IJIEMR Transactions, online available on 4th Sept 2020. Link :http://www.ijiemr.org/downloads.php?vol=Volume-09&issue=ISSUE-09

Title: **REDUNDANT BUG REPORT DETECTION OF USER INTERFACE BUGS UTILIZING DECISION TREE INDUCTION AND INVERTED INDEX STRUCTURE**

Volume 09, Issue 09, Pages: 101-107

Paper Authors

SHEIK SAIDHBI, ASEMRIE YEMATA

REDUNDANT BUG REPORT DETECTION OF USER INTERFACE BUGS UTILIZING DECISION TREE INDUCTION AND INVERTED INDEX STRUCTURE

[1]SHEIK SAIDHBI, [2]ASEMRIE YEMATA
[1]Associate Professor, Department of Information Systems, Faculty of Informatics, University of Gondar, Ethiopia,
[2]Head of the Department, Department of Information Systems, Faculty of Informatics, University of Gondar, Ethiopia.

Abstract:
In different programming framework situations, Bug following framework assumes a fundamental job in recognizing bugs. An engineer as a rule needs to replicate the bug report for similitude cause, a procedure that is repetitive and tedious. To address this issue a few methodologies exist yet there is a lot of degree to improve precision of copy bug identification process. This paper depicts a strategy for copy bug report discovery of UI bugs by utilizing the idea of choice tree enlistment and modified record structure. UI bug is fundamentally shown up from terrible CSS code or from programming code causes ruin for website specialists everywhere throughout the world.
Keywords: Duplicate bug, Decision tree, Inverted index

Introduction

For programming venture engineer who is chipping away at venture a bug archive is going to make an imperative job and furthermore rich asset supplier. By utilizing that archive designer can comprehend in a productive way, he ready to recognize what transforms he have to continue further. Bug announcing anyway an ungraceful appropriated process is utilized to triage, track, and comments.[4] In the bug detailing framework clients and analyzers may report similar imperfections commonly. In the event that this is the issue in this situation it causes an issue as various designers ought not be doled out a similar bug to fathom it make extremely run of the mill. Just some senior engineer can locate the specific bug by perusing title of the bug however the designer is going to peruse the report absolutely, with parcel of conversation among them and with different colleagues and different seniors. Each client needs to speak with an application. The best way to impart is through Graphical User Interface. GUI permits clients to connect with programming either by contributing content or by mouse development. Of the all out code composed 50-60% of the code is committed to Graphical User Interface [1].As it is a UI, clients with various degrees of expertise, information and jargon report bugs. Here there is a significant possibility for duplication on account of the reports composed by the clients is for the most part in normal language. These reports must be converted into specialized phrasing by designers. A significant number of the UI bug reports are copies of existing bug reports. Perceiving copy reports would assist with fixing bugs quicker and diminish time.

Browser(s)	Summary of bug
11	Hovered element still remains inhoverstate after scrolling away.
Firefox	Table-borderedwith an empty<tbody> is missin g borders.
Chrome	Focus ring of image map within a modal is displayed in the wrong location.
Chrome	Incorrect viewport size used for media queries when printing.
Firefox	Button elements with width: 100% become cropped inlong tables.
Internet Explore r 11	CSS border-radius sometimes causes lines of bleed-through of the background-colorof the parent element.

Table source: http://getbootstrap.com/browser-bugs/

Up to 60% of advancement is squandered simply attempting to crush out Internet Explorer explicit bugs which aren't generally a powerful utilization of your time. A bug report must have a title plainly indicating nature of bug and the activity performed must be obviously described. The objective of a bug report is to make it basic and ready to fix the bugs rapidly. Bug reports are made with the expectation that others with the comparable disadvantage will have the option to work together with you on comprehending it with an assurance. Avoiding bugs isn't new: it's as of now done ordinary by clients. When a bug is recognized the client can apparently attempt to maintain a strategic distance from it inside what's to come. Anyway a standard methodology has run of the mill issues. From the start, this methodology of staying away from the bug needs fundamental intellectual procedure that is troublesome if application has parcel of bugs. Be that as it may, it is a significant weight on the memory, in light of the fact that the client needed to recall for each new discharge. In the event that a bug is mounted, the client would furthermore wish to honor that comparably to exploit a previously broken element. The client needs to recognize all the bugs of each application in the event that he/she is survivor of a few applications. Second, the manual way to deal with bug restriction doesn't allow it straightforward for clients to gain from interchange clients.

For example, it would be better if a client may keep away from a bug while not in any event, experiencing it once. This may be done if a client discovers the bug in earlier from different clients. Anyway this only here and there occurs. It isn't completely worthy to anticipate that a client should look over and remember each one of those a large number of bugs in a bug following framework all together that he/she may skip them. Third, the manual way to deal with bug shirking needs the client to look out the conditions under which a bug occurs. Anyway recognizing the bug introduction conditions physically is kind of intense. Additionally, for notifably troublesome bug introduction conditions, there is part to acknowledge by pooling along execution setting from a few clients to work out the settings inside which a bug occurs. By arranging, distinguishing copy bug reports is normally done. Physically triaging takes an economical amount of your time and furthermore the outcomes are probably not going to be finished once the amount of day by day revealed bugs for recognized programming is taken into thought,. In Eclipse, for example, every day two man hours skipped altogether on bug arranging. Via computerizing bug-report duplication an assortment of tests are led to manage this issue. Essentially focusing on the literary highlights of the bug reports, and using of common language process (NLP) procedures to attempt to do printed examination, various bug-report similitude precessions are anticipated, Number of these investigations also utilize clear cut alternative extricated from the fields of bug reports (for example part, form, need, and so forth.).

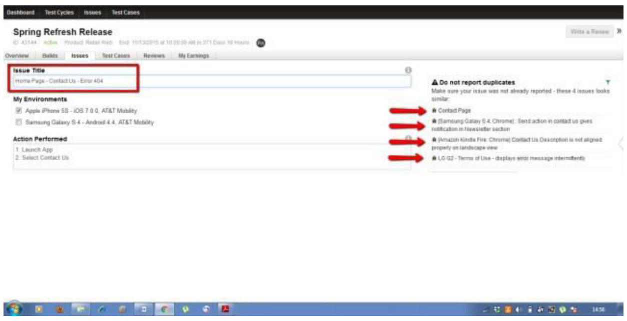
Bug announcing procedure ought to follow these means:

1) Report the bug. While detailing the bug we should make reference to the accompanying in bug report:

- Title of the issue,
- Last Used Environment,
- Action Performed,
- expected result,
- actual result and
- frequency

2) After getting the report analyzers must group the bugs as GUI bugs, Functional Bugs and Technical Bugs.
3) IDENTIFY BUG seriousness as hig,medium,low, Testers must choose to which part issue has a place.
4) Bug survey process must be refreshed to client as Approved, data, mentioned, dismissed, questioned, under audit, finished.
5) Identifying the copy bugs preceding announcing

Each test cycle comprises of a bugs list which was given by all analyzers. To see the rundown of bugs of a specific test cycle, click on the test cycle and afterward on the Issues Tab. On the other hand, when you begin to report a bug, the stage will consequently look through the test cycle's Issue reports for the watchwords and prompts the client for survey when the rundown of copies are found.



Spring Refresh Release

Issue Title: Home Page - Contact Us - Error 404

My Environments
- [✓] Apple iPhone 5S - iOS 7.0.0, AT&T Mobility
- [ ] Samsung Galaxy S 4 - Android 4.4, AT&T Mobility

Action Performed
1. Launch App
2. Select Contact Us

Do not report duplicates
Make sure your issue was not already reported - these 4 issues looks similar:
- Contact Page
- [Samsung Galaxy S 4, Chrome] : Send action in contact us gives notification in Newsletter section
- [Amazon Kindle Fire, Chrome] Contact Us Description is not aligned properly on landscape view
- LG G2 - Terms of Use - displays error message intermittently

Notwithstanding recently detailed bugs, 'Realized bugs' rundown additionally will be available in the test cycles as a connection in the Scope and Instructions tab. If you don't mind check these two records before revealing another bug.

The essential commitments of this paper are as per the following:

1.Decision trees are utilized to order the bugs dependent on certain qualities. We have picked choice trees since it is anything but difficult to get ready information utilizing them.

2.Determining ascribes to discover closeness of bugs andclassifythem.We present point by point investigation of qualities and grouping.

3.We investigate how to assemble an altered record structure and how to recognize copy bugs from file structure

We present the consequences of three run of the mill errands, i.e., improving bug triage by order and foreseeing copy bugs by utilizing decided properties. As far as anyone is concerned, this is the main work to assess the outcomes utilizing choice trees.

II. Related Work

To lessen manual exertion extensive research has been done in the region of discovering programmed copy bug reports.P.Runenson[2] proposed a way to deal with recognize copy bugs utilizing regular language handling methods which centers around tokenization, stemming and stop words with traits, for example, part type and priority.But this methodology is utilizing an off-rack record similitude measure which isn't so successful Sun.et.al[3] utilized Rep recovery capacity to quantify closeness between bug reports.

He expected that copy reports are comparative literarily as well as in different fields, for example, priority,component,severityetc.Propsed another methodology that is an externsion to BM25F a data recovery calculation by joining both literary and unmitigated fields. Alipour et.al.[4] proposed a methodology dependent on non-utilitarian necessities by making wordlists and themes through Latent Dirichlet Allocation(LDA) .To quantify similitude he utilized BM25F,KNN and SVM calculations. Sureka and Jalote proposed a methodology that utilizes N-gram based Information Retrieval Technique to manufacture the capabilities of bug reports[5]. They have indicated that their methodology can recognize bug reports with .more exactness yet they haven't contrasted their methodology and different strategies. Lazar[6] proposed a way to deal with recognize copy bug reports utilizing printed similitude estimates Take Lab is a lot of two frameworks that mechanizes estimating of semantic comparability of short messages utilizing directed AI. After all the highlights are determined a few twofold arrangement techniques including Naive Bayes and Support Vector Machines are rushed to characterize bugs as copy or no copy. They announced an improvement of 3.35 to 6.32% accuracy.Klien [7], expanded past work of alipour [4] presenting a scope of measurements dependent on the subject circulation of the issue reports, depending just on information taken legitimately from bug reports. Specifically, we present a novel metric that quantifies the principal shared subject between two theme archive conveyances. Karan Aggarwal [8] propose a technique to in part robotize the extraction of relevant word records from programming building writing. Assessing this product writing setting technique on true bug reports produces valuable outcomes that demonstrate this semi-mechanized strategy can possibly considerably diminish the manual exertion utilized in logical bug reduplication while enduring just a minor misfortune in precision.

III. Proposed Approach

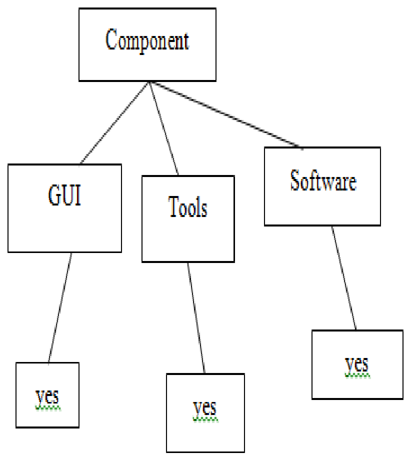
A choice tree is a flowchart-like tree structure, where each inside hub indicates a test on a property, each branch speaks to a result of the test, and each leaf hub holds a class name. The top most hub in a tree is the root hub. [9]

BUGID	2285	1451
Component	GUI	Tools
Priority	Medium	Critical
Type	Defect	Defect
Version	1.5	4
Status	Released	Duplicate
Merge Id		14156

Table: Example Bug Report Information

As we are concentrating on UI bugs in our methodology we intend to develop choice trees for field of segment and order UI bugs among the parts. We have picked choice tree for grouping since it is anything but difficult to enter information into choice trees.

Choice Tree orders the information dependent on preparing set and the qualities in a grouping trait and utilizations it in arranging information. Here we gave preparing set as bug report and the class is segment.



For the most part UI bugs are caused as a result of misalignment, broken pictures, shading, incorrectly measured realistic components and so forth. In the wake of arranging GUI bugs among parts. We wish to order UI bugs relying upon their inclination for example misalignment, shading and so forth.

A few instances of UI bugs are

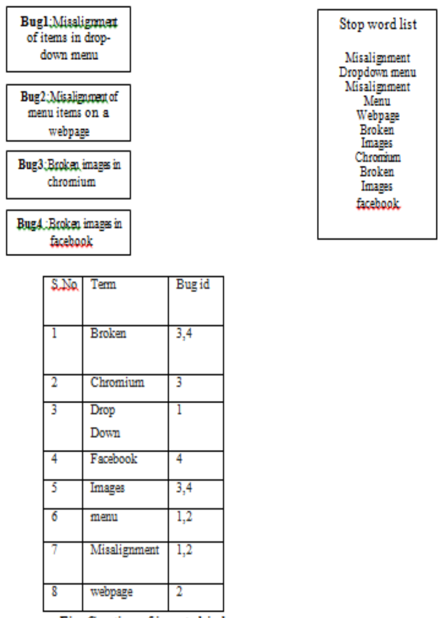
1) Misalignment of things in drop-down menus.

2) Form label causing misalignment in footer

3) Misalignment in music application

All bugs which have the word misalignment are ordered under one group.

In the wake of grouping bugs so as to locate the copy bugs we have utilized a rearranged ordering structure. Ordering structure is utilized on the grounds that ordering encourages quickest recovery Inverted file.



Bug1:Misalignmet of items in drop-down menu

Bug2:Misalignmet of menu items on a webpage

Bug3:Broken images in chromium

Bug4 :Broken images in facebook

Stop word list

Misalignment
Dropdown menu
Misalignment
Menu
Webpage
Broken
Images
Chromium
Broken
Images
facebook

S.No	Term	Bug id
1	Broken	3,4
2	Chromium	3
3	Drop Down	1
4	Facebook	4
5	Images	3,4
6	menu	1,2
7	Misalignment	1,2
8	webpage	2

Fig: Creation of inverted index

From the above ordering structure we discover it is anything but difficult to recover copy bugs .Based on the id field we intend to distinguish copy bugs .For instance the ids 3& 4 rehashes multiple times so we sort them as copy bugs.the id 1 shows up just a single time so it isn't delegated copy.

IV. Conclusion and FrameWork

This paper proposes an improved strategy for distinguishing copy bugs by utilizing choice trees to characterize the bugs and modified filed structure is utilized to gauge likeness and to recognize copy bugs.this paper centers around UI bugs since UI had more importance.decision trees are utilized in light of the fact that information section into the choice trees is simple. Altered file structures encourage quick and simple recovery of copy reports.In future we intend to assemble a trial arrangement for our proposed approach and contrast it with different methodologies with make it exact and financially efficient.

References

1) Atif M. Memon, Martha E. Pollack, and Mary Lou Soffa, "Hierarchical GUI Test Case Generation Using Automated Planning," IEEE Transactions on Software Engineering, vol. 27, no. 2, pp. 144-155, February 2001.

2) P. Runeson, M. Alexandersson, and O. Nyholm, “Detection of duplicate defect reports using natural language processing,” in Software Engineering,2007. ICSE 2007. 29th International Conference on. IEEE, 2007,pp. 499–510.

3) C. Sun, D. Lo, S.-C. Khoo, and J. Jiang, “Towards more accurate retrieval of duplicate bug reports,” in Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering. IEEE Computer Society, 2011, pp.253–262.

4) A. Alipour, A. Hindle, and E. Stroulia, “A contextual approach towards more accurate duplicate bug report detection,” in Proceedings of the Tenth International Workshop on Mining Software

Repositories. IEEE Press, 2013, pp.183–192.

5) A. Sureka and P. Jalote, “Detecting duplicate bug report using character n-gram-based features,” in Proceedings of the 2010 Asia Pacific Software Engineering Conference, 2010, pp.366–374.
6) Alina Lazar, Sarah Ritchey, Bonita Sharif “Improving the Accuracy of Duplicate Bug Report Detection using Textual Similarity Measures”In proceedings ofICSE2014
7) Nathan Klein Christopher S. Corley, Nicholas A. Kraft “New Features for Duplicate Bug Detection “ In proceedings of ICSE2014.
8) Karan Aggarwal, Tanner Rutgers, Finbarr Timbers, Abram Hindle, Russ Greiner, and EleniStroulia “Detecting Duplicate Bug Reports with Software Engineering Domain Knowledge This paper is a preprint accepted to be published at IEEE SANER’15,Montréal
9) https://www.drupal.org/node/2502715

http://getbootstrap.com/browser-bugs/