**SARATHI**: Characterization Study on Regression Bugs and Identification of Regression Bug Inducing Changes: A Case-Study on Google Chromium Project

*Manisha Khattar [manishak@iiitd.ac.in]*

ABB PhD Fellow, Software Analytics Research Lab (SARL)

www.software-analytics.in
M.Tech Thesis Evaluation Committee

**Thesis Adviser**
Prof. Ashish Sureka
Faculty Member at IIIT-Delhi and currently Visiting Researcher at Siemens Corporate Research and Technology
*Faculty In-charge, Software Analytics Research Lab (SARL)*

**External Examiner**
Dr. Atul Kumar
Senior Researcher at Siemens Corporate Research and Technology

**Internal Examiner**
Prof. Rahul Purandare
Faculty Member at IIIT-Delhi
Presentation Outline

1. Research Motivation and Aim
2. Related Work and Research Contributions
3. Experimental Dataset
4. Characterization Study
5. Proposed Approach
6. Experimental Evaluation and Results
7. Limitations and Conclusion
Presentation Outline

1. Research Motivation and Aim
2. Related Work and Research Contributions
3. Experimental Dataset
4. Characterization Study
5. Proposed Approach
6. Experimental Evaluation and Results
7. Limitations and Conclusion
Regression

“Regression” means return to a former or less developed state.
“Regression” means return to a former or less developed state.
Regression Bugs

Regression Bugs are defects which occur, when a previously working software feature or functionality stops behaving as intended.
How Regression Bugs creep in ??

99 little bugs in the code.  
99 little bugs in the code.  
Take one down, patch it around.  

127 little bugs in the code...
How Regression Bugs creep in??

99 little bugs in the code.
99 little bugs in the code.
Take one down, patch it around.

127 little bugs in the code...

Regression:
"when you fix one bug, you introduce several newer bugs."
Regression Testing - The Only Solution

- Enhancement/Fixes and Regression Testing should go hand in hand
But...

A nightmare!

Regression testing

New features testing
But...

Regression Testing in large and complex systems after every change is almost impractical.

*A nightmare!*

Regression testing

New features testing
But...

- Regression Testing in large and complex systems after every change is almost impractical.
- Regression bugs are considered to be inevitable and truism in large and complex software systems[1].
Need for Automation

- The regression hunting process can be extremely time consuming and error-prone.
Need for Automation

▶ The regression hunting process can be extremely time consuming and error-prone.

▶ Automating hunting of regression inducing change will help developers to spend their time fixing bugs rather than tracking down where they were introduced.
Research Aim

- To conduct an in-depth characterization study of regression bugs by mining issue tracking system dataset belonging to a large and complex software system i.e Google Chromium Project
Research Aim

- To conduct an in-depth characterization study of regression bugs by mining issue tracking system dataset belonging to a large and complex software system i.e Google Chromium Project

- To investigate bug report and source-code commit meta-data and content based mining solution, to develop a predictive model for identifying a regression bug inducing change. To validate the proposed model and demonstrate its effectiveness
Presentation Outline

1. Research Motivation and Aim
2. Related Work and Research Contributions
3. Experimental Dataset
4. Characterization Study
5. Proposed Approach
6. Experimental Evaluation and Results
7. Limitations and Conclusion
# Regression Bug Hunting and Location

<table>
<thead>
<tr>
<th>Study</th>
<th>Overview</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bowen 2006[1]</td>
<td>Present a tool PyRet to automate the process of identifying which code addition or patch created the regression</td>
</tr>
<tr>
<td>Johnson 2004[2]</td>
<td>Provide a solution for automated regression hunts for the Linux kernel based on patch sets rather than dates</td>
</tr>
<tr>
<td>Yorav 2008[3]</td>
<td>Present a tool called as CodePsychologist which assists a programmer to locate source code segments that caused a given regression bug</td>
</tr>
</tbody>
</table>
## Related Work and Research Contributions

### Regression Bug Prediction

<table>
<thead>
<tr>
<th>Study</th>
<th>Overview</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mockus 2000[4]</td>
<td>Develop a predictive model to predict the probability that a change to software will cause a failure</td>
</tr>
<tr>
<td>Shihab 2012[5]</td>
<td>Conduct an industrial study on the risk of software changes</td>
</tr>
<tr>
<td>Sunghun 2008[6]</td>
<td>Present an approach to classify a software change into clean or buggy</td>
</tr>
<tr>
<td>Tarvo 2008[7]</td>
<td>Propose a statistical model for predicting software regression</td>
</tr>
<tr>
<td>Tarvo 2009[8]</td>
<td>Present a tool called as Binary Change Tracer (BCT) which collects data on software projects and helps predict regressions</td>
</tr>
</tbody>
</table>
## Characterization Study

<table>
<thead>
<tr>
<th>Researcher</th>
<th>Overview</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lal 2012[9]</td>
<td>Present a study of mining issue tracking system to compare and contrast seven different types of bug reports: crash, regression, security, clean up, polish, performance and usability</td>
</tr>
<tr>
<td>Gegick 2010[10]</td>
<td>Perform an industrial study on identification of security bug reports via text mining</td>
</tr>
<tr>
<td>Zaman 2011[11]</td>
<td>Conduct a case-study on Firefox project and study two different types of bugs: performance and security</td>
</tr>
</tbody>
</table>
Presentation Outline

1. Research Motivation and Aim
2. Related Work and Research Contributions
3. Experimental Dataset
4. Characterization Study
5. Proposed Approach
6. Experimental Evaluation and Results
7. Limitations and Conclusion
### Issue Tracking system

<table>
<thead>
<tr>
<th>S.No</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Start Issue ID</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>End Issue ID</td>
<td>388954</td>
</tr>
<tr>
<td>3</td>
<td>Reporting Date Start Issue</td>
<td>8/30/2008 4:00:21 PM</td>
</tr>
<tr>
<td>4</td>
<td>Reporting Date End Issue</td>
<td>6/26/2014 12:16:25 AM</td>
</tr>
<tr>
<td>5</td>
<td>Number of Issue Reports Downloaded</td>
<td>295202</td>
</tr>
<tr>
<td>6</td>
<td>Number of Issue Reports unable to Download</td>
<td>93571</td>
</tr>
<tr>
<td>7</td>
<td>Number of Labelled Bug Reports</td>
<td>39658</td>
</tr>
</tbody>
</table>
## Version Control System

<table>
<thead>
<tr>
<th>S.No</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Start Revision ID</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>End Revision ID</td>
<td>279885</td>
</tr>
<tr>
<td>3</td>
<td>Reporting Date Start Revision</td>
<td>7/25/2008 7:49:01 PM</td>
</tr>
<tr>
<td>4</td>
<td>Reporting Date End Revision</td>
<td>6/26/2014 1:27:51 AM</td>
</tr>
</tbody>
</table>
Presentation Outline

1. Research Motivation and Aim
2. Related Work and Research Contributions
3. Experimental Dataset
4. Characterization Study
5. Proposed Approach
6. Experimental Evaluation and Results
7. Limitations and Conclusion
Characterization Study

- To understand how regression bugs differentiate themselves from the other bug types
To understand how regression bugs differentiate themselves from the other bug types
Characterization Study

- To understand how regression bugs differentiate themselves from the other bug types

- 51.09% of the labelled bug reports are regression bugs
Characterization Study

To understand how regression bugs differentiate themselves from the other bug types

- 51.09% of the labelled bug reports are regression bugs
- 55% of the regression bugs are high priority (0 [2.8%] and 1 [51.7%]) bugs
Number of Comments

- Number of comments posted serves as a proxy for popularity, user interest and amount of clarification and discussion [12]
Number of Comments

Number of comments posted serves as a proxy for popularity, user interest and amount of clarification and discussion [12]
Number of Comments

- Number of comments posted serves as a proxy for popularity, user interest and amount of clarification and discussion [12]

- Median values for crash, performance, regression and security bugs are 12, 9, 13 and 20 resp.
Closure time

- Closure Time - Time taken to close an opened issue in the issue tracking system

- 50% of the crash and performance bugs are closed within 12 days

- 50% of the regression and security bugs are closed within 8 days
Closure time

- Closure Time - Time taken to close an opened issue in the issue tracking system

![Graph showing closure times for different types of issues: Crash, Performance, Regression, Security]
Closure time

- Closure Time - Time taken to close an opened issue in the issue tracking system

- 50% of the crash and performance bugs are closed within 12 days
Closure time

- Closure Time - Time taken to close an opened issue in the issue tracking system

- 50% of the crash and performance bugs are closed within 12 days

- 50% of the regression and security bugs are closed within 8 days
Number of Stars

- The number of stars on a bug report indicate the number of people who are interested in that issue.
The number of stars on a bug report indicate the number of people who are interested in that issue.

<table>
<thead>
<tr>
<th>Type</th>
<th>Min.</th>
<th>Q1</th>
<th>Median</th>
<th>Mean</th>
<th>Q3</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crash</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2.93</td>
<td>3</td>
<td>224</td>
</tr>
<tr>
<td>Perf.</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3.112</td>
<td>3</td>
<td>185</td>
</tr>
<tr>
<td>Regr.</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3.956</td>
<td>4</td>
<td>703</td>
</tr>
<tr>
<td>Security</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1.335</td>
<td>1</td>
<td>78</td>
</tr>
</tbody>
</table>
The number of stars on a bug report indicate the number of people who are interested in that issue.

<table>
<thead>
<tr>
<th>Type</th>
<th>Min.</th>
<th>Q1</th>
<th>Median</th>
<th>Mean</th>
<th>Q3</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crash</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2.93</td>
<td>3</td>
<td>224</td>
</tr>
<tr>
<td>Perf.</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3.112</td>
<td>3</td>
<td>185</td>
</tr>
<tr>
<td>Regr.</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3.956</td>
<td>4</td>
<td>703</td>
</tr>
<tr>
<td>Security</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1.335</td>
<td>1</td>
<td>78</td>
</tr>
</tbody>
</table>
The number of stars on a bug report indicate the number of people who are interested in that issue.

<table>
<thead>
<tr>
<th>Type</th>
<th>Min.</th>
<th>Q1</th>
<th>Median</th>
<th>Mean</th>
<th>Q3</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crash</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2.93</td>
<td>3</td>
<td>224</td>
</tr>
<tr>
<td>Perf.</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3.112</td>
<td>3</td>
<td>185</td>
</tr>
<tr>
<td>Regr.</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3.956</td>
<td>4</td>
<td>703</td>
</tr>
<tr>
<td>Security</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1.335</td>
<td>1</td>
<td>78</td>
</tr>
</tbody>
</table>

Regression bugs have the **highest** mean amongst all.
Bug Opening and Closing Trends

- Bug opening and closing trend are performance indicators reflecting the characteristics and quality of defect fixing process[13]
Bug Opening and Closing Trends

- Bug opening and closing trend are performance indicators reflecting the characteristics and quality of defect fixing process[13]

- **Bug Opening Trend** - Cumulated number of opened bugs over time
Bug Opening and Closing Trends

- Bug opening and closing trend are performance indicators reflecting the characteristics and quality of defect fixing process[13]

- **Bug Opening Trend** - Cumulated number of opened bugs over time

- **Bug Closing Trend** - Cumulated number of bugs that are resolved and closed over time
Metric for Quality of Bug Fixing Process

- Metric to compute the quality of bug fixing process for one type of bug report in comparison to the quality of bug fixing process for other types of bug reports

\[ BSR(T) = \frac{\Delta_{Secr} + \Delta_{Perf} + \Delta_{Crsh}}{\Delta_{Regr}} \]  

\( BSR(T) \) - Bubble Size for Regression Bugs at a Time \( T \)
\( \Delta_x \) at a time \( T \) represents the number of bugs of type “\( x \)” which are open at that instant of time.
Metric for Quality of Bug Fixing Process

- Metric to compute the quality of bug fixing process for one type of bug report in comparison to the quality of bug fixing process for other types of bug reports

\[
BSR(T) = \frac{\Delta_{Secr} + \Delta_{Perf} + \Delta_{Crsh}}{\Delta_{Regr}}
\]

\(BSR(T)\) - Bubble Size for Regression Bugs at a Time \(T\)

\(\Delta_x\) at a time \(T\) represents the number of bugs of type “\(x\)” which are open at that instant of time.

- More the value of \(BSR(T)\), the better is bug fixing quality of regression bug
Quality of Bug Fixing Process for Regression and Security

**Figure 1**: Line and bubble chart showing the regression bug opening and closing trend

**Figure 2**: Line and bubble chart showing the security bug opening and closing trend
Cumulative Bug Count at 2014 is almost double of that at mid 2012-2013
Decline in Quality of Bug fixing Process for Regression Bugs

- Cumulative Bug Count at 2014 is almost double of that at mid 2012-2013
- There is significant decline in Quality of bug fixing process at 2014.
Decline in Quality of Bug fixing Process for Regression Bugs

- Cumulative Bug Count at 2014 is almost double of that at mid 2012-2013
- There is significant decline in Quality of bug fixing process at 2014.
- As the complexity of system increases quality of bug fixing process declines
Presentation Outline

1. Research Motivation and Aim
2. Related Work and Research Contributions
3. Experimental Dataset
4. Characterization Study
5. Proposed Approach
6. Experimental Evaluation and Results
7. Limitations and Conclusion
Technical Challenges

- Human intensive task
- No record of previous bug inducing changes
- Level of detail provided in bug-report and log messages
Technical Challenges

- Human intensive task
- No record of previous bug inducing changes
- Level of detail provided in bug-report and log messages
Technical Challenges

- Human intensive task
- No record of previous bug inducing changes
- Level of detail provided in bug-report and log messages
Level of Detail

Detailed Description
Level of Detail

Issue 136336: Avatar overlaps with new tab button
1 person starred this issue and may be notified of changes.

Status: Fixed
Owner: rresek@chromium.org
Closed: Jul 2012
Cc: sail@chromium.org, rresek@chromium.org, avs@chromium.org

Reported by dub...@chromium.org, Jul 9, 2012

Chrome 22.0.1281.0 canary on Mac OS X 10.6.8.

When I have a lot of tabs open, the avatar icon partly obscures the new tab button -- see screenshot.
Ground Truth Dataset Establishment

- No previous record of regression inducing revision for fixed regression bugs
- Ground truth Dataset establishment was non trivial
- Only fixed and verified issue reports are considered
- It is a 3 step process
Ground Truth Dataset Establishment

- No previous record of regression inducing revision for fixed regression bugs
- Ground truth Dataset establishment was non trivial
- Only fixed and verified issue reports are considered
- It is a 3 step process
Ground Truth Dataset Establishment

- No previous record of regression inducing revision for fixed regression bugs
- Ground truth Dataset establishment was non trivial
- Only fixed and verified issue reports are considered
- It is a 3 step process
Ground Truth Dataset Establishment

- No previous record of regression inducing revision for fixed regression bugs
- Ground truth Dataset establishment was non trivial
- Only fixed and verified issue reports are considered
- It is a 3 step process
Step 1: Revision ID Extraction

- Mined issue reports and their comments to extract revision ids
- Reference to a revision id generally follows some fixed patterns
- E.g.- r12845, revision 128696, revision=181939, suspecting - 179554, rev. 12456
Step 1: Revision ID Extraction

- Mined issue reports and their comments to extract revision ids
- Reference to a revision id generally follows some fixed patterns
- E.g.- \texttt{r12845}, \texttt{revision 128696}, \texttt{revision=181939}, \texttt{suspecting - 179554}, \texttt{rev. 12456}
Step 1 : Revision ID Extraction

- Mined issue reports and their comments to extract revision ids
- Reference to a revision id generally follows some fixed patterns
- E.g.- r12845, revision 128696, revision=181939, suspecting - 179554, rev. 12456
Step 2: Filter Revision IDs

- Every issue report has a reporting timestamp.
- Every revision has a commit timestamp.
- All revisions having commit timestamp greater than issue reporting timestamp are filtered out.
Step 2: Filter Revision IDs

- Every issue report has a reporting timestamp.
- Every revision has a commit timestamp.
- All revisions having commit timestamp greater than issue reporting timestamp are filtered out.
Step 2 : Filter Revision IDs

- Every issue report has reporting timestamp
- Every revision has a commit timestamp
- All revisions having commit timestamp greater than issue reporting timestamp are filtered out.
Step 3: Heuristics

- Manually inspect several bug reports
- Come up with two heuristics for identifying culprit revision for an issue
  - First Heuristic: Path Overlap
  - Second Heuristic: Num Comments after causing revision mention
Step 3: Heuristics

- Manually inspect several bug reports
- Come up with two heuristics for identifying culprit revision for an issue
  - First Heuristic: Path Overlap
  - Second Heuristic: Num Comments after causing revision mention
Step 3: Heuristics

- Manually inspect several bug reports
- Come up with two heuristics for identifying culprit revision for an issue
  - First Heuristic: Path Overlap
  - Second Heuristic: Num Comments after causing revision mention
Heuristic 1: Path Overlap

- Compute extent of overlap between paths of files modified in bug fixing and bug causing revision.
- Compute bug fixing revision for each bug using regular expression matching.
- Formula for computing overlap:

\[
\text{Overlap} = \frac{\text{Numfilesmatched}}{\min(\text{Numfilescausing}, \text{Numfilesfixing})}
\]
Heuristic 1: Path Overlap

- Issue ID - 5511, Causing Revision ID - 6437, Fixing Revision ID - 11016
Heuristic 2: Num comments after causing revision mention

- If a suspected revision ID is mentioned in comment \( x \) and status of bug changes to fixed within next \( x+8 \) comments and bug is closed within \( x+12 \) comments, then the revision is culprit.
Heuristic 2: Num comments after causing revision mention

Issue ID - 3963

- Regression Causing Revision mentioned
- Fixed in next 2 comments
Heuristic 2: Num comments after causing revision mention

```
#3 ncar...@gmail.com
Idan and I debugged this some today. We confirmed that this is a regression due to
the double-buffering change:
C:\src8\src\chrome\views>svn log tree_view.cc
---
4954 | Sky@google.com | 2008-10-27 20:01:25 +0700 (Mon, 27 Oct 2008) | 6 lines
...;
Regression Causing Revision mentioned

Makes tree double buffered so that it doesn't flicker during resize.
BUG=3784
TEST=none
Review URL: http://codereview.chromium.org/8819

---
Potentially suspicious (to me at least) is the CreateCompatibleDC(NULL) in BitmapPlatformDeviceWin::BitmapPlatformDeviceWinData::GetBitmapDC.
Idan said he had
an idea for how to fix, so he's agreed to take this on.

#4 lafo...@chromium.org
(No comment was entered for this change.)

Labels: Area-Misc Area-BrowserUI Regression Mstone-1.1

#5 id...@chromium.org
This is fixed in r4917

Fixed in next 2 comments
```
Final Ground Truth Dataset consists of 350 issues and their corresponding regression causing revisions

Training Dataset - 300 issues

Test Dataset - 50 issues
Past is a reflection of future

Identify certain features of regression causing revisions that can be used to identify potential revisions that may have regressed the bug.

Features identified are:
- Temporal
- Textual Similarity
Feature Extraction

- Past is a reflection of future

- Identify certain features of regression causing revisions that can be used to identify potential revisions that may have regressed the bug.

- Features identified are:
  - Temporal
  - Textual Similarity
Feature Extraction

- Past is a reflection of future

- Identify certain features of regression causing revisions that can be used to identify potential revisions that may have regressed the bug.

- Features identified are:
  - Temporal
  - Textual Similarity
Temporal Feature

- **Time Difference**(N) in days between the date of report of the issue and the commit date of the bug inducing revision.
  - Mean difference in days is about 26 days
  - 78% of the bugs are reported within 20 days of the induction of the bug.
Temporal Feature

- **Time Difference**($N$) in days between the date of report of the issue and the commit date of the bug inducing revision.
- Mean difference in days is about 26 days
- 78% of the bugs are reported within 20 days of the induction of the bug.
Temporal Feature

- **Time Difference(N)** in days between the date of report of the issue and the commit date of the bug inducing revision.
- Mean difference in days is about 26 days
- 78% of the bugs are reported within 20 days of the induction of the bug.
Temporal Feature

- Box plot for the difference in days for 93.33% of total dataset i.e. time difference less than 100 days.
Character n-gram Approach

- **N-gram** is a continuous subsequence of n items from a given sequence of items

- Word N-grams are a continuous subsequence of words whereas character N-grams are a continuous subsequence of characters

- For example word bi-grams (N=2) and tri-grams (N=3) for phrase “Regression Bug Prediction” are Regression Bug, Bug Prediction and Regression Bug Prediction respectively

- character bi-grams (N=2) and tri-grams (N=3) for word “Regression” are Re,eg,gr,re,es,ss,si,io,ion and Reg,egr,gre,res,ess,ssi,sio,ion respectively
Character n-gram Approach

- **N-gram** is a continuous subsequence of n items from a given sequence of items
- Word N-grams are a continuous subsequence of words whereas character N-grams are a continuous subsequence of characters
- For example word bi-grams (N=2) and tri-grams (N=3) for phrase “Regression Bug Prediction” are Regression Bug, Bug Prediction and Regression Bug Prediction respectively
- character bi-grams (N=2) and tri-grams (N=3) for word “Regression” are Re,eg,gr,re,es,ss,si,io,on and Reg,egr,gre,res,ess,ssi,sio,ion respectively
Character n-gram Approach

- **N-gram** is a continuous subsequence of n items from a given sequence of items.

- Word N-grams are a continuous subsequence of words whereas character N-grams are a continuous subsequence of characters.

- For example, word bi-grams (N=2) and tri-grams (N=3) for phrase “Regression Bug Prediction” are Regression Bug, Bug Prediction and Regression Bug Prediction respectively.

- Character bi-grams (N=2) and tri-grams (N=3) for word “Regression” are Re,eg,gr,re,es,ss,si,io,on and Reg,egr,gre,res,ess,ssi,sio,ion respectively.
Character n-gram Approach

- **N-gram** is a continuous subsequence of n items from a given sequence of items.
- Word N-grams are a continuous subsequence of words whereas character N-grams are a continuous subsequence of characters.
- For example, word bi-grams (N=2) and tri-grams (N=3) for phrase “Regression Bug Prediction” are Regression Bug, Bug Prediction and Regression Bug Prediction respectively.
- Character bi-grams (N=2) and tri-grams (N=3) for word “Regression” are Re,eg,gr,re,es,ss,si,io, on and Reg, egr, gre, res, ess, ssi, si, io, on respectively.
Advantages of Character n-gram

- Ability to Match Concepts: For eg. title of issue ID 128690, it contains word “bookmarks” and Log message of its corresponding regression causing Revision ID 112400 which contains class name BookmarkModelTest.

- Match Term Variation to Common Root: For example, description of issue ID 18749 contains words “print, printed” and its corresponding regression causing Revision ID 20876 also contains words “printing”, “print”.

- Ability to Match Words and their Short-Forms: For example component of issue ID 263160 is “Internals-Network-Cache” and files modified in its corresponding regression causing Revision ID 201493 are in directory “net”.
Advantages of Character n-gram

▶ Ability to Match Concepts: For eg. title of issue ID 128690, it contains word “bookmarks” and Log message of its corresponding regression causing Revision ID 112400 which contains class name BookmarkModelTest.

▶ Match Term Variation to Common Root: For example, description of issue ID 18749 contains words “print, printed” and its corresponding regression causing Revision ID 20876 also contains words “printing”, “print”.

▶ Ability to Match Words and their Short-Forms: For example component of issue ID 263160 is “Internals-Network-Cache” and files modified in its corresponding regression causing Revision ID 201493 are in directory “net.”
Advantages of Character n-gram

- Ability to Match Concepts: For eg. title of issue ID 128690, it contains word “bookmarks” and Log message of its corresponding regression causing Revision ID 112400 which contains class name BookmarkModelTest.

- Match Term Variation to Common Root: For example, description of issue ID 18749 contains words “print, printed” and its corresponding regression causing Revision ID 20876 also contains words “printing”, “print”.

- Ability to Match Words and their Short-Forms: For example component of issue ID 263160 is “Internals-Network-Cache” and files modified in its corresponding regression causing Revision ID 201493 are in directory “net”.
Ability to Match Words Joined Using Special Characters: For example, description of issue ID 125323 contains url chrome://chrome/settings/languages and log message of its corresponding regression causing Revision ID 134134 contains “TEST=browser_tests-gtest_filter=*.
TestSettingsLanguageOptionsPage”
Textual Similarity Features

- 4 textual similarity features are identified
  1. Similarity between Title of a bug report and the Log Message of the revision.
  2. Similarity between the Description of a bug report and the Log Message of the revision.
  3. Similarity between the Cr and Area labels of the issue and the Changed Paths in the VCS.
  4. Similarity between the Title of the issue and the Changed Paths in the VCS.
Textual Similarity Features

- 4 textual similarity features are identified
  1. Similarity between Title of a bug report and the Log Message of the revision.
  2. Similarity between the Description of a bug report and the Log Message of the revision.
  3. Similarity between the Cr and Area labels of the issue and the Changed Paths in the VCS.
  4. Similarity between the Title of the issue and the Changed Paths in the VCS.
Textual Similarity Features

- 4 textual similarity features are identified
  1. Similarity between Title of a bug report and the Log Message of the revision.
  2. Similarity between the Description of a bug report and the Log Message of the revision.
  3. Similarity between the Cr and Area labels of the issue and the Changed Paths in the VCS.
  4. Similarity between the Title of the issue and the Changed Paths in the VCS.
Textual Similarity Features

- 4 textual similarity features are identified
  1. Similarity between Title of a bug report and the Log Message of the revision.
  2. Similarity between the Description of a bug report and the Log Message of the revision.
  3. Similarity between the Cr and Area labels of the issue and the Changed Paths in the VCS.
  4. Similarity between the Title of the issue and the Changed Paths in the VCS.
4 textual similarity features are identified

1. Similarity between Title of a bug report and the Log Message of the revision.

2. Similarity between the Description of a bug report and the Log Message of the revision.

3. Similarity between the Cr and Area labels of the issue and the Changed Paths in the VCS.

4. Similarity between the Title of the issue and the Changed Paths in the VCS.
Similarity between Title of a bug report and the Log Message of the revision
Proposed Approach

Feature Extraction

Similarity between Description of a bug report and the Log Message of the revision

**Issue 79629**: Right click on non-foreground window is ignored

3 people starred this issue and may be notified of changes.

**Status**: Verified
**Owner**: asvilkine@chromium.org
**Closed**: Apr 2011
**OS-Mac**
**Prl.2**
**ReleaseBlock-Beta**
**Type-Bug-Regression**
**Cr-UI**

**Project Member**
**Reported by**: atwilson@chromium.org, Apr 15, 2011

**Chrome Version**: 12.0.733.0
**OS Version**: OS X 10.6.7

**What steps will reproduce the problem?**
1. Open two windows, one overlapping the other.
2. On the background window, right click on a link, in an attempt to open that link in a new window.

**What is the expected result?**
I expect to see a context menu open up to let me open that link. This is what other browsers like Safari do when you right click on a window in the background.

**What happens instead?**
Nothing. Right clicks on non-foreground windows are completely ignored.

Please provide any additional information below. Attach a screenshot if possible.

UserAgentString: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_6_7) AppleWebKit/534.28 (KHTML, like Gecko) Chrome/12.0.733.0 Safari/534.29, gzip(qfe)
Similarity between Description of a bug report and the Log Message of the revision

Revision 80563

Jump to revision: 80563
Author: asvitkine@chromium.org
Date: Wed Apr 6 01:22:27 2011 UTC (3 years, 6 months ago)
Changed paths: 2
Log Message:

[Mac] Don’t route mouse events to the render view host if window is not frontmost.

This change makes Mac Chrome match the behaviour of other browsers on the Mac (tested Firefox and Safari) as well as other Mac applications such as iTunes.

Additionally, this fixes a number of bugs, including:
- Mouse cursor changing when hovering over web content in background windows yet clicks in this case would be eaten to bring the window to the front.
- Mouse cursor NOT changing when hovering web content in background windows when Chrome is not the active application - resulting in inconsistent behaviour.
- http://crbug.com/77928 - resulting in wrong cursor persisting after moving out of the web content area of a background window.

Also, change related unit tests to make the test window the keyWindow so that the tests continue to pass.

BUG=77928
TEST=Open two windows, A and B. Open a webpage in A, then make B the foreground window. Now, hover the cursor over web content in A. The cursor should not change and mouse-over actions should not occur.
R=mark@chromium.org

Description
Similarity between the Cr and Area labels of the issue and Changed Paths in the VCS
Similarity between the Title of the issue and the Changed Paths in the VCS

Issue 72859: ClearBrowserData Overlay doesn't open via Shortcut/Menu/Wrench-Menu
24 people starred this issue and may be notified of changes.

Status: Fixed
Project Member Reported by meh...@chromium.org, Feb 13, 2011

Changed paths:

<table>
<thead>
<tr>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>trunk/src/chrome/browser/resources/options/advanced_options.js</td>
</tr>
<tr>
<td>trunk/src/chrome/browser/resources/options/alert_overlay.js</td>
</tr>
<tr>
<td>trunk/src/chrome/browser/resources/options/clear_browser_data_overlay.js</td>
</tr>
<tr>
<td>trunk/src/chrome/browser/resources/options/content_settings_exceptions_area.js</td>
</tr>
<tr>
<td>trunk/src/chrome/browser/resources/options/options.js</td>
</tr>
<tr>
<td>trunk/src/chrome/browser/resources/options/options_page.js</td>
</tr>
<tr>
<td>trunk/src/chrome/browser/resources/options/search_page.js</td>
</tr>
</tbody>
</table>
Similarity Function

- Use the similarity function suggested by [14]
- Formula for computing the similarity between 2 documents:

\[
SIM(U, V) = \frac{\sum_{u \in U} \sum_{v \in V} \text{Match}(u, v) \times \text{Length}(u)}{|U| \times |V|}
\]

\[
\text{Match}(u, v) = \begin{cases} 
1 & \text{if } u = v \\
0 & \text{Otherwise}
\end{cases}
\]

\[
|U| = \sqrt{f_{u_1}^2 + f_{u_2}^2 + \ldots + f_{u_n}^2}
\]
Similarity Plots
Architecture Diagram
Architecture Diagram
Revision ID Extractor

- Extracts all revision IDs that were committed maximum “N” days before the reporting timestamp of given issue.

- Log messages and modified file paths are extracted for each revision.
Revision ID Extractor

- Extracts all revision IDs that were committed maximum “N” days before the reporting timestamp of given issue.

- Log messages and modified file paths are extracted for each revision
Pre-processing - Issue title, description and log messages are pre-processed to remove noisy data

<table>
<thead>
<tr>
<th>S.No</th>
<th>Stop words/Phrase/Sentences removed</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>“Report ID”, “Cumulative Uptime”, “Other browsers tested”, “Meta information:”, “Thank you”</td>
</tr>
<tr>
<td>3</td>
<td>“What steps will reproduce the problem”, “What is the expected output”, “What do you see instead”, “Kindly refer the screencast for reference”</td>
</tr>
</tbody>
</table>
Feature Extractor

- Character n-gram feature extractor - Converts pre-processed data into bag of character n-grams
Similarity calculator

For each revision ID, this module computes similarity between following 4 features:

1. Title of a bug report and the Log Message of the revision.
2. Description of a bug report and the Log Message of the revision.
3. Cr and Area labels of the issue and the top levels of the Changed Paths in the VCS.
4. Title of the issue and the Changed Paths in the VCS.
Similarity Calculator

- Computes the overall similarity value for each revision expressed as a weighted sum of similarities of each feature

\[ SIM_{SCORE} = W_1 \times SIM(Feature1) + W_2 \times SIM(Feature2) + W_3 \times SIM(Feature3) + W_4 \times SIM(Feature4) \]

(5)

- Higher the similarity score, higher is the rank of the revision
- Top k revision IDs are then recommended to the bug fixer as the most suspected revisions of having caused the bug.
Similarity Calculator

- Computes the overall similarity value for each revision expressed as a weighted sum of similarities of each feature

\[
SIM_{SCORE} = W_1 \times SIM(Feature1) + W_2 \times SIM(Feature2) + W_3 \times SIM(Feature3) + W_4 \times SIM(Feature4)
\]

(5)

- Higher the similarity score, higher is the rank of the revision
- Top k revision IDs are then recommended to the bug fixer as the most suspected revisions of having caused the bug.
Similarity Calculator

- Computes the overall similarity value for each revision expressed as a weighted sum of similarities of each feature

\[
SIM_{SCORE} = W_1 \times SIM(\text{Feature1}) + W_2 \times SIM(\text{Feature2}) + W_3 \times SIM(\text{Feature3}) + W_4 \times SIM(\text{Feature4})
\]

(5)

- Higher the similarity score, higher is the rank of the revision
- Top k revision IDs are then recommended to the bug fixer as the most suspected revisions of having caused the bug.
Presentation Outline

1. Research Motivation and Aim
2. Related Work and Research Contributions
3. Experimental Dataset
4. Characterization Study
5. Proposed Approach
6. Experimental Evaluation and Results
7. Limitations and Conclusion
Appropriate Value of N

- N - Number of lookback days

<table>
<thead>
<tr>
<th>S.No</th>
<th>N</th>
<th>Avg. # Revisions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>N=10</td>
<td>1543</td>
</tr>
<tr>
<td>2</td>
<td>N=15</td>
<td>2340</td>
</tr>
<tr>
<td>3</td>
<td>N=20</td>
<td>3073</td>
</tr>
<tr>
<td>4</td>
<td>N=30</td>
<td>4515</td>
</tr>
</tbody>
</table>
Predictive Power of each feature

- D-L turns out to be the most important feature
- SIM UNEQ turns out to be best weight configuration

<table>
<thead>
<tr>
<th>Label</th>
<th>W(T-L)</th>
<th>W(D-L)</th>
<th>W(C-P)</th>
<th>W(T-P)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIM T-L</td>
<td>1.0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SIM D-L</td>
<td>0</td>
<td>1.0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SIM C-P</td>
<td>0</td>
<td>0</td>
<td>1.0</td>
<td>0</td>
</tr>
<tr>
<td>SIM T-P</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.0</td>
</tr>
<tr>
<td>SIM UNEQ</td>
<td>0.25</td>
<td>0.3</td>
<td>0.2</td>
<td>0.25</td>
</tr>
</tbody>
</table>
Results

Best accuracy for N=20 and K=75

<table>
<thead>
<tr>
<th>N</th>
<th>Top 20</th>
<th>Top 30</th>
<th>Top 50</th>
<th>Top 75</th>
<th>Top 100</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>21</td>
<td>29</td>
<td>30.61</td>
<td>33</td>
<td>33</td>
</tr>
<tr>
<td>15</td>
<td>36.73</td>
<td>42.85</td>
<td>49</td>
<td>51.02</td>
<td>51.02</td>
</tr>
<tr>
<td>20</td>
<td>36.73</td>
<td>41</td>
<td>54</td>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td>30</td>
<td>32.65</td>
<td>42.86</td>
<td>46.94</td>
<td>57.14</td>
<td>59.18</td>
</tr>
</tbody>
</table>
Results

Figure 3: Bubble Plot showing accuracy (percentage) results for different values of N and K
Presentation Outline

1. Research Motivation and Aim
2. Related Work and Research Contributions
3. Experimental Dataset
4. Characterization Study
5. Proposed Approach
6. Experimental Evaluation and Results
7. Limitations and Conclusion
Limitations

- Small size of dataset
- Dependency on previous predictions of bug fixers'
- Level of detail provided in issue reports and revision log messages
- Noisy data in title and description of issue reports and log messages of revisions
- Unable to recommend for Outliers
Limitations

- Small size of dataset
- Dependency on previous predictions of bug fixers’
- Level of detail provided in issue reports and revision log messages
- Noisy data in title and description of issue reports and log messages of revisions
- Unable to recommend for Outliers
Limitations

- Small size of dataset
- Dependency on previous predictions of bug fixers’
- Level of detail provided in issue reports and revision log messages
- Noisy data in title and description of issue reports and log messages of revisions
- Unable to recommend for Outliers
Limitations

- Small size of dataset
- Dependency on previous predictions of bug fixers’
- Level of detail provided in issue reports and revision log messages
- Noisy data in title and description of issue reports and log messages of revisions
- Unable to recommend for Outliers
Limitations

- Small size of dataset
- Dependency on previous predictions of bug fixers’
- Level of detail provided in issue reports and revision log messages
- Noisy data in title and description of issue reports and log messages of revisions
- Unable to recommend for Outliers
Limitations

- Small size of dataset
- Dependency on previous predictions of bug fixers’
- Level of detail provided in issue reports and revision log messages
- Noisy data in title and description of issue reports and log messages of revisions
- Unable to recommend for Outliers
Conclusion

▶ More than 50% of labelled bugs are regression and majority of regression bugs have high priority.

▶ Quality of bug fixing process for Regression bugs decreases as system becomes complex.

▶ Character n-gram based approach for computing textual similarity between issue reports and log messages, files modified can be used for identifying suspected regression causing revisions.

▶ The approach can further be improved by improving pre-processing stage.
Conclusion

- More than 50% of labelled bugs are regression and majority of regression bugs have high priority.

- Quality of bug fixing process for Regression bugs decreases as system becomes complex.

- Character n-gram based approach for computing textual similarity between issue reports and log messages, files modified can be used for identifying suspected regression causing revisions.

- The approach can further be improved by improving pre-processing stage.
Conclusion

- More than 50% of labelled bugs are regression and majority of regression bugs have high priority.

- Quality of bug fixing process for Regression bugs decreases as system becomes complex.

- Character n-gram based approach for computing textual similarity between issue reports and log messages, files modified can be used for identifying suspected regression causing revisions.

- The approach can further be improved by improving pre-processing stage.
Conclusion

- More than 50% of labelled bugs are regression and majority of regression bugs have high priority.

- Quality of bug fixing process for Regression bugs decreases as system becomes complex.

- Character n-gram based approach for computing textual similarity between issue reports and log messages, files modified can be used for identifying suspected regression causing revisions.

- The approach can further be improved by improving pre-processing stage.
References I


References II


References III

Alexander Tarvo.  
Using statistical models to predict software regressions.  
In ISSRE ’08, pages 259–264. IEEE.

Alexander Tarvo.  
Mining software history to improve software maintenance quality: A case study.  

S. Lal and A. Sureka.  
Comparison of seven bug report types: A case-study of google chrome browser project.  
References IV


References V
