統合開発環境のためのアスペクト指向システム

薄井義行† 千葉 滋†

1. はじめに

本発表では我々が開発した統合開発環境のためのアスペクト指向システム Bugdel のデモを行う。Bugdel ではアスペクト指向を利用してテストやデバッグ用コードの指定を行う。アスペクト指向を利用することでソースコードと、これらの追加コードを別々に指定できる。また、テスト、デバッグに特化したアスペクト指向システムであり AspectJ のような従来の汎用的なアスペクト指向システムには無い機能を提供する。

2. アスペクト指向を利用したデバッグ、テストコードの記述

ソフトウェア開発においてテスト、デバッグが必ず行われる。その際、開発中のプログラムにテストのためのコードやデバッグのためにトレースコードを追加する。これらのコードは本来、開発中のプログラムには関係のないものでありソースコードとは別に記述されるべきである。アスペクト指向を利用することで、これらのコードを別に記述できる。

2.1 アスペクト指向とは

アスペクト指向とはロギング処理や同期処理などのような複数のクラスにまたがる処理をアスペクトとしてモジュール化する技術である。アスペクトは主に次の要素から構成される。

Pointcut コードの実行位置を指定

Advice 実行するコードを指定

Pointcut ではプログラム中の任意の位置を指定できるわけではなく、フィールドアクセスやメソッド呼び出しなどのイベントを指定する。汎用的なアスペクト指向システムとして Java を言語拡張したアスペクト指向言語 AspectJ がある。以下に AspectJ で記述したアスペクトのコード例を示す。

```
1 public aspect Logging{
2   pointcut p():
3        call(void Figure.set*(..));
4   before():p(){
5     System.out.println("call set");
6   }
7 }
```

このコードは Figure クラスのメソッドで名前が set で始まるメソッドの呼び出し位置でコンソールへの出力をするためのアスペクトである。

2.2 AspectJ の問題点

AspectJ ではアスペクトをプログラミング言語で記述する。しかし、デバッグのためだけに 2.1 章で示したような pointcut 記述を行うのではプログラマへの負担が多い。

また、汎用的なアスペクト指向システムである反面、デバッグやテストに不向きな点がある。一つめは、任意のソースコードの位置を pointcut で指定することはできない。例えば、ソースコードの 10 行目という pointcut の指定は行えない。二つめは、Advice コード内から pointcut で指定した位置に存在するローカル変数にアクセスできないためローカル変数をトレースするコードを記述できない。

さらに、AspectJ では pointcut を指定する際にメソッド名やフィールド名を指定する。それらの名前を変更した場合 pointcut の対象にならなくなってしまうという問題がある。例えば、call(void Point.setX(int)) という pointcut が指定されている状態で setX メソッドの名前を setterX にした場合、setterX は pointcut の対象にならなくなってしまう。

3. Bugdel

前章で述べた問題点を解決するためにテスト、デバッグに特化したアスペクト指向システム Bugdel を開発した。

Bugdel は統合開発環境である Eclipse のプラグインとして実装されており GUI を利用して pointcut の指定を行う。例えば、ソースコードのクラス名やメソッ

† 東京工業大学 情報理工学研究科 数理・計算科学専攻
Department of Mathematical and Computer Science, Tokyo Institute of Technology

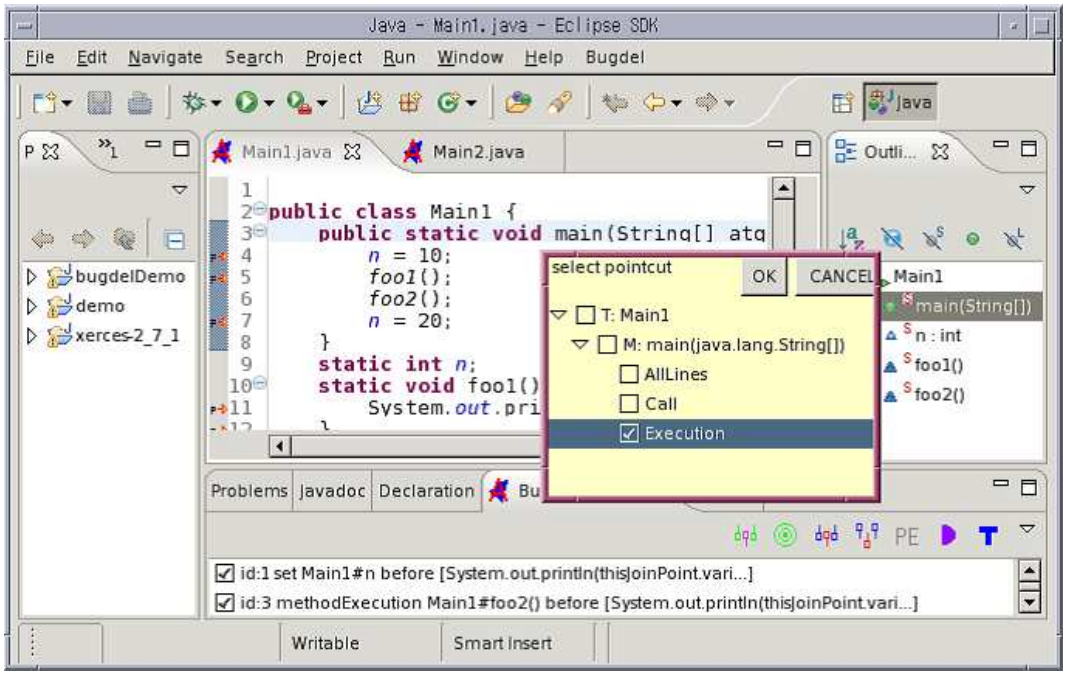

図 1 Bugdel のデモ画面

ド名をマウスでクリックすることで pointcut を指定できる。また、Bugdel が提供するソースコードブラウザを利用して、検索ベースで pointcut の指定ができる。GUI を通して pointcut の指定ができるため、AspectJ に比べて容易に利用できる。

Bugdel はテストやデバッグに特化したアスペクト指向システムであり汎用的なアスペクト指向システムには無い機能を提供する。一つめは、ソースコードの行番号を pointcut で指定する line pointcut を提供する。これにより、ほぼ任意の位置でテスト、デバッグ用コードを実行させることができる。二つめは、メソッド内の全ての行番号を指定する allLines pointcut を提供する。これによりステッ実行を擬似的に行える。三つめは、Advice コード内から pointcut で指定した位置に存在するローカル変数へのアクセスが可能であり、ローカル変数に対するリフレクション機能も提供する。

さらに、Bugdel は pointcut に影響のあるソースコードの変更を監視する機能を持っている。統合開発環境の機能を利用することでクラスやクラスメンバの追加、削除を監視し、pointcut に影響のある変更があった場合に警告を出しユーザに知らせる。ソースコード検索ブラウザを利用して pointcut を指定した場合には、検索の手順を保存する。その後、検索の手順に影響のあるソースコードの変更があった場合には警告を出す。

3.1 デバッグ、テストコードの埋め込み

Bugdel ではアスペクトとして指定したテストやデバッグ用コードの追加をクラスファイルを変換することで行う。そのため、生成したコードの実行には Eclipse や Bugdel は必要なく通常の JVM でテスト、デバッグが行える。

4. Eclipse の x86 用 Solaris への適用

現在（2005 年 11 月 15 日）Eclipse のウェブサイトで公開されている Eclipse の最新バージョンは 3.1.1 であり、さまざまなハードウェアと OS 用の Eclipse が配布されている。しかし、x86 用 Solaris の Eclipse は無い。これは、Eclipse の利用している GUI コンポーネント SWT が対応していないためである。SWT は高速に動作するがハードウェア、OS 依存のネイティブライブラリが必要である。

そこで今回、x86 用 OpenSolaris 上でデモを行うために、x86 用 Solaris に適応させた SWT ライブラリを準備した。x86 用 Solaris の SWT は SWT (spark 用 Solaris) のソースコードを x86 用 Solaris 上で再コンパイルすることで生成できる。生成した SWT ライブラリを使うことで Eclipse が利用可能になる。