided by reporters.



(b) Most helpful for developers vs. provided by reporters.



(c) Most helpful for developers vs. reporters expected to be helpful.

RATING BUG REPORTS

Table 4: Developers rated the quality of ECLIPSE bug reports.

Bug Report	Votes	Rating
Tree - Selection listener stops default expansion (#31021)	3	5.00
JControlModel "eats up" exceptions (#38087)	5	4.8
Search - Type names are lost [search] (#42481)	4	4.50
150M1 withincode type pattern exception (#83875)	5	4.40
ToolItem leaks Images (#28361)	6	4.33
...	...	...
Selection count not updated (#95279)	4	2.25
Outline view should [...] show all project symbols (#108759)	2	2.00
Pref Page [...] Restore Defaults button does nothing (#51558)	6	1.83
[...]<Incorrect /missing screen capture> (#99885)	4	1.75
Create a new plugin using CDT. (#175222)	7	1.57

I20030205

Run the following example. Double click on a tree item and notice that it does not expand.

Comment out the Selection listener and now double click on any tree item and notice that it expands.

```
public static void main(String[] args) {
    Display display = new Display();
    Shell shell = new Shell(display);
    [...] (21 lines of code removed)
    display.dispose();
}
```

(ECLIPSE bug report #31021)

√

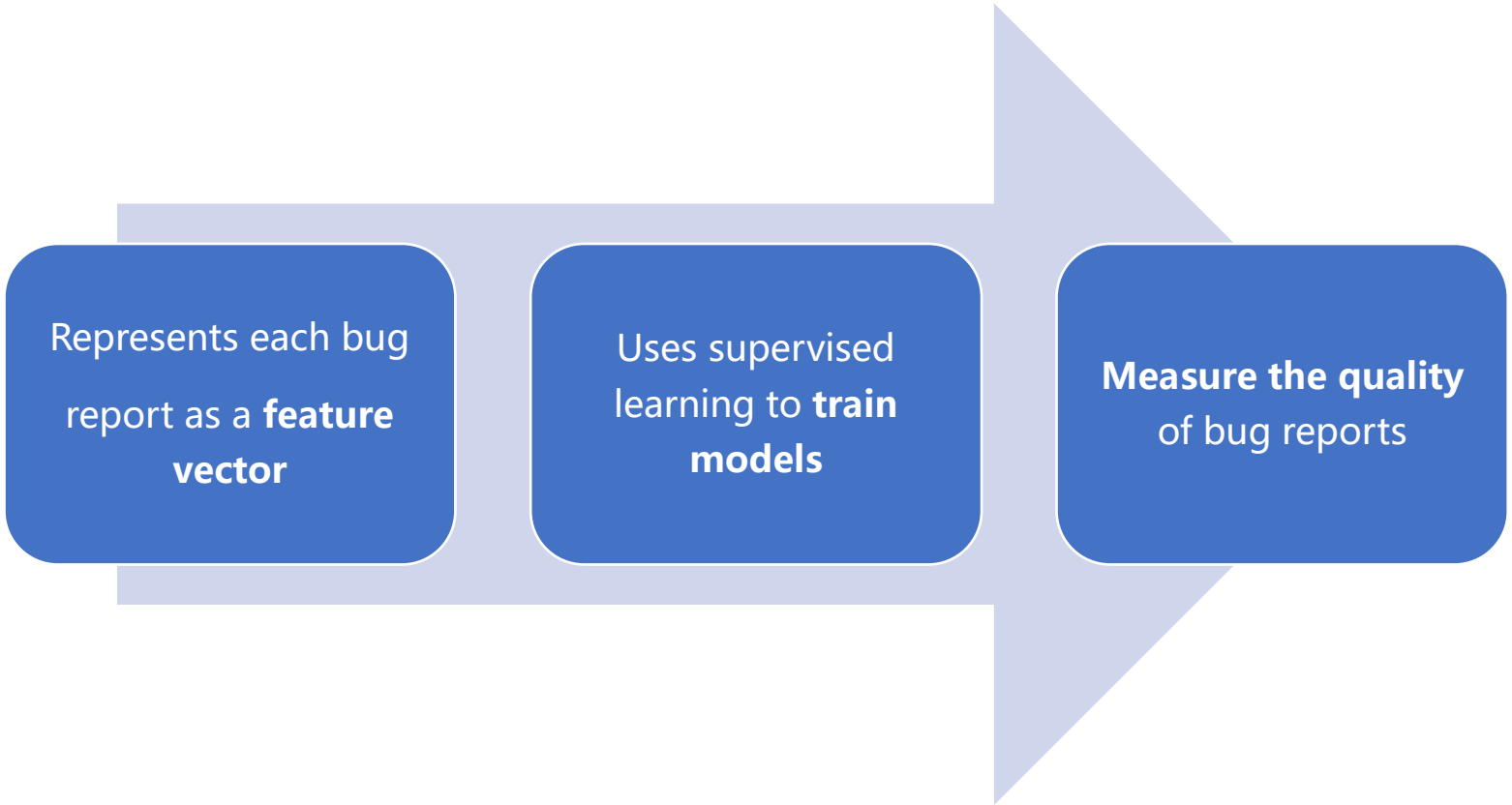
I wand to create a new plugin in Eclipse using CDT. Shall it possible. I had made a R&D in eclipse documentation. I had get an idea about create a plugin using Java. But i wand to create a new plugin (user defined plugin) using CDT. After that I wand to impliment it in my programe. If it possible?. Any one can help me please...

(ECLIPSE bug report #175222)

×

MEASURING BUG REPORT QUALITY

How CUEZILLA works ?



Input Features

Itemizations

- itemization character (such as -,*, or +).

Keyword completeness

- reused the dataset by Andy Ko
- quality-score of bug reports

Code Samples

- island parsing
- recognize declarations, comments, conditional statements, and loops

Stack Traces

- JAVA stack traces
- GDB stack traces
- MOZILLA talkback data

Patches

- regular expressions
- several start lines and blocks

Screenshots

- File tool in UNIX
- Image —— screenshot
- Text —— code examples, stack traces, and patches

Input Features

- Use the **style tool**
- The readability of a text is measured by the number of syllables per word and the length of sentences.
- The higher a readability score, the more complex a text is to read.
- **Seven readability measures**: Kincaid, Automated Readability Index (ARI), Coleman-Liau, Flesh, Fog, Lix, and SMOG Grade

Evaluation Setup

Three Models

- support vector machines (SVM)
- generalized linear regression (GLR)
- stepwise linear regression

two setups

Within project

- leave-one-out cross-validation technique
- Maximize training data

1

Across projects

- Test if models from one project can be transferred to others

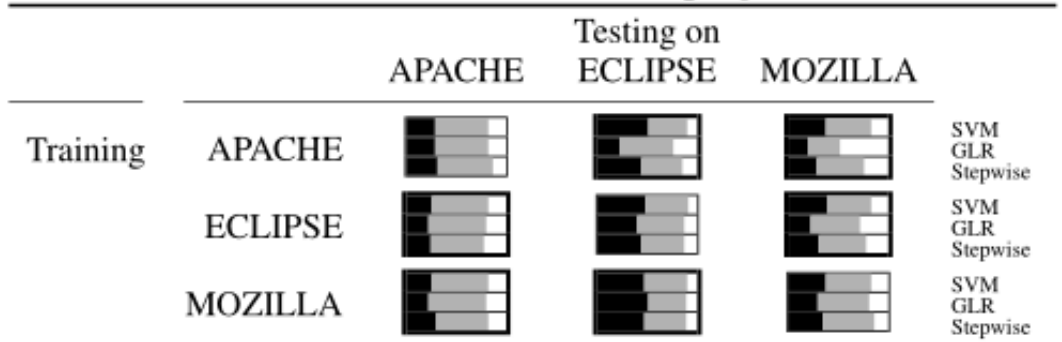
2

Table 6: Leave-one-out cross-validation within projects.

	APACHE	ECLIPSE	MOZILLA
Support vector machine	28% (82%)	48% (91%)	37% (82%)
Generalized linear regression	28% (82%)	40% (87%)	29% (80%)
Stepwise linear regression	31% (86%)	44% (87%)	34% (85%)

Number of unique bugs rated by developers
Number of perfect agreements
Number of off-by-one agreements

Table 7: Validation across projects.



学 习 进 展 & 暑 期 计 划

感 谢 您 的 聆 听

汇报人：王昭丹