モデル変換に基づくエンドユーザ主導の Web アプリケーション開発技法

八木 紀幸　　中所 武司
明治大学大学院理工学研究科ソフトウェア工学研究室

近年のPCの高機能化および低価格化、およびインターネットの普及により、インターネットやイントラネットに接続されるPCは広く普及している。それに伴い、ネットワークを通してWebアプリケーションを活用し、日常の業務処理を行う人々（エンドユーザ)も増えてきている。そうした中、小規模なアプリケーションを情報技術の専門家に作ってもらうのは、コスト的に見合わないケースが多い。小規模な業務であれば、エンドユーザ自らが自分の業務に合ったアプリケーションを設計し作成することが望ましい。本研究では、小規模 Web アプリケーションを対象に，エンドユーザ自身が Web アプリケーションの仕様を記述することで Web アプリケーションのコードを自動生成するモデリングツールを開発した。また、そのツールの適用実験により、エンドユーザによる Web アプリケーション開発が可能であることを確認した。

A Method for Web Application Development by End-Users based on Model Transformation

Noriyuki Yagi　　**Takeshi Chusho**
Software Engineering Laboratory, Graduate School of Science and Technology, Meiji University

The number of PCs on the Internet or intranet is on the increase. And the number of end-users who use Web applications to do their daily works is also on the increase. Thus, it is difficult for end-users to make orders for small-scale Web applications to IT professionals, because of the cost. In such a case, it is a good way that an end-user makes his own small-scale Web application by himself. We developed a Web Application Modeling Tool which generates the Web application code automatically, based on model transformation. It uses a Web application model which should be defined by an end-user. The effectiveness of this method is confirmed by feasibility studies.

1. はじめに

近年のパーソナルコンピュータ(以後 PC)の高機能化および低価格化、および インターネットの普及により、インターネットやイントラネットに接続される PC は広く普及している。それに伴い、ネットワークを通して Web アプリケーションを活用し、日常の業務処理を行う人々(エンドユーザ)も増えてきている。

そうした中、小規模なアプリケーションを情報技術の専門家に作ってもらうのは、コスト的に見合わないケースが多い。小規模な業務であれば、エンドユーザ自らが自分の業務に合ったアプリケーションを設計し作成することが望ましい。しかし、エンドユーザが自らアプリケーションを作成するには、多くの知識や高いスキルが要求されるため容易ではない。

そこで本稿では、エンドユーザによる小規模 Web アプリケーション構築支援として、Web アプリケーションの仕様を記述し、その記述からWebアプリケーションの実行コードを出力するモデリングツールを試作し、その評価を行った。

2. エンドユーザ主導開発の課題と解決策

2.1. 課題

エンドユーザが自分に合った Web アプリケーションを自ら作ることは難しい。その理由としては、Web アプリケーション開発には高いスキルや知識が必要だからである。Web アプリケーション開発に必要な知識は主に以下の3つであると考える。

1. ユーザーインターフェースの記述
2. ロジックの記述
3. データの保存

最初のユーザーインターフェースの記述では、開発

者は HTML や場合によっては JavaScript の知識が必要になる。次のロジックの記述では、プログラミング言語はもちろん、Web アプリケーション開発における作法やフレームワークの使用方法といった知識が必要である。知識だけではなく、ロジックを組むスキルも必要になる。最後のデータの保存では、プログラミング言語を使ってファイルやデータベースを操作してデータの永続化を行うスキルが必要となる。

2.2. 解決策

そこで，エンドユーザに知識があまりなくてもエンドユーザ主導で Web アプリケーションを開発できる環境やツールが必要であると考える。たとえば、フレームワークやコンポーネントを利用した開発やビジュアルツールを利用した開発が挙げられる。

フレームワークやコンポーネントはプログラムを部品化することで、そのプログラムの中身を知らなくても簡単なプログラムを書けるレベルの人であるならば、その部品を利用でき高機能なプログラムを作成できる技術である。この技術を一般的には再利用技術と呼ぶ。この再利用技術をビジュアルツールを用いて利用するビジュアルプログラミングの適用により、エンドユーザであっても簡単なソフトウェア開発は可能であると考えられる。

また、近年ではモデル駆動開発と呼ばれる実装コードの自動生成技術も注目を浴びている。実装コードが自動生成されるメリットは大きいが、モデルの定義の難しさや自動生成率が低い問題もあるために、低コストでの開発は期待できないという問題点がある。

そこで本稿では、ビジュアルプログラミングによって部品を組み合わせることで、Web アプリケーションの実装コードを自動生成させるビジュアルツールの提案をする。

3. モデル駆動開発によるコード自動生成技術

モデル駆動開発(MDD)とは、OMG によって MDA(モデル駆動アーキテクチャ)として提唱されているモデルからコードを自動生成する手法を用いた開発のことをいう。

3.1. モデル駆動開発による開発プロセス

1. 特定のプラットフォームに依存しないモデル(これを PIM:Platform Independent Model と呼ぶ)として開発対象のドメインモデルを構築する。
2. 作成したドメインモデルを元に、プラットフォームに依存するモデル(これを PSM:Platform Specific Model と呼ぶ)である設計モデルを作成する。このフェーズにおいて PIM から PSM を自動生成または半自動生成させる MDA ツールを利用することで、開発にかかる時間が短縮できる。
3. 作成した設計モデルを元に、アプリケーションコードを生成させる。このフェーズにおいても MDA ツールを用いてアプリケーションコードを自動生成または半自動生成させることで、開発にかかる時間が短縮できる。

これらの手順をまとめると、図1のようになる。

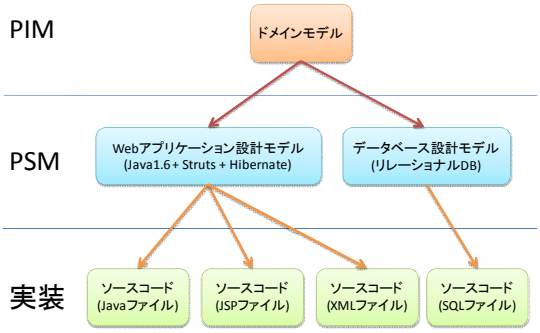

図 1 モデル駆動開発による開発手順例

3.2. モデル駆動開発の利点

ドメインモデルから設計モデルおよび実装コードを、MDA ツールを使い自動または半自動生成させる利点は開発期間の短縮だけではない。ドメインモデルがあり、新しい技術に対応した MDA ツールがある状態であるなら、古い技術による実装から新しい技術による実装への移植が楽に行える。(図 2)

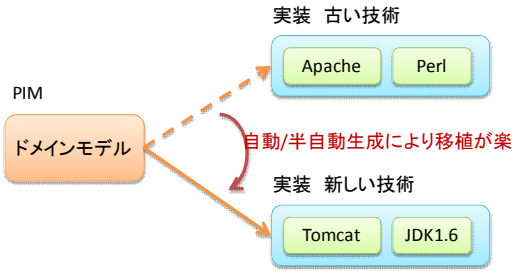

図 2 モデル駆動開発による新しい技術への移植

4. モデル変換に基づくエンドユーザ向け開発ツールの提案

3.2 節で述べたように、MDD によるコード自動生成

技術は非常に魅力的である。しかし MDD は現状の技術において自動生成率はあまり高くなく、UML を用いた詳細なドメインモデルを構築する必要性もあるのでエンドユーザがアプリケーション開発を行うには不向きである。

そこで本研究では、エンドユーザにとってわかりやすい形で、Web アプリケーションの仕様をモデルとして記述させることで、実際に稼動する Web アプリケーションを自動生成するツールを提案する。

ツールの基本的なコンセプトは、「Web アプリケーションモデルから実際に稼動する Web アプリケーションへの自動作成」である。ここでの Web アプリケーションモデルとは、エンドユーザにとってわかりやすい Web アプリケーションの仕様記述である。また、Web アプリケーションモデルは専用のビジュアルモデリングツールを用いて記述を行うことで、エンドユーザが直感的に記述できる。

作成するツールは主に次の3つのツールから成っている。

1. Web アプリケーションモデルを作成する**モデリングツール**
2. Web アプリケーションモデルから Struts2 モデルへの**モデル変換ツール**
3. Struts2 モデルから実行ファイル群への**コード生成ツール**

各ツールとその成果物(データ)の流れを次に示す(図3)。

1. **エンドユーザ**は**モデリングツール**を使い **Web アプリケーションモデル**を作成する
2. 作成した **Web アプリケーションモデル**を入力とし、**モデル変換ツール**を用いて**設計モデル**を生成する
3. 生成された**設計モデル**を入力とし、**コード生成ツール**を用いて **Web アプリケーションの実行ファイル群**を生成する

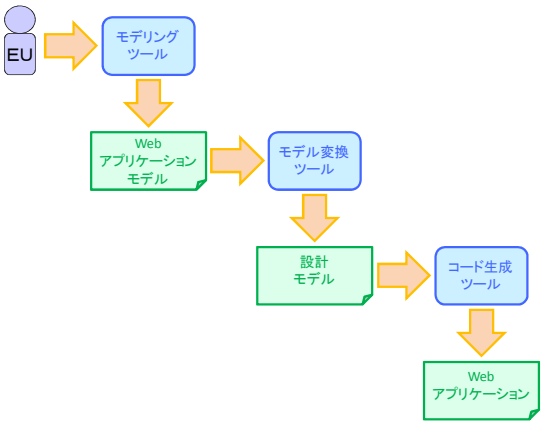

図 3 モデル変換に基づく開発の流れ

5. Webアプリケーションモデリングツール

5.1. Web アプリケーションモデリング言語の設計

モデリングツールによって作成されるモデルのメタモデルを作ることは、モデリング言語の要素や構文の枠組みを作成することになる。本研究では、例題としてイベント管理システムを作成し、それを構成しているオブジェクトをエンドユーザ視点から見つけていくことで、Web アプリケーションモデルのメタモデルを作成した。

5.1.1. 例題アプリケーションの作成

イベント管理システムは、研究室のメンバーのイベントへの出欠の返事の管理をシステム化したいと考えたものである。システムの作成には Rails フレームワークを使用している。イベント管理システムの画面遷移を図 4 に示す。

イベント管理システムはトップページを開くとログイン画面が表示される。トップページにはメンバーの新規登録ページへのリンクが張られている。ログインを行うとイベント一覧が表示される。上部には自分の登録情報の変更画面へのリンクが張られている。イベント一覧の中になるイベント名をクリックすると、そのイベントの詳細情報を見ることができる。さらに同ページからイベントへの出欠の宣言が行える。

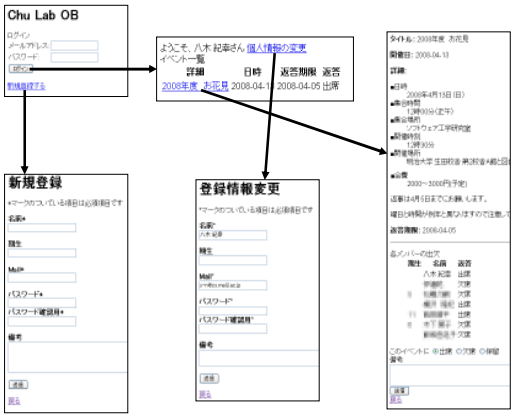

図 4 イベント管理システムの画面遷移

5.1.2. イベント管理システムにおけるメタモデルの抽出

イベント管理システムを利用するユーザーは下記のシナリオに沿ってシステムを利用すると想定する。

1. 利用者がトップページ表示を要求する
2. トップページが利用者に表示される
3. 利用者が新規登録ページ表示を要求する
4. 新規登録ページが利用者に表示される
5. 利用者が新規登録ページの新規登録フォームに記入し送信する
6. 新規登録フォームが利用者情報をデータベースに登録する
7. 新規登録フォームが新規登録完了を利用者に通知する
8. 利用者がログインフォームに記入し送信する
9. ログインフォームがデータベースに利用者を認証する
10. ログインフォームがメンバーページにリダイレクトする
11. 利用者がイベントページ表示を要求する
12. イベントページが利用者に表示される
13. 利用者が返答フォームを記入し送信する
14. 返信フォームが返答情報をデータベースに登録する

上記のシナリオにおいて、ページやフォーム、データベースといった名刺をオブジェクトとして捉え、その振る舞いを図に描き出したものが図 5 となる。なお、データベースに関しては、扱うデータごとに分割することで処理をわかりやすくした。

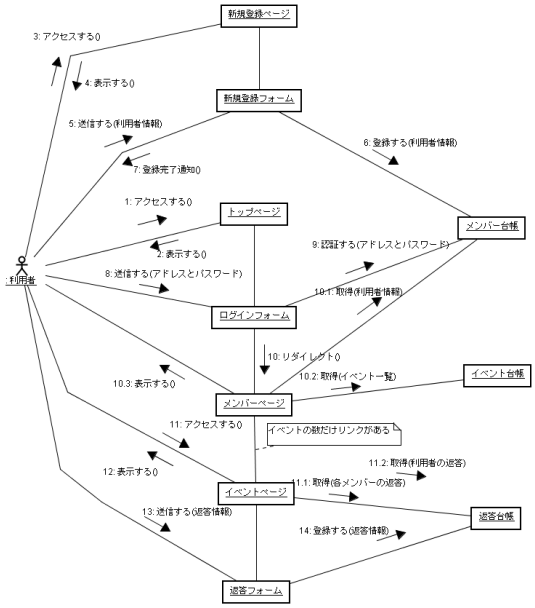

図 5 ユーザーのシステム利用のシナリオ

次に、ページの構成要素に注目し、ページを構成している要素の静的な関係を図に描き出した(図 6)。

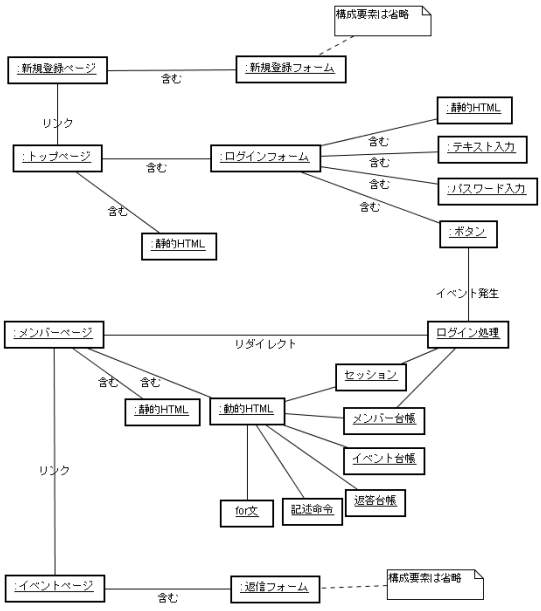

図 6 抽出したオブジェクトの静的な関係

5.1.3. モデリングツールにおけるメタモデル

前節の分析を元に、エンドユーザが作成するモデ

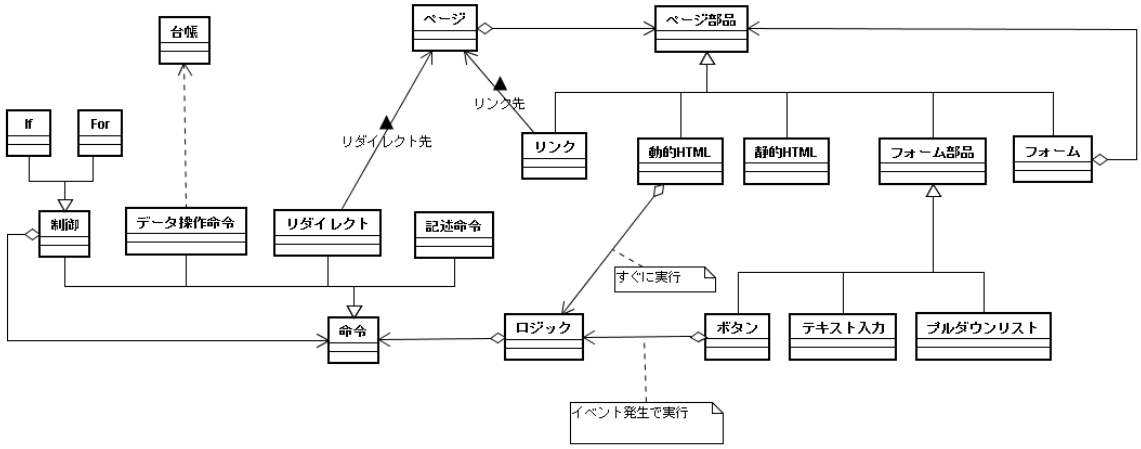

図 7 抽出されたメタモデルのクラス図

ルの構成要素となるモデルを抽出した(図 7)。

ページはページ部品を持っている。ページ部品とは、ページの中に格納することができる部品である。たとえば、リンクやフォームなどはページの中にあるものなのでページ部品のクラスに分類される。リンクにはリンク先となるページを参照する関係がある。フォーム部品はフォームの中で特に意味をなすものであり、パラメータの値を入れるテキストボックスや送信ボタンがこのサブクラスとなる。ボタンは押されたときに実行するロジックを持っている。ロジックは複数の命令を持つことができる。命令は If 文を表現する制御命令やページ遷移命令、データベースを操作するための命令などに分けられる。

5.2. 詳細なロジックの記述方式

エンドユーザ主導開発において、エンドユーザ自らが詳細なロジックを記述することは難しい。エンドユーザによるロジックの記述方式として考えられるのが、スクリプト言語方式と自然言語方式とタイルプログラミング方式である。(図 8)

スクリプト言語方式とは、特定の分野においてよく使われる操作をあらかじめ関数のような形で用意しておくことで、エンドユーザにも簡単にロジックが記述できるというものである。この方式は汎用のプログラミング言語と同様に、記述者は関数の使い方(引数の順番など)といった構文に気を使う必要があるため、エンドユーザが使用するには問題点がある。

自然言語方式とは、普段我々が使っている言語のようなスクリプト言語を用いる方式である。命令が、普段使う言葉になるため、関数の使い方を間違える心配はない。しかし、どの単語をインタープリタが理解できるかを熟知していなければ使用することは難しい。

タイルプログラミング方式とは、あらかじめ用意された命令文の一部を組み合わせることで命令文を作成する方式である。この方式では、自然言語のような理解のしやすいさと、タイプミスによるエラーが起こりにくいというメリットがある。また、ツール側でおかしな組み合わせができないようにすることで、構文エラーも回避できる。

3つの記述方式を述べたが、タイルプログラミング方式がエンドユーザにとって一番記述しやすい方法であると考える。よって本研究において作成するツールでは、ロジックの記述にはタイルプログラミング方式を採ることにした。

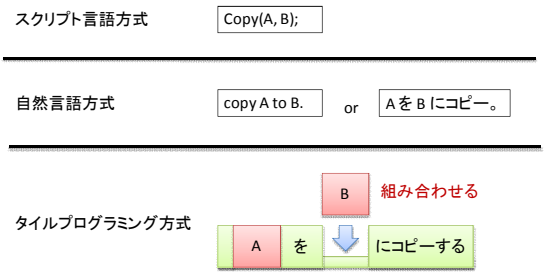

図 8 エンドユーザ向けのロジックの記述方式

5.3. Web アプリケーションモデリングツールの開発

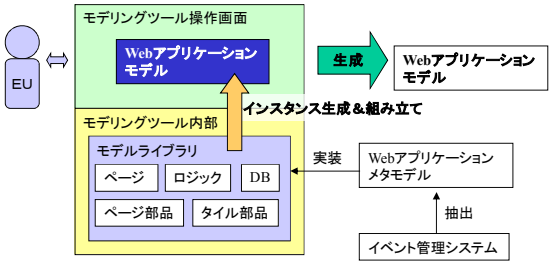

図 9 モデリングツールの概要図

図 9 が開発した Web アプリケーションモデリングツールの概要図である。モデリングツールは 5.1 節で抽出したメタモデルを実装したモデルライブラリを持っており、それらのインスタンスをエンドユーザが組み合わせることで Web アプリケーションモデルを作成する。

図 10 が開発した Web アプリケーションモデリングツールの実行画面である。

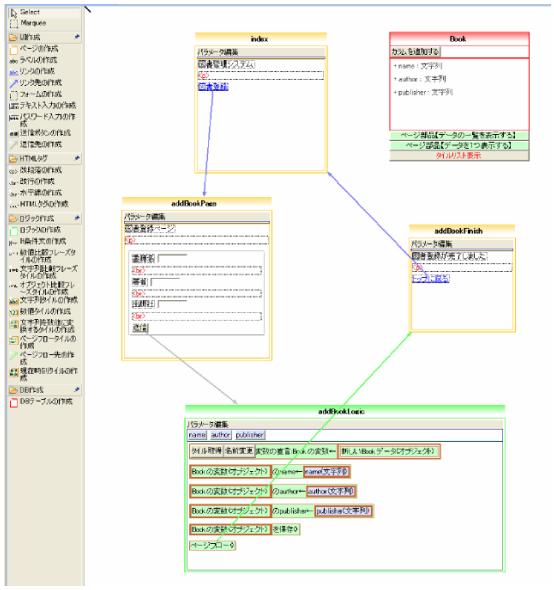

図 10 モデリングツールの実行画面

6. モデル変換ツール

モデリングツールを用いて作成した Web アプリケーションモデルは特定のアーキテクチャを想定していない論理レベルのモデルなので，実際に稼動する Web アプリケーションの設計モデルへとモデル変換を行う。

設計モデルは特定のプラットフォーム上で動くことを前提としたものになる。ここでいうプラットフォームとは、Web アプリケーションを動かすためのプログラミング言語やフレームワークのことをいう。本研究ではこのプラットフォームに Struts2 フレームワークを使う。

6.1. XSLT を用いたモデル変換

Web アプリケーションモデルから設計モデルへの変換のマッピング関係の一部を図 11 に示す。変換の際には、ページから JSP への変換のように単純に変換できるものもあれば、複数の要素から1つの要素へとマージしなければならない複雑な変換を要する場合もある。

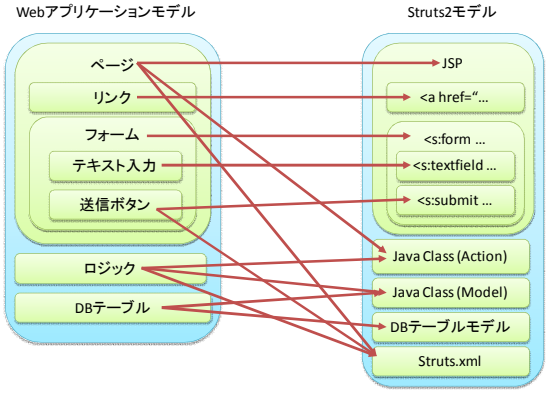

図 11 モデル変換におけるマッピング関係の一部

本研究において作成したツールでは、この2つのモデルの変換を、XSLT を用いて変換することにした。XSLT とは XSL という言語を用い、XML ファイルを入力とし別の XML ファイルを出力する技術である。XML とはデータの意味や構造を記述するためのマークアップ言語である。

6.2. モデル変換ツールの開発

モデル変換ツールの概要図を図 12 に示す。その構成は XSLT プロセッサとモデル変換ルールとなる XSL ファイルから成る。XML 形式で記述された Web アプリケーションモデルを入力とし、XSLT 変換を行うことで XML 形式の Struts2 モデルを出力する。

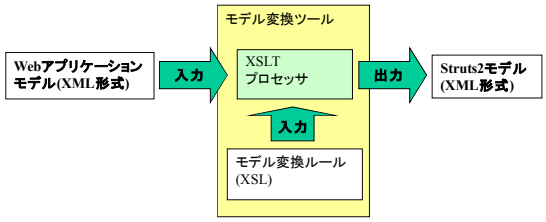

図 12 モデル変換ツールの概要図

7. コード生成ツール

コード生成ツールは設計モデルを元に、実際に稼動する Web アプリケーションの実行ファイル群を生成する。

設計モデルには JSP ファイルならそのファイル名やその中に何を記述するのか、Java ファイルならクラス名や属性やメソッドに何を記述するべきなどが書かれているので、これらの情報を元にコード生成を行うことになる。

7.1. コード生成ツールの開発

コード生成ツールの概要図を図 13 に示す。

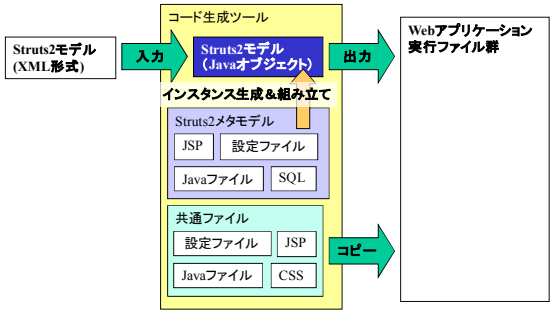

図 13 コード生成ツールの概要図

コード生成ツールでは、Struts2 モデルを表現した XML から Java オブジェクトで作成した Struts2 モデルを構築する。構築した Struts2 モデル（Java オブジェクト）は、コード生成器の役割を果たすことになる。

自動生成される Web アプリケーションに共通しているファイルは、設計モデルには記述されてなく、Web アプリケーション生成時にコード生成ツールによってコピーされる。

8. 適用実験と評価

本研究において作成したツールを使い、例題アプリケーションを作成し評価した。

8.1. 例題アプリケーションの概要

例題アプリケーションとして、大学の研究室単位で使うような図書管理システムを作成した。

作成した図書管理システムの概要を以下に示す。

A) 利用者は図書情報を登録できる
B) 図書情報を登録する際に、ログインしている状態でなければログイン画面が要求される
C) 図書情報登録の際に必須項目に空があればエラー画面を表示する
D) 利用者は図書の貸出記録を付けることができる
E) 貸出記録を付ける際に、ログインしている状態でなければログイン画面が要求される
F) 利用者は図書の返却記録を付けることができる
G) 返却記録を付ける際に、ログインしている状態でなければログイン画面が要求される
H) 利用者は図書の貸出記録一覧を閲覧することができる
I) 管理者は利用者をデータベースに登録することができる
J) 管理者は図書情報を削除することができる

8.2. 例題アプリケーションの作成

作成した図書管理システムの Web アプリケーションモデルは図 14 のような全体像になった。これだけのモデルを記述することで、前述した機能を持つ Web アプリケーションが自動生成されることを確認した。

8.3. 評価

作成した図書管理システムは参考文献[3]の図書管理システムを参考にして作成した。但し、ツール側の機能不足により，利用者情報の管理と認証をディレクトリサーバで行う機能，図書検索機能，図書一覧から図書個別のページへのリンク機能などは作成することが出来なかった。これらの機能はツールの機能を強化することで克服できる問題であると考えている。

実験の結果、作成したツールによって、限定的ではあるが Web アプリケーションが自動生成できることが確認できた。

また、本ツールを使うことで、汎用プログラミング言語を用いるよりも、はるかに容易に Web アプリケーションを作成できることも確認できた。これは、本ツールを使いこなすのにかかる講習時間が、Web アプリケーションを汎用言語で開発できるようになる時間よりもはるかに短い期間でできるものだと予想できるためである。

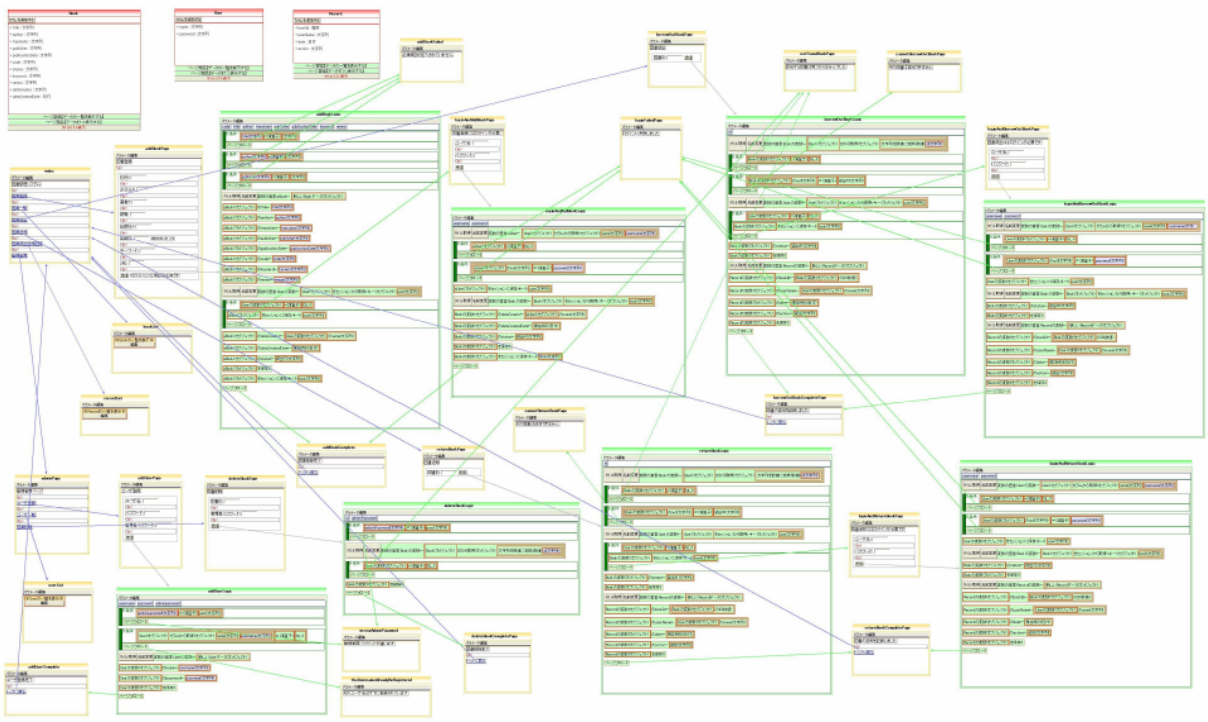

図 14　作成された図書管理システムの Web アプリケーションモデル

9. おわりに

本稿では、エンドユーザによる小規模 Web アプリケーションの開発技法として、エンドユーザにわかりやすい形の Web アプリケーションモデリング言語を提案し、UI、ビジネスロジック、データベースからなる Web アプリケーションを自動生成するビジュアルモデリングツールの試作を行った。

例題として図書管理システムを取り上げ、そのほとんどの機能を、ツールを使い実装することができることを確認した。また、汎用プログラミング言語を用いるよりも容易に Web アプリケーション開発が可能であることを確認した。

参考文献

[1]　八木　紀幸、中所　武司:エンドユーザによる Web アプリケーション開発技法の提案と試作, FIT2007　第 6 回情報科学技術フォーラム.

[2]　八木　紀幸:エンドユーザによる Web アプリケーション作成のためのスクリプト言語の作成と評価, 明治大学理工学部情報科学科 2006 年度卒業論文.

[3]　中所　武司・藤原　克哉　共著:Java による Web アプリケーション入門　-サーブレット・JSP・Struts-,サイエンス社.

[4]　ObserveEclipse, http://www13.plala.or.jp/observe/index.html

[5]　梅澤真史:自由自在 Squeak　プログラミング”, 株式会社ソフト・リサーチ・センター.