Where is Bug Resolution Knowledge Stored?

Gerardo Canfora
Research Centre on Software Technology
Department of Engineering - University of Sannio
Viale Traiano - 82100 Benevento, Italy
canfora@unisannio.it

Luigi Cerulo
Research Centre on Software Technology
Department of Engineering - University of Sannio
Viale Traiano - 82100 Benevento, Italy
lcerulo@unisannio.it

ABSTRACT

ArgoUML uses both CVS and Bugzilla to keep track of bug-fixing activities since 1998. A common practice is to reference source code changes resolving a bug stored in Bugzilla by inserting the id number of the bug in the CVS commit notes. This relationship reveals useful to predict code entities impacted by a new bug report.

In this paper we analyze ArgoUML software repositories with a tool, we have implemented, showing what are Bugzilla fields that better predict such impact relationship, that is where knowledge about bug resolution is stored.

Categories and Subject Descriptors

H.3.1 [**Information storage and retrieval**]: Content Analysis and Indexing; D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement

General Terms

Measurement, Experimentation

Keywords

Mining Software Repositories, Impact Analysis

1. OVERVIEW

In [1] we introduced a method to predict the set of source files impacted by a new bug description submitted to a bugzilla repository. It takes advantage of the impact relationship extracted from CVS commit notes as suggested in [3]. In a set of four case studies, we obtained a top 1 precision ranging between 20% and 78%, and a top 30 recall ranging between 67% and 98%. The method builds, for each source file, a descriptor consisting of free text extracted from the set of fixed bugs and CVS commit notes that previously impacted it. An information retrieval algorithm scores each source file by measuring the similarity between its descriptors and the new bug description. The hypothesis is that the textual data carried by the bug tracking system during the bug fixing activity is a good descriptor of the impacted files to be considered in the impact analysis of future similar bugs. The similarity between descriptors is computed by using a probabilistic model that assumes that each term is associated with a topic, and that a document may be about the topic, or not [5]. The score of a source file descriptor d with respect to a bug is measured by using the following statistic measure about the term occurrences in source file descriptors:

$$S(d, bug) = \sum_{t \in bug} W_d\,(t)$$

where W is a weighting function directly proportional to the term frequency in the source file descriptor, and inversely proportional to the inverse document frequency [5].

A source file descriptor is represented with any combination of the following software repository fields: *notes*, the set of CVS commit notes; *short-descr*, the short bug description; *long-descr*, the long bug description; *comments*, the set of comments submitted by developers during bug resolution.

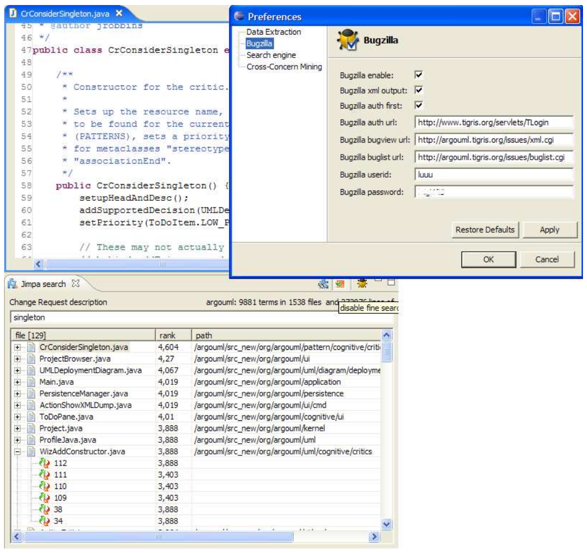
The model has been implemented in a tool, named Jimpa [2], that allows user to write a short explanation of a change and return the set of source files, ranked by their relevance with bug change description. The tool provides the support for setting information retrieval properties such as stop word list, stemmer algorithm, and software repositories fields to be included or excluded from the indexing process. Figure 1 shows a snapshot of the tool.

Data, provided by the tool, is stored in an intermediate database; we have used this database to perform the analysis presented in the following section that shows what are source file descriptor fields that contain more information about bug resolution.

2. MINING RESULTS

ArgoUML is an open-source UML modeling tool implemented in Java. Development started in 1998. The first bugzilla bug has been submitted in January 2000. Currently there are 2018 fixed bugs, 670 of which (about 33%) are referenced in CVS commit notes. The total number of Java source files is 1538. 6% of these files have a reference to more than 10 different bugs, while 40% of files do not have any bug reference.

We have performed a change impact prediction with the method introduced in [1] and with source file descriptors composed in different ways. In particular, we have used, as



```
 * @author jrobbins
 */
public class CrConsiderSingleton e

    /**
     * Constructor for the critic.
     *
     * Sets up the resource name,
     * to be found for the current
     * (PATTERNS), sets a priority
     * for metaclasses "stereotype
     * "associationEnd".
     */
    public CrConsiderSingleton() {
        setupHeadAndDesc();
        addSupportedDecision(UMLDe
        setPriority(ToDoItem.LOW_P

        // These may not actually
```

Bugzilla

Bugzilla enable:	✓
Bugzilla xml output:	✓
Bugzilla auth first:	✓
Bugzilla auth url:	http://www.tigris.org/servlets/TLogin
Bugzilla bugview url:	http://argouml.tigris.org/issues/xml.cgi
Bugzilla buglist url:	http://argouml.tigris.org/issues/buglist.cgi
Bugzilla userid:	luuu
Bugzilla password:	

Change Request description   argouml: 9881 terms in 1538 files and   disable fine searc

singleton

file [129]	rank	path
CrConsiderSingleton.java	4,604	/argouml/src_new/org/argouml/pattern/cognitive/criti
ProjectBrowser.java	4,27	/argouml/src_new/org/argouml/ui
UMLDeploymentDiagram.java	4,067	/argouml/src_new/org/argouml/uml/diagram/deployme
Main.java	4,019	/argouml/src_new/org/argouml/application
PersistenceManager.java	4,019	/argouml/src_new/org/argouml/persistence
ActionShowXMLDump.java	4,019	/argouml/src_new/org/argouml/ui/cmd
ToDoPane.java	4,01	/argouml/src_new/org/argouml/cognitive/ui
Project.java	3,888	/argouml/src_new/org/argouml/kernel
ProfileJava.java	3,888	/argouml/src_new/org/argouml/uml
WizAddConstructor.java	3,888	/argouml/src_new/org/argouml/uml/cognitive/critics
112	3,888	
111	3,403	
110	3,403	
109	3,403	
38	3,888	
34	3,888	

Figure 1: Tool snapshot

source file descriptor, each combination of bugzilla fields in order to put in evidence what is the field whose presence should give a better prediction performance.

Table 1 shows the top 1 precision and top 100 recall. The first is the percentage of cases in which the first retrieved source file is correct, while the second is the percentage of correct source files covered by the first 100 retrieved source files. Results show that the presence, in source file descriptor, of bug resolution comments give always the best performance. In particular, the overall best performance is obtained with a descriptor composed only with bug comments and CVS notes. This leads to consider that short and long descriptions submitted when the bug is discovered contain a partial knowledge about bug resolution, while most of the knowledge is contained in the comments submitted during the bug resolution process. On average, the presence of bug comments information gives an improvement of precision of about 5%.

Precision and recall have been computed using the leave-one-out assessment technique [4, 6] performed over 670 bugs, with short descriptions used as queries. For a given bugzilla bug we have predicted the set of impacted files by using an index without data regarding that bug. The predicted set of files has been then compared with the oracle set, that is the files impacted by that bug, recovered by considering the presence of the Buzilla id number in the revision comments of the files [3].

No evidence has been found for the dependence of prediction performance with other information retrieval parameters, such as, general English stop word list and Porter stemmer algorithm.

3. CONCLUDING REMARKS

Text mining of software repositories integrates information provided by source code analysis and gives new opportunities to support the software development process and to know new aspects about software evolution. It can be used not only for impact analysis but also, for example, to aggregate source files in a topic clusters.

Table 1: Performance dependencies

top 1	top 100	CVS	bug	bug	bug
precision	recall	notes	short decr	long decr	comments
0.232	0.791	×			×
0.212	0.802	×	×		×
0.191	0.795	×	×	×	×
0.163	0.792	×		×	×
0.161	0.790	×		×	
0.145	0.788	×	×	×	
0.126	0.754	×	×		
0.119	0.701	×			

Quality of text and project maturity are two factors that strongly impact every approach that takes advantage on free text stored in software repositories. Sometime CVS comments are used for communication rather that for description purpose and in almost all projects there is an initial period of transition that generates noise in both CVS and Bugzilla repositories. This leads to consider that results and issues obtained by applying data mining algorithms on software repositories can suggest new directions in developing more innovative configuration management and software development tools.

The open source community uses other repository for knowledge sharing, such as: mailing lists, newsgroups, and IRC conversations. They are rich of free text and it should be interesting to investigate how this information can be used in conjunction or as an alternative to CVS and Bugzilla.

4. REFERENCES

[1] G. Canfora and L. Cerulo. Impact analysis by mining software and change request repositories. In *METRICS '05: In Proceedings of the 11th IEEE International Software Metrics Symposium*, Como, Italy, 2005. IEEE Computer Society.

[2] G. Canfora and L. Cerulo. Jimpa: An eclipse plug-in for impact analysis. In *CSMR '06: In Proceedings of the 10th European Conference on Software Maintenance and Reengineering: Tools Demonstration*, Bari, Italy, 2006. IEEE Computer Society.

[3] M. Fischer, M. Pinzger, and H. Gall. Populating a release history database from version control and bug tracking systems. In *ICSM '03: In Proceedings of the 19th International Conference on Software Maintenance*, Amsterdam, Netherlands, 2003. IEEE Computer Society.

[4] K. Fogel and M. Bar. *Cross-Validatory Choice and Assessment of Statistical Predictions (with Discussion)*, volume 36. J. the Royal Statistical Soc., 1974.

[5] K. S. Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: development and comparative experiments. *Inf. Process. Manage.*, 36(6):779–808, 2000.

[6] B. Ribeiro-neto and Baeza-yates. *Modern Information Retrieval*. Addison Wesley, 1999.