TOWARDS AN OPEN-SOURCE TOOL FOR MEASURING AND VISUALIZING THE INTEREST OF TECHNICAL DEBT

Davide Falessi
California Polytechnic State University
San Louis Obispo, CA, USA
dfalessi@calpoly.edu

Andreas Reichel
University for Applied Science Mannheim
Mannheim, Germany
andreas.reichel@stud.hs-Mannheim.de
BASICS – TECHNICAL DEBT

Customer Satisfaction  Market Opportunities

Code Quality  Maintainability
SonarQube helps Project Managers make decisions based on knowledge of Technical Debt

Source: nemo.sonarqube.org Project: JDK7
Basics - Quality Rule Violation

- Quality Rule = empirically validated software engineering principle (e.g., avoiding code clones, high comments frequency, intuitive variable naming, etc.)
LIMITATION OF SONARQUBE

- SonarQube only reports the principal part of Technical Debt

What is the consequence of not spending 4,048 days?
MOTIVATION

How many extra defects occurred, or will occur, due to quality rule violations? → Interest
First tool to quantify and visualize the interest of Technical Debt
MIND

- SonarQube Plugin
- Apache licensed
- Open Source (Hosted at SourceForge.net)
- Supports all programming languages supported by SonarQube
- Written in Java (7K LOC)
MIND

- MIND assumptions and pre-conditions
  - High Linkage between commits and tickets
  - High number of releases
  - High variation in rule violations
  - Git and Redmine
MIND – How it Works
Preliminary Validation

- Is code with more rule violations also more defect prone?
  - Validating and Prioritizing Quality Rules for Managing Technical Debt: An Industrial Case Study (Davide Falessi, Alexander Voegele)
MEASUREMENT APPROACH

- Violation density = Violations / Size in LOC
- Defect proneness = Defects / LOC touched (Defect injection frequency)

Chart showing releases and metrics:
- Violations\textsubscript{i}
- LOC touched\textsubscript{i}
- Size\textsubscript{i}
- Defects\textsubscript{i}

Timeline:
- Release i-1
- Release i
- Release i+1

Time
Validating and Prioritizing Quality Rules for Managing Technical Debt: An Industrial Case Study (Davide Falessi, Alexander Voegele)
CALCULATING THE INTEREST

$$EDP_{ij} = DP_{ij} - DP_{ij}^*$$

Equation 3: Definition of Extra Defect Proneness (EDP) for a specific class (j) in a specific release (i).

- Defect Proneness without Violations has to be estimated.
We use Linear Regression as prediction model

- Standard reliable model for estimating variables

\[ DP_{i,j^*} = f(VD_{i,j^*}) \]

Equation 10: Prediction formula to compute the defect proneness of a class without violation\((j^*)\) in a release \((i)\).
## Training Table for Prediction Model (Sanitized Data)

<table>
<thead>
<tr>
<th>Class ID</th>
<th>Version</th>
<th>Violations Density</th>
<th>Defect Proneness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 1</td>
<td>v1.0</td>
<td>0.052</td>
<td>0.019</td>
</tr>
<tr>
<td>Class 2</td>
<td>v1.0</td>
<td>0.164</td>
<td>0.423</td>
</tr>
<tr>
<td>Class 3</td>
<td>v1.0</td>
<td>0.252</td>
<td>0.238</td>
</tr>
<tr>
<td>Class 4</td>
<td>v1.0</td>
<td>0.013</td>
<td>0.795</td>
</tr>
<tr>
<td>Class 5</td>
<td>v1.0</td>
<td>0.333</td>
<td>0.763</td>
</tr>
<tr>
<td>Class 6</td>
<td>v1.0</td>
<td>0.410</td>
<td>0.760</td>
</tr>
<tr>
<td>Class 1</td>
<td>v1.1</td>
<td>0.778</td>
<td>0.717</td>
</tr>
<tr>
<td>Class 2</td>
<td>v1.1</td>
<td>0.177</td>
<td>0.663</td>
</tr>
<tr>
<td>Class 3</td>
<td>v1.1</td>
<td>0.429</td>
<td>0.307</td>
</tr>
<tr>
<td>Class 4</td>
<td>v1.1</td>
<td>0.853</td>
<td>0.057</td>
</tr>
<tr>
<td>Class 5</td>
<td>v1.1</td>
<td>0.868</td>
<td>0.451</td>
</tr>
<tr>
<td>Class 6</td>
<td>v1.1</td>
<td>0.166</td>
<td>0.774</td>
</tr>
<tr>
<td>Class 1</td>
<td>v1.2</td>
<td>0.232</td>
<td>0.736</td>
</tr>
<tr>
<td>Class 2</td>
<td>v1.2</td>
<td>0.260</td>
<td>0.271</td>
</tr>
<tr>
<td>Class 3</td>
<td>v1.2</td>
<td>0.439</td>
<td>0.229</td>
</tr>
<tr>
<td>Class 4</td>
<td>v1.2</td>
<td>0.051</td>
<td>0.853</td>
</tr>
<tr>
<td>Class 5</td>
<td>v1.2</td>
<td>0.174</td>
<td>0.370</td>
</tr>
<tr>
<td>Class 6</td>
<td>v1.2</td>
<td>0.426</td>
<td>0.548</td>
</tr>
</tbody>
</table>
MIND – User Stories

“As a developer I want to know the interest I will have if I will not apply any refactoring so that I can decide if refactoring activities are worthwhile.”

<table>
<thead>
<tr>
<th>MIND (Managing Technical Debt)</th>
<th>Extra Defect Proneness</th>
<th>Defect Proneness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relative</td>
<td>Maximum per 100LOC</td>
<td>Maximum per 100LOC</td>
</tr>
<tr>
<td>3.6%</td>
<td>0.1</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Quality Of Data

<table>
<thead>
<tr>
<th>Linkage</th>
<th>Estimation Error (MAE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>72.4%</td>
<td>0.0</td>
</tr>
</tbody>
</table>
MIND – User Stories

“As a developer, I want to know the interest I paid, whether increased or decreased over time, so that I can do root-cause analysis.”
CONCLUSION

- First tool for quantification and visualization of the interest of Technical Debt
- MIND computes number of extra defects due to rule violations
  - Using historical Data from Redmine and Git
- The User of MIND is supported in understanding how much of the principal of Technical Debt he should reduce
**Future Questions**

- How to quantify the interest?
  - i.e. Decreased speed of development
THANK YOU VERY MUCH

- MIND deliverables
  - Instruction and Demo: http://goo.gl/Ydjtkq
  - Installation files: http://goo.gl/VvDv75
  - Redmine repository: http://goo.gl/4r7Y1a
  - Source files (Git): http://goo.gl/TWRnat