Towards identifying software project clusters with regard to defect prediction

Marian Jureczko, Wrocław University of Technology
Lech Madeyski, Wrocław University of Technology
Agenda

- Introduction
- Data acquisition
- Study design
- Results
- Conclusions
Introduction
Motivation - Why defect prediction?

20% of classes contain 80% of defects

We can use the software metrics to predict error prone classes and therefore prioritize and optimize tests.
Motivation - Why clustering projects?

• Defect prediction is sometime impossible because lack of training data:
  – It may be the first release of a project
  – The company or the project may be too small to afford collecting training data

• With well defined project clusters the cross-project defect prediction will be possible
Definitions

- **Defect**
  - Interpreted as a defect in the investigated project
  - Commented in the version control system (CVS or SVN)

- **Defect prediction model**

Values of Metrics for a given java class

- WMC = ...
- DIT = ...
- NOC = ...
- CBO = ...
- RFC = ...
- LCOM = ...
  - Ca=...
  - ....
Data acquisition

- 19 different metrics were calculated with the CKJM tool ([http://gromit.iiar.pwr.wroc.pl/p_inf/ckjm](http://gromit.iiar.pwr.wroc.pl/p_inf/ckjm))
  - Chidamber & Kemerer metrics suite
  - QMOOD metrics suite
  - Tang, Kao and Chen’s metrics (C&K quality oriented extension)
  - Cyclomatic Complexity, LCOM3, Ca, Ce and LOC

- Defects were collected with BugInfo ([http://kenai.com/projects/buginfo](http://kenai.com/projects/buginfo))
Data acquisition

• 92 versions of 38 projects were analysed
  – 6 proprietary projects (5 custom build solutions from insurance domain, 1 quality assurance tool)
  – 17 academic projects

• Metrics Repository (http://purl.org/MarianJureczko/MetricsRepo)
Correlation between the number of defects and the values of metrics were calculated.

- Bug & metrics repository
  - Correlation vectors
    - K-Means
      - Clusters
        1st of 2
        2nd of 2
        proprietary A
        proprietary B
        proprietary open
        open
      - Kohonen neural network
Study design - verification of the cluster existence

Training set: C

Model: M_C

Model evaluation:
- \( E(M_C, r: r \in C) \)

- If \( E(M_C, r: r \in C) \) is better than \( E(M_{All}, r: r \in C) \), then C exists.
- If \( E(M_C, r: r \in C) \) is not better than \( E(M_{All}, r: r \in C) \), then C does not exist.

Tests:
- Shapiro-Wilk tests
- Levene's test
- Homogeneity of variance
- T-test
- Wilcoxon matched pairs test
## Results

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Is the cluster model better?</th>
<th>P value (statistical test)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st of 2</td>
<td>YES</td>
<td>0.954</td>
</tr>
<tr>
<td>2nd of 2</td>
<td>NO</td>
<td>-</td>
</tr>
<tr>
<td>proprietary A</td>
<td>NO</td>
<td>-</td>
</tr>
<tr>
<td>proprietary B</td>
<td>YES</td>
<td>0.035</td>
</tr>
<tr>
<td>proprietary / open</td>
<td>YES</td>
<td>0.005</td>
</tr>
<tr>
<td>open-source</td>
<td>NO</td>
<td>-</td>
</tr>
</tbody>
</table>
Results

- Cluster ‘Proprietary B’
  - custom build solutions;
  - heavy weight, plan driven development process;
  - already installed in the customer environment;
  - insurance domain;
  - manual tests;
  - similar development period;
  - use database;
  - proprietary - the same company.

- Cluster ‘proprietary / open’
  - text processing domain;
  - SVN and Jira or Bugzilla used;
  - medium size international team;
  - automatization in the testing process;
  - do not use database
Conclusions

- 92 releases of 38 proprietary, open-source and academic projects were analysed
- 2 methods of clustering were applied
- 6 clusters were identified and the existence of 2 of them were proven
Thank You for Your attention