PREDICTING SOFTWARE CHANGE COUPLING

Robert M. Dondero and Gregory W . Hislop

The iSchool at Drexel
College of Information Science and Technology

INTRODUCTION

Two source code files are **change coupled** if programmers change them together frequently. Excessive change coupling can be a **software maintenance problem**: a programmer might introduce bugs by changing some but not all files of a change coupled set. How can we **predict future change coupling**?

RESEARCH QUESTION

We considered three prediction approaches:

- **Mining of change logs**, i.e., analysis of past change coupling.
- **Analysis of software similarity**, i.e., presence of software clones in source code files.
- **Analysis of software proximity**, i.e., presence of references among the code in source code files.

Which approach best predicts change coupling?

MATERIALS

We used:

- Four large open source code **databases** (Ant, Struts, Tomcat, Xerces) containing many Java source code files and change logs over long periods of time.
- A **Miner** tool, created for the project.
- Three pre-existing **Similarity Detectors** (Duplo, CCFinderX, CPD).
- Two **Proximity Detectors**, created for the project.

DATA COLLECTION

For each database, we:

- Split the database's change log at the **one-half** point.
- Used the Miner and the change log **after** the split to generate a **Reference Set**: file pairs in descending order by amount of change coupling.
- Used the Miner and the change log **before** the split to generate a **Mining Prediction Set**: file pairs in descending order by amount of change coupling.

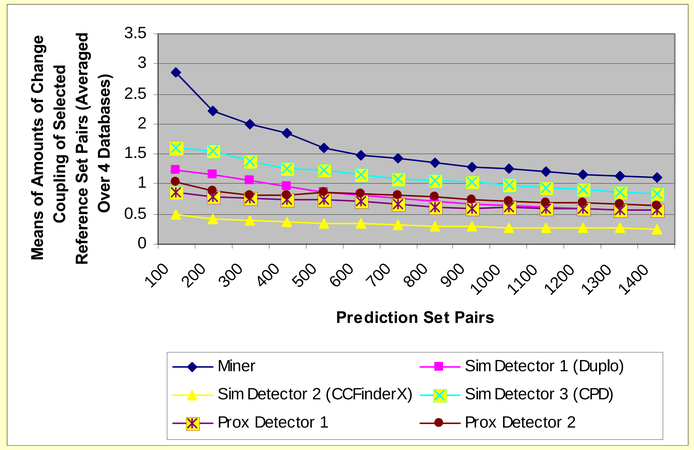
DATA COLLECTION (CONT.)

- Used the Similarity Detectors and the files at the time of the split to generate **Similarity Prediction Sets**: file pairs in descending order by degree of similarity.
- Used the Proximity Detectors and the files at the time of the split to generate **Proximity Prediction Sets**: file pairs in descending order by degree of proximity.

DATA ANALYSIS: PRECISION

For each database and each Prediction Set, we mapped the first x pairs (for x = 100, 200, é , 1400) of the Prediction Set into the Reference Set, thus choosing some Reference Set pairs. We then determined the **average amount of change coupling** of those selected pairs.

RESULTS: PRECISION

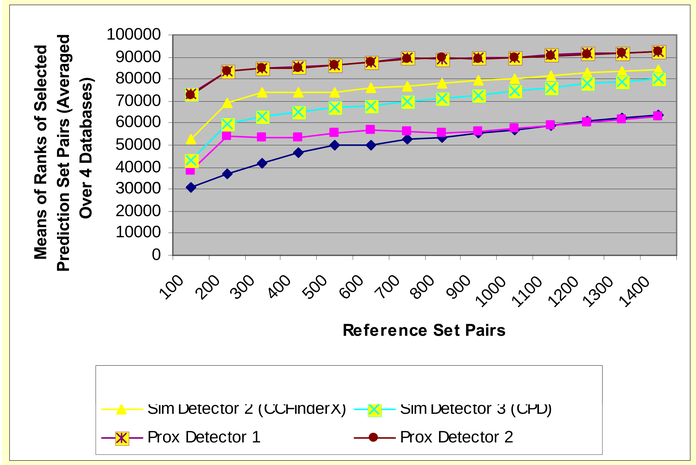


Note that lines **higher** in the graph indicate better performance. A subsequent ANOVA confirmed that the Miner had significantly better precision than CPD (the best Similarity Detector), and CPD had significantly better precision than the Proximity Detectors. Analyses at the **one-quarter** and **three-quarter** points in the change logs yielded similar results.

DATA ANALYSIS: RECALL

For each database and each Prediction Set, we mapped the first x pairs (for x = 100, 200, é , 1400) of the Reference Set into the Prediction Set, thus choosing some Prediction Set pairs. We then determined the **average rank** of those selected pairs.

RESULTS: RECALL



Note that lines **lower** in the graph indicate better performance. A subsequent ANOVA confirmed that the Miner had ñon-significantly betterò recall than Duplo (the best Similarity Detector), and Duplo had significantly better recall than the Proximity Detectors. Analyses at the **one-quarter** and **three-quarter** points in the change logs yielded similar results.

CONCLUSIONS

Which approach best predicted change coupling?

- **Concerning precision: Mining of change logs** had significantly better precision than **analysis of software similarity**, which had significantly better precision than **analysis of software proximity**.
- **Concerning recall: Mining of change logs** had ñon-significantly betterò recall than **analysis of software similarity**, which had significantly better recall than **analysis of software proximity**.