2K-3

Swing/SWT 共通コードジェネレータの開発

大城　繁鷹　　塚本　享治
東京工科大学メディア学部メディア学科

1　はじめに

現在では、ほとんどのクライアントアプリケーションに GUI が実装されているが、それは同時にプログラマー側へ課される負担を増加させている。複雑な GUI プログラムはソースコードの記述を難解にし、後の管理や修復を困難なものにしてしまう。また、アプリケーションを実行する環境によって、開発に利用する言語やライブラリを変更しなくてはならないという問題もある。

本研究では上記の問題を解決するために、「XML で定義した GUI 情報を XSLT 変換することにより、一度に JavaSwing、JavaSWT の２種類のソースコードを同時に生成する」というコードジェネレータを開発した。

2　コードジェネレータの構成

本システムは JavaSwing プログラムを生成する XML2Swing 、JavaSWT プログラムを生成する XML2SWT、それらふたつのシステムを統合・管理している XML2GUI の３つのシステムで成り立っている。

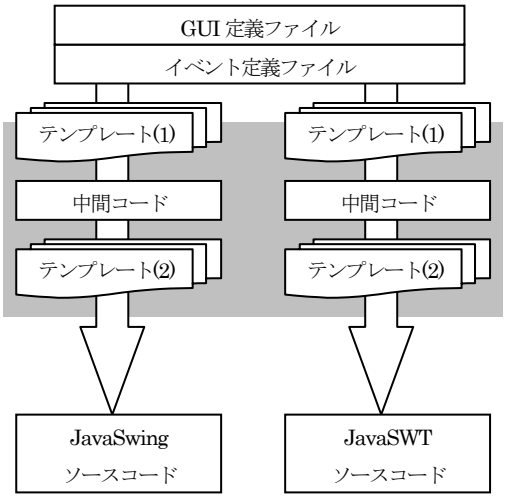

図１：GUI コードジェネレータの処理フロー

“Development of Swing and SWT unified code generator”, by Shigetaka Ooshiro, Michiharu Tukamoto, Tokyo University of Technology

本システムではまず、①GUI とイベントの情報を XML でそれぞれ整理したコードを定義ファイルに記述する。②それをジェネレータに読み込ませると、内部で JavaSwing と JavaSWT の２本の変換ルートに分岐され、③テンプレート(1)による中間コードの生成を経て、④テンプレート(2)によるソースコードの生成がそれぞれのルートで行われる。

3　コードジェネレータの実装

3.1　XML による GUI 情報の定義

GUI とイベントの情報を XML で整理すると以下のようになる。

```
<Window>
  <Button id="B_SAMPLE"
  text="ここを押してください" widht="280" height="120"/>
  <Label id="L_SAMPLE" width="280" height="30"/>
</Window>
```

```
<event id="B_SAMPLE">
  L_SAMPLE.setText("ボタンが押されました");
</event>
```

図２：GUI とイベント情報の XML 表記

タグの要素に各コンポーネントを、属性にそのコンポーネントに関する情報を追加していく。さらにそのタグを GUI の視覚情報に基づいて階層化させる。

3.2　XSLT によるコード変換・第１段階

最初の変換で定義ファイルから XML の中間コードを生成する。GUI を簡易に表記しただけのコードを JavaSwing、JavaSWT のプログラムコードに近い形に整形する。この段階では XSLT で以下の処理を行う。

(1) コアコードの提供

すべてのアプリケーションに共通するパッケージのインポート、メインクラスの生成、メインメソッドの生成、メインウィンドウの生成、デフォルトのレイアウトの設定といったコアコードはあらかじめまとめて準備しておく。

(2) イベントリスナの実装

定義ファイル内を全検索してインターフェイスの実装が必要なコンポーネントが存在した場合、イベントリスナをメインクラスに実装する。

(3) グローバル変数の宣言

定義ファイル内を全検索してイベントに使用される可能性のあるコンポーネントが存在した場合、オブジェクトのIDをグローバル変数として宣言する。

(4) コンポーネントコードの分解

ひとつのコンポーネントに関連したコードはひとつのタグにまとめられているが、これをプログラムのメソッドごとに分解して新たにコードを組み立てる。

(5) コンポーネントの親子関係の決定

各コンポーネントのタグはGUIの視覚表現に基づいて階層的に位置付けられている。Xpathを読み取り、コンポーネントの相対位置を判定して各コンポーネントどうしの親子関係を特定し、コードを組み立てる。

(6) イベント定義ファイルとの結合

最後に、GUI定義ファイル内のコンポーネントのIDでイベントメソッドを作成し、そこへ、同じIDを持つイベント定義ファイル内のイベントコードを挿入する。

3.3 XSLTによるコード変換・第２段階

２度目の変換で中間コードからソースコードを生成する。XMLで記述されているプログラムコードをJavaコードに変換しなおして出力する。

3.4 アプリケーションの完成

最後に出力されたJavaSwing、JavaSWTのソースコードをそれぞれコンパイルして実行する。なお、これらの一連の処理はすべてAntにより自動化されている。

4 統合開発環境の構築

また、このコードジェネレータのシステムを利用するための統合開発環境のアプリケーションを併せて開発した。この開発環境では定義ファイルの作成からコード変換、ソースコードのコンパイルとアプリケーションの実行までのすべてを行うことが可能である。

この開発環境自体も本システムの仕様で開発されたものであり、この開発環境上でまったく同じ開発環境のアプリケーションを自己生成させることも可能である。

図３：統合開発環境

さらに、JavaSwingで開発した開発環境からJavaSWTのアプリケーションを作ることも可能であり、またその逆も可能である。これは変換に関連した一連の処理がbuildファイル上にまとめられており、OSへの直接の命令でAntコマンドを起動しているからである。

5 検証と考察

本システムと開発環境を利用して、実際にいくつかアプリケーションを生成した。直接Javaプログラミングを行った場合と比較して、格段に早く、簡単にプログラミングを行うことが可能であり、かつ完成度の高いアプリケーションの生成が可能であることも確認された。

こうして、簡単なGUIの記述のみでGUIプログラムが生成できることが実証された。また、本システムではJavaSwingとJavaSWTの２種類の仕様のGUIを同時に生成した。このようなタイプのジェネレータは珍しく、今後多方面で利用できると思われる。

6 今後の展望

今回はJavaSwing、JavaSWTといったクライアントアプリケーションの生成のみを行ったが、さらに出力先にJSPなどを追加することによって、大きなWebシステムの一部としてこのコードジェネレータの機能を提供することも可能となるだろう。

参考文献

[1] Jack Harrington，Code Generation In Action，MANNING，2003

[2] 金長源，XML技術を用いたJavaSwingプログラム生成ツールの開発，第66回全国大会論文，2003

[3] 中嶋祐介，XSLTを用いたSwingプログラムジェネレータの開発とその自己生成による検証，第67回全国大会論文，2004