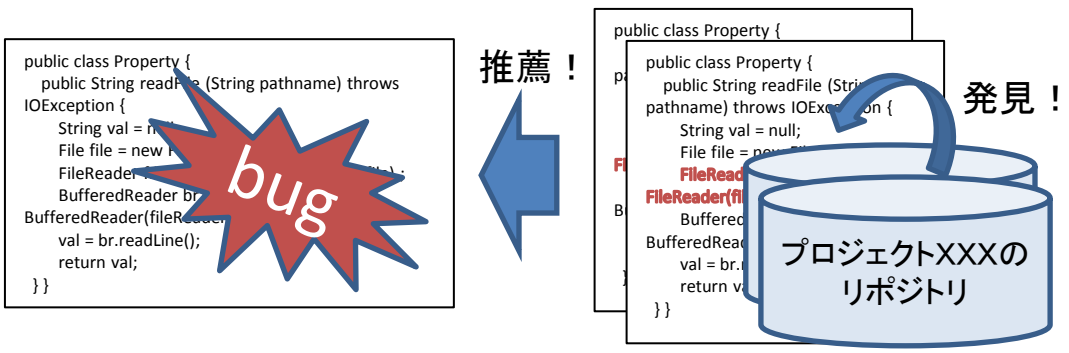
dcNavi: デバッグ方法をアドバイスする関心事指向リポジトリナビゲータ

塩塚 大
九州工業大学

鵜林 尚靖
九州大学

概要

- 目的
  - デバッグにおける***関心事(*表1)***に応じた情報を推薦
- アプローチ
  - 推薦はリポジトリ内のテスト結果，プログラム要素（クラス，メソッド，フィールド），修正パターンを関連付けたグラフである***デバッグ関心事グラフ(DCG)***の探索で実現
  - サポートツール***dcNavi***ではDCGの構築，関連情報の検索を支援



例）今まで見たことない例外発生
修正方法が分からない！

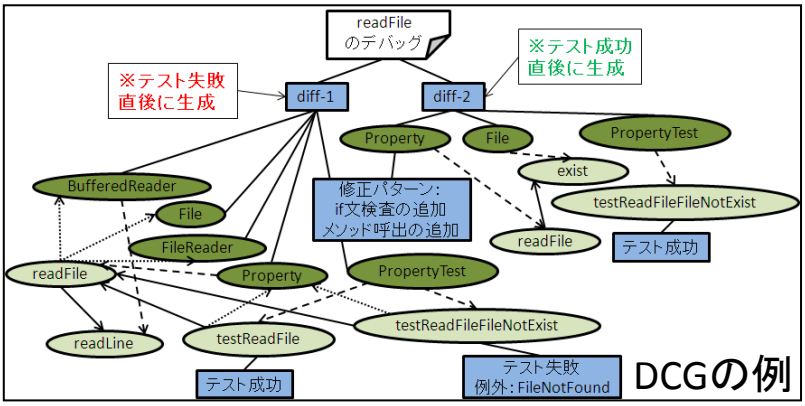
例外情報をもとに修正の参考になりそうなソースコードを推薦！

表1. 提案する推薦一覧

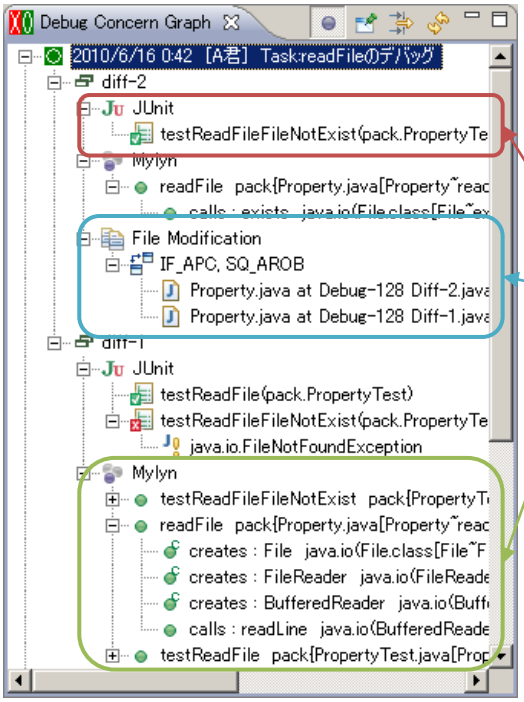
関心事	入力	出力（推薦）
例外やテスト結果をうまく活用できない	例外・テスト結果	関連ソースコード
ライブラリの利用方法が分からない	ライブラリ	ライブラリを利用したメソッド
テスト作成方法が分からない	ライブラリ	関連したテスト
レビューすべき箇所が分からない	対象ソースコード	修正候補箇所

デバッグ関心事グラフ（DCG）

- テスト失敗から成功までの作業をもとに，あるデバッグ作業に関連した情報をグラフで表現
  - グラフのリンクを辿ることで，ある例外が発生した際に関連していたテスト，メソッド，あるいは修正の差分情報の取得が容易となる
  - デバッグの際に頻繁に参考にされているようなソースコード（APIの利用方法などで）の取得が容易となる



dcNavi（debug concern navigator）



統合開発環境Eclipseのプラグインとして作成
以下の3機能をもつ．

1．関連情報収集機能
開発者のデバッグ作業を監視して以下の3情報を収集しDCGを構築する

- ***テスト結果，***
- ***修正パターン***
  （メソッドのパラメータの変更など＊）
- ***作業に関係したプログラム要素，***

2．関連情報推薦機能
次項で説明

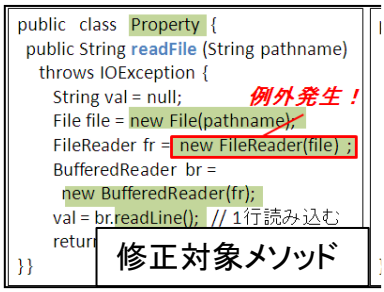
3．既存リポジトリからの移行機能
既存のSVNなどのバージョン管理システムからDCGを生成する

*Pan, K., Kim, S. and Whitehead, Jr. E. J.: Toward an understanding of bug fix patterns. Empirical Software Engineering, pp.286-315, 2009.
代表的な修正のパターンを構文の変更に基づいて27種類に分類

dcNaviの利用例1

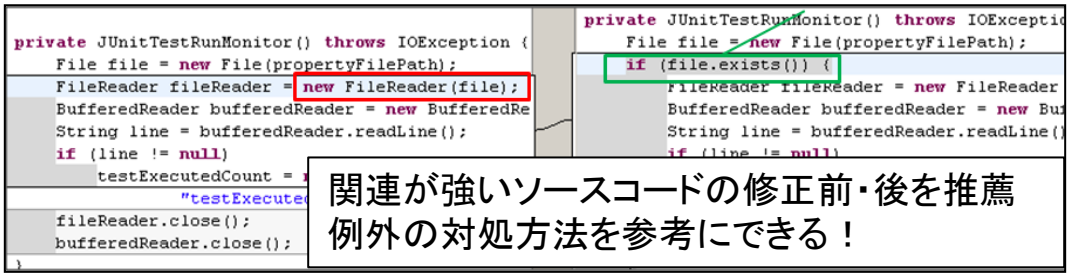
ケース1：例外が発生したので修正例を知りたい

⇒ 発生した例外を入力とし，過去に同様の例外が発生した際の修正方法を推薦



dcNavi上で発生した例外を選択

修正箇所

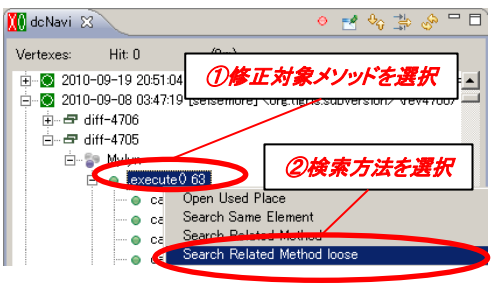


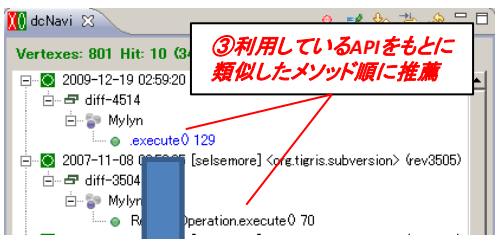
関連が強いソースコードの修正前・後を推薦
例外の対処方法を参考にできる！

dcNaviの利用例2

ケース2：使い慣れていないAPIを利用した際にバグに遭遇

⇒ 修正対象メソッドを入力とし，内部で利用しているAPIの過去の修正例を推薦






```
                .getConflictNew());
IFile conflictOldFile = (IFile) File2Resource
        .getResource(svnResource.getStatus()
                .getConflictOld());
IFile conflictWorkingFile = (IFile) File2Resourc
        .getResource(svnResource.getStatus()
                .getConflictWorking());
if (conflictWorkingFile == null) {
    conflictWorkingFile = resource;
}

IPreferenceStore preferenceStore = SVNUIPlugin.g
        .getPreferenceStore();
if (preferenceStore
        .getBoolean(ISVNUIConstants.PREF_MERGE_U
```

リビジョン4706*の実際の修正
同じような修正（if文の追加）を行っている

```
IFile conflictOldFile = (IFile) File2Resource
        .getResource(svnResource.getStatus()
                .getConflictOld());
IFile conflictWorkingFile = (IFile) File2Resource
        .getResource(svnResource.getStatus()
                .getConflictWorking());
if (conflictWorkingFile == null) {
    conflictWorkingFile = resource;
}

MergeFileAssociation mergeFileAssociation = null;
try {
    mergeFileAssociation = SVNUIPlugin.getPlugin()
} catch (BackingStoreException e) {
```

リビジョン4515*の修正
IF-APC修正パターン（if文の追加）

*どちらのリビジョンも
Eclipseのプラグイン
subclipseの一部.
修正ではマーク箇所
が追記された.

評価実験

- オープンソースを対象とし推薦量（個数）と推薦時間（ms）を確認
  - [対象] 統合開発環境Eclipseのプラグインで，とくにMylynプラグインに関連した9個のオープンソースプロジェクトに対して実施
  - [方法] バグを含んでいたメソッドに対して、修正の参考になりそうなメソッドをどれだけ（個数）、どのくらいの時間（ms）で推薦できるかを確認．推薦元として「自プロジェクトのリビジョンの4/5 + 他の8プロジェクトの全リビジョン」の修正情報を用いて，自プロジェクトの残り1/5のリビジョンのバグを含んでいたメソッドに推薦を実施
  - [環境] CPU Atom N280(1.66GHz), RAM1GB, OS Windows XP
  - [結果] 推薦量（上限を30個に制限），推薦時間ともに十分であった

サンプルプロジェクト	リビジョン(bug fix)	LOC	修正対象メソッド数	平均推薦数	1メソッドの平均推薦時間（ms）
com.google.code.projecthosting.mylynconnector	18(2)	2743	1	30	5.2
com.industrialtsi.mylyn	50(2)	10953	0	0	-
com.itsolut.mantis	537(107)	18536	15	19.6	6.5
ch.ethz.origo.mylyn	3761(867)	5769	4	30	6.1
org.qcmylyn	223(80)	13117	50	26.7	5.7
org.svenk.redmine	423(134)	14729	70	24.1	5.4
org.eclipse.mylyn.rememberthemilk	70(13)	7259	6	5.7	2.5
org.lcx.scrumvision	431(14)	43263	3	11.6	8.4
org.tigris.subversion	4745(923)	167100	91	24.5	5.8

今後の課題

- 履歴数を増量しての評価
  - 今回は、9つのオープンソースプロジェクトのバージョン管理システムの履歴をもとに推薦を実施．さらに利用する履歴を増やした場合に，推薦量や推薦時間がどのように変化するか確認する．また，今回はドメインをEclipseプラグインのMylynに関連したプラグインに限定して実験をおこなった．他のドメインについても評価が必要．
- 推薦の質についての評価
  - 推薦された修正例が，実際にデバッグをする上でどの程度効果的であるかを評価する．
    - 推薦された修正方法と実際におこなった修正方法の一致度で推薦の質を評価したところ，平均して約8%の修正対象に実際に行われた修正方法を推薦できた
- 推薦結果のフィードバックの利用
  - 推薦されたものが役立った場合と、そうでない場合のフィードバックを利用し、推薦内容を調整する．