トップエスイー修了制作

複数言語対応のソースコード処理系フレームワーク ～テストカバレッジ測定への適用例～

早稲田大学 鷲崎研究室　　坂本 一憲　　kazuu@ruri.waseda.jp

開発における問題点

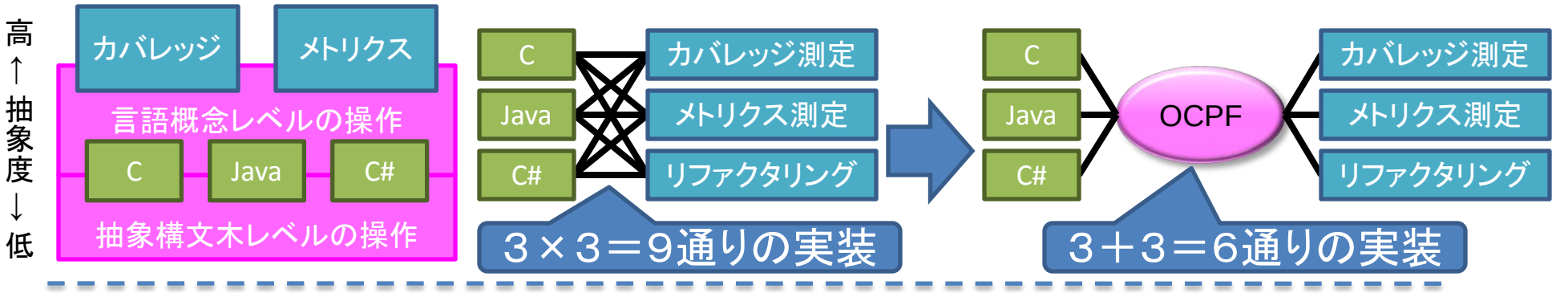
ソースコードを対象とした処理ツールが言語依存で，言語毎にツールが存在する．そのため，ツールの P1)**ノウハウ共有が困難**な点，複数言語を用いるプロジェクトでP2)**統一的に利用できない**点，新言語への対応等P3)**開発コストが高い**点，言語仕様の変更への追従等P4)**保守コストが高い**点で問題がある．

手法・ツールの提案による解決

複数プログラミング言語対応のソースコード処理フレームワーク：OpenCodeProcessorFramework (OCPF)を提案する．共通処理/言語固有/処理系固有の3部分に整理して，処理系固有と言語固有を分けることで，S1)**処理の実装を言語非依存**，S2)**言語の対応を処理非依存**にする．さらに，共通処理を提供することで，S3,4)**開発と保守コストを低減**する．

Open Code Processor Framework の概要

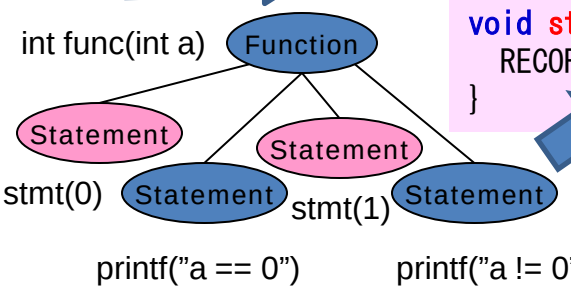
共通処理部（コールドスポット）：言語と処理系の両方に非依存．OCPFが想定する言語概念レベルの操作をインタフェースで規定（ステートメントの追加や削除等）．抽象構文木（AST）レベルの操作を提供（ノード追加や削除，置換）
言語固有部（ホットスポット）：各プログラミング言語固有の処理．各言語の言語概念レベルでの操作を実装．
処理系固有部（ホットスポット）：各ソースコード処理系固有の処理．提供される言語概念レベルの操作を利用．



カバレッジ測定への適用例

```
int func(int a) {
  if (a == 0) {
    printf("a==0");
  }
  else if (false) {
    printf("a!=0");
  }
}
```

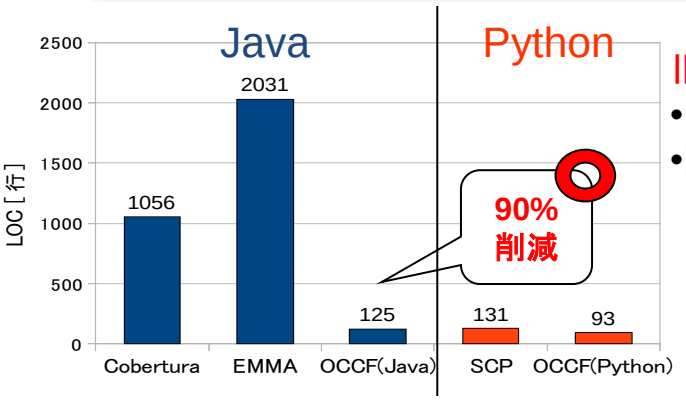
AST上で挿入




```
int func(int a) {
  if (branch(0, a == 0)) {
    stmt(0);
    printf("a == 0");
  }
  else if (branch(1,false)) {
    stmt(1);
    printf("a != 0");
  }
}
void stmt(int id) {
  RECORD_STATEMENT(id);
}
```

測定用コード
- 自動的に挿入
- カバレッジ情報出力
- 副作用なし

適用例への評価



旧バージョンで評価
- カバレッジ測定
- 開発コスト低減
  - 埋込処理のLOC
  - 90%程度削減

実装内容	達成数	平均時間
新カバレッジ追加(OCCF)	4人	13.5分
Python3への対応(OCCF)	4人	47.5分
新カバレッジ追加(SCP)	0人	4時間以上
Python3への対応(SCP)	0人	—

- 機能拡張実験
  - Python2への新カバレッジ
  - Python2からPython3へ変更