社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2006－FI－82（5）
2006－DD－54（5）
2006／3／22

# 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

近年の GUI の発展は、開発者へ課される負担の増加に繋がっている。ソースコードは複雑になり後の管理や修復を困難なものにしてしまい、また、実行環境によって開発手法を変更しなくてはならないという問題もある。

本研究ではこの２つの問題を解決するために「GUI を XML で管理したスクリプトから複数のソースコードを自動で生成する」というコードジェネレータを開発した。XSLT を変換処理に用い、実際に JavaSwing と JavaSWT のソースコードを生成することでシステムの有用性を検証した。

# Development of Swing and SWT unified code generator

Shigetaka Ooshiro　Michiharu Tukamoto
Tokyo University of Technology

Recently Development of GUI leads to increase of developer's burden. A source code becomes complicated and will make manegement and restoration difficulty. Furthermore, it must be changed development technique by execution environment.

In this study, We developed a code generator to solve these two problems. It generates plural source codes from the script which managed GUI by XML automatically. We really inspected utility of a system by generating a source code of JavaSwing and JavaSWT using XSLT for conversion processing.

## 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。しかし、その発展は同時にプログラマー側へ課される負担を増加させてもいる。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。

こうした状況を打開するため、「XML を使って GUI 情報を管理する」という動きが最近活発化している。本研究ではそれらの諸研究を踏まえ、「わかりやすい GUI プログラミング」と「複数の GUI ライブラリへの対応」を実現するコードジェネレータを開発した。このシステムは、XML で定義した GUI 情報を XSLT 変換することにより、JavaSwing、JavaSWT の２種類のソースコードを同時に生成する、というものである。本稿では、このシステムの仕組みを説明するとともに、より効率的に GUI プログラミングを行うにはどうすればよいのか、という問題について述べる。

## 2. GUI プログラミングの現状

現在、GUI プログラミングが抱えている主な問題は以下の２点である。

(1)複雑なソースコードの記述

第１の問題点は、ソースコードの記述の仕方にある。たったひとつボタンを作成するだけで複数行のコードの記述を要し、後に修正を加える際の

再利用性の低下にも繋がる。さらに、プログラムのコアコードとデザインに関わるコードが混在することも開発において効率が悪い。

(2)複数の開発・実行環境の存在

第 2 の問題点は、PC の OS や携帯電話の機種ごとにプログラムの実行環境がいくつも存在することである。この場合、実行環境ごとに開発手法を変更する必要があり、開発者の負担が増加する。

## 3. XML をベースにしたコードジェネレータ

### 3. 1. コードジェネレータの導入

こうした状況を打開するために本研究では GUI プログラムの開発支援ツールとして図1のようなコードジェネレータを開発する。

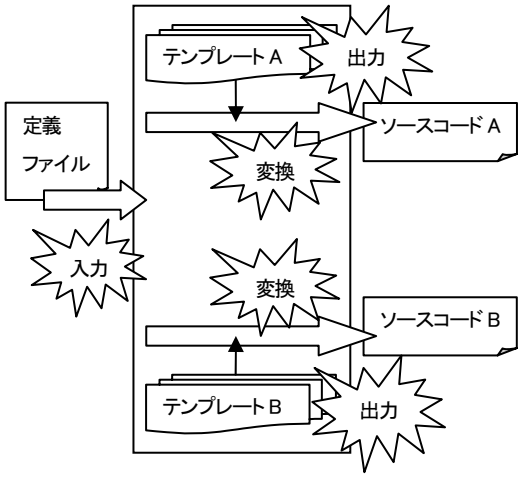


図1　コードジェネレータの構成

コードジェネレータとは定義ファイル上にある決められたフォーマットで必要最小限の情報を記述し、それを内部でテンプレートを適用させて変換処理を行うことで、結果としてプログラムのソースコードを出力するツールである。この仕組みを利用することで、プログラマーは「複雑なソースコードの記述」を行う必要がなくなる。また、ジェネレータ内の変換ルートを 2 分化して同時に 2 種類以上のソースコードを生成することができるようにする。こうして「複数の開発・実行環境の存在」にとらわれる必要もなくなる。

### 3. 2. XML の導入

本システムでは定義ファイルの記述に XML を用いる。XML 自体の記述、及び XSLT を利用した変換が容易に行えるという点に加えて、GUI を XML で表現するという本研究ならではの以下のようなメリットもある。

例えばボタンをひとつ作成する際、複数行にわたる乱雑したコードを、リスト 1 のようにひとつのタグにまとめることができる。

リスト1　ボタンのプロパティの XML 化

```
<Button    id = ”button”
           text = “This is a Button.”
           width=”200”
           height=”40”
           layout=”vertical”
           action=”pushedEvent”
/>
```

GUI はウィンドウの上にパネルが、そのパネルの上にボタンが……というように階層的に組み立てられている。これを XML で表現するとリスト 2 のようになる。

リスト2　GUI 階層の XML 化

```
<Window>
    <MenuBar/>
    <PanelA>
        <Button1/>
        <Button2/>
    </PanelA>
</Window>
```

### 3. 3. コードジェネレータの仕組み

実際に構築するコードジェネレータは図 2 のような構成になっている。①まずユーザーは GUI 情報を定義した XML ファイル(1種)とイベント情報を定義した XML ファイル(JavaSwing 用と JavaSWT 用の計 2 種)を用意する。②ジェネレータはこの 3 種の定義ファイルを読み込むと、JavaSwing 変換ルートと JavaSWT 変換ルートに処理過程を分岐する。③ここで 1 度目の変換作業が発生し、ひとつ目の XSLT テンプレートによる変換でそれぞれに XML の中間コードが生成される。④さらに 2 度目の変換作業が発生し、ふたつ目の XSLT テンプレートによる変換でそれぞれに Java のソースコードが生成される。なお、この一連の工程は Ant によりすべて自動化されている。

## 4. Swing/SWT コードジェネレータの実装

本稿では「クリックするとラベルに文字が表示

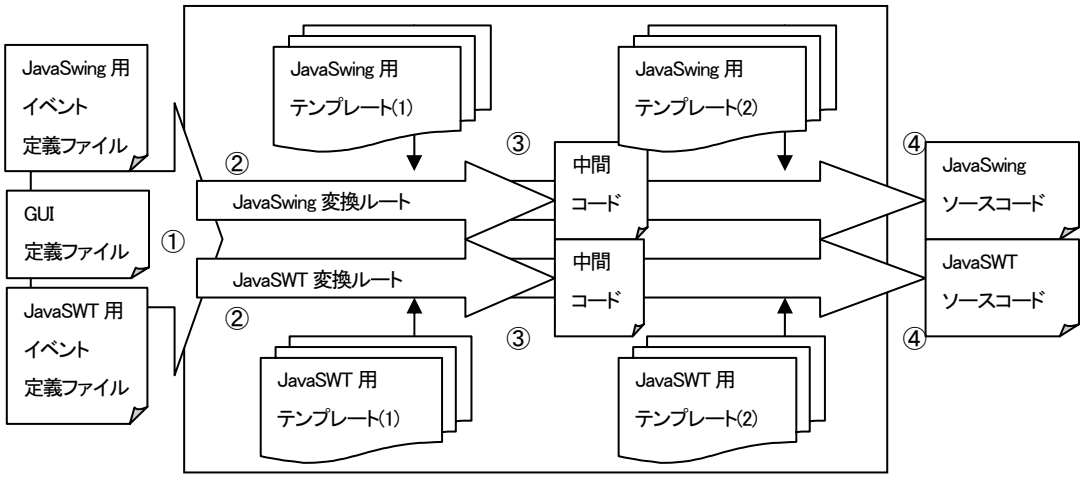


**図2　Swing/SWT コードジェネレータの構成**

されるボタンをひとつ作成する」という情報定義から、JavaSwingとJavaSWTのソースコードを同時に生成する過程を例に、コードジェネレータのシステムを説明していく。

### 4. 1. GUI 情報の XML 定義

目的とするアプリケーションの GUI、及びその階層関係を図 3 に、各々のコンポーネントが持つプロパティを表 1 にまとめる。

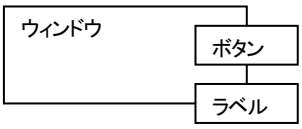


**図3　サンプルプログラムの GUI イメージ**

**表1　各コンポーネントのプロパティ**

| | |
|---|---|
| ウィンドウ | ID、タイトル、横幅、縦幅 |
| ボタン | ID、テキスト、横幅、縦幅、イベント |
| ラベル | ID、テキスト、横幅、縦幅 |

これらの情報をもとにサンプルプログラムをリスト 3 のような XML で表現することができる。

### 4. 2. イベント情報の XML 定義

イベント情報はアプリケーションに柔軟性を持たせるために JavaSwing、JavaSWT のコードをそのまま記述する。

**リスト4　サンプルプログラムのイベント定義**

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

リスト 4 のコードは ID が B_SAMPLE のボタンを押したたときにイベントが発生することを示している。なお、JavaSwing と JavaSWT ではイベントに使用するメソッドが異なるため、それぞれイベント定義ファイルを作成する必要がある。

### 4. 3. プログラムコードへの整形

定義ファイルは GUI の視覚情報をもとに簡潔に表現された XML のコードであるが、ジェネレータはこれをソースコードと同じロジックの XML で整形し直し、中間コードとして出力する。なお、以降の変換過程は JavaSwing と JavaSWT で共通のため、説明は JavaSwing のみで行う。

**リスト3　サンプルプログラムの GUI 定義**

```
<Window id="W_SAMPLE" title="サンプルアプリケーション" width="300" height="120">
  <Button
    id="B_SAMPLE" text="ここを押してください" width="280" height="30" event="PUSH"/>
  <Label id="L_SAMPLE" text="" width="280" height="30">
</Window>
```

中間コードは以下のプロセスを経て生成される。

(1)コアコードの生成

GUI 定義内の Window タグを整形する。どのアプリケーションを作る際にも共通して必要となる、各種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をひとつのフォーマットとしてまとめて提供する。

(2)イベントリスナの実装

GUI定義内を全検索してevent属性を持つコンポーネントを発見した場合にメインクラスにインターフェイスを組み込む。サンプルの場合、Button タグが PUSH イベントを発生することを突き止め、ActionListener を実装する。

(3)グローバル変数の宣言

GUI 定義内を全検索してイベントを受け取る可能性のあるコンポーネントを発見した場合にそのオブジェクトをグローバル変数に切り替える。サンプルの場合、Label タグが存在することを突き止め、メインメソッドを生成する前にオブジェクトを宣言しておく。

(4)コンポーネントに関連したコードの分解

GUI 定義内の Button タグ、Label タグを整形する。ひとつのタグにまとめられているコンポーネントのプロパティをソースコードで表現されるメソッドのレベルにまで分解する。

(5)コンポーネントの親子関係の決定

GUI 定義内の Button タグ、Label タグを整形する。各コンポーネント間の親子関係は、定義ファイルではタグを階層化させて視覚的に再現していたが、ソースコードではメソッド中にどのコンポーネントがどのコンポーネントに貼り付けられているのかをはっきりとコードに書き表さなければならない。したがって、XPath を用いて自身の上の階層、下の階層のコンポーネントの ID を取得した後に、それぞれのパターンに合わせた結合メソッドを用いてコードを組み立てる必要がある。

**リスト5　GUI の親子関係の決定**

```
<Add id="B_SAMPLE" parent="Window"/>
<Add id="L_SAMPLE" parent="Window"/>
```

(6)イベント定義ファイルとの結合

GUI 定義内でのイベントを発生するコンポーネントの ID と、イベント定義内でのイベントの内容が記述されていコードのIDが一致した場合、中間コードにイベントメソッドを追加する。

### 4.4. ソースコードへの変換

最後に XML の中間コードを Java のソースコードへ変換する。

### 4. 5．アプリケーションの実行

コードジェネレータは結果として JavaSwing、JavaSWT の 2 種類のソースコードを出力する。これらをコンパイルし、実行するとそれぞれ以下のようになった。

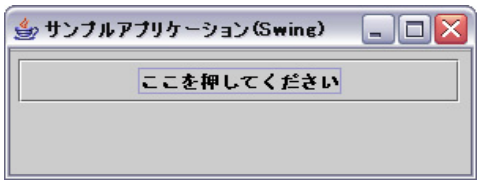


**図4　サンプルアプリケーション(JavaSwing)**

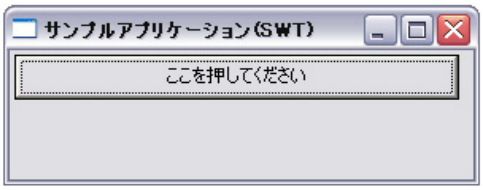


**図5　サンプルアプリケーション(JAvaSWT)**

単一の GUI 定義を元にして作られているため、最終的に画面に現れるアプリケーションの形もまったく同じものである。また、ボタンを押したときのイベントの動作も確認された。

### 5. 本システムのための統合開発環境の構築

GUI プログラムの開発者がこのジェネレータを利用するための統合開発環境アプリケーションを併せて構築した。

**図6　統合開発環境アプリケーション**

この開発環境アプリケーションはリスト3とリスト4の3種の定義ファイル、それらにより生成される2種のソースコードを同時編集できる一般的なタブ形式のテキストエディタである。この開発環境アプリケーション上では「定義ファイルの記述」から「ソースコードの生成」「プログラムのコンパイル」「アプリケーションの実行」までの一連の操作のすべてを行うことができる。実際に開発環境アプリケーションを使って先ほどのサンプルアプリケーションを生成している状態が図6である。この開発環境アプリケーションはさらに以下の2つの要件を満たしている。

(1) JavaSwing/JavaSWT での実装

この開発環境アプリケーションは JavaSwing、JavaSWT の 2 種類のソースコードを出力するが、この開発環境アプリケーション自体も JavaSwing もしくは JavaSWT を使って実装したものである。

(2) ant コマンドの実行

コードジェネレータにおける「XSLT 変換」「Java コンパイル」「Java 実行」といった OS への命令は、Ant ビルドファイル内に各タスクごとにまとめられている。コードジェネレータは利用者がAntコマンドを実行しそれを呼び出すことで動作している。対して、開発環境アプリケーションは開発環境アプリケーション自身が exec コマンドにAntコマンドを引数として渡しそれを呼び出すことで動作している。

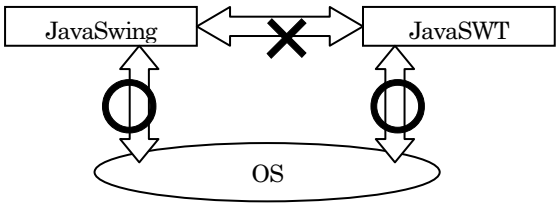


**図7　Ant を経由したコード生成**

直接 JavaSwing と JavaSWT 間で情報を交換することは難しい。しかしAntを経由してOSにすべての命令を任せてしまうことで、図7のように間接的には自在な変換を行うことができるようになるのである。図 8 は JavaSwing で構築した開発環境アプリケーションからJavaSWTのサンプルプログラムを実行している状態である。同じようにJavaSWTで構築した開発環境アプリケーションから JavaSwing のサンプルプログラムを実行することも可能である。このように処理に Ant を経由させることによって、本研究で開発したコードジェネレータは GUI ライブラリの枠にとらわれない相互の変換をも実現しているのである。

**図8　JavaSwing-JavaSWT の相互変換**

## 6. 開発環境アプリケーションの自己生成

前述したように、この開発環境アプリケーション自体が JavaSwing もしくは JavaSWT のいずれかを使用して実装している。したがってこの開発環境アプリケーションを構成する GUI 及びイベントの情報を定義ファイルに記述し、ジェネレータを通して変換することで、図9のように開発環境アプリケーションから同じ開発環境アプリケーション自身を自己生成することも可能である。

**図9　開発環境アプリケーションの自己生成**

こうして生成された開発環境アプリケーションに、再び同じ定義を記述するとまた同じ開発環境アプリケーションが生成される。このループは半永久的に繰り返すことができると思われる。

## 7. 既存言語との比較

前述したようにXMLを使ってGUIプログラミングを行う試みは多数存在し、代表的なものに

Mozilla.org の XUL(XML User Interface Language)[1]や Microsoft の XAML(eXtensible Application Markup Language)[2]がある。ここでは、それらとこのジェネレータとの相違点を比較してみる。

(1)定義を簡易に行っている

通常、ジェネレータはより汎用性のあるソースコードを組み立てるために XML 以外の CSS や JvaScript などを併せて定義に必要とする。しかし、本研究では利用者の利便性を考えて定義ファイルを XML1 種のみに限定した。

(2)解析を簡易に行っている

通常、ジェネレータはより正確なソースコードを組み立てるために正規表現などを用いた高度な字句解析、構文解析、意味解析を変換に必要とする。しかし、本研究では開発と後の拡張を容易に行うためにすべての解析を XSLT に任せている。

(3)ターゲットを限定していない

他プロジェクトが出力先のターゲットとしてひとつのパターンのみを想定している代わりに強力な機能を実装させているのとは逆に、このプロジェクトは実装する機能に制限を持たせる代わりに出力先のターゲットを複数想定させることができたのである。

## 8. 検証と考察

このジェネレータを利用することで「複雑なソースコードの記述」を要することなく、「複数の開発・実行環境の存在」を考慮することなく GUI プログラミングを行うことができる。ここではさらにシステムの詳しいメリット・デメリットを検討しながら今後の展望について考察する。

(1)変換対象の拡張は可能か

このジェネレータは、次々と変換対象を追加していくことをあらかじめ想定して作られている。WEB アプリケーションでもリッチクライアントなアプリケーションでも、テンプレートさえ用意すればどんなプログラム言語でも作り出すことができる。例えば、JSP のソースコードを生成するためには図 10 のように JSP 用のテンプレートを作成し、JSP 変換ルートを追加で設ければよい。

(2)テンプレートの仕様に問題はないか

テンプレートを作成するためには、それぞれの GUI ライブラリのロジックをよく理解し有効なアルゴリズムを検討しなければならない。本研究で開発したジェネレータもまだまだ利用できる機

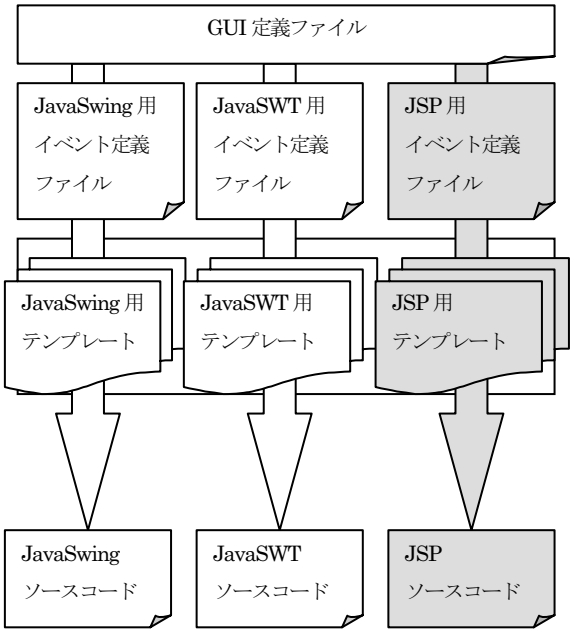


**図 10　JSP 変換ルートの追加**

能に乏しく、JavaSwing や JavaSWT の API を完全に再現するためには膨大な労力が必要となることが予想される。

また、テンプレートは「一般的によく使用されるであろう」フォーマットを自動で生成するものである。そのため手動でソースコードを記述した際に比べると、不自然なコードや非効率なコードの組み立て方が出現することもある。さらに、こちらが想定していない特殊なコンポーネントやメソッドの使用には対応できないという問題もある。

(3)GUI の統合には限界がないか

本研究の目標のひとつは、マルチプラットフォームなアプリケーションの作成にある。本研究でターゲットとした JavaSwing と JavaSWT では最終的に画面に表示されるアプリケーションの「見た目」をほぼ同じものにできた。しかし今後変換対象を追加していくことを考えた場合、それがどこまで可能なのかは疑問である。例えば今回の JavaSwing と JavaSWT の 2 種では「タグ」コンポーネントをサポートしているが、ここにタグを持たない JSP を追加した場合の処理をどうすべきか。各 GUI ライブラリの共通性は変換対象の増加に伴って薄れてしまう。

(4)完成度をどう高めていくのか

開発したジェネレータは研究のための試作であり、公開を前提とした場合にはさらに細かい改良を加える必要があるだろう。定義ファイルの記述

方法の改良や、出力されるソースコードのインデントの追加などユーザビリティの向上が望まれる。特に開発環境アプリケーションには必要最小限の機能しか実装されていないため、まだまだ改良の余地は多く残されている。

(5)システムへ組み込むことはできるか

開発したジェネレータは現在、単体でのみ稼動する。しかし一般的にコードジェネレータとは図11 のように、データベースへのアクセスを伴う WEB アプリケーションなど、一連のシステムのユーザーインターフェイス層に組み込まれて使われることが多い。このジェネレータにおいても、アクセス手段である定義ファイルを作成する機能をサポートすることで、そうしたシステムを構築することが可能である。これは本研究でもっとも展望が期待される点である。

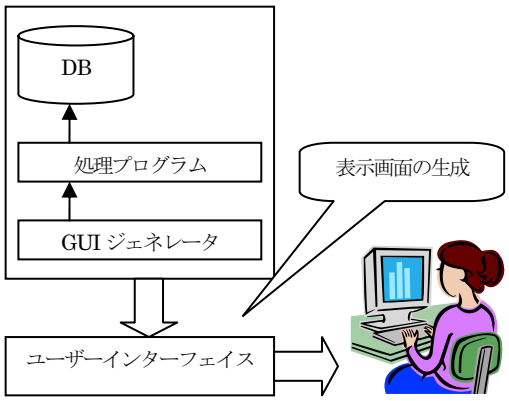


**図 11　システムの中のコードジェネレータ**

## 9. おわりに

本研究で開発したコードジェネレータは「わかりやすい GUI プログラミング」と「複数の GUI ライブラリへの対応」を可能とした。GUI プログラミングのひとつの手法として新しい提案ができたのではないかと思う。

## 10. 参考文献

[1] Mozilla.org, XML User Interface Language (XUL) Project, http://www.mozilla.org/projects/xul/

[2] Microsoft, コントロールと XAML, http://www.microsoft.com/japan/msdn/longhorn/introducing/longhornch03.asp

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

[4] 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

[5] 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