アスペクト指向を応用したディペンダブル基盤ソフトウェア

千葉　滋
東京工業大学 大学院 情報理工学研究科

概要

ソフトウェアをディペンダブルに動作させるための基盤ソフトウェアの研究をおこなっている。近年の web アプリケーションの普及によって、過負荷時に web アプリケーション・サーバの安定性を確保することが求められている。従来技術で作られた web アプリケーション・サーバは、利用者からのリクエストが殺到すると急激に性能が悪化して、最悪の場合はシステム・ダウンに至る。そのような急激な性能悪化は望ましくない。ディペンダブルなシステムとするためには、リクエスト数の増加に対して性能が比例的に悪化する程度におさえる必要がある。我々は web アプリケーション・サーバ自体の改良により信頼性を高める技術と、アスペクト指向プログラミングによりアプリケーション・ソフトウェアのプログラムを変換し、信頼性を高めるための機能を自動的に埋め込む技術の組み合わせにより、この問題を解決しようとしている。

1　今年度の主な成果

今年度、我々はまずアスペクト指向システム GluonJ を試作した。このシステムは通常の Java 言語で書かれたモジュール同士を XML 言語で書かれたアスペクト記述にしたがって結合する機能を提供する。これによって、例えばディペンダビリティを高める機能を提供するモジュールを、アプリケーション本体の機能を提供するモジュールに結合し、ソフトウェア全体をディペンダブルにすることが可能になる。ディペンダブル機能を提供するモジュールをアプリケーション本体に、従来の技法で組み合わせようとすると、アプリケーション本体側のプログラムをそれが可能になるように準備しておく必要があり、必ずしも容易ではなかった。アスペクト指向技術を用いると、アプリケーション側のプログラムに特段の準備をほどこさなくても、そのような組合せが可能になる。

我々はさらに GluonJ の応用研究として、実際的な Web アプリケーションを開発し、それに対してディペンダブル機能をアスペクト指向技術で組み込む実証実験をおこなった。この Web アプリケーションは、全国の河川の水位情報をリアルタイムにわかりやすく表示するアプリケーションを模している。例えば特定の河川の水位を時系列にそってグラフ化して表示しする機能などをもつ。このアプリケーションは、台風接近時などにアクセスが殺到すると、システムが過負荷状態になり、河川の水位情報の取得処理が遅れがちになり、正しい情報を発信できなくなってしまう。

このような状態を避けるためには、水位情報の取得処理のような大切な処理は、優先度を高くして実行しなければならない。しかしこのアプリケーションは、類似の商用システムに似せて Java 言語で書かれているため、大切な処理を実行しているスレッドの優先度を選択的に上げるような制御はできない。また、やはり類似の商用システムに似せて、このアプリケーションは汎用のアプリケーションフレームワーク上に構築してある。したがって、仮に Java 言語がスレッドの優先度を細かく制御する機能を持っていたとしても、アプリケーションフレームワークもそれに対応していなければ、アプリケーションにとって大切な処理を優先的に実行するようなことはできない。

そこで我々は一般的な Java 言語で書かれたアプリケーションであっても、大切な処理を優先的

に実行できるようにし、システムが過負荷状態であっても、最低限の処理品質を維持できるようにした。具体的には、システムの負荷状態によっては、自主的に処理を一時停止して、大切な処理が先に実行されるように配慮するモジュールを作成し、これを GluonJ を使ってアプリケーション本体に結合した。GluonJ を使うことで、処理の優先度を制御するモジュールの存在を考えずに書かれたアプリケーション本体のプログラムは変更せずに、優先度を制御するモジュールを組み合わせることが可能になった。

2 GluonJ

プログラムは一度完成した後に機能の追加や変更が必要になることが多い。そのためプログラムを後から簡単に変更できるように設計することは重要である。コンポーネントを用いた開発はプログラムを後から簡単に変更できるよう開発するための手法の 1 つである。コンポーネントとは関連する機能を 1 つにまとめた部品のことで、コンポーネントを用いた開発とはコンポーネントを組み合わせてプログラムを作るという開発手法である。

コンポーネントを組み合わせてプログラムを開発するときには、コンポーネントの内部を変えずにコンポーネントの組み合わせを変更できるようプログラムを設計するべきである。コンポーネントの組み合わせの変更は、プログラムを後から変更するときに必要となる。もしコンポーネントの組み合わせを変えるたびにコンポーネントの内部を変更しなければならないとすると、そのコンポーネントの再利用性は高いとは言えない。

GluonJ は、コンポーネントの内部、つまりそのプログラムを変更せずに、コンポーネントの組み合わせを変えられるようにするプログラミング・システムである。基本的なアイデアは、アスペクト指向プログラミングに基づくが、AspectJ 言語などと異なり、GluonJ ではいわゆるアスペクトをコンポーネントの実装には用いない。コンポーネントの組み合わせにだけアスペクトを用いる。これを強調するため、GluonJ ではいわゆるアスペクトのことを glue コードと呼ぶ。また glue コードは、AspectJ 言語のアスペクトと同等のものではなく、インタータイプ宣言やポイントカット・アドバイスというアスペクト指向プログラミングのための基本機能だけでなく、いわゆる dependency injection の機能も備えている。

2.1 コンポーネントの組み合わせ

例として、オンライン書店プログラムの一部で、利用者の注文をデータベースへ送信するプログラムを考える。このプログラムは OrderingService コンポーネントと MySQLOrderDAO コンポーネントからなる。OrderingService コンポーネントの addShippingRateAndOrder メソッドは利用者が購入しようとしている本の総額を調べ、総額が一定以下ならば総額に送料を足して発注管理データベースに注文を送信する。注文の送信には MySQLOrderDAO コンポーネントの order メソッドを用いるとする。

このプログラムに対する変更として，以下の 2 つの変更を考える.

- 変更 1: データベースシステムの変更

  注文発注プログラムが利用するデータベースを別のものに置き換える。新たに利用するデータベースにはアクセスを最適化するためのヒントを与えることができるとする。新たに利用するデータベースにアクセスするためのコンポーネントは OracleOrderDAO として提供され、このコンポーネントの sendHint メソッドを呼ぶことで、最適化のヒントを与えるとする。

- 変更 2: 安全性の強化

  セキュリティを強化するため、プログラムを実行するユーザにデータベースへのアクセス権があるか、アクセス前に検査する。アクセス権の検査は、アクセスの直前に SecurityManager コンポーネントの check メソッドを呼ぶことでおこなう。SecurityManager コンポーネントは、SecurityManager クラスのファクトリメソッドを呼んで

生成する。

2.2 Java 言語による実装

一般的なオブジェクト指向言語である Java を用いた実装では、コンポーネントを結びつけるコードが、コンポーネントの実装内部に含まれてしまう。このためコンポーネントの組み合わせを変更するには、コンポーネントの実装も修正しなければならないことが多々ある。

具体的に上の例を使って説明する。まず OrderingService コンポーネントの変更前の元のプログラムは次のようになるだろう。

```
public class OrderingService {
  private MySQLOrderDAO dao;   // 宣言
  public OrderingService() {
    dao = new MySQLOrderDAO(); // 初期化

  }
  public void addShippingRateAndOrder (
      User user, Book[] books) {
    int total = 0;
    for (int i = 0; i < books.length;
                                  i++)
      total += books[i].getPrice();
    if (total < 1500)
      total += getShippingRate(user);
    // 呼出し
    dao.order(user, books, total);
  }
}

public class MySQLOrderDAO {
  public void order (User user,
      Book[] books, int total) { ... }
}
```

OrderingService コンポーネントと MySQLOrderDAO コンポーネントとを結びつけるコードは次の 3 箇所である。(1) dao フィールドの宣言、(2) コンストラクタ内で dao フィールドの値を初期化しているコード、それから (3) addShippingRateAndOrder メソッドの最後におこなう order メソッドの呼び出しである。

これらのコードは、コンポーネントの組み合わせを変える際には修正が必要になる。例えば先に述べた変更を加えると、これらのコードを次のように修正することになる。

- 変更 1: データベースの変更

  dao フィールドの型を OracleOrderDAO に変更。MySQLOrderDAO のコンストラクタ呼び出しを OracleOrderDAO のコンストラクタ呼び出しに変更。order メソッドの呼び出しの直前に dao フィールドの指す OracleOrderDAO コンポーネントの sendHint メソッドの呼び出しを追加。

- 変更 2: 安全性の強化

  OrderingService コンポーネントに SecurityManager 型のフィールドを追加。追加したフィールドを初期化するコードを追加。addShippingRateAndOrder メソッドの冒頭に、追加したフィールドが指すコンポーネントの check メソッドの呼び出しを挿入。

コンポーネントの組み合わせを変えるたびに、コンポーネントの実装を修正していては、コンポーネントを使う利点が生きない。完成したコンポーネントの修正にはコストがかかる。少なくとも、実装を修正したらコンポーネントの単体でのテストを最初からやり直さなければならない。

2.3 GluonJ による実装

我々が開発した GluonJ を使って実装すると、コンポーネントを結びつけるコードはコンポーネントの実装の外に分離できる。先のプログラムを GluonJ で書き直したものを下に示す。

```
public class OrderingService {
  private User userdata;
  private Book[] booksdata;
  private int total;
  public void addShippingRateAndOrder (
      User user, Book[] books) {
    total = 0;
    for (int i = 0; i < books.length;
                                  i++)
      total += books[i].getPrice();
    if (total < 1500)
      total += getShippingRate(user);
    userdata = user;
    booksdata = books;
  }
}
```

SubCa の定義は Java 言語で実装した場合と変わらない。

GluonJ で実装した Comp コンポーネントのプログラムには、SubCa コンポーネントを結びつけるためのコードが含まれていない。それらのコードは別途、以下の glue コードの中に XML 言語で記述する。

```
<glue>
  <reference>
    MySQLOrderDAO OrderingService.
        aspect = new MySQLOrderDAO();
  </reference>
  <behavior>
    <pointcut>
      execution(void OrderingService.
          addShippingRateAndOrder(..))
    </pointcut>
    <after>
      MySQLOrderDAO.aspectOf(this)
              .order(userdata,
                     booksdata, total);
    </after>
  </behavior>
</glue>
```

一般に、コンポーネントを結びつけるコードは、先に述べたようにフィールドの宣言、その値の初期化、そのフィールドがさすコンポーネントのメソッドの呼び出しの 3 つなので、GluonJ はそれぞれに言語機構を用意している。<reference> タグがフィールドの宣言と初期化、<behavior> タグがメソッドの呼び出しを記述するのに使われている。

<behavior> タグは、AspectJ 言語のポイントカット・アドバイス機構にほぼ対応する。一方、<reference> タグは AspectJ 言語のインタータイプ宣言に対応する。ただし AspectJ 言語の場合と異なり、GluonJ 言語の <reference> タグは、新規のフィールドを追加するだけでなく、コンポーネント内の既存のフィールの初期値を変更することもできる。

このようにコンポーネントを結びつけるコードを分離することで、コンポーネントの組合せを変える際に、コンポーネントの実装を修正する必要がなくなる。先に述べたデータベースの変更や、安全性の強化も、glue コードを修正するだけですむ。

3 ディペンダビリティへの応用

GluonJ を用いると、過負荷時のディペンダビリティを考慮せずに開発された Web アプリケーションに、ディペンダビリティの制御をおこなうモジュールを組み込むことができる。このとき、元のアプリケーションには一切変更を加えなくてよいのが特徴である。

我々は、Linux、Java、Tomcat, JBoss といったごく標準的な OS やフレームワークを用いて構築した web アプリケーションに、GluonJ を使って実際にディペンダビリティの制御をおこなうモジュールを組み込めることを確かめた。組み込みに当たっては、Web アプリケーション本体を開発した開発者とは別の開発者が作業した。この際、glue コードの記述量や書きやすさが問題となる。実際、細かい制御をおこなおうとすると、web アプリケーション内の多数の場所からディペンダビリティの制御をおこなうモジュールを呼び出さなければならない。このため glue コードの記述量自体はけして少なくなかった。しかし、我々は glue コードの記述作業を軽減するため、制御モジュールの呼び出しをおこなうべき箇所の発見を手助けするツールも開発し、全体として従来の Java 言語だけで開発する場合と比べ、相対的に少ない作業量でディペンダビリティの制御をアプリケーションに追加できるようにした。

参考文献

[1] S. Chiba, R. Ishikawa. Aspect-Oriented Programming beyond Dependency Injection. ECOOP 2005 – Object-Oriented Programming, LNCS 3586, Springer, 2005, pp.121-143.

[2] 石川零・千葉滋. アスペクト指向プログラミングと Dependency Injection の融合. Dept. of Math. and Comp. Sciences Research Reports C-220, Tokyo Institute of Technology, February, 2006.

[3] 日比野秀章・光来健一・千葉滋. J2EE アプリケーションにおけるアプリケーションレベルスケジューリング. 第 3 回ディペンダブルソフトウェアワークショップ (DSW'06), 日本ソフトウェア科学会, 東京大学, 2006 年 1 月.