Javaソースコードの CDI(Code Inspection) の開発
～フローグラフとその処理の設計実現～

M2007MM031 立岑将輝
指導教員：沢田篤史

1　はじめに

我々はコードインスペクションツール (CDI ツール) の開発をおこなった.
ソフトウェアの品質を保証するための静的検証技術のひとつにインスペクションがある．特に，プログラムに対するインスペクションをコードインスペクションと呼び，プログラムの欠陥を発見する．CDI ツールは，意味解析，制御フロー解析，およびデータフロー解析などの静的解析をおこなうことで，欠陥を発見するソフトウェアである.
このようなソースコードを入力とする静的解析ツールでは，フローグラフに対して処理をおこなう機構が存在する．CDI ツールで扱うフローグラフとは，ソースコードをもとにソフトウェアの制御やデータの流れを表現したものである.
CDI ツールの製品系列には，製品毎で共通する部分と変動する部分が存在する．共通部分は，フローグラフのような製品ごとに仕様の差異がなく，開発の途中で仕様の変更が起こらない箇所である．変動部分は，解析処理のような製品ごとに多種多様であり，開発の途中で追加変更が起こる箇所である.

系統的に再利用をおこなうための開発方法論としてプロダクトラインソフトウェアエンジニアリング (PLSE)[4] がある．PLSE を適用することによって，製品ごとのソフトウェア部品の再利用が容易になる．CDI ツールの再利用部品開発では，製品系列の統一アーキテクチャに基づいて部品を開発する．CDI ツールの製品開発では，開発した再利用部品を組み合わせて，さまざまな種類の製品を開発する.
ソフトウェア部品の再利用技術としてアプリケーションフレームワーク (フレームワーク) がある．フレームワークは，特定の用途や目的のためのシステムが共通的に必要となる骨格や機能を提供するものである．個別のシステムごとに異なる部分のみが未定義となっているので，開発者はこの未定義部分であるホットスポットに対して開発をおこなう.
PLSE に基づく開発では，フレームワークのフローズンスポットを製品ごとに共通する再利用部品として開発をおこない，ホットスポットは製品ごとに変動する再利用部品として開発をおこなう.

ソフトウェアの制御やデータの流れを検査する CDI ツール開発で解決すべき問題には，フローグラフに対する処理機構および処理の追加，変更に柔軟な構造がある.
制御やデータの流れを検査する CDI ツールでは，フローグラフに対して処理をおこなう．フローグラフに対する処理は，フローグラフのノードを連鎖的に辿って処理をする場合が多い．検査は多種多様に存在するので，検査ごとに走査処理を実現することは冗長である．以上から，CDI ツールにはフローグラフに対する処理の機構が必要となる.
CDI ツールは製品ごとに検査の種類が異なり，ひとつの製品で様々な種類の検査が存在する．一方，フローグラフは製品や検査の種類に依存せず，製品系列で共通して扱うことができる．以上から，CDI ツールには製品系列で共通的なフローグラフをもとに，変動的な解析処理を柔軟に追加，変更できるような構造が必要となる.

本研究の目的は，フローグラフ処理機構を持つ CDI ツールのフレームワークを実現することである.
フローグラフ処理機構を持つ CDI ツールのフレームワークを実現することによって，制御やデータの流れを解析する検査の実現が容易になる.
製品ごとに変動的な再利用部品としてホットスポットを実現する．製品開発では，フレームワークのフローズンスポットを CDI ツールの共通する基盤として再利用し，変動的な再利用部品を選択する．共通部分と変動部分を再利用可能部品として実現することによって，CDI ツールを製品ごとに新規開発する必要がなくなる.

問題を解決するためのアプローチは，抽象構文木処理機構を持つ CDI ツールをもとに，フローグラフ処理機構を持つフレームワークに拡張することである.
設計のもとになる CDI ツールは，抽象構文木に対して解析処理をおこなう．フローグラフ処理機構を持つツールでは，制御フロー解析とデータフロー解析の成果物である制御フローグラフとデータフローグラフに対して解析をおこなう.
事例検証として，Java1.5 の言語仕様に基づくソースコードの CDI ツール開発をおこなう.

次に研究手順を示す.

1. フローグラフ処理機構を持つ CDI ツールの分析
2. 抽象構文木処理機構を持つ CDI ツールとの比較
3. フレームワークの設計
4. フレームワークの妥当性を考察

2　背景技術

プロダクトラインソフトウェアエンジニアリング

系統的に再利用をおこなうための開発方法論としてプロダクトラインソフトウェアエンジニアリング (以下，PLSE) がある．PLSE は対象ドメインの分析結果をもとに核資産を定義し改良を繰り返しながら，核資産から部品を組み合わせてさまざまな製品開発をおこなうプロセスを規定している．核資産とは，要求分析の結果，再利用可能

部品，部品化の手法およびアーキテクチャなどの開発に必要な項目全ての総称である．
PLSE は，核資産開発，製品開発，および管理の 3 つの活動で構成されている．
核資産開発は対象のドメイン分析，ソフトウェアアーキテクチャ構築，再利用部品構築などをおこなう．製品開発は製品仕様をもとに核資産から必要となる部品を選択統合し，製品となるソフトウェアを開発する．管理では，PLSE に基づいた開発をおこなうための組織やプロジェクトなどの体制管理をおこなう．

3 PLSE に基づく CDI ツールの開発プロセス

我々は，PLSE をメタ技術として捉えて，言語処理技術，プログラミング言語，ソフトウェアアーキテクチャ技術およびオブジェクト指向技術などのベース技術を選択・統合した．PLSE によってソフトウェア部品の再利用が容易になる．CDI ツールを事例とした PLSE に基づく開発プロセスを図 1 に示す．

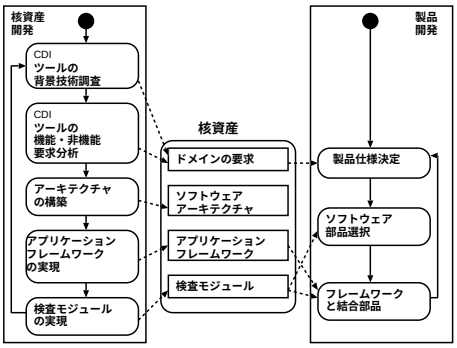

図 1 開発プロセス

核資産開発で，ソフトウェアアーキテクチャ，フレームワークおよび検査モジュールの開発と改良をおこなう．検査モジュールとは，CDI ツールの検査をおこなうモジュールのことである．フレームワークの改良をおこなう場合は，改良元のフレームワークに対して影響を与えないように拡張する．製品開発では，核資産から検査モジュールを選択して，フレームワークと統合して CDI ツールを開発する．

4 フローグラフ処理機構を持つ CDI ツールの分析

フローグラフ処理機構を持つ CDI ツールの構成要素から求められるソフトウェア構造を分析する．さらに，CDI ツールを開発する際に，共通する部品と変動する部品を整理する．

4.1 構成要素

フローグラフ処理機構を持つ CDI ツールの構成要素はフローグラフおよびそれに対する解析処理の 2 つである．
フローグラフには，制御フローグラフとデータフローグラフがある．ソースコードをもとにそれぞれソフトウェアの制御の流れとデータの流れをグラフ構造で表現したものである．
制御文はプログラムにおける処理の分岐や合流を表す．While 文や Do 文のような制御文は，プログラムの処理の分岐や合流の起点となる．
文要素をノードとすると制御文をフローグラフの分岐や合流とすることができるようになるので，我々は文要素をフローグラフのノードとして扱う (図 2)．

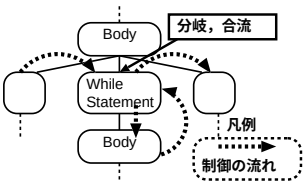

図 2 制御文 (WhileStatement)

フローグラフに対する解析処理は，フローグラフ全体または一部を走査するような処理が多く，ノード間を制御の流れに沿って連鎖的にノードを参照するものが多い．走査順序は解析処理ごとに共通する．または，共通しない場合でも複数の走査の組合せで解析処理を実現することができる．解析処理における共通的な箇所は走査順序であり，変動的な箇所はノードごとの処理である．
フローグラフに対する解析処理には統一した走査処理の機構が求められる．さらに，フローグラフはグラフ構造であるので，走査が循環しないような仕組みが必要となる．

4.2 共通部分と変動部分の分析

CDI ツールは，ソフトウェアの検査をおこなう解析処理の種類が多く，それぞれの処理は独立してフローグラフを参照する．さらに，検査に対する追加・変更の頻度は高いので解析処理の追加・変更の頻度は高い．
一方，処理の対象となるデータは Java1.5 の言語仕様に基づいたソースコードをもとに生成される．言語仕様の変更はないことから処理の対象となるデータも変更はない．
以上の特徴からフローグラフ処理プログラムには解析処理の追加・変更に対して柔軟な構造が求められる．この際に，解析処理は変更容易性が求められ，フローグラフは必ずしも変更容易性が求められるわけではない．フローグラフ処理プログラムにおける普遍箇所はフローグラフであり，変動箇所は解析処理である．

5 抽象構文木処理機構を持つ CDI ツールとの比較

フレームワーク設計のもとになる CDI ツールの静的構造を図 3 に示す．

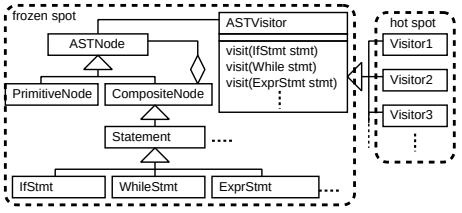

図 3 CDI ツールの静的構造

Visitor パターンを適用することで抽象構文木から処理を分離しているので，処理の変更に対して柔軟な構造である．さらに，Interpreter パターンを適用して走査順序をノードに定義しているので，抽象構文木に対する解析処理は共通の走査処理を再利用することができる．
フローグラフ処理機構を持つ CDI ツールと抽象構文木処理機構を持つ CDI ツールで異なる箇所は対象となるデータ構造と走査処理である．フローグラフを処理する機構を実現する場合は，抽象構文木と走査処理を変更する必要がある．

6 フレームワークの実現

6.1 フローグラフの定義

フローグラフの情報を取得できるようにするために，抽象構文木の文要素に対してフローグラフに必要となるメソッドを加える (図 4)．

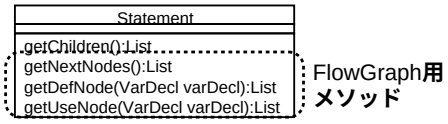

図 4 フローグラフのノード

文要素だけでは制御やデータの流れが表現できないものに関しては，新たにフローグラフ用のノードとして定義して置き換える．
例えば，for 文に対して制御がうつった場合，一回目であれば初期化式が評価された後に条件式が評価される．二回目以降の場合は初期化式が評価されずに，条件式が評価される．我々が構築した抽象構文木を単純にもちいると，初期化式，または条件式のどちらを評価すべきかを判断する処理をおこなわなければならない．
図 5 に for 文の初期化式，条件式，および更新式をフローグラフ用のノードとして定義した例を示す．

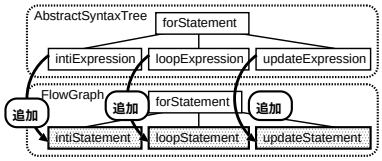

図 5 for 文の式をフローグラフ用のノードに変換

6.2 フローグラフに対する処理の定義

フローグラフに対する処理の追加変更に柔軟な構造にするために，CDI ツールのアーキテクチャと同様に Visitor パターンを適用し，データ構造からデータ構造に対する処理を分離する．Visitor パターンを適用することによってフローグラフに対する処理を分離することができるようになり，フローグラフに対する処理の追加変更に柔軟な構造になる．
フローグラフ用の Visitor クラスを定義し，サブクラスに処理を実装する．サブクラスには，ノードの型ごとの処理を記述する．

6.3 フローグラフに対する走査の定義

フローグラフに対する走査処理を定義するために，CDI ツールのアーキテクチャと同様に Interpreter パターンを適用し，解析処理から走査処理を分離する．分離した走査処理は，フローグラフのノードに対して定義する．Interpreter パターンを適用することによって解析処理から走査処理が分離され，解析処理の内部に走査処理を実装する必要がなくなる．
フローグラフはグラフ構造であるので走査の循環を防ぐために，一度処理をおこなったノードは処理をしないような機構にする．グラフ探索 [6] に一般的に用いられている深さ優先探索を用いた場合の例を図 6 に示す．

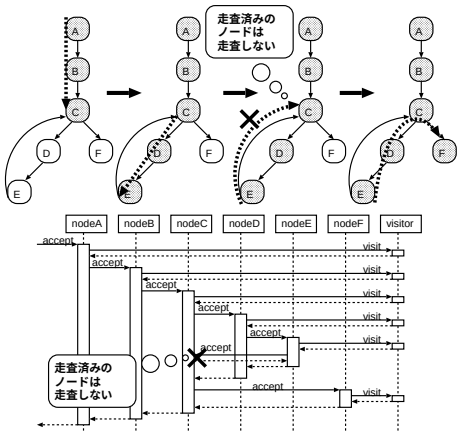

図 6 フローグラフに対する走査

6.4 実現したフレームワーク

実現したフレームワークを図 7 に示す．

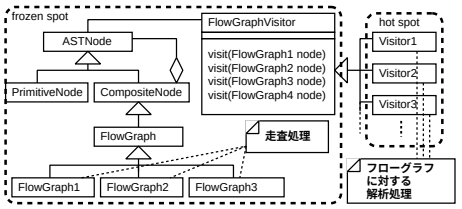

図 7 実現したフレームワーク

実現したフレームワークのホットスポットはフローグラフに対する解析処理である．フローズンスポットはフローグラフと走査処理である．
フローグラフ処理プログラムの開発をおこなう際は，ホットスポットである解析処理の実現をおこなう．具体的には，フローグラフ用の Visitor のサブクラスを定義し，各ノードに対する解析処理を記述する．変更の頻度が低いフローグラフや走査処理は解析処理ごとに共通であるので再利用される．

7 考察

実現したフレームワークの有用性を示すために次の 3 つを考察する．

- ホットスポットとフローズンスポットの妥当性
- 解析処理の追加・変更に対する柔軟性

7.1 ホットスポットとフローズンスポットの妥当性

実現したフレームワークのホットスポットは解析処理であり，フローズンスポットはフローグラフおよび走査処理である．
ソースコードに対する検査は解析処理でおこなう．本開発は PLSE に基づく開発であり，ソースコードに対する検査は新規開発や仕様変更を繰り返しおこなわれる．ノードに対する処理順序を定義した走査処理に変更はなく，解析処理の実現で再利用する．フローグラフは，Java1.5 の言語仕様に基づくソースコードをもとにソフトウェアの制御やデータの流れを表したものである．Java1.5 の言語仕様に変更は起きないことからフローグラフの仕様変更は起きない．さらに，フローグラフは複数の解析処理で共有される．
以上から，解析処理はホットスポットとして妥当であり，フローグラフとその走査処理はフローズンスポットとして妥当であることがわかる．

7.2 解析処理の追加・変更に対する柔軟性

実現したフレームワークはフローグラフから処理を分離している (図 8).

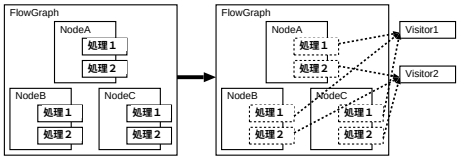

図 8 ノードに対する処理の分離

新たな検査の開発をおこなう場合は解析処理の実現をおこなう．検査内容の変更をおこなう場合は解析処理に対して変更をおこなう．フローグラフから解析処理を分離したことによって，フローグラフに対する変更はなく，解析処理である Visitor のサブクラスを追加変更するだけでよい.

解析処理の変更

検査の解析処理や処理対象とするノードを変更する場合は，対象とする visit メソッドを追加変更するだけでよい (図 9).

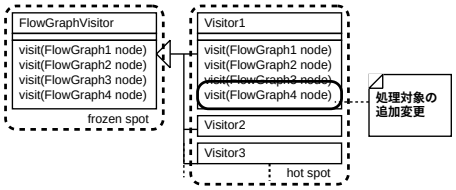

図 9 解析処理の変更

この際，修正箇所は visit メソッドに局所化されており，解析対象となるノードに対して変更をおこなう必要はない.

解析処理の追加

新たに検査を実現する場合は，Visitor クラスのサブクラスを追加して，そのサブクラスに対してノードごとの処理を記述するだけでよい (図 10).

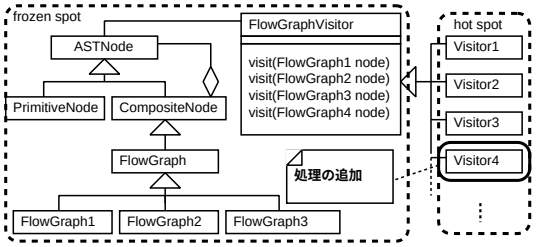

図 10 解析処理の追加

この際，それぞれのノードに対する処理は Visitor のサブクラスに局所化されており，ノードに対して処理を記述する必要はない.

8 おわりに

本研究では，フローグラフ処理機構を持つ CDI ツールのフレームワークを実現した．フレームワークの実現では，CDI ツール開発において共通に再利用できる箇所と変動的に追加変更をおこなう箇所を明確にした．本開発プロセスは PLSE に基づいており，これらの再利用部品を核資産として扱う．核資産開発では，これらの再利用部品の実現・改良をおこなった．製品開発では，共通的な再利用部品を中心に，変動的な再利用部品を組み合わせて CDI ツールを実現した．実現したフレームワークは，本開発の様な PLSE に基づく開発で有効であることを確認した.

参考文献

[1] 岸知二, 野田夏子, 深澤良彰, ソフトウェアテクノロジーシリーズ 4. (全 12 巻) ソフトウェアアーキテクチャ, 共立出版, 2005.

[2] 中田育男, ” コンパイラの構成と最適化”, 朝倉書店, 1999.

[3] Arnold Ken, Gosling James, and Holmes David, *JAVA PROGRAMINNG*, Addison-Wesley, 2006.

[4] L. M. Northrop, ”SEI’s Software Product Line Tenets”, *IEEE Software*, Vol.19 No.4, pp32-40, 2002.

[5] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, *Design Patterns Elements of Reuseable Object-Oriented Software*, Addison-Wesley, 1995.

[6] 茨木俊秀, アルゴリズムとデータ構造２１世紀を指向した電子・通信・情報カリキュラムシリーズ, 昭晃堂, 1989.

[7] 蜂巣吉成, 山本晋一朗, 阿草清滋, オブジェクト指向言語のための細粒度システム依存グラフ, 情報処理学会論文誌, pp.1851-1860, Apr. 1999.