# SA-6-5　自動生成プログラミングツールによるＷｅｂアプリケーション開発

坂本光範†　岩根典之 ††　吉田 誠†

# Web-based Application Development by Automatic Program Generator

Mitunori Sakamoto,　Noriyuki Iwane,　Makoto Yoshida
† 沖ソフトウェア(株),　†† 広島市立大学
† Oki Software Co. Ltd ,　†† Hiroshima City University

## １．まえがき

ソフトウェア開発を効率的に行う手法として、オブジェクト指向に基づいたソフトウェアコンポーネントの再利用による方式が注目されている[2]。 そして、近年Ｅ－ビジネスの普及に伴い、迅速なソフトウエア開発がますます重要視されている。 そこで、著者らは、アプリケーションに関する情報から自動的にコードを生成させるツールを開発して[1]、Webアプリケーションの開発に適用し、さらなる開発の効率化を試みた。 ツールは、ASP/COMアーキテクチャとJavaアーキテクチャ上のコードを自動生成することができる。これらのコードを自動生成する本プログラミングツールにおける出力ファイルの対応関係を表1に示す。

表１ ツールが出力するファイル

| | ﾌﾟﾚｾﾞﾝﾃｰｼｮﾝ層 | ﾋﾞｼﾞﾈｽ・ﾛｼﾞｯｸ層 |
|---|---|---|
| JAVA ｱｰｷﾃｸﾁｬ | JSP ﾌｧｲﾙ | Java ﾌｧｲﾙ(Bean) |
| ASP/COM ｱｰｷﾃｸﾁｬ | ASP ﾌｧｲﾙ | C++ｿｰｽﾌｧｲﾙ(COM) |

## ２．コード自動生成に基づいたプログラミング方式

実際にアプリケーションを開発する場合、再利用可能なコンポーネント以外に、アプリケーションに特化したコードを多くコーディングしなければならない。 アプリケーションに特化したコードの中には、クラスによる再利用可能なコンポーネントとして構築することは困難であるが、良く似たコード・パターン（反復したコード）が多く存在している。 本ツールのコード自動プログラミングは、この良く似たコード・パターンを簡易な文字列やツールのユーザインタフェースに対応させて、それらをアプリケーション情報として入力することで、コードを自動的に生成している。 ツールが持っていないコード・パターンについては、自動的に生成することができないので、自動生成後、コーディング作業を行う。 ツールへ入力するアプリケーションの情報をどのような形式で何をパラメータとするか、またどのようなユーザインタフェースで入力するかによって、ツールの特性が大きく変わってくる。 本ツールは、Web アプリケーションをビジネスロジック層とプレゼンテーション層に分離して、それぞれに適した形式でアプリケーションに関する情報を入力するようにした。

## ３．ビジネス・ロジック層のコード自動生成

ビジネス・ロジック層のコード自動生成を行うために、クラス名やメソッド名の他にメソッド内で実行すべきロジックをアプリケーション情報として入力する。 表２に、ロジック定義の例を示す。

表２　ロジック定義の例

| | |
|---|---|
| ロジック名 | SQL |
| コマンド名 | select |
| パラメータ | * from zaiko_table<br>where ｺｰﾄﾞ=code 変数 |

当該定義による、java ソース生成事例を以下に示す。

```
Class.forName("xxxxx");
Connectionconn=DriverManager.getConnection(URL,props);
conn.setAutoCommit(false);
Statement stmt=conn.createStatement();
String sql="select * from zaiko_table  where コード="+code;
ResultSet rset=rstmt.executeQuery(sql);
while(rset.next()){
      :
      }
```

注：同様に C++ｺｰﾄﾞ の自動生成も可能である。

## ４．ツールを使用した Web アプリケーション開発例

図１のアプリケーションのコードを自動生成するための、ビジネス・ロジック層のアプリケーション情報を図２に、またプレゼンテーション層のプログラムを自動生成するためのアプリケーション情報を図３に示す。

開発手順は、図２の内容を記述したファイルを作成し、ツールに入力して、Java Bean のファイルを自動生成させる。また、図３の下線部のデータをツールに入力し、JSP

ファイルを自動生成させる。　この例は、簡単な機能であるため、100%自動生成可能であり、それぞれをコンパイルしてそのまま実行することができる。


図１　実行画面

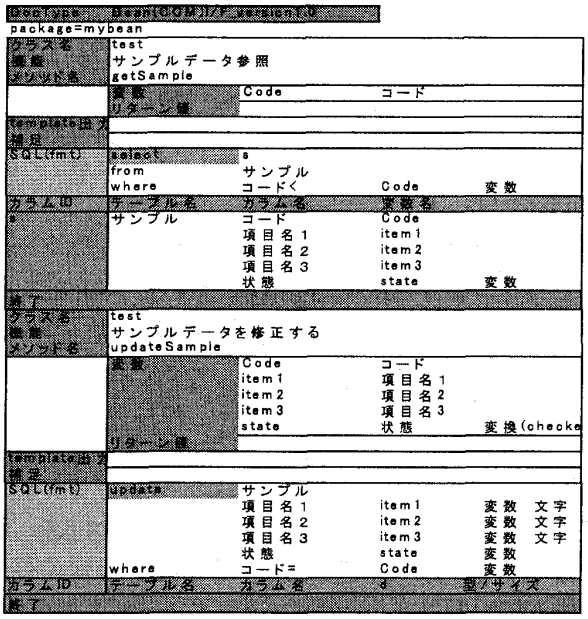


図２　ビジネス・ロジックロジック層のアプリケーション情報

●共通
〈ファイル名〉　Jsp2. jsp
〈タイトル〉　java基本的な表の表示と更新
●使用するビジネス・ロジックの定義
〈クラス名〉　test
〈importパス〉　mybean. *
〈変数名〉　rs
●表の定義
〈Beanの変数名〉　rs
〈参照メソッド〉 getSample("40");
〈アップデートメソッド〉
updateSample(Code, item1, item2, item3, state);
〈表のカラム定義〉

| コード | 項目名１ | 項目名２ | 項目名３ | 状態 |
|---|---|---|---|---|
| Code | item1 | item2 | item3 | state |
| Bean | Bean | Bean | Bean | Bean |
| num | char | char | char | |

〈コードのアトリビュート〉
style="text-align:right" readonly size=10
align="right"
〈項目名１のアトリビュート〉
size=50
align="center"

図３　プレゼンテーション層のアプリケーション情報

## ５．評価

表３に評価結果（ＡＳＰ／ＣＯＭとして測定）を示す。

表３　評価結果(ASP/COM)

| | 機能数 | 総ステップ数 | 追加ステップ数 | 自動生成率 |
|---|---|---|---|---|
| 画面作成(ASP ファイル) | 10 画面 | 6.3k | 40 | 99.4% |
| | 8 画面 | 7k | 0 | 100% |
| ビジネスロジック(COM メソッド) | 14 | 1.4k | 40 | 97.2% |
| | 12 | 1.2k | 40 | 96.9% |
| | 32 | 3.2k | 4 | 99.9% |

ASP ファイルの自動生成率(自動的に生成したステップ数*100/総製造ステップ数)は 99%、COM のソースコードは 98%であった。ほとんどコーディングなしにソフトウェアの製造ができたといっても差し支えない値である。ただし、他の実システムへの応用例として、アプリケーション固有のユーザインタフェースが多く存在し、固有のコードパターンの割合が多い場合、自動生成率 20%～50%の範囲となるケースも有り得た。今後、コード自動生成できるパターンを増やすことによって、更に効率向上可能と考えている。

**参考文献**

[1]M.Yoshida, M.Sakamoto,Experimental Results of Pattern-Based Automatic Program Generator, The 2002 Symposium on Applications and the Internet, 28th,Jan.,2002

[2]P.Herzum, O.Sims, Business Component Factory: A Comprehensive Overview of Component-Based Development for The Enterprise, 2000, John&Sons.Inc.