業務アプリケーション向けリバースツールの設計と実装

嶋田 大輔† 渋谷 健介† 大村 裕† 早川 裕志†
NEC 共通基盤ソフトウェア研究所†

1 はじめに

近年の業務アプリケーションは、ビジネスで求められる多様な要求を実現するため、大規模となる傾向がある。現在では生産性と保守性の向上のために、アプリケーションフレームワークを利用した Layer 構成を取ることが多い。

しかし、このような構成では、モジュールの独立性を高められるが、モジュール間の関係がフレームワークの基底クラスや設定ファイルに隠蔽され、全体構成が把握しづらくなるため、保守コストが増加する問題がある。このような背景から、Layer 構成のアーキテクチャの抽出方法に関する研究がある[1]。

本研究では、フレームワークを用いた Java アプリケーションを対象とし、ソースコードのみではなく、フレームワークの基底クラスと設定ファイルに定義されたクラスの依存関係を含めて抽出する機能を開発した。またリバース結果に対し、Layer 情報の付与や不要なクラスの排除を行うルールを指定可能にすることで、アプリケーションの主要な構造を把握可能なリバースツールを試作した。

2 従来のリバースツールの課題

本章では、従来のリバースツールで業務アプリケーションを解析する際の課題を述べる。

- **課題 1**
  モジュール依存関係の一部は、フレームワークの基底クラスに隠蔽されたり、設定ファイルに定義されている。従来のツールでは、プログラムのソースファイルのみを入力とするため、基底クラスや設定ファイルを含めた依存関係を復元できない
- **課題 2**
  アプリケーションの全てのクラスが表示されるため、構造の理解には不要なクラスまでもが表示され、全体構造を把握しづらい

図 1 に従来のツール[2]で業務アプリケーションのリバース結果の例を示した。

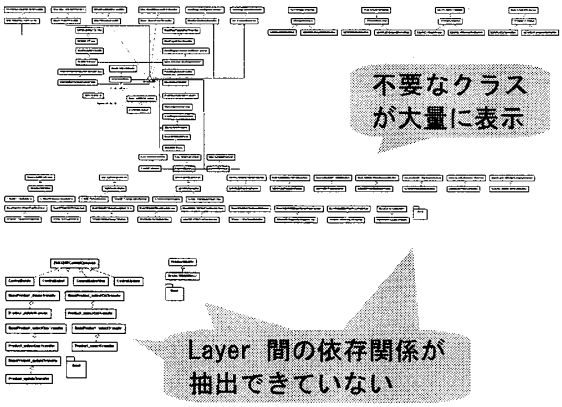

図 1：従来のリバース結果の例

依存関係の不明箇所や、クラスの表示個数が多く、全体構造が把握困難であることが分かる。

3 業務アプリケーション向けリバースツール

本章では、本研究で試作したリバースツールについて述べる。

ソースコードと設定ファイルから依存関係を抽出し、フレームワークごとにルールを用意することで、全体構造を理解する手掛かりとなるビューを表示可能にしている。

図 2 に本ツールの構成を示す。

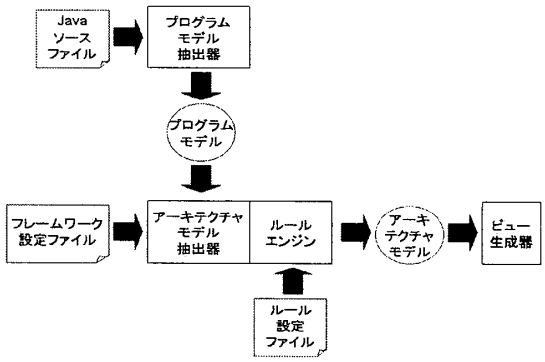

図 2：本ツールの構成

本ツールは、モデル抽出器、ルールエンジン、ビュー生成器から構成されている。

ツール内部では、リバース結果をプログラムモデル、アーキテクチャモデルで表現し、表示形式に合わせてビュー形式ファイルを生成する。

Design and Implementation of Reverse Engineering Tool for Business Applications
†SHIMADA, Daisuke (d-shimada@ak.jp.nec.com)
†SHIBUYA, Kensuke (k-shibuya@bq.jp.nec.com)
†OHMURA, Yuu (y-ohmura@cb.jp.nec.com)
†HAYAKAWA, Masashi (m-hayakawa@ce.jp.nec.com)
Common Platforms Software Research Laboratories, NEC Corporation(†)

ビューは、クラス図、画面遷移図※1、コンポーネント配置図※2を扱っている。

3．1　アプリケーションの俯瞰的な可視化

図３に本ツールで業務アプリケーションの構造を可視化した結果を示した。

Layer 構成のアプリケーションの全体構造を把握するため、画面ファイル、画面イベントアクションクラス、ロジッククラス、データアクセスクラス等の構造の骨格となるクラス間の関係の抽出と、クラスが所属する Layer 情報の付与を行っている。

また、大規模なアプリケーションでは、構造とは直接関係のないユーティリティ等のクラスが多く存在するため、それらをルールによって排除することで、より全体構造を把握し易くすることを目指している。

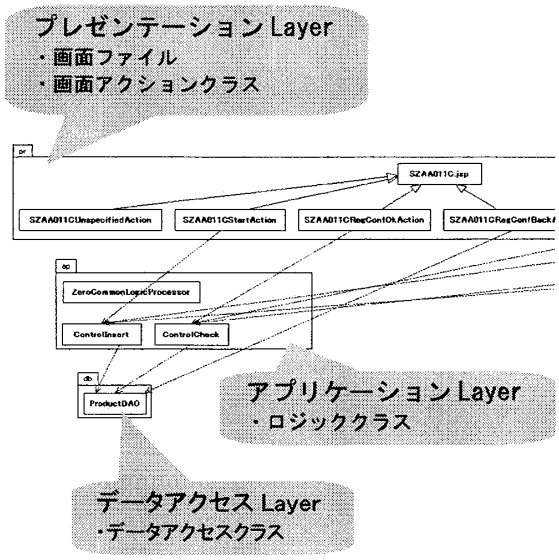

図３:本ツールのリバース結果の例(クラス図)

図１と比較して、主要クラス間の依存関係の表示、Layer 情報の付与、不要クラスの排除を行い、アプリケーションの全体構造が把握し易くなっていることが分かる。

3．2　アプリケーション構造の規約違反検出

アプリケーションの構造には、そのアーキテクチャごとに規約が存在する。近年では製造を別の拠点で行うオフシェア形態を取ることも多いため、アーキテクチャを設計した際の規約を実装で守られているかどうかの受入検査コストが生じる。

※1 UML ステートマシン図形式で表現
※2 UML クラス図形式で表現

そこで、規約をツールで検出可能にすることで、受入検査や品質確認のコストを削減できるといえる。本ツールのリバース結果に対し、規約の違反を自動検出可能にすることで、アプリケーションの構造上の品質確認を容易にすることを試みた。図４では、図３に対し違反箇所を色付けして表示している。

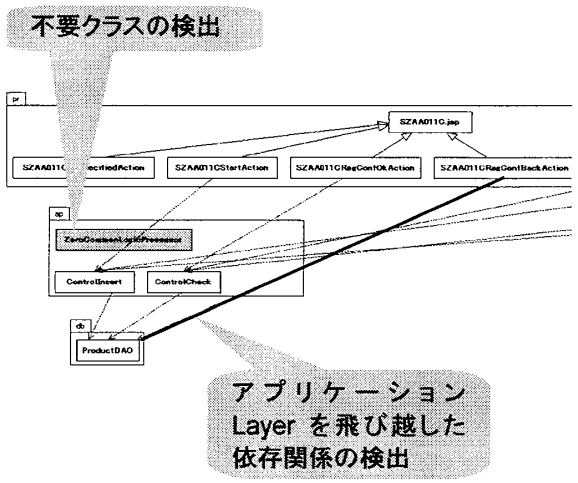

図４:規約違反の検出結果の例(クラス図)

検出結果として、Layer を飛び越したクラス間の依存関係と、他のクラスから利用されない不要なクラスを検出したことが分かる。

4　おわりに

本稿では、フレームワークを利用した業務アプリケーションから、全体構造を把握可能なリバースツールを提案し試作を行った。さらに、リバース結果に対し、規約違反の検出機能を試作し、アプリケーションの構造上の品質確認を容易にした。そして、本ツールを典型的な業務アプリケーションを用いて評価し、有効性を確認した。今後は、対象のフレームワークの種類を広げ、本ツールの有効性の範囲を確認する予定である。

参考文献

[1] 方学芬、玉井哲雄「システム理解のための分散システムアーキテクチャの抽出」情報処理学会第 137 回ソフトウェア工学研究会、2002 年 3 月

[2] 株式会社チェンジビジョン、システム設計支援ツール JUDE、 http://jude-users.com/ja/