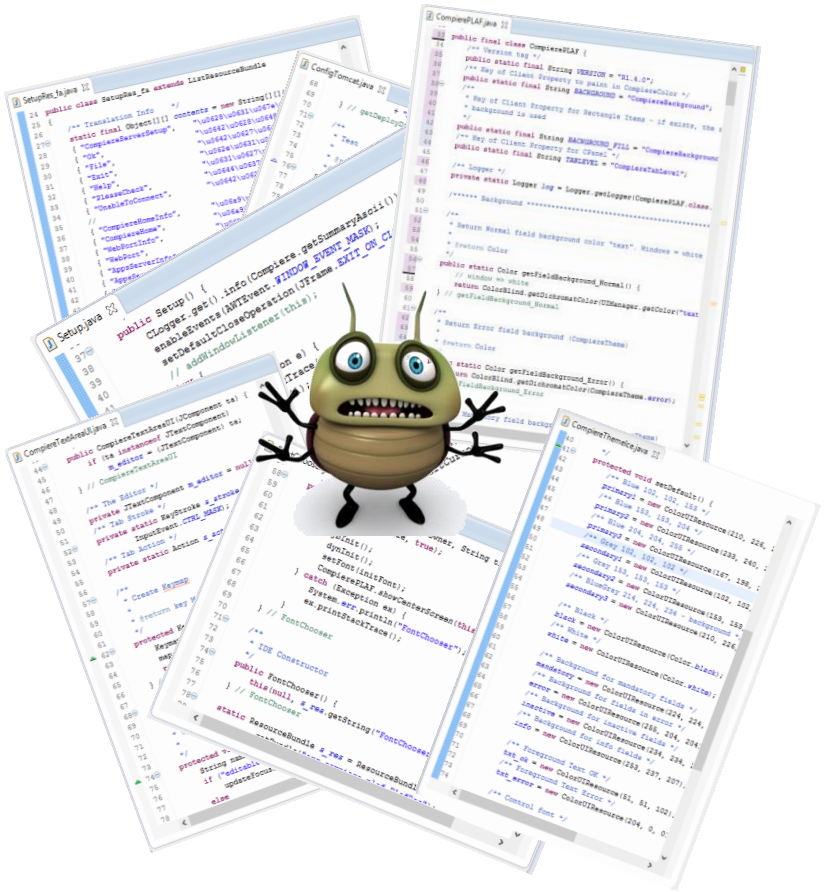
On the Relationship between the Vocabulary of Bug Reports and Source Code

Laura Moreno
Wathsala Bandara
Sonia Haiduc
Andrian Marcus

ICSM'13
ERA track

Bug Location

- Determine where to fix a bug
- Important task during software evolution
- Text Retrieval (TR) based approaches have been proposed to automate this task



TR-based Bug Location

Bug 12118 - **[CVS EXTSSH] extssh should tell user if ssh server is not of a compatible protocol**

Status:	RESOLVED FIXED	**Reported:**	2002-03-22 09:51 EST by Jean-Michel Lemieux CLA
Product:	Platform	**Modified:**	2002-11-06 10:41 EST (History)
Component:	Team	**CC List:**	0 users
Version:	2.0		
Hardware:	PC Windows 2000	**See Also:**	
Importance:	P3 normal (vote)		
Target Milestone:	2.1 M3		
Assigned To:	Boris Shingarov		
QA Contact:			
URL:			
Whiteboard:			
Keywords:			
Depends on:			
Blocks:			
	Show dependency tree		

Attachments			
Patch (1.42 KB, patch) 2002-10-23 19:50 EDT, Boris Shingarov	*no flags*	Details | Diff	
Add an attachment (proposed patch, testcase, etc.)		View All	

Jean-Michel Lemieux CLA 2002-03-22 09:51:49 EST

```
It would be nice if...
When the ssh client initializes a connection to the
server, it already gets the servers revision. It can
throw an exception and tell the user that the
connection method can not operate with SSH2.
```

TR-based Bug Location: Assumption

Bug 12118 - [CVS EXTSSH] extssh should tell user if ssh server is not of a compatible protocol

Status: RESOLVED FIXED
Reported: 2002-03-22 09:51 EST by Jean-Michel Lemieux CLA
Modified: 2002-11-06 10:41 EST (History)
CC List: 0 users
See Also:

Product: Platform
Component: Team
Version: 2.0
Hardware: PC Windows 2000
Importance: P3 normal (vote)
Target Milestone: 2.1 M3
Assigned To: Boris Shingarov
QA Contact:
URL:
Whiteboard:
Keywords:
Depends on:
Blocks:
Show dependency tree

Attachments		
Patch (1.42 KB, patch) 2002-10-23 19:50 EDT, Boris Shingarov	no flags	Details | Diff
Add an attachment (proposed patch, testcase, etc.) View All		

Jean-Michel Lemieux CLA 2002-03-22 09:51:49 EST

It would be nice if
When the ssh client initializes a connection to the server, it already gets the servers revision. It can throw an exception and tell the user that the connection method can not operate with SSH2.

```
public class Client {
    // client identification string
    private static final String clientId = "SSH-1.5-J

    // server identification string
    private static String serverId = null;

    // maximum outgoing packet size
    private static final int MAX_CLIENT_PACKET_SIZE =

    // packet types
    private static final int SSH_MSG_DISCONNECT = 1;
    private static final int SSH_SMSG_PUBLIC_KEY = 2;
    private static final int SSH_CMSG_SESSION_KEY = 3
    private static final int SSH_CMSG_USER = 4;

    // cipher names
    private static String[] cipherNames = {
        "None", "IDEA", "DES", "3DES", "TSS", "RC

    // cipher types
    private static int SSH_CIPHER_NONE = 0;
    private static int SSH_CIPHER_IDEA = 1;
    private static int SSH_CIPHER_DES = 2;

    // authentication methods
    private final int SSH_AUTH_RHOSTS = 1;
    private final int SSH_AUTH_RSA = 2;
    private final int SSH_AUTH_PASSWORD = 3;
    private final int SSH_AUTH_RHOSTS_RSA = 4;

    private String host;
```

Research Questions

RQ1 Is the vocabulary of bug reports reflected in the code?

RQ2 What is the code location of the shared terms between bug reports and patched classes?

RQ3 Is the number of shared terms between bug reports and classes an adequate measure for bug location?

Methodology

- Data previously used in TR-based bug location (corpus)

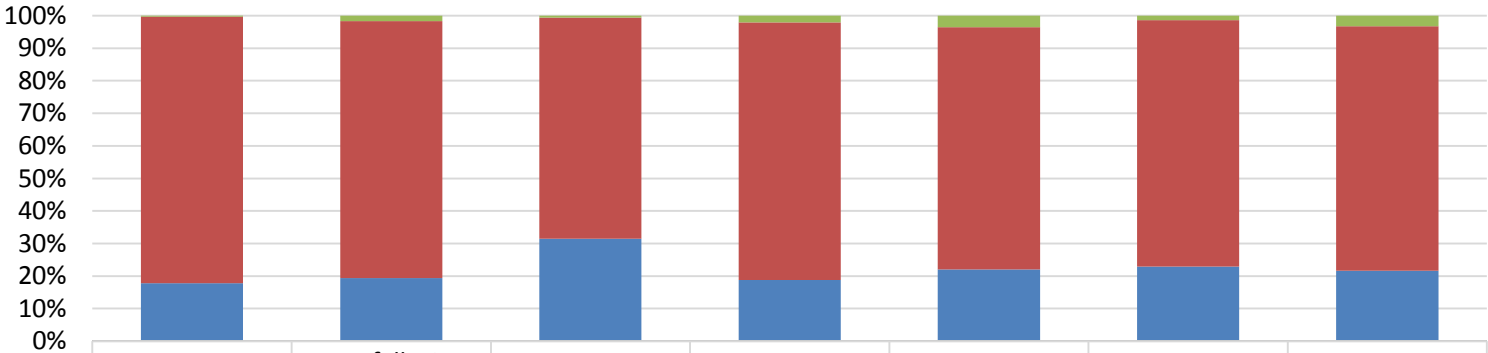
System	# of classes	# of bug reports	# of patched classes
ADempiere 3.10	1896	16	16
Art of Illusion 2.4.1	570	10	13
aTunes 1.10	439	17	22
Eclipse 2.0	7689	13	14
Eclipse 3.0	22980	40	74
JEdit	801	18	27
Total	**34375**	**114**	**166**

Methodology

- Data previously used in TR-based bug location (corpus)
- Documents as bags of words
  - Bug reports (title + description) and classes (identifiers + comments + literals)
  - Identifier splitting, stop word removal, stemming
- Different measures on pairs of documents <*bug report*, *class*>

RQ1 Is the vocabulary of bug reports reflected in the code?

- Measure: common terms between bug reports and classes
- Object: full set, i.e., 1,077,074 pairs <bug report, class>



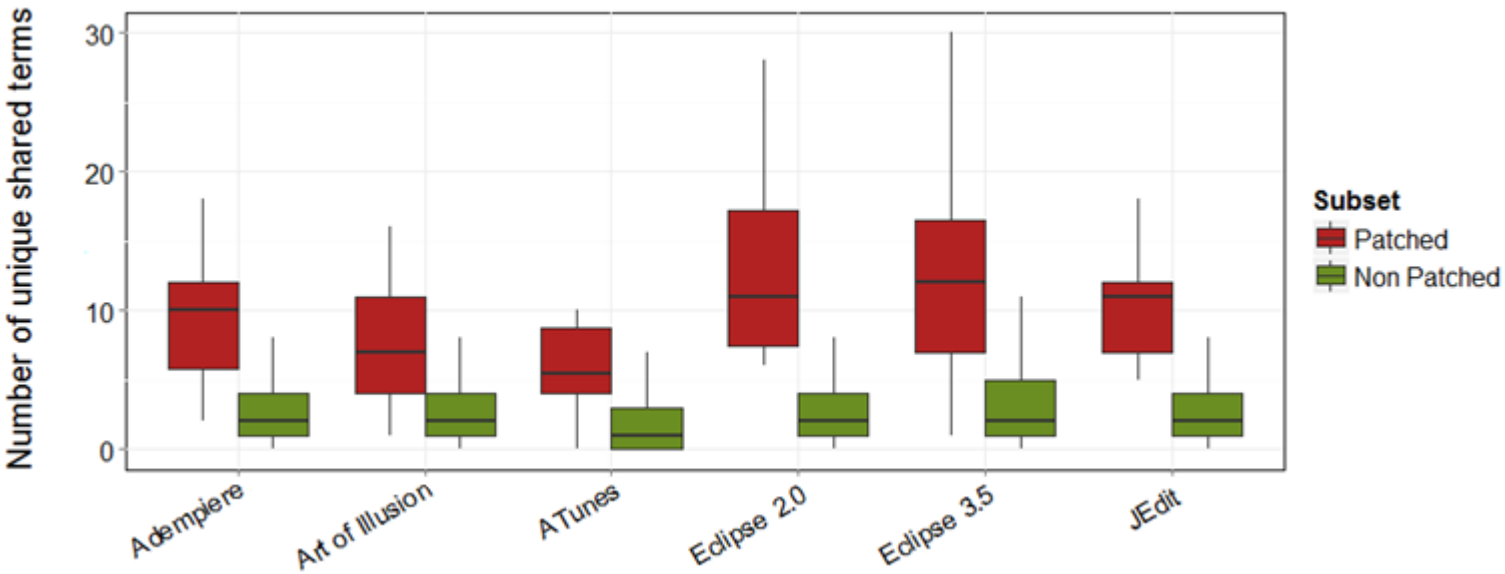
	ADempiere 3.10	Art of Illusion 2.4.1	aTunes 1.10	Eclipse 2.0	Eclipse 3.5	JEdit 4.2	All
More than 13 terms	0.29%	1.70%	0.56%	2.09%	3.50%	1.32%	3.22%
1-13 terms	81.89%	78.91%	67.94%	79.09%	74.47%	75.76%	75.10%
0 terms	17.82%	19.39%	31.50%	18.83%	22.03%	22.92%	21.68%

Bug reports share vocabulary with a large number of classes in a software system

RQ1

Is the vocabulary of bug reports reflected in the code?

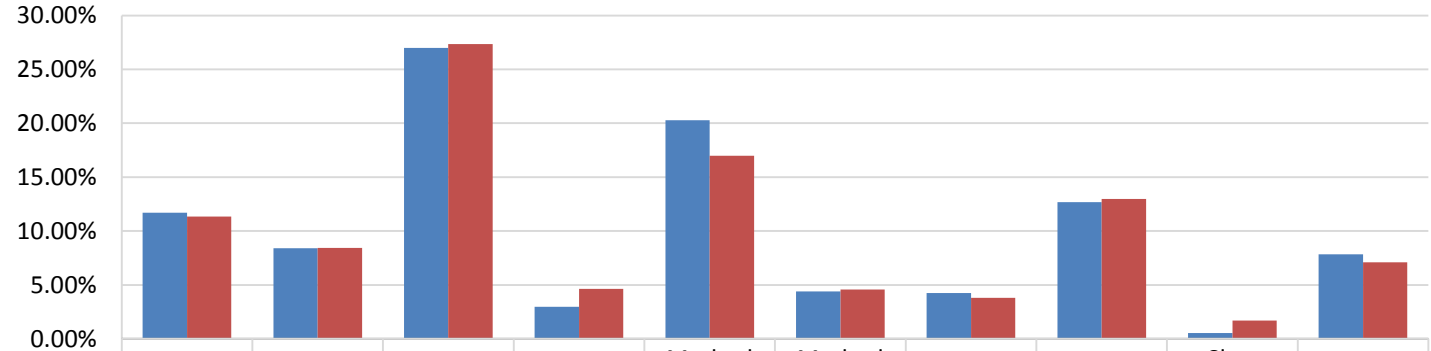
- Measure: common terms between bug reports and classes
- Object: patched subset, i.e., 166 pairs <*bug report*, *corresponding patched class*>, and non-patched subset



Bug reports have more terms in common with the patched classes than with the non-patched classes

RQ2 What is the code location of the shared terms?

- Measure: percentage of terms and contribution to shared terms by code location
- Object: patched subset, i.e., 166 pairs <*bug report*, *corresponding patched class*>

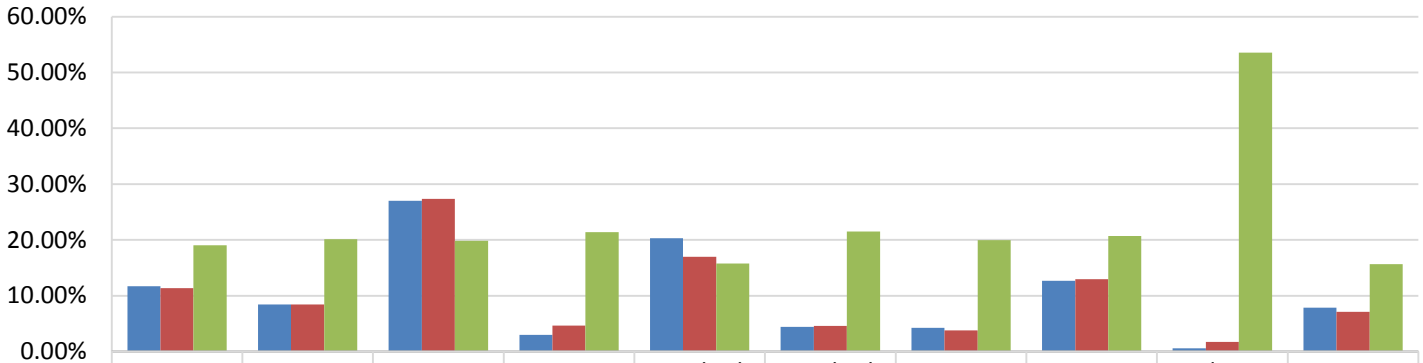


	Argument	Attribute	Comment	Literal	Method call	Method name	Parameter	Type	Class Name	Variable
Avg. % of terms	11.71%	8.40%	26.98%	2.99%	20.28%	4.40%	4.25%	12.69%	0.55%	7.86%
Avg. contrib. to shared terms	11.34%	8.45%	27.33%	4.64%	16.99%	4.58%	3.81%	12.97%	1.70%	7.11%

The more verbose a code location is, the more it contributes to the common vocabulary between a patched class and its respective bug report

RQ2 What is the code location of the shared terms?

- Measure: percentage of shared terms by code location
- Object: patched subset, i.e., 166 pairs <*bug report*, *corresponding patched class*>

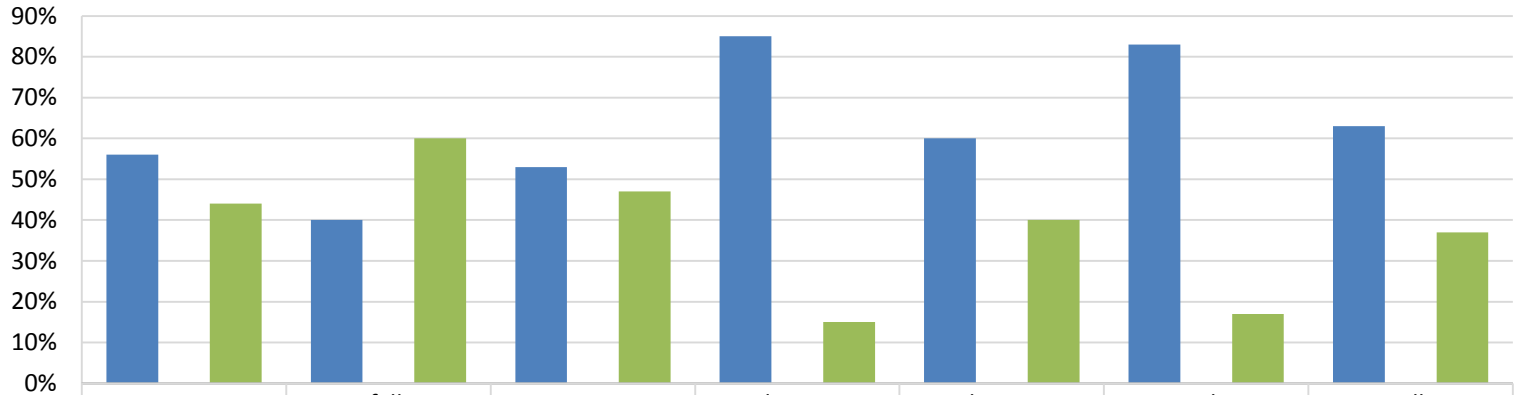


	Argument	Attribute	Comment	Literal	Method call	Method name	Parameter	Type	Class Name	Variable
Avg. % of terms	11.71%	8.40%	26.98%	2.99%	20.28%	4.40%	4.25%	12.69%	0.55%	7.86%
Avg. contrib. to shared terms	11.34%	8.45%	27.33%	4.64%	16.99%	4.58%	3.81%	12.97%	1.70%	7.11%
Avg. % of shared terms	19.06%	20.10%	19.86%	21.40%	15.78%	21.52%	19.97%	20.69%	53.57%	15.68%

The names of the patched classes are more likely to share terms with the respective bug reports than other locations

RQ3 Is the number of shared terms an adequate measure for bug location?

- Measure: effectiveness of LSI and Shared Terms
- Object: full set using bug reports as queries

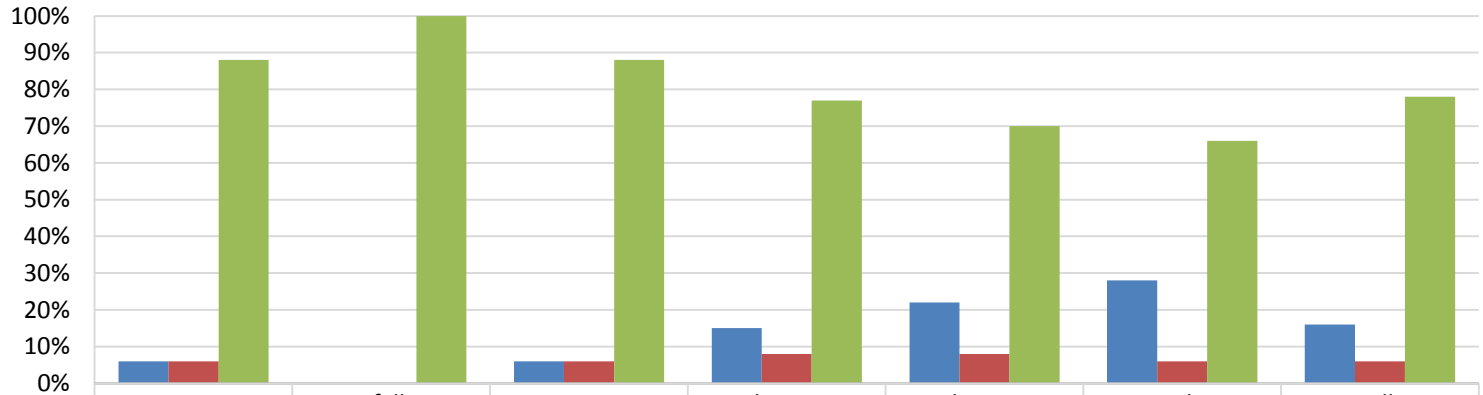


	ADempiere	Art of Illusion	aTunes	Eclipse 2.0	Eclipse 3.5	JEdit	All
ST better than LSI	56%	40%	53%	85%	60%	83%	63%
ST equal to LSI	0%	0%	0%	0%	0%	0%	0%
ST worse than LSI	44%	60%	47%	15%	40%	17%	37%

The number of shared terms between bug reports and classes supports bug location better than LSI

RQ3 Is the number of shared terms an adequate measure for bug location?

- Measure: effectiveness of Lucene and Shared Terms
- Object: full set using bug reports as queries



	ADempiere	Art of Illusion	aTunes	Eclipse 2.0	Eclipse 3.5	JEdit	All
ST better than Lucene	6%	0%	6%	15%	22%	28%	16%
ST equal to Lucene	6%	0%	6%	8%	8%	6%	6%
ST worse than Lucene	88%	100%	88%	77%	70%	66%	78%

The number of shared terms between bug reports and classes supports bug location better than LSI, yet not as well as Lucene

In Summary

- Is the vocabulary of bug reports reflected in the code?
  - Bug reports share terms with a large number of classes in a software system
  - The number of shared terms is higher for patched classes than for the rest of classes
- What is the code location of the shared terms between bug reports and patched classes?
  - Class names (most likely)
- Is the number of shared terms between bug reports and classes an adequate measure for bug location?
  - Well, it works better than complex TR techniques as LSI but not as well as Lucene

The question...

How can TR-based concept location techniques benefit from these findings?