Systematic Mining of Software Repositories

Lecture 5 - Retrospective

Harald Gall

University of Zurich, Switzerland
http://seal.ifi.uzh.ch
@ LASER summer school 2014
2009
Roundtable on the Future of Mining Software Archives
## 2009 Future of Mining Software Repos

Based on a Software Roundtable published in IEEE Software 2009

<table>
<thead>
<tr>
<th>Vision statement</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Answer Commonly Asked Project Questions</td>
<td>partly</td>
</tr>
<tr>
<td><strong>Michael W. Godfrey</strong></td>
<td></td>
</tr>
<tr>
<td>Software Repositories: A Strategic Asset</td>
<td>yes</td>
</tr>
<tr>
<td><strong>Ahmed E. Hassan</strong></td>
<td></td>
</tr>
<tr>
<td>Create Centralized Data Repositories</td>
<td>yes</td>
</tr>
<tr>
<td><strong>James Herbsleb</strong></td>
<td></td>
</tr>
<tr>
<td>Embed Mining in Developer Tools</td>
<td>partly</td>
</tr>
<tr>
<td><strong>Gail C. Murphy</strong></td>
<td></td>
</tr>
<tr>
<td>Help Developers Search for Information</td>
<td>partly</td>
</tr>
<tr>
<td><strong>Martin Robillard</strong></td>
<td></td>
</tr>
<tr>
<td>Deploy Mining to Industry</td>
<td>ongoing</td>
</tr>
<tr>
<td><strong>Audris Mockus</strong></td>
<td></td>
</tr>
<tr>
<td>Let Us Not Mine for Fool’s Gold</td>
<td>ongoing</td>
</tr>
<tr>
<td><strong>David Notkin</strong></td>
<td></td>
</tr>
</tbody>
</table>
2013
MSRconf revisited
A Trend Analysis on Past MSR Papers, by Serge Demeyer et al., MSR 2013

- RQ 1: Which are the popular and outdated research topics? (by text analysis, with n-grams)
- RQ 2: Which are the frequently and less frequently cited cases?
- RQ 3: Which is the popular and emerging mining infrastructure?
- RQ 4: What is the “actionable information” which we are deemed to uncover?
RQ 1: Popularity of topics

Fig. 1: Frequency of typical terms related to software engineering topics. (Note the threshold line at $0.1 \times 10^{-3}$)
RQ 2: Frequently cited cases

Fig. 2: Frequently and less frequently cited cases
RQ 3: SCM’s

Fig. 3: Frequently cited source configuration management systems (SCM)
CfP of MSRconf:

- Goal is “to uncover interesting and actionable information about software systems and projects”
The LNCS book chapter
Revisiting Mining Studies
Katja Kevic (UZH), Stefanie Beyer (AAU), Ilias Rousinopoulos (AAU), Sven Amann (TUD)

- what’s a mining study: setup, resources, machinery, ..
- what sources (archives) can be used for what kind of study (a catalog)
- what questions have been addressed so far
- what questions and conclusions (answers) so far
- which studies can be automated in terms of tooling and infrastructure
- what is a benchmark for mining studies
A retrospective overview of topics addressed in the lectures
The Screening Plant of a SW Miner
Which data sources?

- Evolution analysis data repositories à la **PROMISE**
  - Flossmole, Sourcerer, Ultimate Debian DB
  - Provide benchmark (raw) data
- Interactive online **web platforms** that provide various analyses
  - Boa, FOSSology, Alitheia core, Ohloh
  - Analyses offered by design
  - Data produced is best used within the system
- **Industrial** project data (not widely accessible!)
What kind of studies?

- **Source code**
  - Which entities co-evolve/co-change?
  - How to identify code smells or design disharmonies?

- **Bugs and changes**
  - Who should / how long will it take to fix this bug?
  - When do changes induce fixes?
  - Predicting bugs and their components?

- **Project and process**
  - Do code and comments co-evolve?
  - Who are the experts of a piece of code?
Example: Bug Prediction

- Using Code Churn vs. Fine-Grained Changes
- Predicting the Types of Code Changes
- Predicting the Method
- Using the Gini Coefficient for Bug Prediction
- Using developer networks for Bug Prediction
Learn a prediction model from historic data

Predict defects for the same project

Hundreds of prediction models / learners exist

Models work fairly well with precision and recall of up to 80%.

Performance of bug prediction

<table>
<thead>
<tr>
<th>Predictor</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-Release Bugs</td>
<td>73.80%</td>
<td>62.90%</td>
</tr>
<tr>
<td>Test Coverage</td>
<td>83.80%</td>
<td>54.40%</td>
</tr>
<tr>
<td>Dependencies</td>
<td>74.40%</td>
<td>69.90%</td>
</tr>
<tr>
<td>Code Complexity</td>
<td>79.30%</td>
<td>66.00%</td>
</tr>
<tr>
<td>Code Churn</td>
<td>78.60%</td>
<td>79.90%</td>
</tr>
<tr>
<td>Org. Structure</td>
<td>86.20%</td>
<td>84.00%</td>
</tr>
</tbody>
</table>

Example: Code Ownership

C. Bird, N. Nagappan, B. Murphy, H. Gall, P Devanbu, Don't touch my code! Examining the effects of ownership on software quality, ESEC/FSE '11
Performance/Time variance

Workflows & Mashups
Conclusions

- Bug predictions do work
- Cross-project predictions do not really work
- Data sets (systems) need to be “harmonized”
- Data preprocessing and learners need to be calibrated
- Studies need to be replicable (systematically)
- Periods of stability vs. drift