6N-3

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

中嶋　祐介　　鎌倉　隆司　　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

近年のクライアントアプリケーションはそのほとんどが GUI を用いて構築されているが、GUI アプリケーション開発における問題点として、実装段階におけるプログラム内にコンポーネント情報が混在することが挙げられる。そのため、コード内のコンポーネントの再利用性が低く、仕様変更にも柔軟に対処することができない。

そこで、我々は JavaSwing アプリケーションの開発を題材にコードジェネレータ[1](SwingGUI ジェネレータ)を開発することによって、コンポーネントのレイアウト情報とロジックを分離し、再利用性を高めた。

しかし、SwingGUI ジェネレータでは GUI しか生成することができず完成度が低かった。そこで、さらにイベント処理をコードジェネレータに実装し、完成度を高めたので報告する。

2　SwingGUI ジェネレータの問題点とその解決方針

SwingGUI ジェネレータの機能を拡張するにあたって問題となる点とその解決方針を述べる。

2.1　SwingGUI ジェネレータの問題点

SwingGUI ジェネレータを利用した場合に以下の 3 つの問題点が挙げられる。

（1）XML の記述の煩雑さ
（2）イベント処理が扱えない
（3）JavaSwing しか生成できない

2.2　問題点の解決方針

（1）～（3）の問題点をそれぞれ以下のような方針で解決した。本研究では特に（2）に力をいれた。

（1）XML の記述を GUI を持った開発サポートツールを開発する。それを使い XML を自動的に生成することで XML の記述の煩雑さが解消できる。
（2）イベント処理を実装することで完全にコードジェネレータのみで Swing プログラムを生成することができ、完成度が高まる。
（3）Swing 以外に SWT を生成するコードジェネレータ(SWTGUI ジェネレータ)を開発することで機能の拡張を図る。

3　Swing プログラムジェネレータの開発

SwingGUI ジェネレータ[2]の機能を拡張した Swing プログラムジェネレータを開発する際の設計方針を述べる。

3.1　生成のワークフロー

Swing プログラムジェネレータでコードを生成するまでには 4 種類のユーザ定義ファイルを受け取り、中間ファイルを生成しながら XSLT 変換を行う。

1 回目の変換では Transform1.xsl を用いて 4 種類のユーザ定義ファイルを 1 つの XML ファイルにまとめる処理を行う。2 回目の変換では Transform2.xsl を用いて 1 回目の変換で生成された XML2Swing.tmpxml を解析し、必要なコンポーネントの変数宣言文を作成する。3 回目の変換では Transform3.xsl を用いて 2 回目の変換で生成された XML2Swing.tmp2xml を解析し、コンポーネントにレイアウト情報とイベント処理を関連付ける。4 回目の変換で Transform4.xsl を用いて 3 回目の変換で生成された XML2Swing.tmp3xml をコンパイル可能な形に整形する。計 4 回の変換を行い、目的の Java ソースコードを生成する。(図 1)

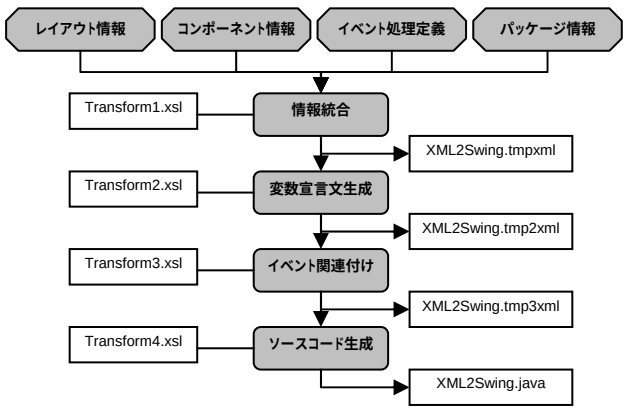

図 1　コード生成の流れ

"Development of Swing program generator and its self-generator",
by Yusuke NAKASHIMA, Ryuji KAMAKURA, Michiharu TUKAMOTO, Tokyo University of Technology

3.2 イベント処理の実現方法

図 2 はボタンを押した時に警告ダイアログを表示するプログラムを生成する XML ファイルの一部である。このようにコンポーネントとイベントの関連付けを実現するためコンポーネント情報用 XML ファイル(図 2.Button.xml)とイベント処理定義用 XML ファイル(図 2.JOptionPane.xml)に<Command>要素を持たせ、両方の<Command>要素が一致した場合のイベントを処理するように設計した。

Button.xml の<Command>要素と JOptionPane.xml の<Command>要素が一致するため、btn1 のコンポーネントが押された場合警告メッセージが表示されるようになっている。

また、<Action>要素に ActionListener や MouseListener など、どの種類のイベントなのかを記述すると動作が切り替わる。

Button.xml

```
<?xml version="1.0" encoding="Shift_JIS"?>
<ButtonComponent>
    <Button name="btn1">
        <Caption>Button</Caption>
        <Command>push</Command>
    </Button>
</ButtonComponent>
```

JOptionPane.xml

```
<?xml version="1.0" encoding="Shift_JIS"?>
<JOptionPane>
    <ShowMessageDialog>
        <Command>push</Command>
        <Action>ActionListener</Action>
        <Component>cp</Component>
        <Message>Button is pushed.</Message>
        <Title>Warning</Title>
        <MessageType>WARNING_MESSASGE</MessageType>
    </ShowMessageDialog>
</JOptionPane>
```

図 2　コンポーネントとイベントの関連付け

4　自己生成

SwingGUI ジェネレータにイベント処理を実装することによって Swing プログラムジェネレータで Swing プログラムジェネレータ自身を生成することができるようになる。これを自己生成と呼ぶ。図 3 は自己生成による検証の流れを示した図である。

具体的には Swing プログラムジェネレータのアルゴリズムを備えたエディタ(Swing 開発サポートツール)を開発し、それを用いて Swing 開発サポートツール自身を生成することで自己生成した。

自己生成に必要な機能として以下の 3 つがある。

・GUI を生成できる

・イベントハンドラを生成できる

・Ant スクリプトを生成できる(オプション)

これら 3 つを Swing 開発サポートツールで生成することができれば自己生成できたと言える。

自己生成を行う目的は Swing 開発サポートツールの機能と Swing プログラムジェネレータを改良できるかを検証するためである。

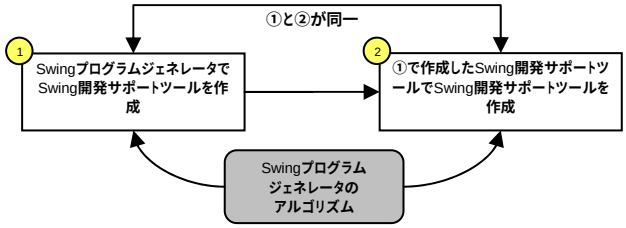

図 3　自己生成の流れ

5　検証

検証のため、Swing 開発サポートツールを用いて自身を生成する実験を行った。

実験の結果、自己生成を行うことができ、Swing プログラムジェネレータの完成度の高さや有効性を示すことができた。また、Swing プログラムジェネレータと SWTGUI ジェネレータとを Ant スクリプトを用いて連携させることにより両方のコードを生成することができた。

6　おわりに

開発した Swing プログラムジェネレータによって、完全にコードジェネレータのみで Swing プログラムを生成することが可能となった。

しかし、まだ実装していないコンポーネントやイベントクラスがあるため、それらを実装し、さらに完成度を高める必要がある。また、コード変換にかかる時間を短縮するために XSLTC[3]を用いる方法も検討したい。

謝辞

最後に本研究に関して貴重な助言をいただいた本学橋本勉教授、棟上昭男教授、及び金沢典子講師に感謝いたします。

参考文献

[1]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 回全国大会論文集, 2003

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

[3]XSLTC, http://xml.apache.org/xalan-j/xsltc_usage.html

[4]関口宏司, Apache Ant, 技術評論社, 2004