Trends in Engineering and Technology (NCTET-2K17)
International Journal of Advanced Engineering Research and Science (IJAERS) *Special Issue-3*
https://dx.doi.org/10.22161/ijaers/nctet.2017.24 *ISSN: 2349-6495(P) | 2456-1908(O)*

Choosing Historical Fixes using Association Rules for fixing the bugs

T.MAMATHA
Assistant Professor
Dept of CSE
SNIST, Ghatkesar
Hyderabad,INDIA
mamathat7@gmail.com

Dr.B. RAMA SUBBA REDDY
Professor & HOD
Dept of CSE,
SV College of Engineering
Tirupati, India,
Email:rsreddyphd@gmail.com

A BALARAM
Associate Professor
Dept of CSE
CMRIT, Medchal
Hyderabad,INDIA
balaram.balaram@gmail.com

ABSTRACT: *In automation debugging fixing of the bug plays a vital role which takes much time to fix the bug depending of the severity of the bug. In this context many approaches have been proposed to automatically detect and patch the software code, the strategies are limited to a set of identified bugs that were thoroughly studied to define their properties, so to reduce the cost and to save the time. We propose a new approach, in this approach when we are developing a new project which have 70% code is similar to the existing project then using the fixes of previously generated bug . Once a bug is reported, it is automatically compared to all previous patches using Association rules techniques. Based on this comparison, we recommend a fix actions, identified from past fix examples, that may be suitable as hints for software developers to address the new bug.*

Key words: Association Rules, Bug, Fixes.

1.INTRODUCTION

100% bug-free software is impossible to produce as we have to accept some percentage of bugs and release to the customer [1]. When the consequences of the software is catastrophic then it leads to huge financial losses for businesses. Thus, fixing bugs is an important task in software development for continuously improving the software quality. Fixing bugs is however time-consuming and cost-sensitive. Thus constraints on resources, including time, manpower and testing environment, often lead project developers to release software that still contain many bugs [2].
The contributions of this paper are as follows:
In this paper we introduce a new approach which mainly save the time and cost for fixing the bugs which improving the fix rate of bugs through automatic bug technique. This approach leverages historical information extracted from software development artifacts including bug reports and bug fix links, to produce candidate fix actions for newly reported bugs. We rely on Association Rules technique. Association rules helps to choose the correct patches for the bugs within very less time where the developers can benefits a lot in fixing the bugs with less span of time.

2.RELATED WORK

Research on recommender systems for software repair (i.e.,bug fixing) is part of a larger research agenda on software bugs. Currently, research is being done to find new bugs either statically (e.g. with abstract interpretation) or dynamically (e.g. with fuzz testing). There is also research to support the process of handling software bugs: how to automatically prioritize bug reports following criticality criteria? How to automatically assign bugs to competent developers? What are the code commits that are related/linked to a given bug report? Then, there is work on enhanced debugging which consist of approaches and tools to help localizing buggy pieces of code (fault localization), the causal dependencies (e.g. with program dependency graphs).

Other works includes approaches implemented within AutoFix-E [6], have proposed to leverage contracts inside programs to automatically patch program source code, For Spreadsheet applications Abraham and Erwing[8] proposed an idea where if an user is given expected value than for an error prone formula it suggest to change the value in the cell to get expected value.. Than the tools generate fixes which changes the failing runs patterns to passing runs. Sometimes tool may delete or insert the existing transactions.

Our earlier work is for college placement process, For this we used Cause effect graph and decision table, which helps college students and placement people the procedure of placement and status. Again our work extend to finding the Root Cause[3] with the help of Cause effect Graph and fixed the fixes with the help of fix schemas. Previous paper is on Ranking of fixes for choosing the best fix and the

https://dx.doi.org/10.22161/ijaers/nctet.2017.24 *ISSN: 2349-6495(P) | 2456-1908(O)*

current paper helps the developers for choosing the appropriate fix in less time with low cost.

3. HISTORICAL FIXES

After knowing the root cause then finding the fix is very crucial part in automated debugging, for this we proposed a approach of using historical fixes . If the bug is already caught in previous projects and used the suitable fix for that bug. If same bug is repeated in the future project so instead of again going of finding the fix for a bug just choose the appropriate patch for that bug using association rule. Associate rules help in choosing the accurate fix for the buggy code. The procedure and sample example is taken shown how the association rules been used.

3.1. Procedure:

when developers completes their coding and given the module to the testing, the testing team will execute all the test cases and found any bug then they report the same to developers so the developers checks the piece of code and try to find the root cause for the given bug, .

For the detected bug, to choose the correct root cause we used the cause effect graph [3][4] after finding the root cause we choose the appropriate patches by using association rules as follows

3.2. Association rules:

Here in our paper we used association rules to relate our bugs to patches which help in finding the suitable fix.

Association Rules are used to describe associations or correlations among a set of BUGID's and suitable patches

- applications:
  - basket data analysis, cross-marketing, catalog design, clustering, data preprocessing, genomics, etc.

- rule format: LHS ⇒ RHS [support, confidence]

Body ==> *Consequent* [*Support* , *Confidence*]

- *Body*: represents the examined data.
- *Consequent*: represents a discovered property for the examined data.
- *Support*: represents the percentage of the records satisfying the *body* or the *consequent*.
- *Confidence*: represents the percentage of the records satisfying both the *body* and the *consequent* to those satisfying only the *body*.

Data should be provided in transactional form

- each record consists of BUGID and information about all Patches.

BUGID	FIXES
BUGID	FIX1,FIX2,FIX3
...	.....

Table-1: Format

Using historical patches for future projects which have similar bugs, for this bugs we already had patches which have been fixed and validated so using the same patches for future projects will give you accurate results in fixing the bug. To do so we used some simple bugs and patches to associate each other in fixing the bugs. A *patch* is a piece of software designed to update a computer program or its supporting data, to fix or improve it. patches usually called bug fixes or bug fixes, and improving the usability or performance. Although meant to fix problems, poorly designed patches can sometimes introduce new problems. In some special cases updates may knowingly break the functionality, for instance, by removing components for which the update provider is no longer licensed or disabling a device. In the below example, we shown how we are used association rule principles in fixing the patches. The first Table-1 describes the format we used to relate bugid's with patches, Second Table-2 which describes the buggy code with BUGID and Bug type.. The following tables we used to show association between bugs and bugfixes.

Example :

Table-2:Buggy Table

https://dx.doi.org/10.22161/ijaers/nctet.2017.24 *ISSN: 2349-6495(P) | 2456-1908(O)*

BUGID	Type of BUGNAME	Buggy code

BUG001	**Recursion error**	// Insert a value into an ordered linked list
		void insert(lnode*& curr, int val) {
		if (curr == NULL)
		curr = new lnode(val, NULL);
		else if (lnode->val > val)
		curr = new lnode(val, curr->next);
		else {
		curr = curr->next;
		insert(curr, val);
		}
		}

BUG002	**Dyslexic**	int minval(int *A, int n) {
		int currmin = MAXINT;
		
		for (int i=0; i<n; i++)
		if (A[i**]** > **currmin)**
		currmin = A[i];
		return currmin;
		}

BUG003	**Improper initialization**	int minval(int *A, int n) {
		int ***currmin;***
		
		for (int i=0; i<n; i++)
		if (A[i] < currmin)
		currmin = A[i];
		return currmin;
		}

BUG004	**Parameter mismatch**	char string1[10] = ***"Hello";*** char string2[10]; strcpy(string1, string2);

Bug005	**Model error**	1. // Return pointer to the node storing "val" if any;
		NULL otherwise
		void find(listnode **curr, val) {
		while (*curr != NULL)
		if ***(*curr-**>val == val) return;
		else
		*curr = ****curr-**>next;
		}

BUG006	**Off-by-one error**	int i; int array[5]; int j; for (i=0**; *i<=5***; i++) cin >> array[i];

Table-3: Fixing patches for Buggy code

https://dx.doi.org/10.22161/ijaers/nctet.2017.24 *ISSN: 2349-6495(P) | 2456-1908(O)*

BUGID	FIXES

BUG001	// Insert a value into an ordered linked list
	void insert(lnode*& curr, int val) {
	  if (curr == NULL)
	  curr = new lnode(val, NULL);
	  else if (lnode->val > val)
	  curr = new lnode(val, curr->next);
	  else {
	  insert(curr->next, val);
	  insert(curr, val);
	  }
	}

BUG002	int minval(int *A, int n) {
	  int currmin = MAXINT;
	
	  for (int i=0; i<n; i++)
	  if (A[i] > currmin)
	  currmin = A[i];
	  return currmin;
	}

BUG003	int minval(int *A, int n) {
	  int **currmin=0**;
	
	  for (int i=0; i<n; i++)
	  if (A[i] < currmin)
	  currmin = A[i];
	  return currmin;
	}

BUG004	char string1[10**;** char string2[10] = **"Hello"**; strcpy(string1, string2);

Bug005	// Return pointer to the node storing "val" if any; NULL otherwise
	void find(listnode **curr, val) {
	  while (*curr != NULL)
	  if **(*curr)**->val == val) return;
	  else
	  *curr = **(*curr)->**next;
	}

BUG006	int i; int array[5]; int j; for (i=0***; i<=4***; i++) cin >> array[i];

https://dx.doi.org/10.22161/ijaers/nctet.2017.24 *ISSN: 2349-6495(P) | 2456-1908(O)*

4. Conclusion:

Our paper mostly helps developers in fixing the bugs in less time with low cost because the fixes we are used from the historical fixes, these fixes are framed in the form of association rules. The Table-2 and Table-3 shows how we relate the bugid with patches.

REFERENCES

[1] E. Joyce, "Is error-free software possible?" Datamation, February 1989.

[2] C. Weiss, R. Premraj, T. Zimmermann, and A. Zeller, "How long will it take to fix this bug?" in Proceedings of the Fourth International Workshop on Mining Software Repositories, May 2007.

[3] Ms.T.Mamatha, A.Balaram, Dr.D S R Murthy et al (2015), Automated Detection of Root Cause and Fixing of Programs, International Journal of Electrical and Computing Engineering, Vol.1, Issue.4, Journal Proceedings.

[4]DhanammaJagli, T.Mamatha T,Swetha Mahalingam,Namrata Ojha, International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.6, November 2012

[5] Mrs T.Mamatha, Dr. M Chandra Mohan, A BalaRam, 2017 IEEE 7th International Advance Computing Conference, paper titled Ranking of FIXes for Choosing the Best Fix

[6] Y. Wei, Y. Pei, C. A. Furia, L. S. Silva, S. Buchholz, B. Meyer, and A. Zeller "Automated fixing of programs with contracts," in ISSTA, 2010.

[7] A. Bachmann and A. Bernstein. When process data quality a_ects the number of bugs: Correlations in software engineering datasets. In MSR'10, pages 62{71, Cape Town, South Africa, May 2010. IEEE Computer Society.

[8] R. Abraham and M. Erwig, "Goal- directed debugging of spreadsheets", IEEE Symposium on Visual Languages and Human-Centri Computing,pp. 37–44, 2005.

Authors:

Mrs.T.Mamatha is an Assistant Professor of CSE Department in SreeNidhi Institute of Science & Technology since from 2011. She Received her M.Tech from JNTU-Anantapur in Software Engineering. Her Graduation is from JNTU-Hyderabad in Computer Science and Engineering. She has more than 9 yrs of teaching experience. She has published 6 research papers in International Journals and 3 international Conferences. Her Area of Interest is Software Engineering & Software Testing. she is a member of IAENG(International Association of Engineers)

Mr.Dr. B Rama Subba Reddy is a Professor & HOD in SV College of Engineering. He received his Ph.D from S V University, Tirupati. He has more than 19yrs of Teaching experience. He has published 14 research papers in International Journals. His Area of Interest is Data Mining.

Mr. A.BalaRam is an Associate professor of CSE Department in CMR Institute of Technology, Hyderabad. He has more than 10 years of teaching experience. He Received his M.Tech from JNTU- Anantapur in Software Engineering. His Graduation is from JNTU-Hyderabad in Computer Science and Engineering. He has published 12 research papers in National and International Journals and Conferences. His area of interests Software Engineering, Image Processing, Net work Security and Cryptography. He is a member of IAENG(International Association of Engineers).