

ワークフロー図から Struts 定義ファイルの生成

床田 紘美、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

ビジネスプロセスをモデル化する BPMN、UML や MDA といった考え方が注目されている。このような観点から、本稿では、モデル化の対象として、物品情報管理システムを題材としてとりあげ、ビジネスの手順を描いたワークフロー図から Struts[1] の振る舞いを規定している定義ファイルを生成するコード生成システムを試作したので報告する。

2. コード生成システムの概要

2.1. コード生成システムの目的

業務担当者と技術者の認識の差を減らすためにワークフロー図が用いられる。このワークフロー図からできるだけ自動的にプログラムが作成できれば、業務とプログラムのセマンティックギャップの少ない効率の良いシステムが実現できる。

2.2. ワークフロー図から Struts 定義ファイル生成手順

ビジネスシステムは処理変更が頻繁におこる。ビジネスフローをそのままコーディングしていると大幅な変更を伴う。そこで、GUI 環境でワークフロー図を作成すると、そのワークフロー図からシステムを自動生成できるという変換スクリプトを開発する。その手順を表したフローが、図 1 である。システムは Struts を用いて実装していく。図中の番号は、以降に詳細に説明している章節番号である。

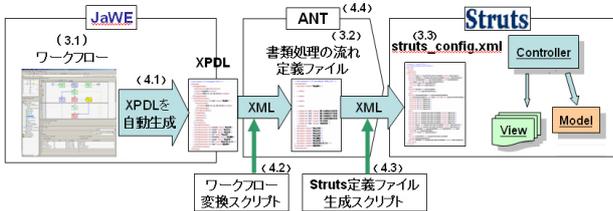


図 1 コード生成システムのフロー

3. プロセス記述の変換モデル

同じワークフローが、次のように 3 段階に変換されていく。

3.1. ワークフロー図の記述

図 2 は、研究室の物品購入から納品までの流れをワークフロー図として記述したものである。

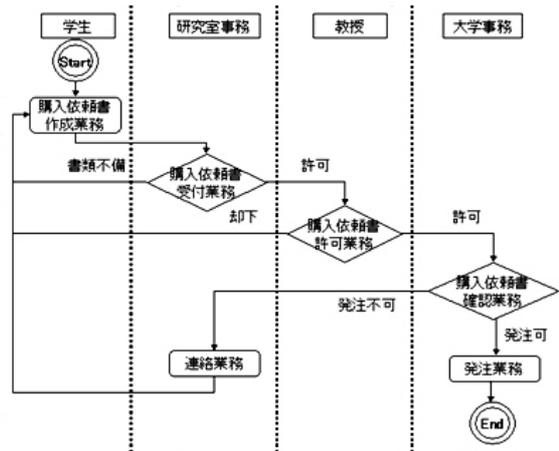


図 2 物品購入フロー図

このワークフロー図を、ワークフローグラフィカルエディタである JaWE[2]を用いて入力した。

3.2. ワークフロー記述

3.1 で記述したワークフロー図を元に、ビジネスの処理の流れをワークフロー記述言語で記述した (図 3)。この XML ファイルには、ビジネス処理手順や携わる者の役割などをプロセスごとに記述した。これを「書類の流れ定義ファイル」と呼ぶ。

```
<process name="物品購入">
  <role><name>学生</name></role>
  <role><name>研究室事務</name></role>
  <role><name>教授</name></role>
  <role><name>大学事務</name></role>
  <state id="n2" name="購入依頼書作成業務">
    <assigned-role>学生</assigned-role>
    <transition id="tra2" name="success">n3</transition>
  </state>
  <state id="n3" name="購入依頼書受付業務">
    <assigned-role>研究室事務</assigned-role>
    <transition id="tra4" name="success">n4</transition>
    <transition id="tra3" name="failure">n2</transition>
  </state>
  <state id="n4" name="購入依頼書許可業務">
    <assigned-role>教授</assigned-role>
    <transition id="tra5" name="success">n5</transition>
    <transition id="tra9" name="failure">n2</transition>
  </state>
  <state id="n5" name="購入依頼書確認業務">
    <assigned-role>大学事務</assigned-role>
    <transition id="tra7" name="success">n7</transition>
    <transition id="tra6" name="failure">n6</transition>
  </state>
  <state id="n7" name="発注業務">
    <assigned-role>大学事務</assigned-role>
  </state>
  <state id="n6" name="連絡業務">
    <assigned-role>研究室事務</assigned-role>
    <transition id="tra8" name="failure">n2</transition>
  </state>
</process>
```

図 3 物品購入フロー記述

3.3. Struts 定義ファイル

3.2 で生成された「書類の流れ定義ファイル」を元に Struts で実行可能な Struts 定義ファイル(図 4)を作成した。

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<!DOCTYPE struts_config (View Source for full
doctype...)>
<struts-config>
<form-beans>
<form-bean name="n2" type="app.n2" />
<form-bean name="Document"
type="app.Document" />
:
<global-forwards>
<global-forward name="Document"
path="/Document.do" />
<global-forward name="n2" path="/n2.do" />
:
</global-forwards>
<action-mappings>
<action path="/n2"
type="org.apache.struts.actions.ForwardAction"
parameter="/pages/n2.jsp" />
:
<action path="/Document"
type="app.DocumentAction">
<forward path="/pages/DocumentSelect.jsp" />
</action>
<action
path="/DocumentSelectSubmit"
type="app.DocumentSelectSubmitAction"
name="DocumentSelectSubmitForm"
scope="request"
validate="true"
input="/pages/DocumentSelectSubmit.jsp">
<forward name="n2" path="/pages/n2.jsp" />
<forward name="n3" path="/pages/n3.jsp" />
<forward name="n4" path="/pages/n4.jsp" />
<forward name="n5" path="/pages/n5.jsp" />
<forward name="n6" path="/pages/n6.jsp" />
<forward name="n7" path="/pages/n7.jsp" />
</action>
:
</struts-config>
```

図 4 物品購入フロー Struts 定義ファイル

4. 変換モデルに基づくコード生成

4.1. XPD L ファイルの生成

ワークフロー図(図 2)を、JaWE を用いて記述すると、XPD L で記述されたファイルが自動的に生成される。

4.2. 「書類の流れ定義ファイル」の生成

4.1 で生成された XPD L ファイルを 3.2 で定義したワークフロー記述言語に変換するために XSLT で記述した「ワークフロー変換スクリプト」を作成した。この XSLT スクリプトで変換をおこなうと、「書類の流れ定義ファイル」(図 3)が生成される。

4.3. Struts 定義ファイルの生成

4.2 で生成された「書類の流れ定義ファイル」を Struts 定義ファイルである struts_config.xml(図 4)に変換するために XSLT で記述した「Struts 定義ファイル生成スクリプト」を作成した。

4.4. ANT による変換スクリプト

4.2 で述べた「書類の流れ定義ファイル」生成、4.3 で述べた Struts 定義ファイルの生成といった一連の流れを、ANT スクリプトとして記述した。

この ANT スクリプトが、図 1 のワークフロー図から Struts 定義ファイルを生成するコード生成システムの骨格である。

5. 実験と考察

5.1. システムの試用実験

コード生成システムの完成度を検証するために、物品情報管理システムを生成する実験を以下の手順でおこなった。

まず、4. で開発した ANT スクリプトを用いて Struts 定義ファイルを生成した。次に、この Struts 定義ファイルに合わせて、Struts 用の部品を作成した。この部品とは、アクションサーブレット、JSP、アクションフォーム Bean であり、Struts が提供しているクラスである。本システムでは、画面遷移の管理を Struts 定義ファイルで制御した。また、Struts 定義ファイルで記述できない、役割や書類処理手順の管理を「書類の流れ定義ファイル」で制御した。

5.2. 考察

ワークフロー変更に対するシステムの完成度や部品の独立性についての考察をおこなった。

ビジネス手順が変更になった場合でも、ワークフローを変更するだけで struts 定義ファイルが生成され、一貫性を保ったシステムの変更が可能であった。また、ビジネス処理において特徴的な書類の分岐や非同期的な処理は、システム実装の際に、Struts の ActionServlet を拡張して記述することで実現できた。以上から、ワークフロー図から Struts 定義ファイルを生成する変換スクリプトの有効性を確認できた。

6. おわりに

複雑なビジネスフローに対応できるようにシステムの完成度を高める必要がある。また、ワークフローを描く際に、様々なグラフィカルエディタにも対応させ、応用の幅を広げたい。

参考文献

- [1]. Struts : <http://struts.apache.org/>
- [2]. JaWE : <http://jawe.objectweb.org/>