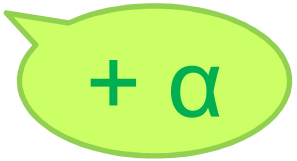
コードクローンをテンプレートとして用いて識別子のバリエーションを求める手法の提案



神谷年洋

産業技術総合研究所 サービス工学研究センター

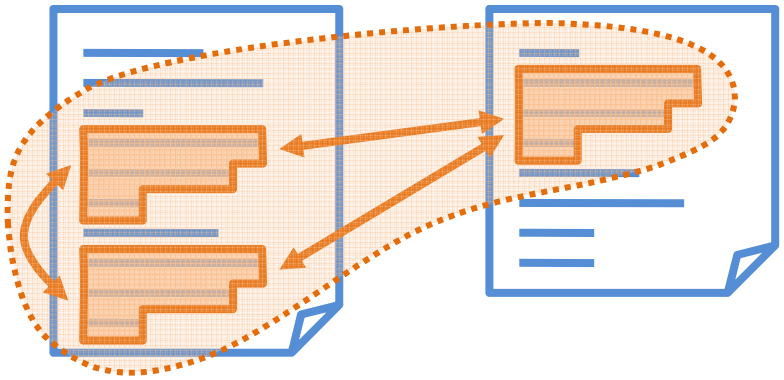
最適化研究チーム

1. コードクローンとは
2. コードクローン検出ツールCCFInderXにおける「類似」
3. アイデア
4. 応用
5. ツールVaci（仮称）
6. ケーススタディ
7. 関連研究
8. まとめと課題

1. コードクローンとは

- コードクローンとは、ソースコード中の同一または類似した部分
  - 典型的には、開発者がソースコードをコピー＆ペーストすることによって作られる
- 保守性を下げる原因になると言われている
- 特に大規模なソフトウェアのソースコードでは問題

- 類似するコード断片のペアをクローンペア
- クローンペアの推移閉包をクローンクラス



コードクローン検出手法

- 何らかの意味・手段で「類似」しているコード断片を見つける
  - 文字列としてみたときの類似(n-gram)
  - 構文の類似
  - 識別子の類似
  - ASTの類似（同一の部分木）
  - 特徴メトリクスの類似（ファンイン・ファンアウト、再クロマチック数など）

2. CCFinderXにおける「類似」

- コードクローン検出ツールCCFinderX
- 構文の類似＋識別子の一致（P-match）

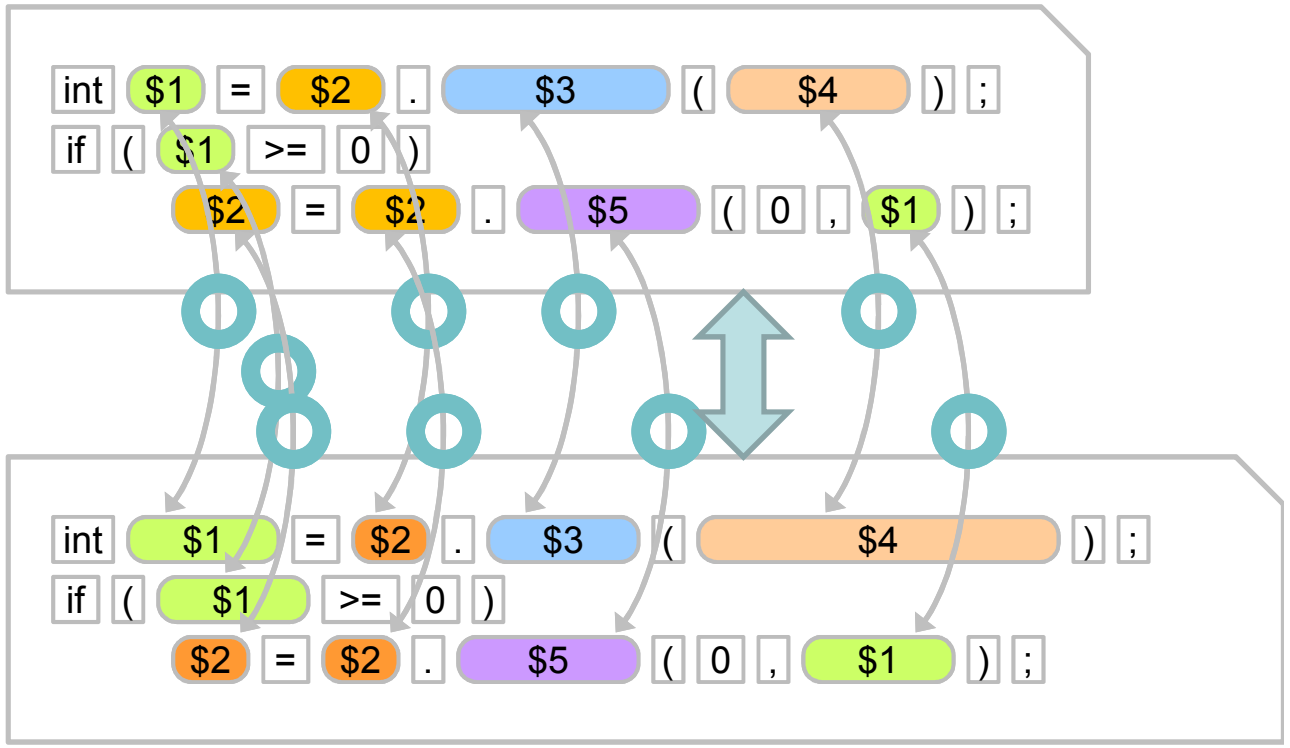
```
int pos = fpath.lastIndexOf(extension);
if (pos >= 0) {
        fpath = fpath.substring(0, pos);
}
```

類似

```
int endPos = line.indexOf(commentStartMark);
if (endPos >= 0)
        line = line.substring(0, endPos);
```

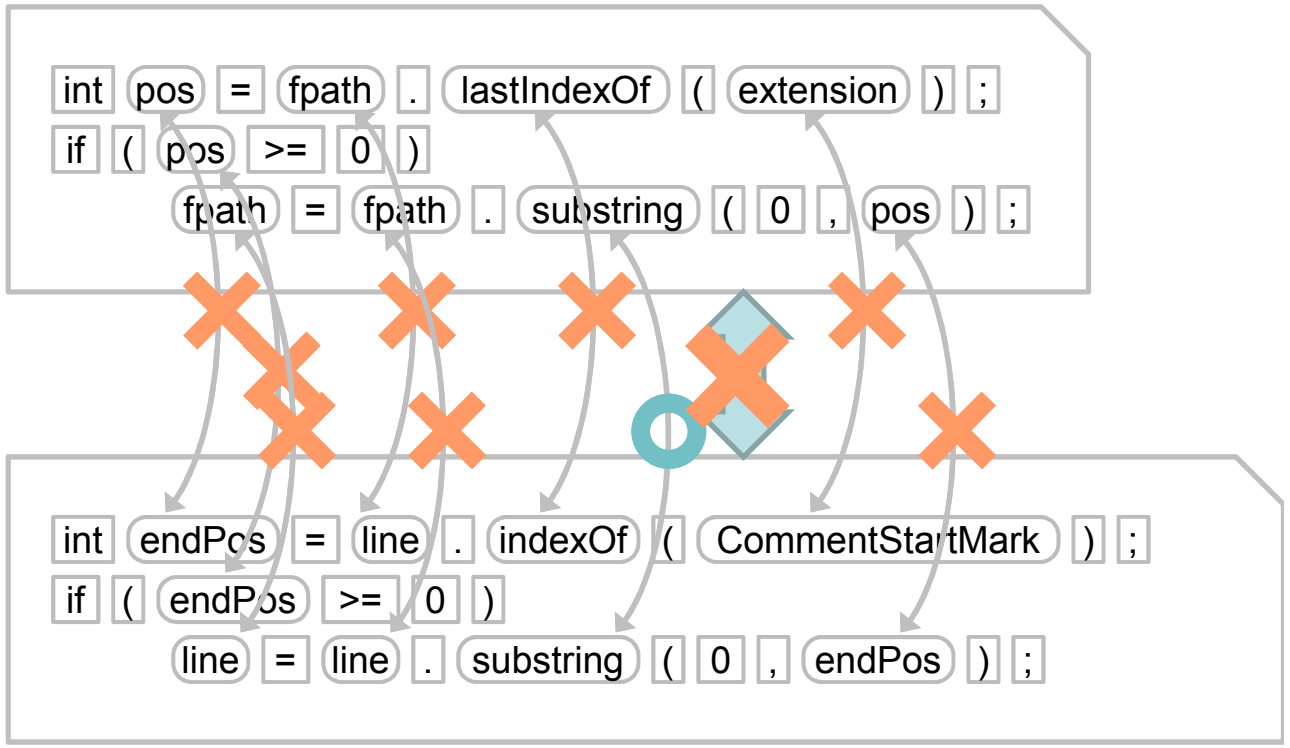
P-match

- 識別子の種類を表すパラメータに置換する
- パラメータが出現するたびに比較



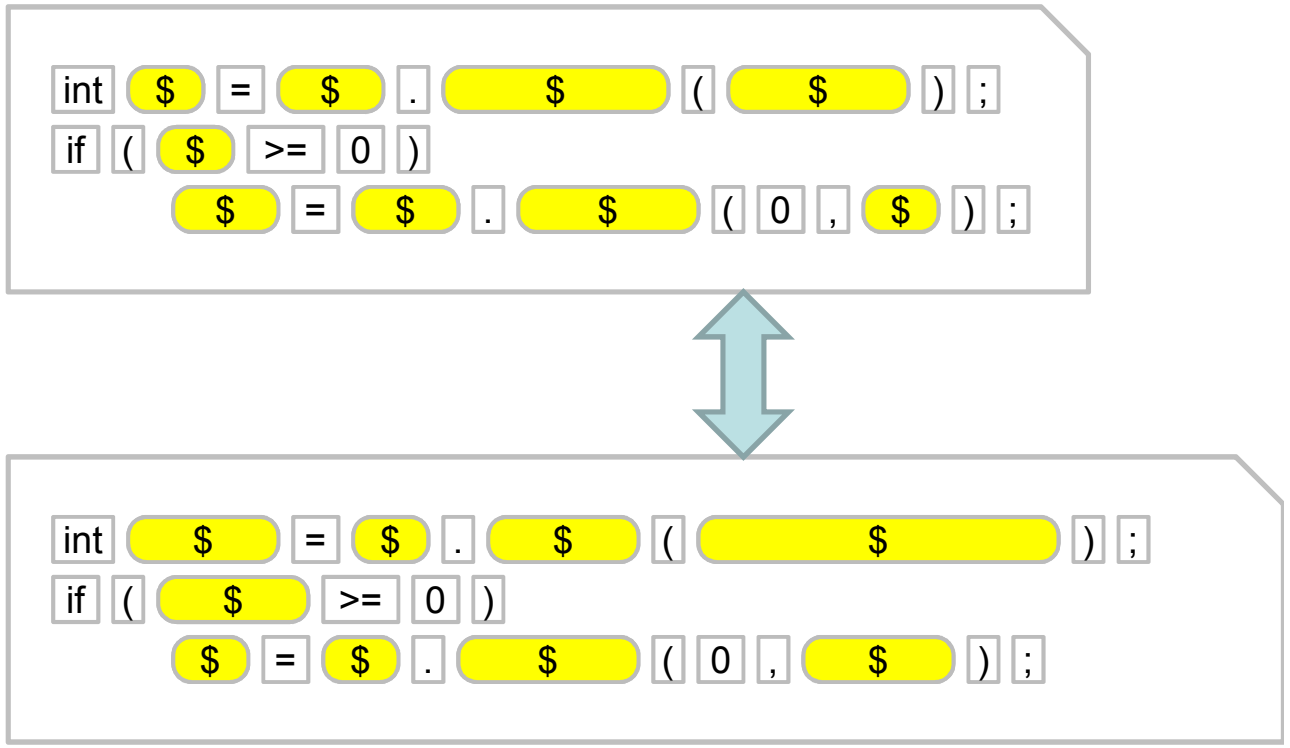
完全一致

- 識別子が出現するたびに比較
- 文字列として同じか



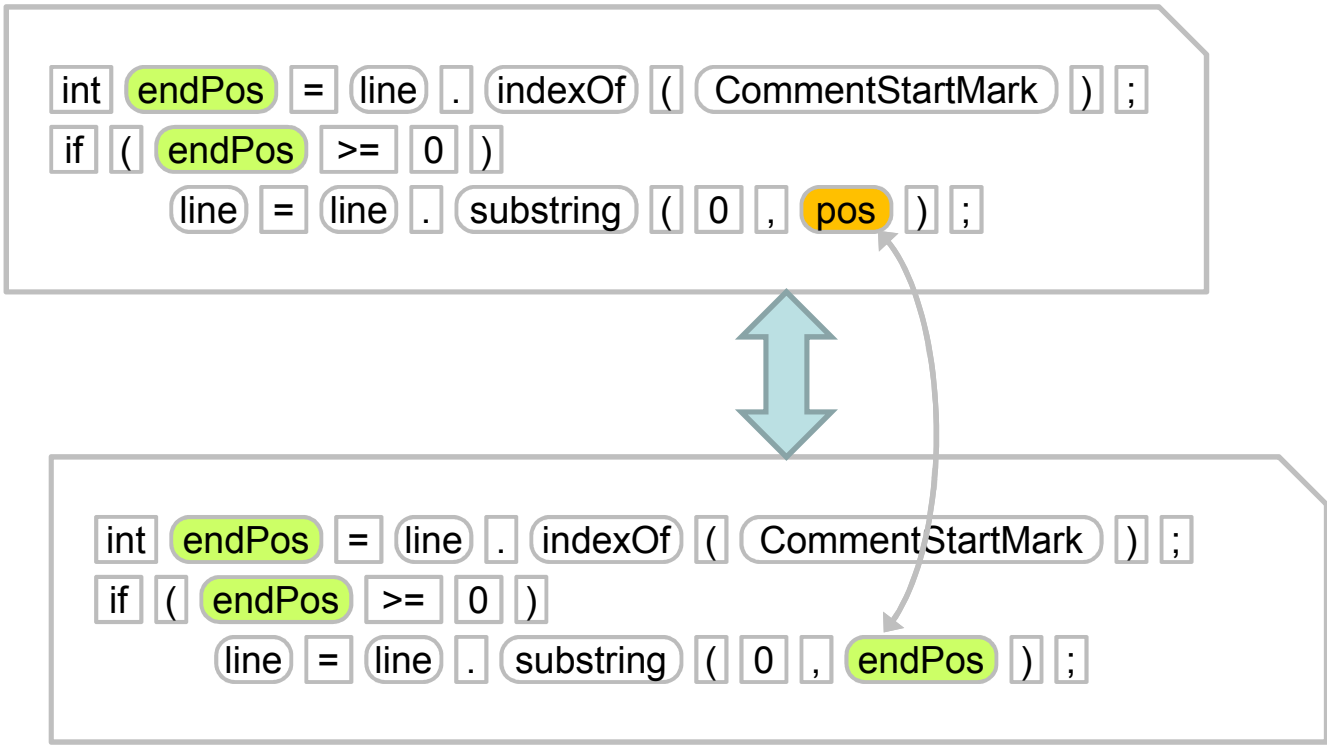
単純なパラメータ化

- すべての識別子の単一のパラメータに置換する



単純なパラメータ化

- ・・・一貫性のない書き換えでも一致と見なされる



3. アイデア

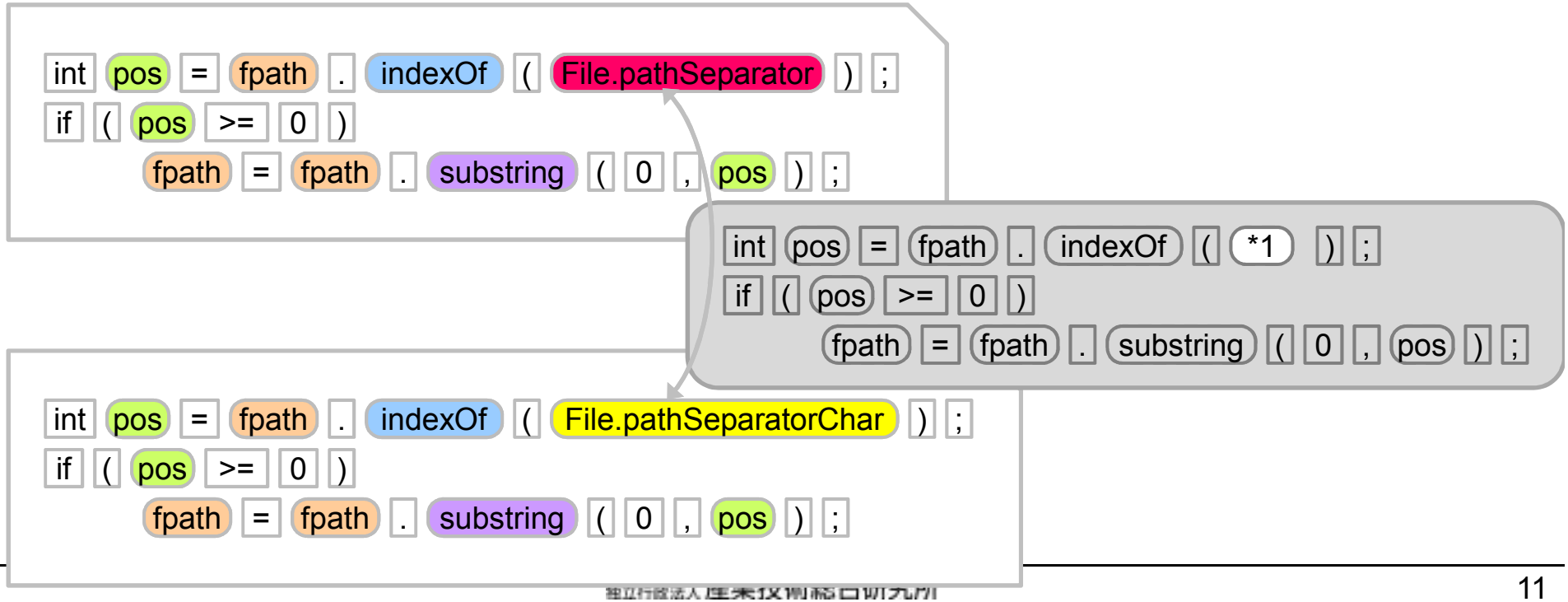
- 識別子がほぼ同じで、ちょっとだけ違っているようなコードクローンから同義語のリストを作る
- 同義語になっている識別子は同じであるとみなすようなマッチングはP-matchより精度が良くなるのでは？

道具立て: テンプレート

- P-matchのコードクローンのコード断片の

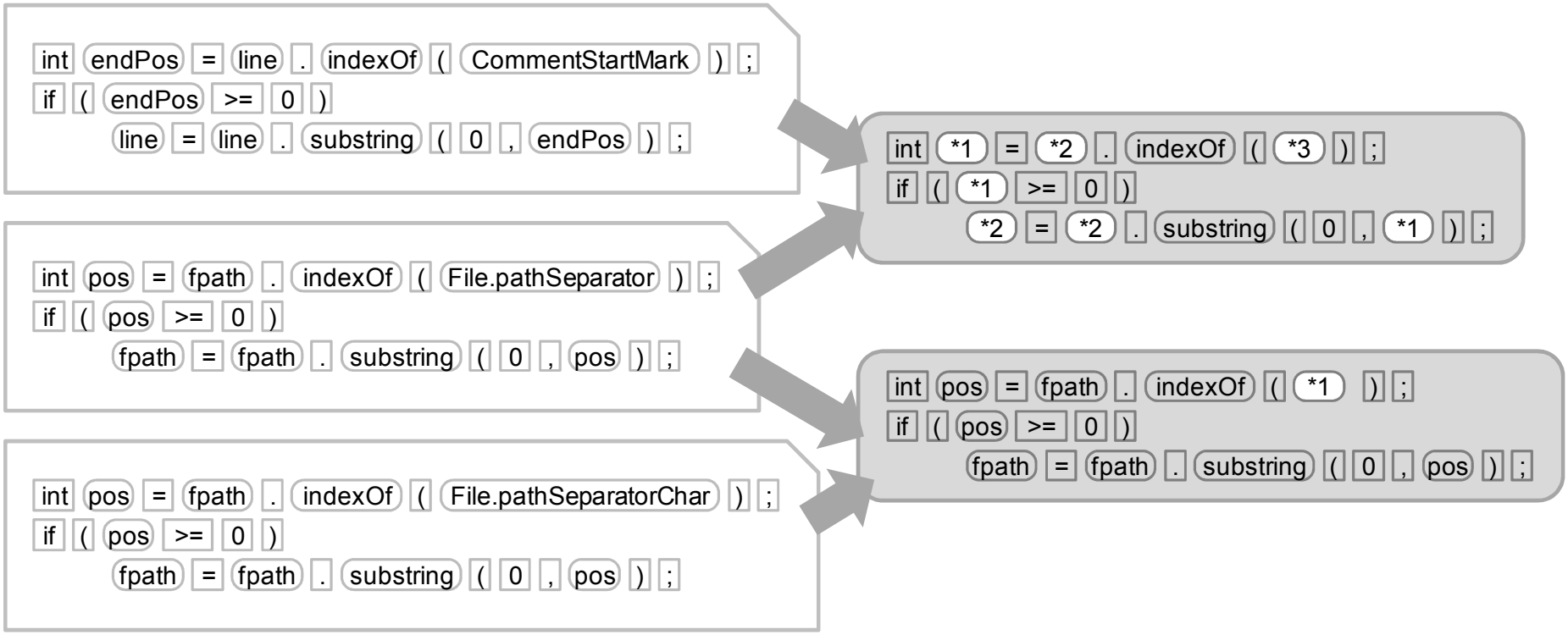
同じ識別子 → 共通部分

異なる識別子 → バリエーション部分 → 同義語



バリエーション部分の「大きさ」

- 同義語っぽさを評価するための尺度として使えるかも（今回は大きさ1のもののみ利用）

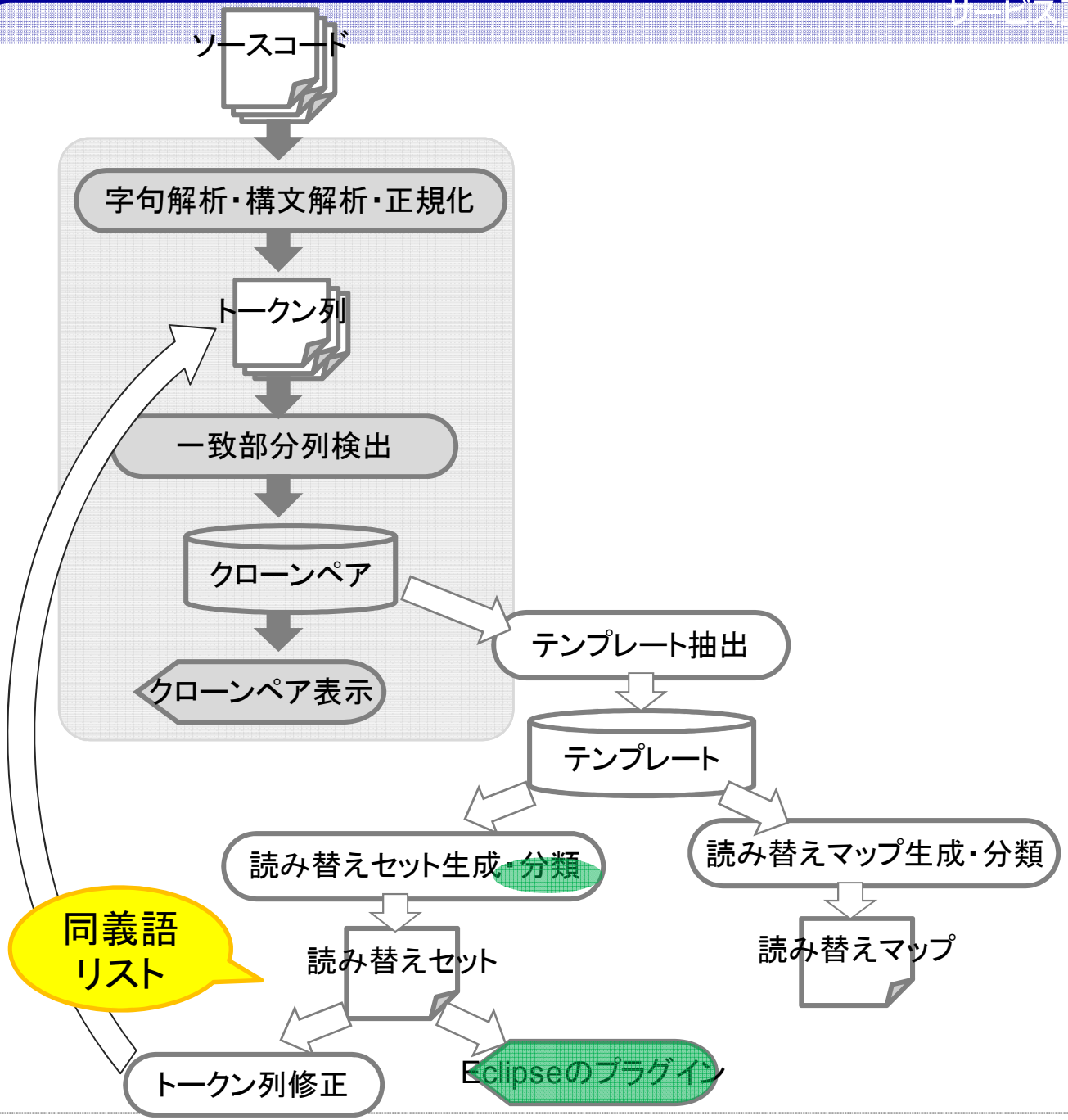


4. 応用

- 単一のプロダクトに適用して
  - 同義語のリストにより名前の揺れを調べる(#1)
  - (前述)同義語のリストを用いることで、コードクローンの検出精度を改善する(#2)
- 単一のプロダクトの複数のバージョンに適用して
  - バージョン間の名前の変遷を調べる(#3)
- 複数のプロダクトに適用して・・・
  - → 今回ははっきりとした結果は出ず

5.ツールVaci(仮称)

- Variation Analyzer of Context-sharing Identifiers
  - 投稿時のIVACaT(Identifier-Variation Analyzer with Clone as Template)から改名、機能強化
- 機能
  - テンプレートを抽出
  - テンプレートを解析
    - n-gramとウォード法によるクラスタリング
    - GUI, Eclipseのプラグイン
  - コードクローンを再検出


ソースコード
字句解析・構文解析・正規化
トークン列
一致部分列検出
クローンペア
クローンペア表示
テンプレート抽出
テンプレート
読み替えセット生成・分類
読み替えマップ生成・分類
同義語
リスト
読み替えセット
読み替えマップ
トークン列修正
Eclipseのプラグイン

6. ケーススタディ

- 応用#1, #2, #3のそれぞれに対応するケーススタディを行った

- （以降で示すデータは、投稿後に修正したツールによるもの）

ケーススタディ#1: 名前の揺れを調べる

ステップ

1. 読み替えクラス(同義語リスト)を作る
   - バリエーション部分の識別子
2. クラスを分類する
   - 名前空間の違い、数字の違い、短い名前、共通の接頭辞、共通の接尾辞、大小文字および単複の違い、短縮形を含むもの、少数派を含むもの、その他

- OpenOffice 2.2.1のソースコードに適用した

結果

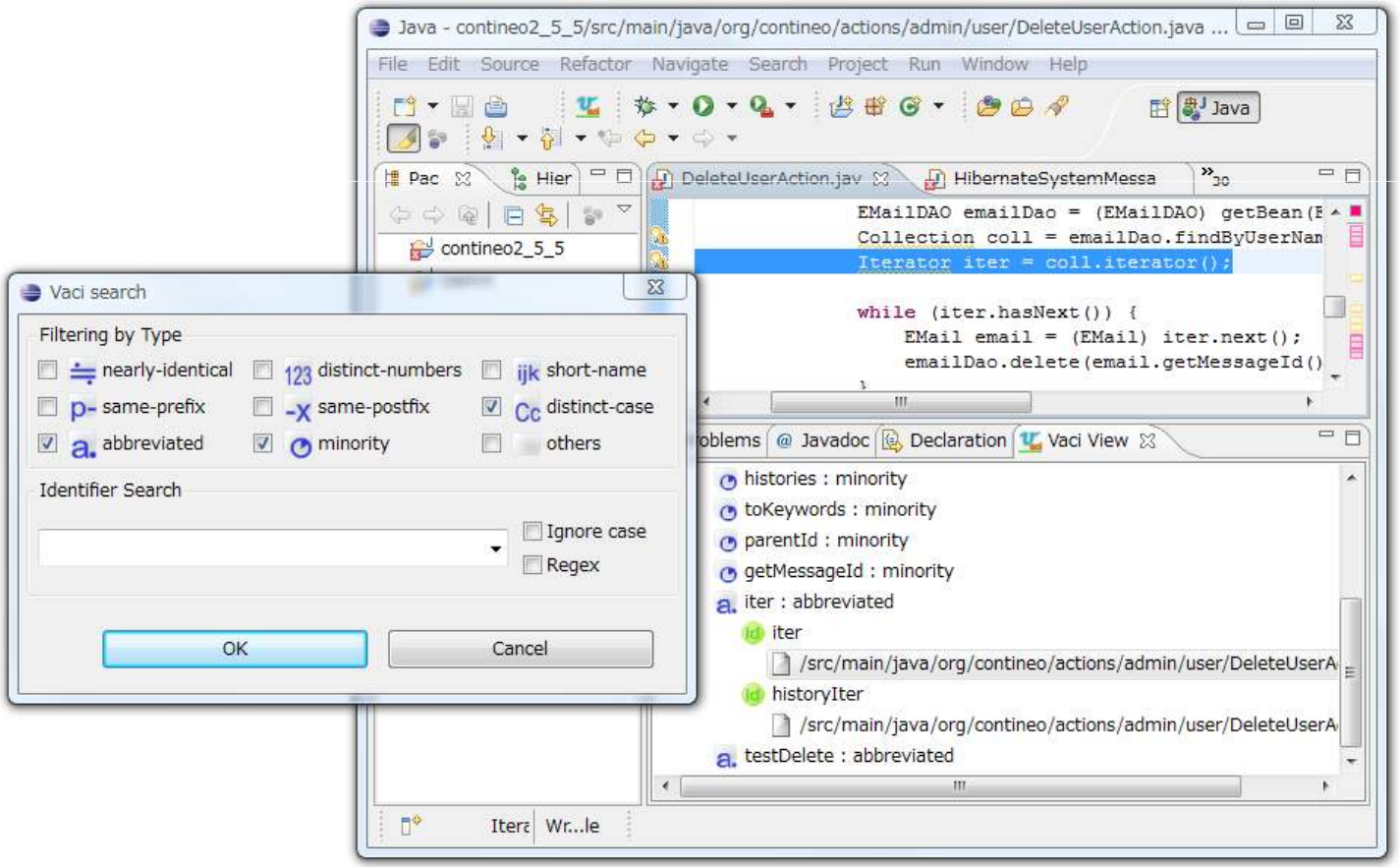
分類	検出例	クラス数	識別子数
名前空間の違い	{ findPos, detail::findPos, ::gdetail::findPos ::binfilter::internal::findPos	3193	9191
数字の違い	{ append_006_Int32_Boun append_007_Int64_Bounde }	540	1594
短い名前	{ PosSize::X, PosSize::Y, PosSize::WIDTH, PosSize::HEIGHT }	25	15953
共通の接頭辞	{ Add_Constant, Add_Function, Add_Property }	7030	22847
共通の接尾辞	{ MoveHdl, MenuSelectHdl, AddCommandsHdl, ToolbarSelectHdl }	3510	9981
大小文字・単複の違い	{ OTYPECOLLECTION, OTypeCollection, cppu::OTypeCollection, ::cppu::OTypeCollection }		
短縮形を含む	{ GetForbiddenCharacterTbl, getForbiddenCharacterTable }		
少数派を含む	{ CFormatEtc, XModifyListener_t, XListEntryListener_t, XValidityConstraintListener_t }	348	4135
その他		4827	17307

分類「共通の接頭辞」に属するものが最多

名前の揺れと見なせるものが多く含まれる

Vaciプラグイン

- Eclipseのプラグイン
- 分類、検索



ケーススタディ#2: コードクローン検出精度を改善する

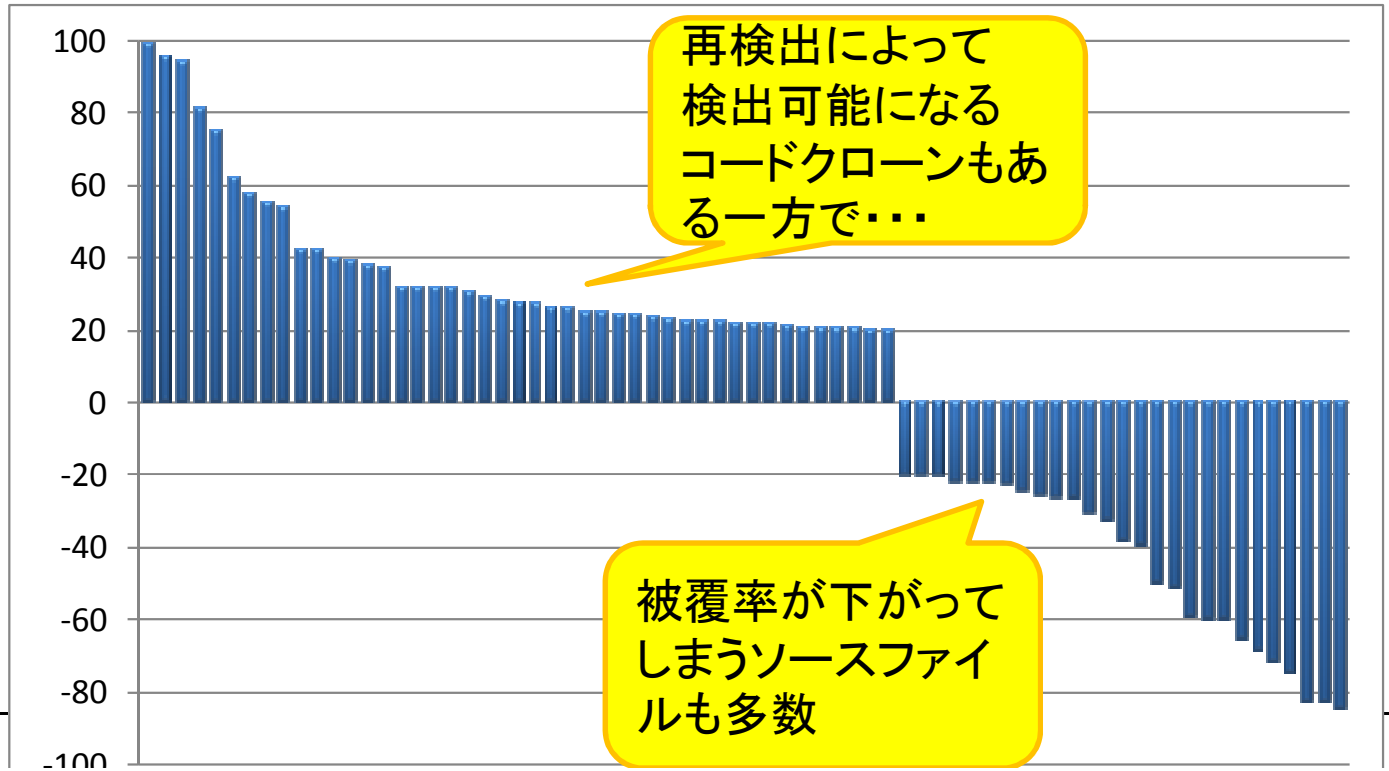
ステップ

1. 読み替えクラスごとに、含まれる識別子を、一の識別子に書き換える(トークン列修正)
2. 修正されたトークン列からコードクローンを検出する(再検出)

- ケーススタディ#1に引き続き、OpenOffice 2.2.1のソースコードに適用した

結果

- コードクローンによる被覆率が20ポイント以上変化したソースファイルが72個あった
  - (論文のデータの修正版)



```
263
264     nItemCount = lNewMenu.getLength();
265     for( nItem=0; nItem<nItemCount; ++nItem )
266     {
267         nPropertyCount = lNewMenu[nItem].getLength();
268         for( nProperty=0; nProperty<nPropertyCount; ++nProperty )
269         {
270             lNewMenu[nItem][nProperty].Value >>= sPropertyValue;
271
272                 sOut.appendAscii ( "New/"                               );
273                 sOut.append      ( (sal_Int32)nItem                     );
274                 sOut.appendAscii ( "/"                                  );
275                 sOut.append      ( lNewMenu[nItem][nProperty].Name      );
276                 sOut.appendAscii ( " = "                                );
277                 sOut.append      ( sPropertyValue                       );
278                 sOut.appendAscii ( "¥n"                                 );
279         }
280     }
281
282     sOut.appendAscii("¥n-----------------------------------¥n");
283
284     nItemCount = lWizardMenu.getLength();
285     for( nItem=0; nItem<nItemCount; ++nItem )
286     {
287         nPropertyCount = lNewMenu[nItem].getLength();
288         for( nProperty=0; nProperty<nPropertyCount; ++nProperty )
289         {
290             lWizardMenu[nItem][nProperty].Value >>= sPropertyValue;
291
292                 sOut.appendAscii ( "Wizard/"                            );
293                 sOut.append      ( (sal_Int32)nItem                     );
294                 sOut.appendAscii ( "/"                                  );
295                 sOut.append      ( lNewMenu[nItem][nProperty].Name      );
296                 sOut.appendAscii ( " = "                                );
297                 sOut.append      ( sPropertyValue                       );
298                 sOut.appendAscii ( "¥n"                                 );
299         }
300     }
301
```

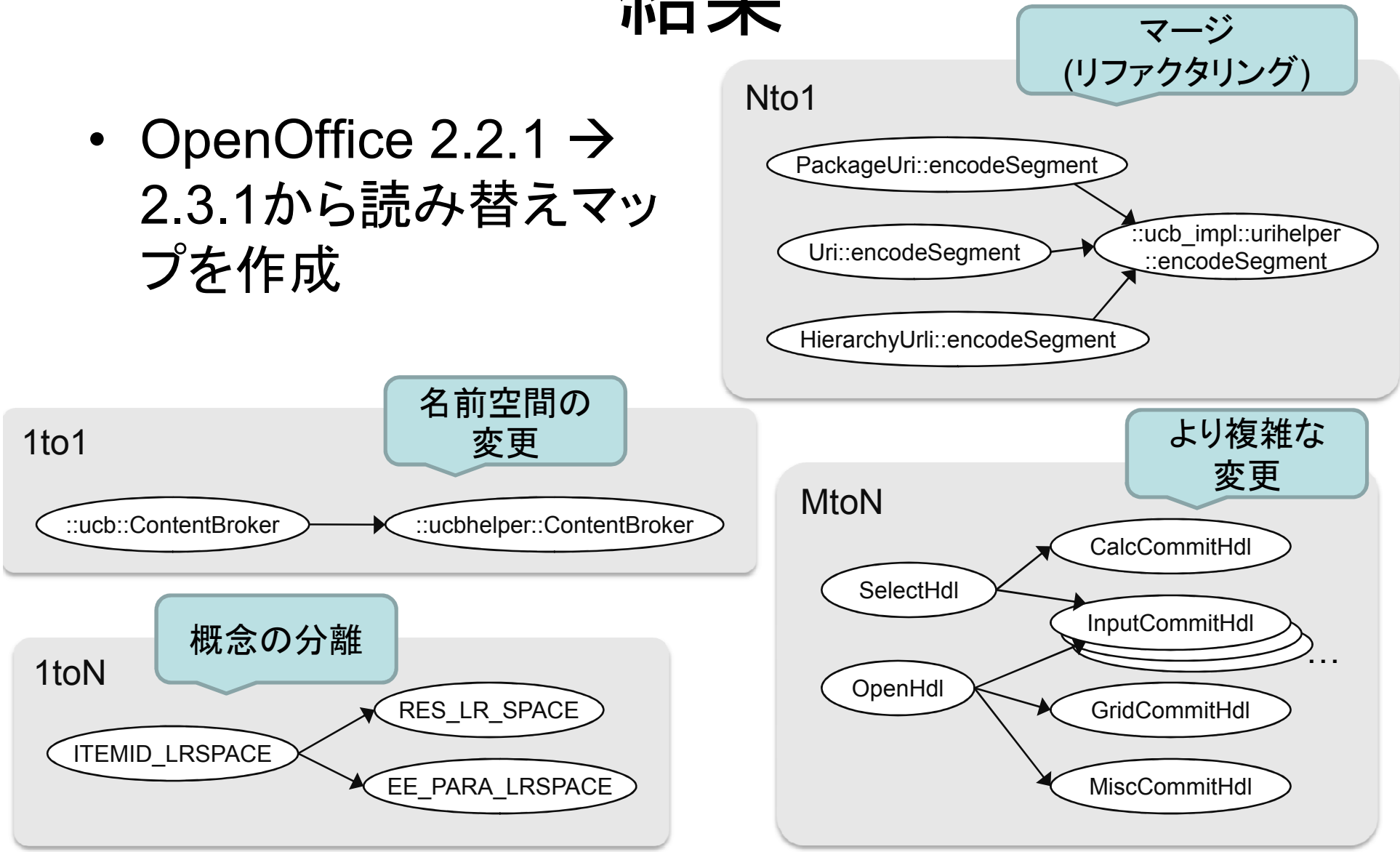
ケーススタディ#3: バージョン間の名前の遷移を調べる

ステップ

1. システムの旧バージョン(O)と新バージョン(*N*)のコード断片の間のテンプレートを求める
2. バリエーション部分から読み替えペアを作る
   - Oの識別子 → *N*の識別子
3. 読み替えペアからグラフを作り、その島を読み替えマップとする
4. 読み替えマップを1to1、1toN、Nto1、MtoNに分類する

結果

- OpenOffice 2.2.1 → 2.3.1から読み替えマップを作成



7. 関連研究

識別子の分析

[3] B. Caprile and P. Tonella, “Restructuring Program Identifier Names”, Proc. IEEE ICSM’00. pp. 97-107, Oct. 2000.

[4] F. Deißenböck and M. Pizka, “Concise and Consistent Naming”, Proc. IWPC 2005, pp. 97-106. May 2005.

コードのパターンや、コード修正のパターンの抽出

[6] M.W. Godfrey, et al. “Using Origin Analysis to Detect Merging and Splitting of Source Code Entities”, IEEE TSE, vol. 31, no. 2, pp. 166-181. Feb. 2005.

[18] 中山, 松下, 井上, “ソースコードの差分を用いた関数呼び出しパターン抽出手法の提案”, 情報処理学会研究報告, Vol.2006, No.35, 2006-SE-151, pp.49-56, 2006.

[19] Y. Yii, et al. “Token Comparison Approach to Detect Code Clone-Related Bugs”, 電子情報通信学会 技術研究報告, SS2007-57～75, Vol.107, No.505, pp.85-90, 2008.

7. 関連研究 つづき

コードクローン技術の応用

[13] 川口, 松下, 井上, “版管理システムを用いたクローン履歴分析手法の提案”, 電子情報通信学会論文誌D, Vol.J89-D, No.10, pp.2279-2287, 2006.

[15] Z. Li, et al. “CP-Miner: Finding Copy-Paste and Related Bugs in Large-Scale Software Code”, IEEE TSE, vol. 32, no. 3, pp 176-192, March 2006.

8. まとめと課題

まとめ

- 識別子のバリエーションを求める手法
  - 同義語のリストにより名前の揺れを調べる
  - 同義語のリストを用いることで、コードクローンの検出精度を改善する
  - バージョン間の名前の変遷を調べる
- プロトタイプを実装し、ケーススタディを行った

課題

- 手法の改良
- 実験による評価
- ツールの改良
- さらなる応用（できれば）