Nirikshan: Mining Bug Report History for Discovering Process Maps, Inefficiencies and Inconsistencies

Monika Gupta, Ashish Sureka
monikag@iiitd.ac.in and ashish@iiitd.ac.in

Indraprastha Institute of Information Technology
New Delhi, India
Presentation Outline

• Research Motivation
• Research Aim
• Related Work and Research Contributions
• Research Methodology and Framework
  • Process Discovery
  • Performance Analysis
  • Inconsistencies: Conformance Verification
• Conclusion
“If one cannot measure it, one cannot improve it”
Research Motivation

- Extract knowledge from event logs recorded by an information system.
- Event logs (e.g. transaction logs) with four fields:
- Tools and framework for process mining:
  - ProM [1] (Open Source)
  - Disco [2] (Commercial)
- Takes logs to:
  - Discover process,
  - Perform compliance verification,
  - Identify inefficiencies and imperfections.

Software Repositories:

• Artifacts generated by the tools during software evolution and archived for future reference.

• Rich data available.

• Uncover interesting and actionable information for process improvement.

For example:
Issue Tracking System (ITS),
Version Control System,
Code Review etc.
Research Motivation

Issue Tracking System:

• Information system to support business process of issue reporting, tracking and management.

• Process is defined to structure and streamline issue management activities.

• Runtime process may not conform to design process, may have inefficiencies and inconsistencies.

• Motivated by need to mine process data generated by ITS to Identify inefficiencies and inconsistencies.

• Improve productivity, process capability and efficiency [3].

Example: Bugzilla[4]: Mozilla, Eclipse

Jira[5]: Facebook, Adobe

5. https://www.atlassian.com/software/jira
Steps to reproduce

Actual results

Expected results
### Bugzilla@Mozilla – Activity log for bug 600028: Set up Preview to allow it to see the RockYou guide

**Back to bug 600028**

<table>
<thead>
<tr>
<th>Who</th>
<th>When</th>
<th>What</th>
<th>Removed</th>
<th>Added</th>
</tr>
</thead>
<tbody>
<tr>
<td>stephen.donner</td>
<td>2010-09-27 14:45:04 PDT</td>
<td>CC</td>
<td>jbalogh</td>
<td></td>
</tr>
<tr>
<td>mrz</td>
<td>2010-09-29 10:55:13 PDT</td>
<td>Assignee</td>
<td>server-ops</td>
<td>jeremy.orem+bugs</td>
</tr>
<tr>
<td>oremj</td>
<td>2010-09-29 16:33:12 PDT</td>
<td>Status</td>
<td>NEW</td>
<td>RESOLVED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Resolution</td>
<td>---</td>
<td>FIXED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Last Resolved</td>
<td>2010-09-29 16:33:12</td>
<td></td>
</tr>
<tr>
<td>stephen.donner</td>
<td>2010-09-29 17:27:48 PDT</td>
<td>Status</td>
<td>RESOLVED</td>
<td>VERIFIED</td>
</tr>
</tbody>
</table>
Nirikshan  *(Sanskrit Word)*

*To Review or Investigate*

Investigation of bug history data generated by Bugzilla issue tracking system for Mozilla project to discover process maps, identify inefficiencies and inconsistencies.
Research Aim

• Application of process mining platforms such as Disco and state-of-the-art algorithms for discovering process maps from ITS event logs.

• Develop a generic algorithm for quantitatively measuring the compliance (conformance checking) between the design time and the runtime (reality) process model.

• Define a process mining framework “Nirikshan” and conduct a case-study on popular open-source projects like Firefox and Core from Mozilla foundation.

• Performance analysis of various process-related activities and identify inefficiencies and imperfections.
## Related Work and Research Contributions

<table>
<thead>
<tr>
<th>Author</th>
<th>Year</th>
<th>Repository</th>
<th>Objectives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rubin <em>et al.</em></td>
<td>2007</td>
<td>Subversion logs of the ArgoUML project. (OSS)</td>
<td>Linear Temporal Logic (LTL) Checking (Conformance Analysis), Social Network Discovery, Performance Analysis, Petri Net Discovery</td>
</tr>
<tr>
<td>Akman <em>et al.</em></td>
<td>2009</td>
<td>Software Configuration Management of industry project.</td>
<td>Analyzed and compared the effectiveness of four process discovery algorithms on software process. Analyzed discrepancies between real time and design time process.</td>
</tr>
<tr>
<td>Knab <em>et al.</em></td>
<td>2010</td>
<td>ITS of EUREKA project SERIOUS (CSS).</td>
<td>Interactive approach to visualize effort estimation and process lifecycle patterns in ITS to detect outliers, flaws and interesting properties.</td>
</tr>
<tr>
<td>Poncin <em>et al.</em></td>
<td>2011</td>
<td>aMSN and GCC bug repositories, mail archives, SVM</td>
<td>Combined different repositories for analysis using a prototype, FRASR. Role classification and Bug life cycle construction using ProM.</td>
</tr>
<tr>
<td>Sunindyo <em>et al.</em></td>
<td>2012</td>
<td>Red Hat Linux ITS</td>
<td>Framework for collecting and analyzing data from bug reporting system, conformance checking to improve process quality.</td>
</tr>
</tbody>
</table>
## Related Work and Research Contributions

<table>
<thead>
<tr>
<th>Author</th>
<th>Year</th>
<th>Repository</th>
<th>Objectives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Poncin <em>et al.</em></td>
<td>2011</td>
<td>aMSN and GCC bug repositories, mail archives, SVM</td>
<td>Combined different repositories for analysis using a prototype, FRASR. Role classification and Bug life cycle construction using ProM.</td>
</tr>
<tr>
<td>Sunindyo <em>et al.</em></td>
<td>2012</td>
<td>Red Hat Linux ITS</td>
<td>Framework for collecting and analyzing data from bug reporting system, conformance checking to improve process quality.</td>
</tr>
</tbody>
</table>

- Identified process mining perspective: Control flow, Time, Organizational etc.
- Major emphasis on Organizational perspective
- Conformance verification not in-depth

---

Related Work and Research Contributions

Process mining event log from ITS to discover:

- **Runtime process** (as-is process)
  - Analyze activities, transitions, events and unique traces

- **Inefficiencies**
  - Reopen analysis
  - Bottleneck identification
  - Self-loops and Back-forth analysis

- **Inconsistencies**
  - Algorithm to compute degree of conformance between design and runtime process

- **Case-study** on ITS data for Mozilla (Firefox and Core)
## Experimental Dataset Details for **Core** and **Firefox**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project</td>
<td>Mozilla Firefox and Core</td>
</tr>
<tr>
<td>Reporting First issue</td>
<td>1 January 2012</td>
</tr>
<tr>
<td>Last issue creation date</td>
<td>31 December 2012</td>
</tr>
<tr>
<td>Date of extraction</td>
<td>14 July 2013</td>
</tr>
<tr>
<td>Total issues in 2012</td>
<td>1,11,234</td>
</tr>
<tr>
<td>Issues not authorized for access</td>
<td>15,638</td>
</tr>
<tr>
<td>Issues without history</td>
<td>3,149</td>
</tr>
<tr>
<td>Total issues for Firefox</td>
<td>12,234</td>
</tr>
<tr>
<td>Total issues for Core</td>
<td>24,253</td>
</tr>
<tr>
<td>Total activities for Core</td>
<td>88,396</td>
</tr>
<tr>
<td>Total activities for Firefox</td>
<td>40,233</td>
</tr>
</tbody>
</table>

**Research Methodology and Framework**

Feb 20, 2014
Research Methodology and Framework

NIRIKSHAN

Bugzilla Issue Tracking System
XML-RPC
JSON-RPC
RAW DATA (MySQL)
Data Extraction

Schema Mapping
Case ID, Timestamp
Activity, Resource
EVENT LOG
Data Transformation

DISCO
PROM
Process Mining

1
Discovery
Process Model
2
Performance
Analysis
3
Verification
Conformance

Challenge:
Produce a log conforming to the input format of process mining tool.
Research Methodology and Framework

Identify important events in lifecycle:

- Open Status
  - *New, Unconfirmed, Assigned, Reopened*
- Component Reassignment
- Developer Reassignment
- QA Reassignment
- Closed Resolution
  - *Fixed, Invalid, Wontfix, Duplicate, Worksforme, Incomplete*
- Verified

- Data compilation, add *Reported* as the first event in the workflow.
- Resolve *Same timestamp* issue of QA-reassignment, Comp-Reassignment and Dev-reassignment.

<table>
<thead>
<tr>
<th>BugID</th>
<th>Activity as Event</th>
<th>Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>714508</td>
<td>REPORTED</td>
<td>2012-01-01 12:43:50</td>
</tr>
<tr>
<td>714509</td>
<td>REPORTED</td>
<td>2012-01-01 13:21:59</td>
</tr>
<tr>
<td>714509</td>
<td>UNCONFIRMED</td>
<td>2012-01-01 13:22:00</td>
</tr>
<tr>
<td>714509</td>
<td>WORKFORME</td>
<td>2012-01-03 07:46:09</td>
</tr>
<tr>
<td>714509</td>
<td>FIXED</td>
<td>2012-01-03 08:23:04</td>
</tr>
<tr>
<td>714509</td>
<td>COMP-REASSIGN</td>
<td>2012-01-11 14:34:07</td>
</tr>
<tr>
<td>714509</td>
<td>QA-REASSIGN</td>
<td>2012-01-11 14:34:08</td>
</tr>
<tr>
<td>714509</td>
<td>DEV-REASSIGN</td>
<td>2012-01-13 20:40:31</td>
</tr>
<tr>
<td>714512</td>
<td>REPORTED</td>
<td>2012-01-01 14:04:59</td>
</tr>
<tr>
<td>714532</td>
<td>UNCONFIRMED</td>
<td>2012-01-01 17:23:40</td>
</tr>
<tr>
<td>714532</td>
<td>REPORTED</td>
<td>2012-01-01 17:33:59</td>
</tr>
<tr>
<td>714532</td>
<td>COMP-REASSIGN</td>
<td>2012-01-01 19:23:05</td>
</tr>
<tr>
<td>714532</td>
<td>QA-REASSIGN</td>
<td>2012-01-01 19:23:06</td>
</tr>
<tr>
<td>714532</td>
<td>INCOMPLETE</td>
<td>2012-01-12 16:08:03</td>
</tr>
<tr>
<td>714532</td>
<td>VERIFIED</td>
<td>2012-01-13 01:14:35</td>
</tr>
<tr>
<td>714532</td>
<td>DUPLICATE</td>
<td>2012-01-13 01:48:46</td>
</tr>
<tr>
<td>714534</td>
<td>REPORTED</td>
<td>2012-01-01 17:55:59</td>
</tr>
<tr>
<td>714534</td>
<td>UNCONFIRMED</td>
<td>2012-01-01 17:56:00</td>
</tr>
<tr>
<td>714534</td>
<td>REPORTED</td>
<td>2012-01-01 19:10:59</td>
</tr>
<tr>
<td>714537</td>
<td>REPORTED</td>
<td>2012-01-01 18:44:59</td>
</tr>
<tr>
<td>714543</td>
<td>UNCONFIRMED</td>
<td>2012-01-01 19:51:00</td>
</tr>
<tr>
<td>714543</td>
<td>REPORTED</td>
<td>2012-01-01 19:50:59</td>
</tr>
<tr>
<td>714543</td>
<td>DUPLICATE</td>
<td>2012-01-02 24:22:00</td>
</tr>
<tr>
<td>714544</td>
<td>UNCONFIRMED</td>
<td>2012-01-01 20:00:00</td>
</tr>
<tr>
<td>714544</td>
<td>REPORTED</td>
<td>2012-01-01 19:59:59</td>
</tr>
<tr>
<td>714544</td>
<td>INCOMPLETE</td>
<td>2012-01-02 16:15:40</td>
</tr>
<tr>
<td>714575</td>
<td>REPORTED</td>
<td>2012-01-02 03:51:59</td>
</tr>
<tr>
<td>714576</td>
<td>UNCONFIRMED</td>
<td>2012-02-02 02:24:00</td>
</tr>
</tbody>
</table>

BugID as CaseID

Data imported to DISCO

Research Methodology and Framework

Feb 20, 2014
Research Methodology and Framework

NIRIKSHAN

Data Extraction

Data Transformation

Process Mining

Discovery
- Process Model

Performance Analysis

Verification
- Conformance

Case ID, Timestamp
- Activity, Resource

EVENT LOG

DISCO

PROM
"What is really happening?"

- Obtain **Runtime process map**.
- Event log are reflection of reality.
- Ability to directly perceive how a process is actually working.
- Leads to better results in the process improvement project.
SNAPSHOT OF MOZILLA BUG HISTORY (BUG ID 600028)
Discovered Process Map for Firefox with frequency as labels

UNIQUE TRACES:

- Complete sequence of events in lifecycle
- Total unique traces:
  - Core: 1164,
  - Firefox: 622
- 80% of the cases covered with:
  - 2% traces (Core)
  - 3% traces (Firefox)
• Less percentage of bugs closed with **Verified** state.
• A good number of bugs marked as **Duplicate, Invalid and Worksforme** which is not desired.
EVENT ANALYSIS

- Majority of the cases have 3 events in lifecycle.
- Reported → New/Unconfirmed → Resolved
- **Stable** as very few have more than 10 events in the lifecycle.
Inefficiencies:

• **Reopen Analysis**
  • If a fair number of fixed bugs are reopened, it could indicate instability in the software system \[9\].
  • Increase maintenance costs, degrade overall user-perceived quality of the software and lead to unnecessary rework by busy practitioners \[10\].

• **Self Loop and back-forth analysis**
  • Indicator of deeper problems.

• **Bottlenecks**
  • Identify most time consuming transitions of process.
  • Identify the cause of delay in end-to-end process.

• Several *Wontfix* and *Worksforme* marked bugs getting reopened. Efforts should be made to better understand the problem and avoid priority disagreements.

• *Duplicate* should be marked carefully in Core.

• Many *Fixed* are reopened. Proper understanding of root cause and minimum testing is required.
Performance Analysis: Self Loops, Back-forth

Self Loops:

Maximum loops for:
- Developer reassignment, and Component reassignment

Some Cases:
- Number of loops > 1, causing undesired delay.

Back-forth: \( \text{A} \leftrightarrow \text{B} \)

- Activities frequently involved in back-forth pattern: Unconfirmed, Component-reassignment, Developer-reassignment, QA-reassignment and Reopen

- Regression or Imperfection
  - Disagreement between developers

Feb 20, 2014
Performance Analysis: Bottlenecks

Unconfirmed

9.3 days

33.4 days

New

New

17.1 days

17.4 days

Assigned

Assigned

Efficient

Duplicate

Duplicate

Fixed

Fixed

Any State

High

Worksforme, Wontfix, Incomplete

Feb 20, 2014
“Do we do what was agreed upon?”

Design Process Model v/s Discovered Runtime Process Model

Calculate Fitness Metric (FM) as:

\[ FM = \frac{\sum_{i=1}^{N} (F_i \times V_i)}{\sum_{i=1}^{N} (F_i)} \]

where,

- \( F_i \) = Frequency of unique trace \( i \)
- \( V_i \) = Valid bit for trace \( i \)
- \( N \) = Number of unique traces

FM for:

- Core = 0.86
- Firefox = 0.91

If FM < 1 then:
If FM < 1 then:

Inconsistent Transition Frequency Matrix (ITF) = $(TF - TF \cdot A)$
Where $TF$ = Transition Frequency Matrix
$A$ = Adjacency Matrix

Total Inconsistent Transition
[ Core: 2412, Firefox: 739 ]

Highest frequency of Inconsistent Transition
[ Core: 1643, Firefox: 305 ]

Most frequent Inconsistent Transition
[ Reported → Assigned ]
Conclusion

• Runtime process map discovered for 12234 Firefox and 24253 Core issues.

• Analyzed distribution of activities, transitions, events in lifecycle and unique traces.

• Self-loops and back-forth transitions are noticed.

• Reopening of many Wontfix and Worksforme labelled issues observed.

• Bottlenecks are identified.

• Conformance checking algorithm and metrics proposed.
THANK YOU!

QUESTIONS ...

Feb 20, 2014
1. https://bugzilla.mozilla.org


