Process Cube for Software Defect Resolution

Monika Gupta and Ashish Sureka

Indraprastha Institute of Information Technology, Delhi (IIITD), India

monikag@iiitd.ac.in, ashish@iiitd.ac.in
Presentation Outline

1. Research Motivation and Aim
2. Related Work and Research Contributions
3. Research Methodology and Framework
4. Research Methodology and Framework
5. References
Research Motivation and Aim

Introduction to Process Cube

1. Stand-alone process analysis is the common way of analyzing processes in today’s process mining approaches [1].

2. There may be multiple variants of same process influenced by multiple factors like different working patterns, diverse business requirements and change with time (concept drift). Therefore, it is of great importance to cluster cases based on various attributes and compare process mining results [2].

3. Process cube is a multidimensional data structure where events and process models are organized using different dimensions to facilitate comparison of multiple processes [1].
Introduction to Process Cube

1. Process Cube View (PCV) is defined using Process Cube Structure (PCS) and Event Base (EB).

2. PCS defines the dimensions of the cube, and EB is the complete set of attributes used for dimension mapping and materialized sublog creation.
Issue Resolution Process

1. ITSs (like Bugzilla\(^1\) and Mantis\(^2\)), PCRs (like Gerrit\(^3\) and Rietveld\(^4\)) and VCSs (like SVN\(^5\) and Mercurial\(^6\)), are workflow management systems jointly supporting the bug reporting and resolution process in software maintenance.

2. An issue reported in ITS is characterized by many attributes like *Priority*, *Type* of issue, *Component* to which issue pertains and *OS*.

\(^1\)http://www.bugzilla.org/
\(^2\)http://www.mantisbt.org/
\(^3\)http://code.google.com/p/gerrit/
\(^4\)http://code.google.com/p/rietveld/
\(^5\)http://subversion.apache.org/
\(^6\)http://mercurial.selenic.com/
Research Motivation

1. This work is motivated by the need to compare and understand the differences in process for cases with different characteristics along various dimensions.

2. We believe it is important to consider how differences along various dimensions produce process variants.
Research Aim

1. To define PCV for software defect resolution process.

2. To investigate applications of OLAP operations like slice, dice, roll-up and drill-down on process cube, thus, compare process along various dimensions.

3. To discover process model using state-of-the-art algorithms for each process cell. Also evaluate and compare metrics from multiple perspectives like control flow, time (bottleneck), and organizational to identify inefficiencies and suggest modifications for improved performance.

4. To conduct a case-study on a popular FLOSS project, Google Chromium to illustrate the usefulness of proposed approach and compare process mining results across various process variants.
Presentation Outline

1. Research Motivation and Aim
2. Related Work and Research Contributions
3. Research Methodology and Framework
4. Research Methodology and Framework
5. References
Data Warehousing Environment in Software Development

1. Ruiz et al. present a DW environment to support the implementation of a process measurement and analysis program in a CMM Level 2 organization [3]. DW provides a centralized and unified view of all projects and stores project-related data to support monitoring of software development according to the defined metrics [3].

2. Colaço et al. propose a data warehousing approach for software development data analysis [4]. They present a dimensional model for software change history and analyze its data using two tools: a change history miner and a corrective maintenance dashboard [4].
Silveira et al. present a software process data warehousing architecture, SPDW+ as a solution to frequent, seamless, and automated capturing of software quality metrics, and their integration in a central repository for full range of analysis [5].
Process Mining Software Repositories

1. Gupta et al. present an application of integrating and process mining three software repositories (ITS, PCR and VCS) from control flow and organizational perspective [6].

2. Kim et al. propose a distributed workflow mining approach to discover workflow process model incrementally amalgamating a series of vertically or horizontally fragmented temporal work-cases [7].

3. Poncin et al. present a framework called as FRASR (FRamework for Analyzing Software Repositories) that facilitates combining and matching of events across multiple repositories like mail archives, subversion and bug repositories followed by assignment of role to each developer [8].

4. Song et al. apply process mining technology to common event logs obtained from five information systems for behavior pattern mining [9].
Process Cubes

1. Aalast et al. formalize the notion of process cubes where events and process models are organized using different dimensions which allows interactive analysis and exploration of process [1].

2. Mamaliga et al. developed an initial prototype for process cubes, ProCube using the process mining framework ProM and the Palo OLAP toolset to allow comparison of multiple processes [2]. ProCube plugin in ProM creates sublogs per cell on-the-fly (unlike traditional process mining approaches) and visualizes the discovered process models, social networks and dotted charts [2].
Novel and Unique Research Contributions

1. While there has been work done in the area of data warehousing for software development, to the best of our knowledge, the study presented in this paper is first work on application of OLAP model to define process cube notion for software defect resolution.

2. While stand-alone defect resolution process has been process mined from multiple perspectives, this paper presents a novel framework to facilitate comparison between various process variants by applying OLAP operations on process cube.
Novel and Unique Research Contributions

1. We conduct an *in-depth empirical analysis on Google Chromium project (open-source) data extracted from Google issue tracker, Rietveld PCR and Subversion VCS* to demonstrate the application and effectiveness of our approach. We compare process mining results for different issue *Type* such as Security and Performance. We present results for runtime process discovery, activity analysis, transition analysis and bottleneck identification.
Presentation Outline

1. Research Motivation and Aim
2. Related Work and Research Contributions
3. Research Methodology and Framework
4. Research Methodology and Framework
5. References
Research Methodology and Framework

## Experimental Dataset

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>First ITS issue creation date</td>
<td>1 Jan 2009</td>
</tr>
<tr>
<td>Last ITS issue creation date</td>
<td>31 Dec 2012</td>
</tr>
<tr>
<td>Total extracted ITS issues</td>
<td>177926</td>
</tr>
<tr>
<td>Total closed ITS issues</td>
<td>145326 (81.67% of extracted)</td>
</tr>
<tr>
<td>Total closed ITS issues with lifecycle spanning 3 IS</td>
<td>39550 (27.21% of closed)</td>
</tr>
<tr>
<td>Total closed ITS issues with type <em>Security</em></td>
<td>2271 (1.56% of closed)</td>
</tr>
<tr>
<td>Total closed ITS issues with type <em>Performance</em></td>
<td>716 (0.49% of closed)</td>
</tr>
</tbody>
</table>
**APSEC 2014, Process Cube for Software Defect Resolution**

**Research Methodology and Framework**

**Data Extraction, Integration and Transformation**

---

**ITS BUG ID**

- **Issue 88294**: Default printing settings are always "two sided"
- **Issue 7285039**: PrintPreview: [WIN] Fix the default duplex print setting. (Closed)

**PCR ISSUE ID**

- **Description**: PrintPreview: [WIN] Fix the default duplex print setting.
- **ITS ISSUE ID**: kmadhus@chromium.org
- **Committed**: http://src.chromium.org/viewvc/chrome?view=rev&rev=92154
- **Patch Set 1**: Total comments: 2
- **Patch Set 2**: Fixed nit

**PEER CODE REVIEW SYSTEM**

**VERSION CONTROL SYSTEM**

**VCS REVISION ID**

- **Revision 92154**
- **Jump to revision**: 92154
- **Author**: kmadhus@chromium.org
- **Date**: Tue Jul 12 12:55:14 2011 UTC (2 years, 10 months ago)
- **Changed paths**: 1
- **Log Message**: PrintPreview: [WIN] Fix the default duplex print setting.

**PCR ISSUE ID**

- **codereview.chromium.org**
- **chromiumcodereview.appspot.com**
Case and Event Attributes

1. **Case Attributes**: Properties associated with a case which remain constant for all the events pertaining to same case. For example: CaseID, Type, State, Closed Status, Priority, OS, Reporter, Owner, Component and Reported Time of an issue.

2. **Event Attributes**: Properties characterizing each event in lifecycle of an issue like Activity, Actor (resource) and Timestamp (when the activity is performed).
List of Activities and Significance.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Issue Tracking System Activities</strong></td>
<td></td>
</tr>
<tr>
<td>I_Creation</td>
<td>Issue reported in ITS</td>
</tr>
<tr>
<td>I_Open</td>
<td>Open bug status label</td>
</tr>
<tr>
<td>Fixed</td>
<td>Resolved as Fixed</td>
</tr>
<tr>
<td>Invalid</td>
<td>Illegible, spam etc.</td>
</tr>
<tr>
<td>Duplicate</td>
<td>Issue has been reported in another bug, or shares the same root cause as another bug.</td>
</tr>
<tr>
<td>WontFix</td>
<td>Can’t reproduce, Working as intended, Obsolete</td>
</tr>
<tr>
<td>Verified</td>
<td>Resolution verified</td>
</tr>
<tr>
<td>I_Closed</td>
<td>ITS progress ends</td>
</tr>
<tr>
<td><strong>Code Review System Activities</strong></td>
<td></td>
</tr>
<tr>
<td>C_Creation</td>
<td>Patch reported in PCR</td>
</tr>
<tr>
<td>C_Reviewed</td>
<td>Code review process ends</td>
</tr>
<tr>
<td><strong>Version Control System Activities</strong></td>
<td></td>
</tr>
<tr>
<td>V_COMMIT</td>
<td>Code change committed</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td><strong>Case Attributes</strong></td>
<td>CaseID</td>
</tr>
<tr>
<td></td>
<td>Unique ITS issue ID</td>
</tr>
<tr>
<td></td>
<td>Type [10]</td>
</tr>
<tr>
<td></td>
<td>Issue type</td>
</tr>
<tr>
<td></td>
<td>State</td>
</tr>
<tr>
<td></td>
<td>Issue is in process or closed</td>
</tr>
<tr>
<td></td>
<td>Closed Status&lt;sup&gt;7&lt;/sup&gt;</td>
</tr>
<tr>
<td></td>
<td>Final issue resolution label</td>
</tr>
<tr>
<td></td>
<td>Priority</td>
</tr>
<tr>
<td></td>
<td>Importance of fixing an issue</td>
</tr>
<tr>
<td></td>
<td>OS</td>
</tr>
<tr>
<td></td>
<td>Operating System to which issue pertains</td>
</tr>
<tr>
<td></td>
<td>Reporter</td>
</tr>
<tr>
<td></td>
<td>Person who reports issue</td>
</tr>
<tr>
<td></td>
<td>Component&lt;sup&gt;8&lt;/sup&gt;</td>
</tr>
<tr>
<td></td>
<td>Component to which issue pertains</td>
</tr>
<tr>
<td></td>
<td>Reported Time</td>
</tr>
<tr>
<td></td>
<td>Published time for issue</td>
</tr>
<tr>
<td></td>
<td>Owner</td>
</tr>
<tr>
<td></td>
<td>Person who manages issue</td>
</tr>
<tr>
<td><strong>Event Attributes</strong></td>
<td>Activity</td>
</tr>
<tr>
<td></td>
<td>Task performed</td>
</tr>
<tr>
<td></td>
<td>Timestamp</td>
</tr>
<tr>
<td></td>
<td>Time when activity is performed</td>
</tr>
<tr>
<td></td>
<td>Actor</td>
</tr>
<tr>
<td></td>
<td>Person who performs activity</td>
</tr>
</tbody>
</table>
### Sparsity | Domain
--- | ---
— | Integer
No | Regression, Security, Crash, Performance, Usability, Polish, Cleanup
No | Open, Closed
No | Fixed, Verified, Duplicate, WontFix, Invalid, External Dependency, FixUnreleased, Icebox
Hierarchy | Class (High or Low) → Value (0 and 1; 2 and 3)
No | All, Chrome, Linux, Mac, Windows
Clustering | Low, Medium, High : Based on frequency of issues reported
Clustering | Content, Internals, Platform, UI, OS: Abstract to broad category
Hierarchy | Year → Month → Day → Time
Clustering | Long Term, Short Term Contributors: Duration and productivity based [11]
— | Activities captured for issue resolution progression [6]
— | Datetime
Presentation Outline

1. Research Motivation and Aim
2. Related Work and Research Contributions
3. Research Methodology and Framework
4. Research Methodology and Framework
5. References
OLAP Operations on a Sample 3-D Process Cube

1. SLICE along State

2. DICE along Status and Type

3. ROLL UP along Status

4. COMPARE two Process Cells

Figure 1: Sample 3-D Process Cube to Compare Process Mining Results for Performance and Security Type Issue Resolution
Process Cube Operations

1. **Slice**: Selects single value for one of the dimension, thus, reducing total number of dimensions by one. We slice process cube along *State* dimension and obtain a subcube with value for *State* as closed, that is, only closed issues are considered for analysis.

2. **Dice**: Defines a subcube by performing selection of single or multiple dimensions thereby, reducing number of members along selected dimensions. We obtain a subcube by selecting *Type* as Performance and Security, and *Status* as Fixed, Invalid, Duplicate, WontFix and Verified. All the cases with final status (like External Dependency, FixUnreleased and Icebox grouped as Others for the sake of presentation) apart from these 5 are filtered.
Process Cube Operations

1. **Roll Up**: Summarizes data along a dimension. For example, we roll up and combine all the selected *Closed Status* values (from Step 2). As a result, no partitioning based on final *Closed Status*. This reduces number of cells along the rolled up dimension.

2. **Drill Down**: Represents data at more specialized level of hierarchy with more number of members along the dimension. It is opposite of roll up.
Process model for Performance issues
Process model for Security issues
Percentage of total ITS issues with total PCR issues in lifecycle
Closed status distribution for Performance and Security issues

- Fixed
- WontFix
- Verified
- Duplicate
- Invalid

Performance and Security issues distribution:

- Fixed: 90%
- WontFix: 10%
- Verified: 5%
- Duplicate: 3%
- Invalid: 2%
General Bottleneck

Experimental results demonstrate transitions which take lot of time delaying an individual process. For example, following results are few actionable information for the project team:

1. We notice that the mean time to reopen and fix closed issues ($I_{Closed} \rightarrow Fixed$) is very high, that is, 29.8 weeks for Security issues. At the same time this transition is present in many cases indeed, it emphasizes the need to minimize issue reopen.

2. In case of Security issues, it takes mean time of around 3 months to directly fix an issue that is, transition from $I_{Creation}$ to $Fixed$. On the other hand, sequence of transitions as $I_{Creation} \rightarrow$
General Bottleneck

1. $I_{\text{Open}} \rightarrow Fixed$ takes mean time of around 2 months which indicates that proper assignment of issue before jumping to the resolution reduces overall delay.

2. Reopening of Performance issues, that is, $I_{\text{Closed}} \rightarrow I_{\text{Open}}$ takes mean time of 54.9 days which is undesirable and degrades the overall user perceived quality of the bug fixing process.

3. Resolution of an issue as $WontFix$ or $Invalid$ is taking significantly high mean time of around 21 and 17 weeks respectively for Performance issues. It indicates inefficiency in identifying the issues which are not worth fixing because of several reasons like insufficient information.

4. Control transfer from ITS state $I_{\text{Open}}$ to PCR state $C_{\text{Creation}}$ is taking 11.3 and 15 days for Security and Performance respectively. It indicates scope of improvement in making transition smooth between different IS.
Conclusions

1. We model process cube with 9 dimensions for defect resolution process. We handle sparsity for selected dimension ($D_{sel}$) by meaningful clustering and hierarchies.

2. We perform a set of OLAP operations (slice, dice and roll-up) to eventually compare Performance and Security issues.

3. We obtain process model with 11 states which helps to visualize differences. Empirical analysis reveals interesting patterns like more Performance issues have patches, high percentage of Security issues are Fixed.
Conclusions

1. We identify high number of unique traces for both Types with majority cases having 3 or 4 events in the lifecycle.

2. We find exceptional control transfer in 18% of Performance issues. We observe frequent reopen cases (anti-patterns) for Security issues and presence of code review cycles (signature patterns).
Presentation Outline

1. Research Motivation and Aim
2. Related Work and Research Contributions
3. Research Methodology and Framework
4. Research Methodology and Framework
5. References
References

Wil MP van der Aalst. 
Process cubes: Slicing, dicing, rolling up and drilling down event data for process mining. 

Tatiana Mamaliga. 
*Realizing a Process Cube Allowing for the Comparison of Event Data.* 

Duncan Dubugras A Ruiz, Karin Becker, Taisa C Novello, and Virginia S Cunha. 
A data warehousing environment to monitor metrics in software development processes. 

Methanias Colaço, Manoel Mendonça, and Francisco Rodrigues. 
Data warehousing in an industrial software development environment. 
References II


References III


References IV


References VII

Thomas Zimmermann, Nachiappan Nagappan, Philip J Guo, and Brendan Murphy.
Characterizing and predicting which bugs get reopened.

Studying re-opened bugs in open source software.

Anne Rozinat and Wil MP van der Aalst.
Conformance checking of processes based on monitoring real behavior.

Christine A Halverson, Jason B Ellis, Catalina Danis, and Wendy A Kellogg.
Designing task visualizations to support the coordination of work in software development.
References VIII


