

An Ethnographic Study of Copy and Paste Programming Practice in OOPL



Miryung Kim



Lawrence Bergman



Tessa Lau



David Notkin

プログラマのコピペ作業を観察し，コピペの使い方のパターンを紹介し，考察する

Introduction

- コードのコピペは様々な場面で使われる
 - 自身のコードや他人のコードのコピペ
- 一般的にコピペ作業は悪い作業方法と言われている
- プログラムの行動を記録し，どのようにコピペを行っているかを調べる
 - コピペのパターンを調査する
 - 将来的にツールを作りコピペ作業をサポート

Ethnographic Study : Observation

- IBM T.J.Watson研究所のチームに対して2つの調査をおこなう

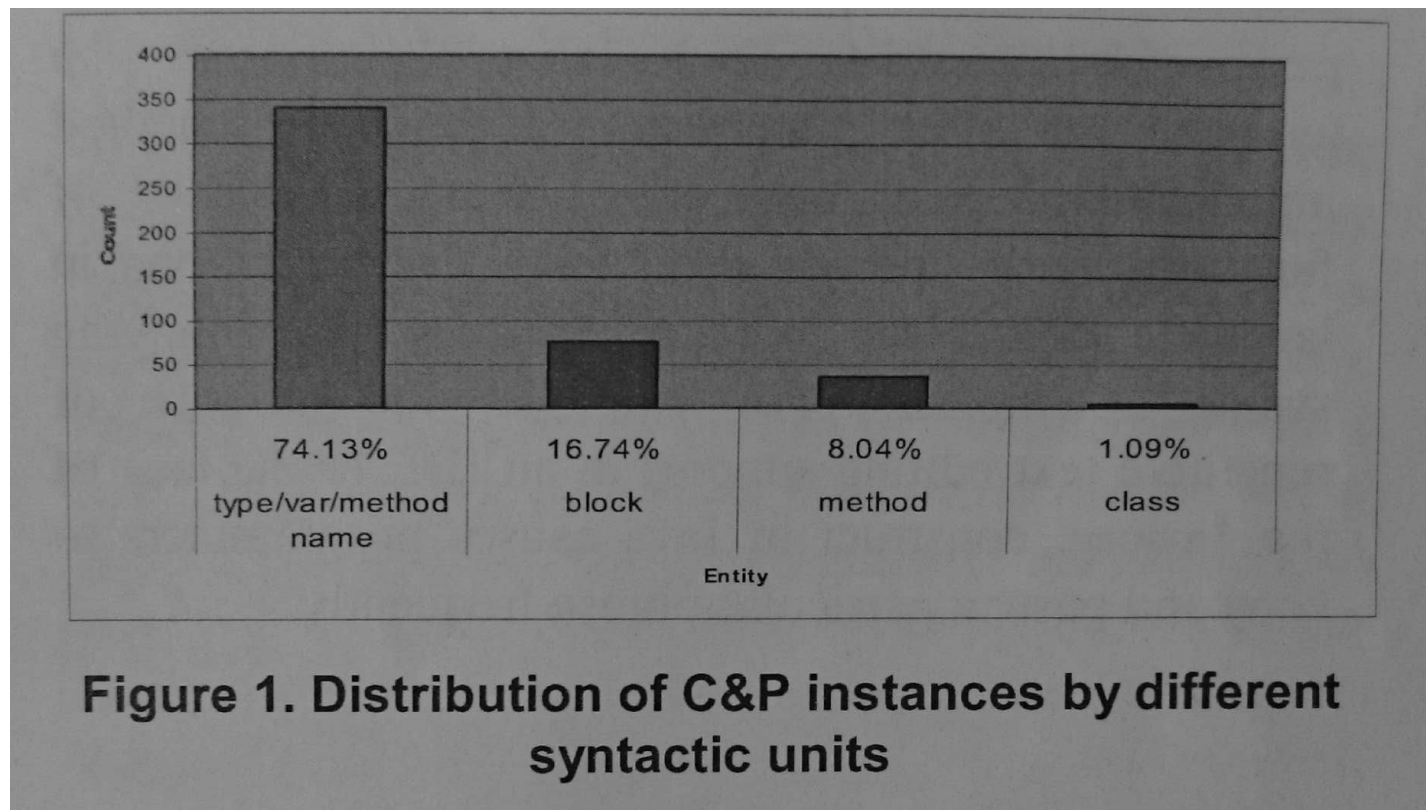
Table 1. Observational study setting

	Direct Observation	Observation using a logger and a replayer.
Subjects	Researchers at IBM T. J. Watson	
Number of Subjects	4	5
Total Coding Hours	About 10 hrs	About 50 hrs.
Interviews	Questions asked during observation	Twice after analysis (30 mins ~1 hour/each)
Programming Languages	Java, C++, and Jython	Java

- プログラミングの様子を後ろから見る
- ツールによって作業をモニタリング

Ethnographic Study : Statistics

- 合計で460のコピペ作業が確認された
 - 1時間あたりの平均は16, 中央値は12



Ethnographic Study : Analysis

- 3つの観点から分析を行う
 - Intention View
 - プログラマがどのような意図をもってコピーをおこなうのか
 - Design View
 - 設計に合わせるためにコピーを行うのではないか
 - Maintenance View
 - 重複コードをメンテナンスにおいてどのような影響を与えるのか

Intention View

- リファクタリング
 - 場所移動・まとめなおす
- 構造的に似ているものを再利用
 - 例：ループの構造

```
For (Iterator it=messages.iterator(); it.hasNext();) {  
    Message curr= (Message) it.next();  
    IFile markFile=  
    WorkspaceUtils.getFile(curr.getFirstLocation().getClassName());  
    .....}  
for (Iterator it=messages.iterator(); it.hasNext();) {  
    Message curr= (Message) it.next();  
    MessageLocation loc = curr.getLastLocation();  
    IFile markFile=  
    WorkspaceUtils.getFile(loc.getFirstLocation().getClassName());  
    .....}
```

Figure 4. Semantic template: Copying a loop construct and modifying the inner logic

Design View

- どのような設計に基づいてコピペをおこなっているのか
 - 一貫したコードが求められる場合
 - 例：log関連のコード

```
if (logAllOperations) {  
    try {  
        PrintWriter w = getOutput();  
        w.write("$$$$$");  
        ..  
    } catch (IOException e) {  
    }  
}
```

Figure 5. Duplicated code: logging concern

Maintenance View

- 重複コードがどんなメンテナンスタスクをもたらすのか
 - 短期的に発生するタスク
 - コピペによる，命名の重複
 - 不適切な場所へのコピペをおこなっている
 - 長期的に発生するタスク
 - コードの再構成（リファクタリング）が必要
 - 同時変更が必要となるときがある

Insights

- コードの重複を避けるのにも限界がある
- プログラマはコードを作り直すことを決定する際、自身のコピペ履歴を使う
- コピペの依存関係はメンテナンスするうえで価値がある
- プログラマが大きなコード片をコピペする際には構造的な再利用を意図している

Proposed Tools

- コピペの可視化
 - コピーされた箇所, ペーストされた箇所
- 構造的なテンプレートを抜き出す
- コピペに対する警告や注意を喚起する
- 具体的なリファクタリングを提案する

Threats to Validity

- 今回の調査は OOPL(Object Oriented Programing Languages)を対象としている
- 対象とした開発者が，小さなチームに所属しているので，大規模開発にも当てはまるとは言えない

Conclusion

- 開発状況をツールを使い記録した
- 本研究によってコピペパターンとコピペを行う意図を体系的に示した
- 調査を基にした支援ツールを提案した

所感

- 具体的にどのような理由でコピペをしているかを知ることができた
- コピペはよく使われる作業方法かそうではないかはあまりわからなかった
 - 統計量の説明がよくわからなかった