# 2012年度ソフトウェア工学分野の先導的研究支援事業

# コードクローン分析に基づくソフトウェア開発・保守支援に関する研究

大阪大学　大学院情報科学研究科

楠本真二

# 発表内容

- 背景
- コードクローン
- 研究目的
  - 4つのテーマ
- 研究内容
  - テーマ毎に，概要と成果
- まとめ

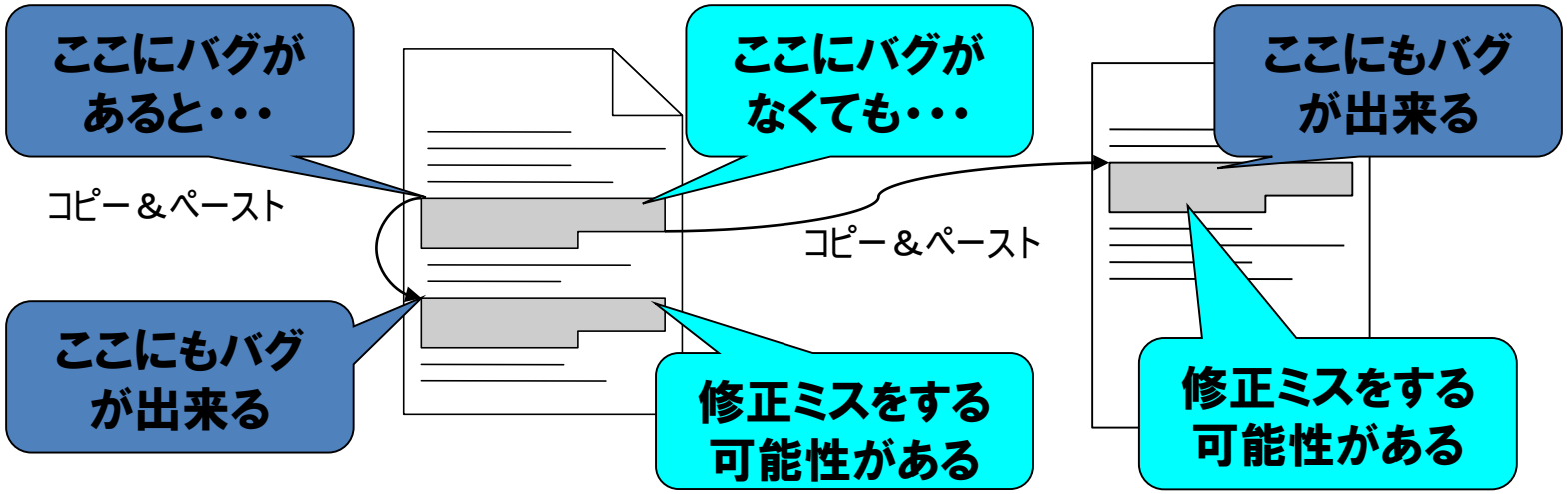
# 研究背景

- ソフトウェアシステムは社会基盤として必須のもの．
  - 現代社会で人々の日々の暮らしを支える
  - 例：銀行オンラインシステム、株取引、航空管制、自動改札、ネット販売、エンジン制御、携帯電話、…
- 情報サービス業の売上げの多くがソフトウェア開発関連業務であり，重要な産業分野
  - 平成23年：約19兆円
- ソフトウェア開発プロジェクトの成功率はあまり高くない
  - 特許庁の情報システム開発の失敗により約55億円の損失が発生（2012年1月）

**ソフトウェア開発に対する技術面，管理面での更なる改善が必要**

# コードクローンとは(1)

- 直観的な定義：ソースコード上に存在する同一，または，類似したコード片
- 主にコピー&ペーストによって発生
  - コードクローンに関係するバグ修正・機能変更時に手間がかかる
  - 潜在的な(見つけにくい)不具合を埋め込みやすい



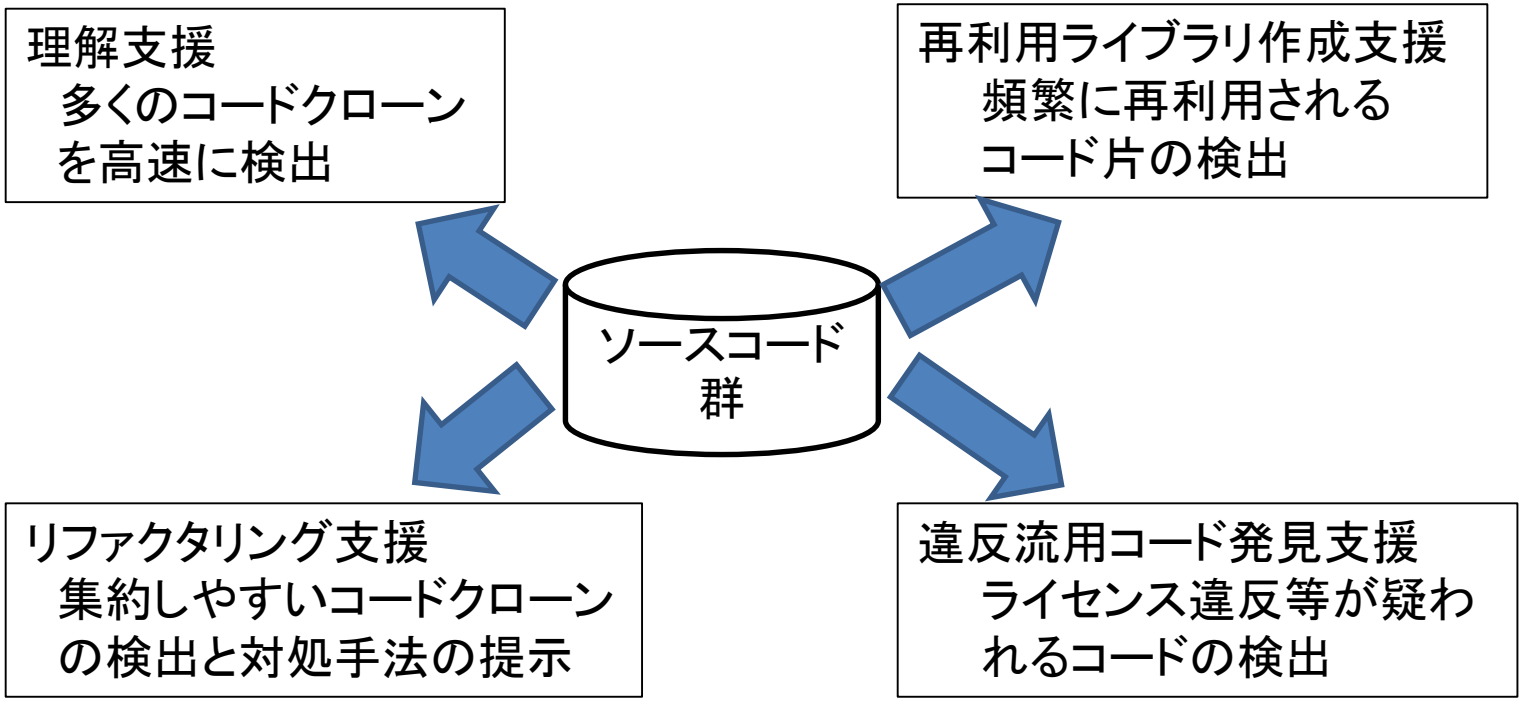
# コードクローンとは(2)

- ソフトウェア開発，保守を阻害する問題の一つとして研究が盛んに行われているテーマ
  - ソフトウェア工学国際会議でのセッションテーマ，コードクローンに特化した国際会議の開催（IWSC）．
- 研究成果の開発現場への普及状況は不十分
  - ソースコード解析サービスとしてのビジネス
  - 企業内でのローカルな利用

一般的なコードクローン検出/分析手法の提示だけでは不十分で，利用目的や状況に特化した手法の開発とその有用性の評価結果を合わせて提示する必要がある

# 本研究の目的

- **ソフトウェア開発や保守の様々な活動，状況（コンテキスト）に応じた支援を行う．具体的には，幾つかのコンテキストに応じたコードクローン検出手法の開発と検出されたコードクローンに対する対策手法の開発を行う．**



# 研究体制

管理責任者

大阪大学
大学院情報科学研究科
研究科長/教授
井上克郎

研究責任者

大阪大学
大学院情報科学研究科
教授
楠本真二

(R1) ソースコード理解支援
楠本真二
土屋達弘
岡野浩三

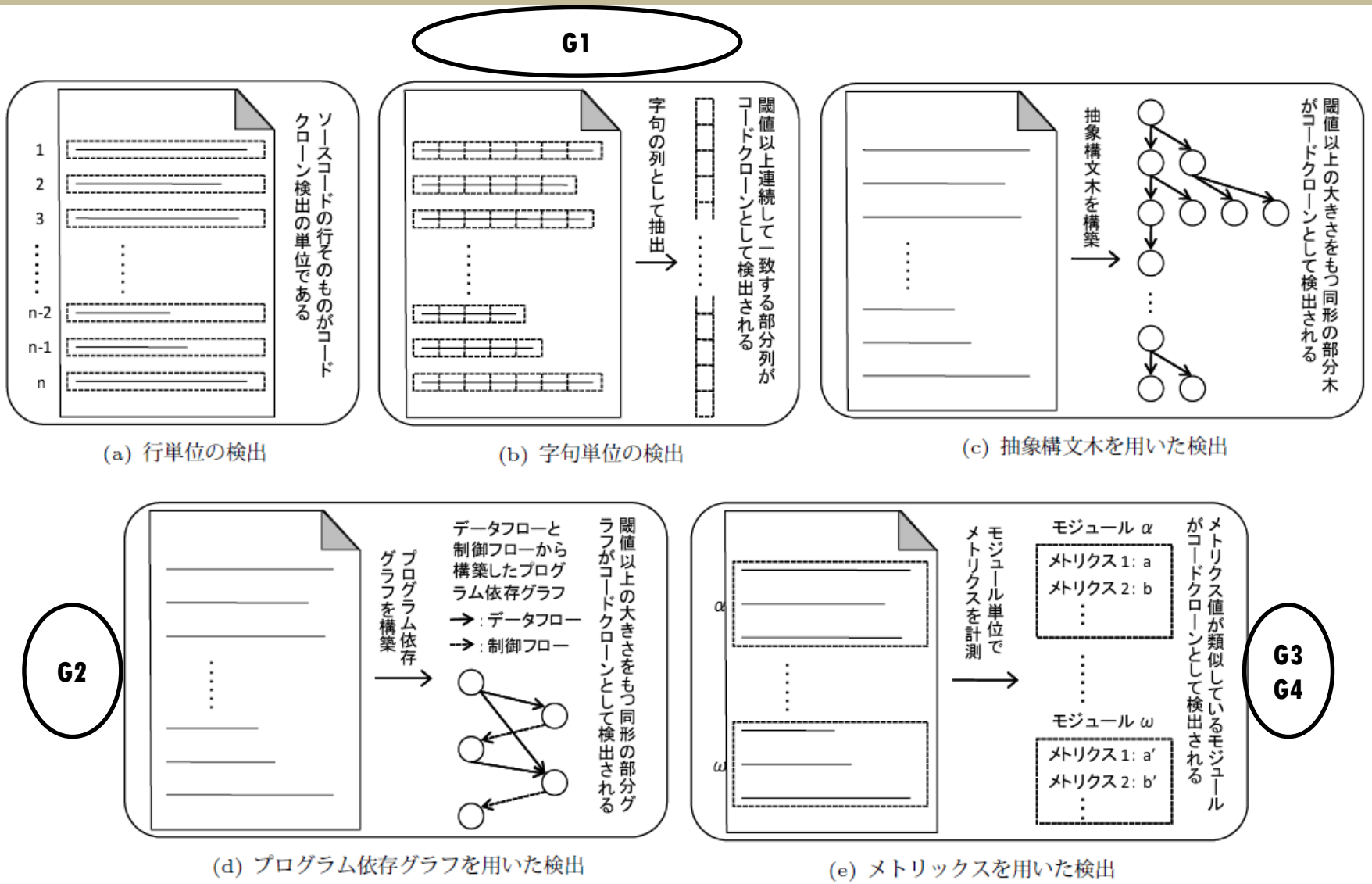
(R2) リファクタリング支援
井上克郎
松下誠
石尾隆

(R3) 再利用ライブラリ・違反コード検出支援
楠本真二
肥後芳樹

# コードクローンの定義

- コードクローン検出手法により異なる定義を持つ.
  - 行単位
  - 字句単位
  - 抽象構文木
  - プログラム依存グラフ
  - メトリクス

肥後, 楠本, 井上, "コードクローン検出とその関連技術," 電子情報通信学会論文誌D, vol.91-D, no.6, pp.1465-1481, June 2008




(a) 行単位の検出　(b) 字句単位の検出　(c) 抽象構文木を用いた検出

(d) プログラム依存グラフを用いた検出　(e) メトリックスを用いた検出

肥後, 楠本, 井上, "コードクローン検出とその関連技術," 電子情報通信学会論文誌D, vol.91-D, no.6, pp.1465-1481, June 2008

# 成果

下記4テーマについて，プロトタイプシステム開発を行い，評価実験を実施.

（G1）ソースコード理解支援
（G2）リファクタリング支援
（G3）再利用ライブラリ作成支援
（G4）違反流用コード発見支援

適用対象

- オープンソースソフトウェア
- IT Spiral実プロジェクトデータ

# IT Spiral実プロジェクトデータ

- IT Spiral：
  - 文部科学省「先導的ITスペシャリスト育成推進プログラム」の一つ．関西圏9大学院が連携して，実践的ソフトウェア工学の教育を行う．
- 実プロジェクト教材
  - 和歌山大学の教務情報管理システム（の一部）を発注し，開発した際に得られた現実の開発データを収集
  - 収集データ
    - 要求仕様書，品質要求書，UMLモデル図，画面設計書，プログラムソースコード(Java)，テスト計画書・仕様書，内部/外部レビュー記録，プロジェクト管理記録
  - 開発期間3ヶ月，総データ量450MB

# (G1) ソースコード理解支援

- 細粒度でなるべく多くのコードクローンを高速に検出する手法の開発.
  - 既存のコードクローン検出ツールと比較して，冗長なコードクローンをなるべく含まず，処理が高速で，検出したコードクローンの質が高い（再現率，適合率が高い）コードクローン検出ツール
- キーアイデア
  - 字句単位のコードクローン検出手法
  - 繰り返し部分の折りたたみ
- 検出対象
  - JavaとC言語のプログラム
- 評価
  - 幾つかのソフトウェアに適用して，従来手法と提案手法で検出されたコードクローンの質（再現率・適合率）を比較

$$\text{再現率} = \frac{\text{検出コードクローン中の正解}}{\text{全正解コードクローン}}$$

$$\text{適合率} = \frac{\text{正解コードクローン}}{\text{検出コードクローン}}$$

# (G1) 繰り返し部分の例

```
1:  package org.eclipse.jdt.internal.eval;
...
(略)
...
16:  public class EvaluationResult {
...
(略)
...
160:  switch (evaluationType) {
161:        case T_CODE_SNIPPET:
162:              buffer.append("Code snippet");
163:              break;
164:        case T_IMPORT:
165:              buffer.append("Import");
166:              break;
167:        case T_INTERNAL:
168:              buffer.append("Internal problem");
169:              break;
170:        case T_PACKAGE:
171:              buffer.append("Package");
172:              break;
173:        case T_VARIABLE:
174:              buffer.append("Global variable");
175:              break;
176:  }
```

(a) EvaluationResult.java

```
1:  package org.eclipse.jdt.
      internal.core.search.matching;
...
(略)
...
17: public class FieldReferencePattern extends
      MultipleSearchPattern {
...
(略)
...
235:  switch(matchMode){
236:      case EXACT_MATCH :
237:            buffer.append("exact match, ");
238:            break;
239:      case PREFIX_MATCH :
240:            buffer.append("prefix match, ");
241:            break;
242:      case PATTERN_MATCH :
243:            buffer.append("pattern match, ");
244:            break;
245:  }
```

(b)  FieldReferencePattern.java

# (G1) 成果 (1)

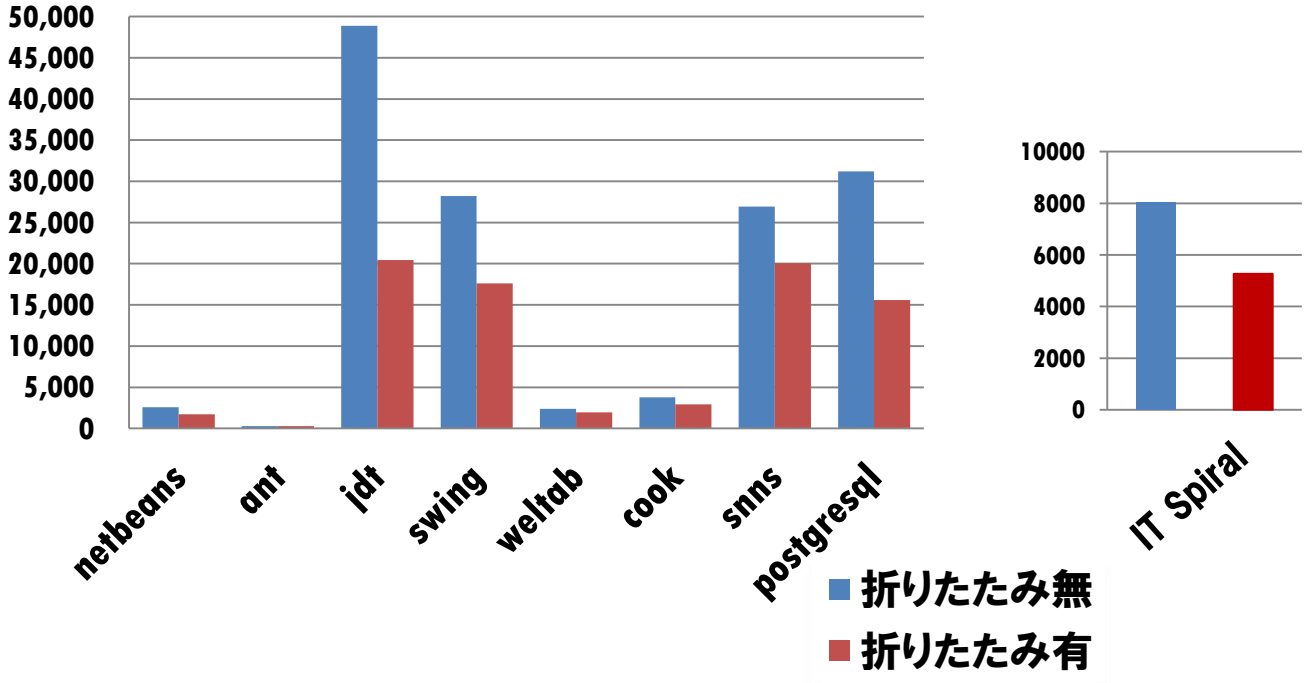
- 適用対象

  8種類のオープンソース

  IT Spiral実プロジェクトデータ（Java，約3万行）

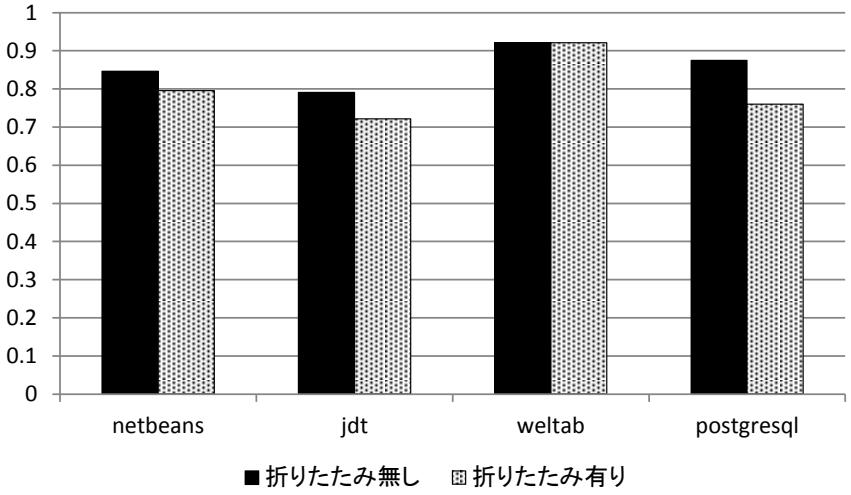
- 分析結果(1)

  折りたたみ後，クローンペア数が削減された

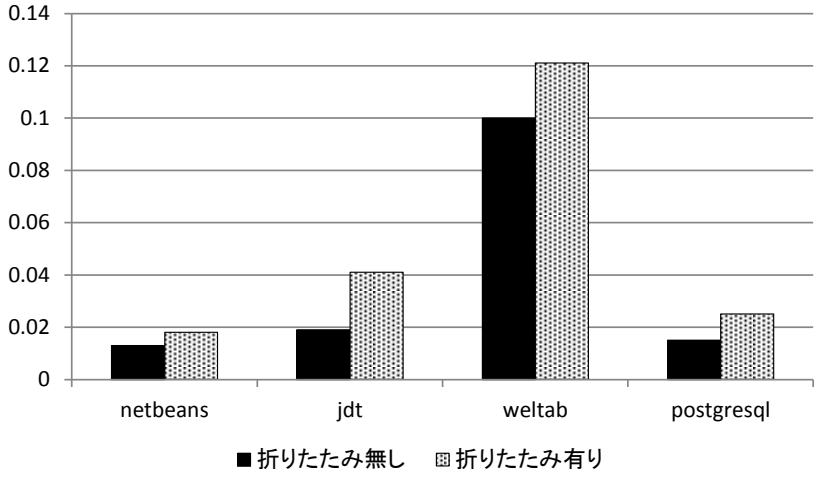


# (G1) 成果 (2)

- 分析結果(2)
  - netbeans,jdtcore, wltab, postgresqlに対する詳細分析(再現率と適合率)




再現率




適合率

# (G1) 成果 (3) 検出例

```
1:  package org.eclipse.jdt.internal.core.jdom;
…
（略）
…
21: class DOMMethod extends DOMMember
      implements IDOMMethod {
…
（略）
…
469:  protected void offset(int offset) {
470:     super.offset(offset);
471:     offsetRange(fBodyRange, offset);
472:     offsetRange(fExceptionRange, offset);
473:     offsetRange(fParameterRange, offset);
474:     offsetRange(fReturnTypeRange, offset); }
```

(a) DOMMethod.java

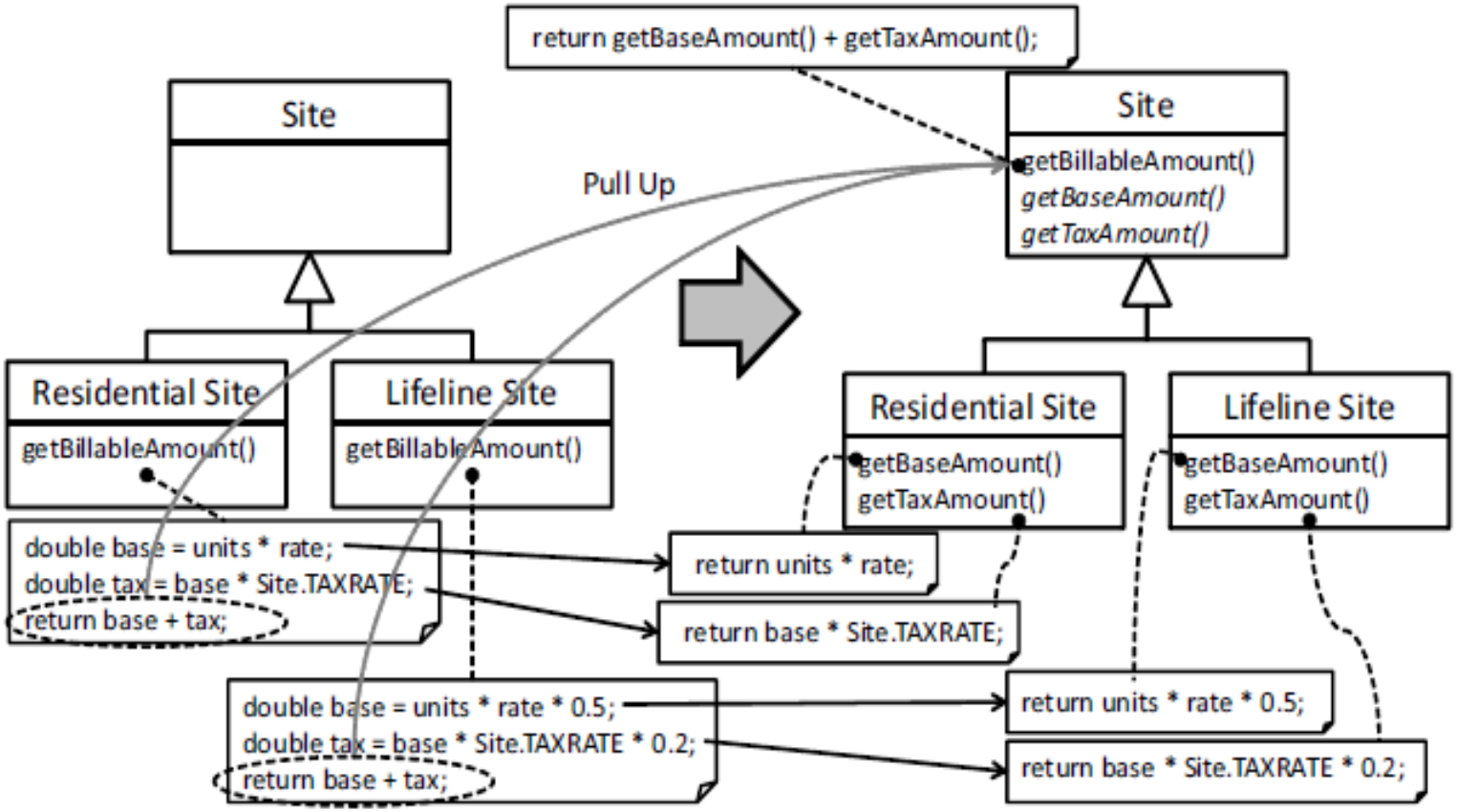
```
1:  package org.eclipse.jdt.internal.core.jdom;
…
（略）
…
25: class DOMType extends DOMMember
      implements IDOMType {
…
（略）
…
483:  protected void offset(int offset) {
484:     super.offset(offset);
485:     offsetRange(fCloseBodyRange, offset);
486:     offsetRange(fExtendsRange, offset);
487:     offsetRange(fImplementsRange, offset);
488:     offsetRange(fInterfacesRange, offset);
489:     offsetRange(fOpenBodyRange, offset);
490:     offsetRange(fSuperclassRange, offset);
491:     offsetRange(fTypeRange, offset); }
```

(b) DOMType.java

# (G2) リファクタリング支援

- Template Method パターンを適用し，リファクタリングを支援する手法の開発
  - Template Method パターン：共通の親クラスを持つ類似メソッドを対象とし，メソッド間で共通の処理を親クラスに記述し，メソッドごとに異なる処理を子クラスに記述する．
- キーアイデア
  - プログラム依存グラフを用いたコードクローン検出手法の応用
    - ユーザ定義名のみが異なるコードクローン
    - 意味的に同じ処理で表現方法が異なるコードクローン
- 検出対象
  - Javaプログラム
- 評価
  - 幾つかのソフトウェアに提案手法を適用して，検出したコードクローンが実際に集約可能かどうかを確認

# (G2) Template Methodパターン適用例




・Site を共通の親クラスとする2つのクラスの間に類似メソッドgetBillableAmount () が存在.
・共通部分を親クラスに引き上げ，異なる部分をそれぞれ新たなメソッドとして抽出.

・多態性によってそれぞれのクラスに応じたgetBaseAmount () 及びgetTaxAmount () が呼び出され，適用前後で振る舞いが保たれる

# (G2) 成果(1)

- 適用対象

  Apache Ant, Argo UML, Apache Synapse

  IT Spiral実プロジェクトデータ

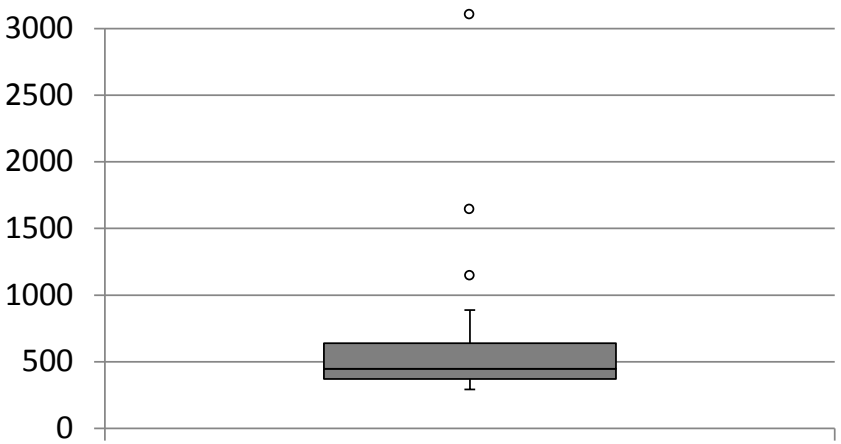
- 結果(1)

  テンプレートメソッドパターンが適用可能なコードクローンを検出できた.

| Target Systems | LOC | # of Candidates | Elapsed Time[s] |
|---|---|---|---|
| Apache Ant | 212,401 | 226 | 237 |
| Argo UML | 328,582 | 486 | 1,080 |
| Apache Synapse | 58,418 | 45 | 95 |
| IT Spiral | 31,948 | 9 | 45 |

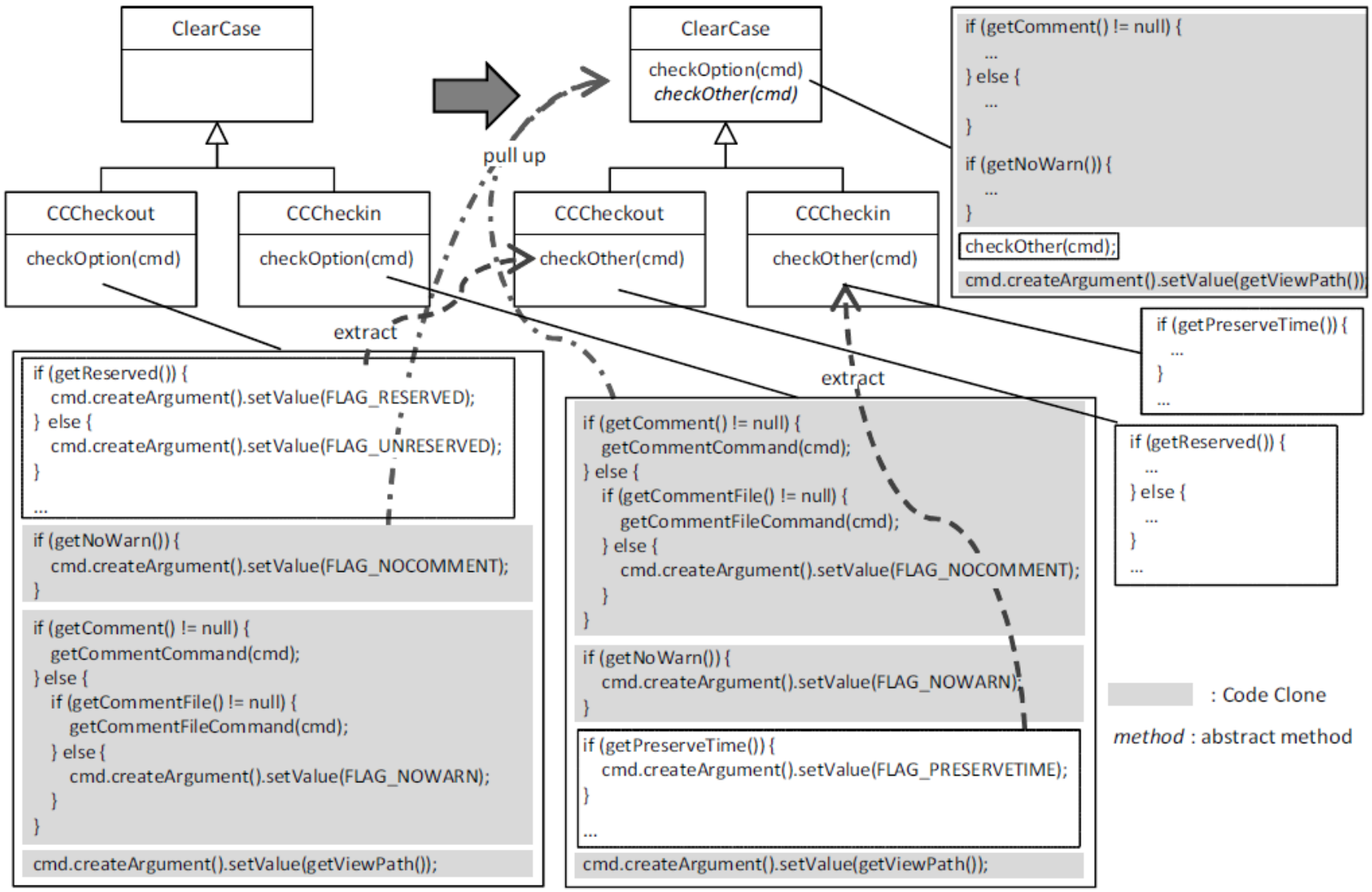
# (G2) 成果(2)

- 結果(2)
  - Apache Synapseから検出した45個の候補に対して，テンプレートメソッドパターンを適用して集約を行った．
  - 結果，集約前後での動作が変わらないことを確認した．




集約実施に要した時間

# (G2) 成果(3) 検出例


ClearCase
CCCheckout
checkOption(cmd)
CCCheckin
checkOption(cmd)
pull up
ClearCase
checkOption(cmd)
checkOther(cmd)
CCCheckout
checkOther(cmd)
CCCheckin
checkOther(cmd)
extract
extract
if (getReserved()) {
cmd.createArgument().setValue(FLAG_RESERVED);
} else {
cmd.createArgument().setValue(FLAG_UNRESERVED);
}
...
if (getNoWarn()) {
cmd.createArgument().setValue(FLAG_NOCOMMENT);
}
if (getComment() != null) {
getCommentCommand(cmd);
} else {
if (getCommentFile() != null) {
getCommentFileCommand(cmd);
} else {
cmd.createArgument().setValue(FLAG_NOWARN);
}
}
cmd.createArgument().setValue(getViewPath());
if (getComment() != null) {
getCommentCommand(cmd);
} else {
if (getCommentFile() != null) {
getCommentFileCommand(cmd);
} else {
cmd.createArgument().setValue(FLAG_NOCOMMENT);
}
}
if (getNoWarn()) {
cmd.createArgument().setValue(FLAG_NOWARN);
}
if (getPreserveTime()) {
cmd.createArgument().setValue(FLAG_PRESERVETIME);
}
...
cmd.createArgument().setValue(getViewPath());
if (getComment() != null) {
...
} else {
...
}
if (getNoWarn()) {
...
}
checkOther(cmd);
cmd.createArgument().setValue(getViewPath());
if (getPreserveTime()) {
...
}
...
if (getReserved()) {
...
} else {
...
}
...
: Code Clone
method : abstract method

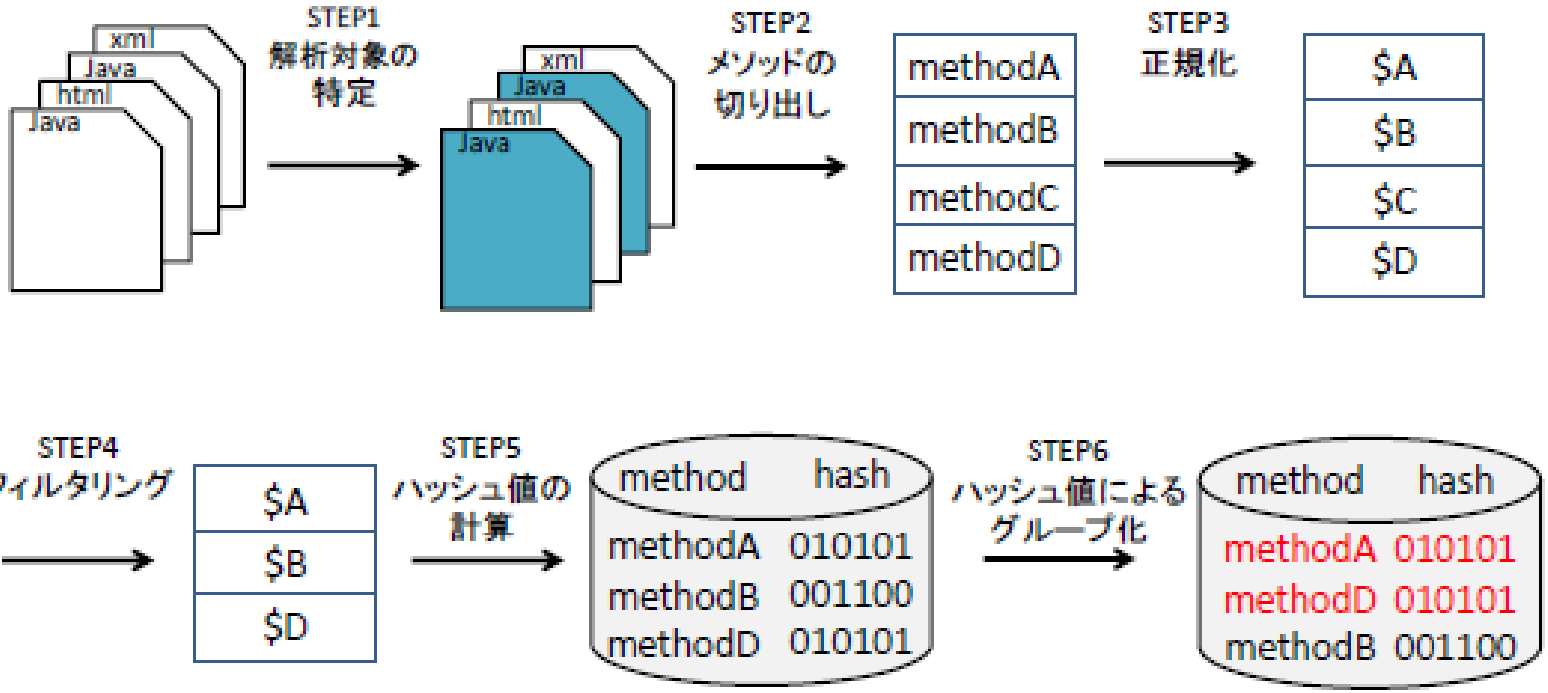
# (G3) 再利用ライブラリ作成支援

- 大規模なソースコード群から，適切な大きさ（例えば，メソッド，関数単位）のコードクローンを高速に検出するための手法とツールを開発．
- キーアイデア
  - 解析対象とするメソッドに対してハッシュ値を計算し，その値が一致するものをコードクローンと判定する．
- 検出対象
  - Javaプログラム
- 評価
  - オープンソースソフトウェア群へ適用し，メソッド単位の再利用コードを検出

# (G4) 違反流用コード発見支援

- ライセンス違反等が疑われるコードの検出を大規模なソースコード群を対象に行う手法の開発.
- キーアイデア
  - 流用の単位を，メソッドとする.
  - 解析対象とするメソッドに対してハッシュ値を計算し，その値が一致するものをコードクローンと判定する.
- 検出対象
  - Javaプログラム
- 評価
  - オープンソースソフトウェア群へ適用し，メソッド単位の再利用コードを検出

# (G3) と(G4)で用いるコードクローン検出手順



# (G3)(G4) 成果　(1)

- 適用対象
  - UCI source code data set*
    - ファイル数：200万，総行数：約4億行
  - IT Spiral　実プロジェクトデータ

- 結果(1)
  - UCI source code data set
    - メソッドクローンの集合：　約81万（要素メソッド数：約300万）
    - 検出時間：約5時間（1CPU,4core(2.00GHz)，メモリ8GByte）

＊ C. Lopes et al.，Uci source code data sets. http://www.ics.uci.edu/ lopes/datasets/.

# (G3)(G4) 成果　(2)

- 複数のソフトウェアに共通していた上位100種類のメソッドクローンを調査.
- 発生原因
  - ソースコードの流用
  - 汎用的な処理を行うメソッド
  - 抽象クラスの継承，インターフェースの実装

| メソッドの種類 | | 検出数 |
|---|---|---|
| GUI 関連 | AbstractTableModel | 15 |
| | その他 Table 関連 | 10 |
| | SwingWorker | 7 |
| | その他 | 8 |
| 64bit-encorder,decorder | | 13 |
| FileFilter | | 7 |
| ResourceBundle | | 4 |
| その他 | | 36 |

# (G3)(G4) 成果　(3)

- ライブラリ化の例
  - Table に関する処理を行うクラスで宣言されているソートを行うメソッド
  - Java1.6でライブラリ化されていた.

```
244 public void n2sort() {
245  for(int i = 0; i < getRowCount(); i++) {
246   for(int j = i+1; j < getRowCount(); j++) {
247    if (compare(indexes[i], indexes[j]) == -1) {
248     swap(i, j);
249    }
250   }
251  }
252 }
```

(a) sun　TableSorterクラス

```
219 public void n2sort() {
220  for (int i = 0; i < getRowCount(); i++) {
221   for (int j = i+1; j < getRowCount(); j++) {
222    if (compare(indexes[i], indexes[j]) == -1) {
223     swap(i, j);
224    }
225   }
226  }
227 }
```

(b) Perham　TableSorterクラス

# (G3)(G4) 成果　(4)

- 流用特定の例
  - (a)にはライセンス記述有り，(b)には無し．
  - (a) stringConverterを宣言し，put メソッドでstringConverter を呼出
  - (b) put メソッドの内部で処理を記述

```
39 private static Converter stringConverter =
new Converter() {
40   public Short convert(Object o) {
41    return parseShort(((String) o));
42   }
43 };

(中略)

49 public Object convertFrom(Object in) {
50   if (!CNV.containsKey(in.getClass())) throw
new ConversionException("cannot convert type: "
51     + in.getClass().getName() + " to: " +
Short.class.getName());
52   return CNV.get(in.getClass()).convert(in);
53 }

(中略)

62    CNV.put(String.class,
63     stringConverter
64    );
```

(a) mvel　ShortCHクラス

```
17 public Object convertFrom(Object in) {
18   if (!CNV.containsKey(in.getClass())) throw
new ConversionException("cannot convert type: "
19     + in.getClass().getName() + " to: " +
Short.class.getName());
20   return CNV.get(in.getClass()).convert(in);
21 }

(中略)

30    CNV.put(String.class,
31     new Converter() {
32      public Short convert(Object o) {
33       return Short.parseShort(((String) o));
34      }
35     }
36    );
```

(b) mvflex　ShortCHクラス

# (G3)(G4) 成果　(5)

- IT Spiral　実プロジェクトデータのソースコード中に，UCI source code data setのソースコードとメソッドクローンになっているものがあるかどうか確認

- IT Spiral: 208メソッド
- クローンとして検出されたメソッド数：17
  - IT Spiral内：14
  - IT Spiral-UCI 間：3
    - 共通しているプロジェクトが少ない(2, 3)
    - 処理内容がif文+ return文

# まとめ

## 主な結果

- ソフトウェア開発や保守の様々な活動，状況（コンテキスト）に応じた支援を目的とし，4つのコンテキストに応じたコードクローン検出手法の開発と検出されたコードクローンに対する対策手法を提案した．
  - (G1) ソースコード理解支援
  - (G2) リファクタリング支援
  - (G3) 再利用ライブラリ作成支援
  - (G4) 違反流用コード発見支援

## 今後の課題

- 評価実験の継続
- 適用対象の拡大