社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
SS2005-75(2005-12)

SPINによるStrutsアプリケーションの動作検証を目的としたモデル生成手法の提案

藤原 貴之† 岡野 浩三† 楠本 真二†

† 大阪大学大学院情報科学研究科
〒 560–8531 大阪府豊中市待兼山 1–3
E-mail: †{t-fujiwr,okano,kusumoto}@ist.osaka-u.ac.jp

あらまし WEBアプリケーションの開発において，ページ遷移異常が発生しないように設計することは重要である．開発に広く用いられているフレームワークの1つであるStrutsでは，このような異常を検出するための機能が用意されている．ページ遷移異常の原因として，ユーザ入力のタイムアウトによる紛失や，二重送信による重複などが考えられる．本稿では非決定的に異常を発生する状況下でもシステムが正常動作することを検証するための手法として，ページ遷移異常の原因となる振る舞いを，WEBアプリケーション自身の振る舞いに含めてモデル化する手法を提案する．
キーワード Struts，モデル検査，PROMELA，SPIN

A Framework to Generate SPIN based Verification Codes for Struts WEB Applications

Takayuki FUJIWARA†, Kozo OKANO†, and Shinji KUSUMOTO†

† Guraduate School of Information Science and Technology Osaka University
Machikaneyama 1–3, Toyonaka City, Osaka, 560–8531 Japan
E-mail: †{t-fujiwr,okano,kusumoto}@ist.osaka-u.ac.jp

Abstract We propose methods which model Struts-based WEB application and an assistance tool. For WEB application, designing correct page-transitions is very important. Users occasionally give invalid input such as double-clicking the submit button, time-out and so on. They cause page-transition error. Struts, which is one of the most popular frameworks for WEB application, has functions to detect such errors. In order to detect the error in advance, the tool based on our methods automatically generates a model in PROMELA from struts-config.xml. The code includes WEB application's behavior as well as functions to detect such page-transition errors.
Key words Struts, Model Checking, PROMELA, SPIN

1. まえがき

近年，WEBアプリケーション機能が多様化し，複雑さも増してきている．このようなシステム開発の際，設計段階での検証技術もより重要になると考えられる．WEBアプリケーションの設計において，ページ遷移が仕様に沿うことは特に重要であり，形式的技術による検証支援が有効である可能性は高い．

本稿では動作検証を目的として，WEBアプリケーション開発の際に広く用いられているStrutsフレームワーク[11]に着目し，その振る舞いを利用したモデル生成手法を提案する．StrutsはMVC設計モデル[10]を採用しており，ビジネスロジックをModel，サーブレットをController，画面をViewに対応させることで，それぞれを独立して開発することができる．しかし，Controllerではビジネスロジック終了後の遷移先，Viewでは各画面が呼び出すアクション名を定義しているだけであり，片方だけでは初期画面から終了画面までの制御フローを一貫して把握することができない．文献[7]ではこの問題を解決するため，Struts設定ファイル（struts-config.xml）に対して，画面に関する記述の追加を提案している．しかし設定ファイルの肥大化はStrutsが持つ問題点の一つであるため，本稿の提案手法ではStrutsを必要十分に抽象化したモデルを対象にする．

WEBアプリケーションは，ユーザの誤操作等を受けても正常に動作すべきである．例えばSubmitボタンを誤って二度押してしまった場合，同じ処理を繰り返さない適切な設計が望まれる．Strutsは二重送信の検出機能を持つが，検出時の処理方法は開発者の設計方針に委ねられている．本稿ではそのような

ソフトウェア設計を支援するため，非決定的にエラーを発生するような状況も表現できるモデルを提案する．従来，不具合はモデルの振舞いから発見するものであったが，通信モデル上の不適切な振る舞いとして取り込んだ所に新規性があると考える．

モデルの記述には PROMELA [2] を用いた．PROMELA では，複数の状態遷移プロセスが，通信用のチャネル（キュー）を通してデータを送受信する振る舞いを表現できる．本稿では，サーバのモデルを Struts 設定ファイル（struts-config.xml），そしてクライアントのモデルを画面設計書から生成する手法について述べる．またモデリング支援のため，これらのモデルをある程度まで自動的に生成するツールも試作した．

PROMELA で記述したモデルの振る舞いは検証系 SPIN [2], [3] を用いて，自動的に検証することができる．SPIN はプロセスの飢餓状態やデッドロックの検出機能を持つ他，線形時相論理（LTL）を用いてより詳細な条件を検証する能力も持つ．ただしモデル検査では SPIN に限らず検証時の状態爆発回避が大きな課題である．検証コストを下げる手法の一つとして，非同期通信モデルを同期通信モデルに置き換える方法がある．文献 [1] では元のモデルの性質を保持したまま置換可能な場合の条件が示されており，本稿の提案モデルはこの手法を応用している．

関連研究として文献 [4]〜[8] 等が挙げられる．文献 [4], [5] は WEB アプリケーション全般の設計を対象としている．また文献 [6] では Web アプリケーションの構成要素を個々にモデリングし，モデルの集合から Struts アプリケーションを生成する手法について述べている．文献 [7], [8] ではそれぞれ設定ファイルの拡張によりアプリケーションの振る舞いを明示する手法，モデリングの際に抽象化した部分をスタブで補う手法を提案している．我々の提案手法では Struts を利用した WEB アプリケーションを対象とし，開発の初期段階から存在することが多い設定ファイルに対し，特別な処理を施すことなくモデルを自動生成する点がこれらと異なる．

本稿の構成は以下の通り，2. で提案する手法およびツールについて述べ，3. で提案手法を適用する例題の説明を行う．4. では実験結果および考察をまとめ，最後に 5. でまとめと今後の課題について述べる．

2. 提案するモデリング手法

本章ではモデリングの対象となる Struts の構成およびモデル記述言語 PROMELA の概要に触れた後，モデリング手法を提案する．ついで，試作したツールの概要について述べる．

2.1 Struts の構成

Struts は Jakarta プロジェクト [11] が開発しているオープンソースのソフトウェアで，Java による WEB アプリケーション開発のためのフレームワークである．MVC というソフトウェア設計モデルに基づいており（図 1），WEB アプリケーションを構築する際に必要となる諸機能を提供する．Struts に含まれる要素は主に次のようなものである．

- クライアントおよび JSP：ユーザからの操作を受け，サーバにリクエストを送信する．またサーバからのレスポンスを受け，画面を切り替える．
- アクションサーブレット：クライアントからのリクエストを受信する．リクエストの種類に応じて適切なリクエストプロセッサを選択し，リクエストを渡す．1 つの Struts アプリケーションについて 1 つだけ存在する．以下，サーブレットと略す場合がある．
- リクエストプロセッサ：Struts を構成する要素の中でも中心的な役割を果たすクラスである．サーブレットから渡されたリクエストを受け，リクエスト内容をコンテキストの適切なスコープにあるアクションフォームに保存する．既存のアクションフォームがない場合は新たに生成し，コンテキストの適切なスコープに保存する．また具体的な処理を実行するためのアクションを呼び出す．モジュールの単位であり，1 つの Struts アプリケーションについて 1 つ以上存在する．
- アクションフォーム：リクエスト情報等を格納しておくクラスであり，リクエストプロセッサによって生成される．
- アクション：リクエストの内容を処理するためのクラスである．ほとんどの場合は処理の順序やエラー発生時の動作のみを制御する．
- コンテキスト：様々なデータを格納するための要素である．データの存在時間が異なる 4 つのスコープを持つが，ここでは request と session のみを扱う．request スコープにあるアクションフォームは，ユーザのリクエストに対してサーバがレスポンスを返した時点で消去される．session スコープにあるアクションフォームは，ユーザとのセッションが途切れるまで保存される．

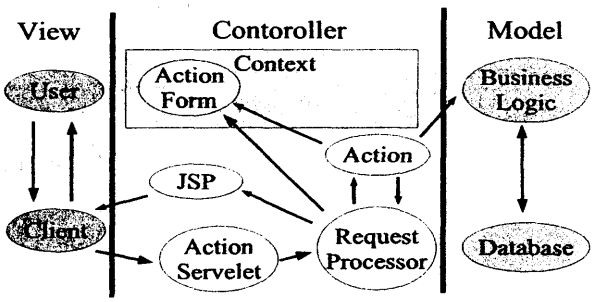

図 1 Struts の概要

```
<struts-config>
  <form-beans>
    <form-bean name="form" type="Form">
  <form-beans>
  <action-mappings>
    <action path="/index" type="Index"
            name="Form" scope="request">
      <forward name="init" path="index.jsp"/>
    </action>
  </action-mappings>
</struts-config>
```

図 2 struts-config.xml の例（抜粋）

2.2 Struts 設定ファイル（struts-config.xml）

2.1 で述べたように，Struts ではリクエストプロセッ

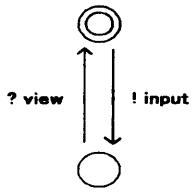

図 3 User の状態遷移モデル

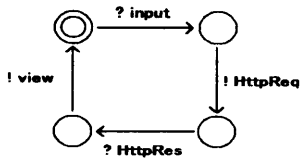

図 4 Client の状態遷移モデル

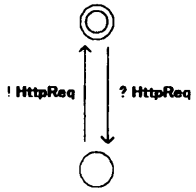

図 5 Network の状態遷移モデル

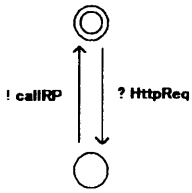

図 6 Servelet の状態遷移モデル

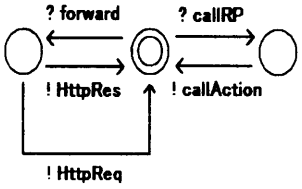

図 7 RequestProcessor の状態遷移モデル

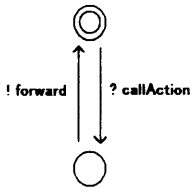

図 8 Action の状態遷移モデル

サが中心的な役割を果たしており．設定ファイル（struts-config.xml）にはその動作を決定するための情報が書かれている．`action-mappings` タグで「`abc.do`」という形のアドレスを「`/abc`」と関連付けており，(図 2) の例では `index.do` というアドレスが呼び出された場合に`/index` が対応し，`Index` アクションを呼び出す仕組になっている．この `Index` アクションは `form` と名づけられたアクションフォーム `Form` を利用して処理を行い，リクエストプロセッサに `init` という遷移情報を返す．最後にリクエストプロセッサはその遷移情報に対応する `index.jsp` をクライアントに返す．提案手法では Struts が本質的に状態機械の振る舞いを持つことを利用している．

2.3 モデリングの方針

ここでは 2 つのモデルを与える．単純なモデルと異常な動作を考慮したモデルである．

2.3.1 単純なモデル

まえがきでも述べたように，モデル記述言語 PROMELA はデータ，チャネル（キュー），プロセス（状態機械）の三要素を持つ．設定ファイルや画面設計書等を元に，我々は Struts アプリケーションの各要素を次のようにモデル化した．

プロセスとして扱うもの

（1） ユーザ（図 3）：画面を通して入力を行い（`!input`），次の画面に遷移するのを待つ（`?view`）動作を繰り返す．

（2） クライアント（図 4）：4 状態のプロセスである．ユーザからの入力を受け（`?input`），その内容をネットワークに送信（`!HttpReq`）する．その後リクエストプロセッサから遷移情報を受け（`?HttpRes`），画面が遷移したことをユーザに知らせる（`!view`）．各画面には複数のリクエスト候補が存在し，ユーザからの入力を受けた際にどのリクエストがサーバに送信されるのかは非決定的に選択される．

（3） ネットワーク（図 5）：クライアントからリクエストを受け（`?HttpReq`），サーブレットへ送信する（`!HttpReq`）．

（4） サーブレット（図 6）：クライアントから命令を受信し（`?HttpReq`），適切なリクエストプロセッサを呼び出す（`!callRP`）．

（5） リクエストプロセッサ（図 7）：チャネルを 2 つ持つ．一方はサーブレットからのリクエストを受信（`?callRP`）するためのもので，受けたリクエストは適切なアクションへと渡される（`!callAction`）．もう一方はアクションからの戻り値を受信（`?forward`）するためのものであり，戻り値に応じてクライアントに次画面への遷移を送信（`!HttpRes`）するか，次のアクションを呼び出す（`!HttpReq`）．他のモジュールにあるアクションを呼び出す場合もあるため，提案モデルではサーブレットに命令を送信することでモジュール選択機能を持たせた．

（6） アクション（図 8）：リクエストプロセッサからの呼び出しを受け（`?callAction`），遷移情報を返す（`!forward`）．

データとして扱うもの

（1） コンテキスト：アクションやリクエストプロセッサ等，多くのプロセスから参照される可能性があるため，外部変数として宣言する．構造体として定義し，内部には request スコープと session スコープを表現する 2 つの構造体型変数を持つ．各スコープの内部には，アクションの数と同じだけのアクションフォームを格納しておく．session スコープのみ，二重送信検出用のトークンを持つ．

（2） アクションフォーム：bool 型変数として扱う．コンテキスト内に存在する場合は真，存在しない場合は偽で表す．

チャネル（キュー）として扱うもの

プロセス間で通信するために，各プロセスが持っているものとする．キューの長さは検証時の状態爆発を避けるため，できる限り短い方が良い．文献 [1] ではキューの長さを 0 にしてもモデルの性質を失わない条件を次のように与えている．

> システムが持つ全ての状態プロセスについて，ある状態へ遷移するために通るパスはその全てが受信であるか，送信である（ただし，ε 遷移はどちらの場合に含まれても良い）．

本モデルはこの条件に一致しているため（図 3〜8 参照），チャネルの長さは全てのプロセスについて 0 に設定できる．

以上により，Struts アプリケーションの簡単なモデルを作成することができる．なお，ビジネスロジックは検証時のコストを抑えるため，提案モデルでは省略する．

2.3.2 異常な動作を考慮したモデル

二重送信やタイムアウト等の異常な振る舞いは，ネットワークプロセスに図 9 のような記述を加えて表現する．ネットワークプロセスに外乱を加えた場合，クライアントおよびリクエストプロセッサのモデルにも変更を加える必要がある．以下ではこれらのモデルを変更する方法を述べる．

図 9 のコードの 2 行目では，ネットワークプロセスが

```
do
::client_net?request -> 
  if
  ::net_searvlet!request;   /*正常時*/
  ::net_searvlet!request;
    net_searvlet!request;  /*二重送信時*/
  ::skip;                  /*タイムアウト時*/
  fi;
od
```

図 9 外乱を発生するネットワークの PROMELA（抜粋）

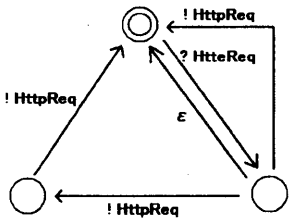

図 10 外乱を発生するネットワークプロセス

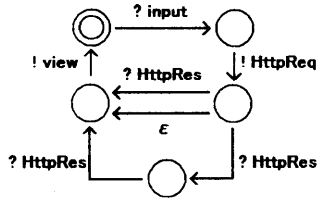

図 11 外乱発生を前提としたクライアントプロセス

client_net というチャネルからの受信を待っており，request という変数に受信内容を保存することを意味している．その後で表れる if 文では各分岐に条件節が無く，クライアントから受信したリクエストが以下のいずれの処理を受けるのかは非決定的であることを示す．

（1） そのままサーブレットに送信
（2） 二重化してサーブレットに送信
（3） 消失

無限ループにより，ネットワークプロセスは以上のような動作を繰り返すことになる．また変更したネットワークプロセスの状態図は図 10 である．この状態図では?HttpReq がリクエストの受信を表しており，正常時は 1 度，二重送信時は 2 度リクエストを送信（!HttpReq）して初期状態に戻る．また ε 遷移はタイムアウト時の状態遷移である．

クライアントプロセスの変更（図 11）

ネットワークプロセスがタイムアウトを発生した場合，クライアントおよびユーザのプロセスは応答を待ち続けることになる．しかし現実ではエラー画面が表示され，ユーザは新たにリクエストを送信し直すことができる．このためクライアントプロセスではサーバ側から一定時間以上応答が無い場合には現在の状態をユーザに返すこととし，デッドロックを回避する（図 11 中の ε 遷移）．またネットワークプロセスが二重送信を発生した場合，リクエストプロセッサプロセスから返されるレスポンスも二重化されている．この場合，現実には二重の処理が実行された後に画面遷移し，ユーザは操作を続行できる場合が多い．従ってクライアントプロセスの受信も二重化する必要がある（図 11 でリクエスト受信（?HttpRes）を 2 度行う遷移が該当）．PROMELA コードでは bool 型の外部変数 wsend を宣言しておき，クライアントプロセスは wsend の値を評価して受信回数を決定する．

リクエストプロセッサプロセスの変更

二重送信が発生した場合は前節で述べた条件の限りではなく，チャネルの長さが 0 の場合は次の条件の両方が成り立つ状況下でデッドロックを発生する．

（1） アクションはリクエストプロセッサに遷移情報を送信できない
（2） リクエストプロセッサはアクションにリクエストを送信できない

この状況では，リクエストプロセッサは最初のリクエストを受けてアクションを呼び出し，アクションは処理を完了して遷移情報を返そうとするが，その間にリクエストプロセッサは 2 番目のリクエストを受けて状態遷移しており，アクションからの遷移情報を受信できる状態ではなくなっている．また同様のことが次の条件の両方が成り立つ状況下でも発生する．

（1） リクエストプロセッサがサーブレットにリクエストを送信できない
（2） サーブレットがリクエストプロセッサにリクエストを送信できない

この状況では，まずリクエストプロセッサが最初のリクエストを受信してアクションを呼び出す．次にアクションは処理を終えてリクエストプロセッサに遷移情報を返す．このときの遷移情報によってリクエストプロセッサがサーブレットにリクエストを送信しようとする間に，サーブレットでは 2 番目のリクエストをリクエストプロセッサに送信しようとしており，デッドロックが発生している．

これらの状況に対処するため，リクエストプロセッサが持つ 2 つのチャネルは命令を 1 つ分だけ保持できるように変更した．

2.4 モデル自動生成ツール

我々は利用者の負担を軽減するため，提案手法に基づいたモデルを自動的に生成するツールを試作した．ツールの入力は Struts 設定ファイル（struts-config.xml）および，各 JSP が呼び出すアクション名の情報を記述した XML ファイルである．後者はツール利用の際，別途作成する必要がある．ツールはこれらの入力内容を反映し，Promela で記述されたモデルを出力する．このモデルでは LTL 記述による制約や，適切なプロセスに assertion を与えることにより SPIN 上で動作検証が可能になる．P という性質を検証したい場合，P に対する検証結果が valid である場合は性質が満たされることを示し，invalid である場合は性質が満たされないことを意味する．

3. 適用例題と実験

例題を用い，正常な状況および二重送信発生時の状況でのページ遷移について検証した．文献 [4] では検証系 SMV [9] を活用し，Next 演算子によるページ遷移の検証方法が述べられている．SPIN で Next 演算子を用いる場合は Partial Order

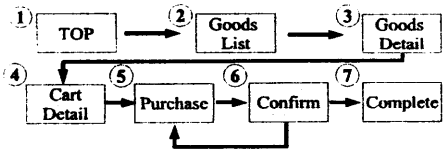

図 12　画面遷移図（メニューからの遷移情報を除く）

```
if
::viewState==7 -> assert(pre_view==6);
::else -> skip;
fi
```

図 13　正常動作時のページ遷移検証用 assertion

```
if
::viewState==7 -> assert(wsend==false);
::else -> skip;
fi
```

図 14　二重送信発生時の動作検証用 assertion

Reduction [2] を利用することができず，多大な検証コストが必要になる．そのため SPIN では通常，Next 演算子を用いない．従って同様の検証には若干の工夫が必要となる．以下ではまず例題の概要を説明し，ついで 2 つの検証項目に対する方針を述べる．

3.1　適用例題について

我々は提案手法の有効性を確かめるため，日本ユニシス・ラーニング株式会社の新入社員研修で開発された次のような WEB アプリケーションを対象に実験を行った．なお検証実験の対象は開発で用いられた設計書等から生成されたモデルであり，研修で開発された最終成果物を直接検証したわけではない．

- WEB 上で品物を購入するためのアプリケーション
- 論理設計書に画面遷移図が示されている
- 画面設計書に各画面が呼び出すアクション名等の情報が示されている
- リクエストプロセッサ数 1，アクションフォーム数 8，アクション数 9

画面遷移図（図 12）には Top 画面から購入完了（Complete）画面までのページ遷移情報が含まれている．また画面設計書には各画面が呼びだすアクション名に加え，設定ファイル（struts-config.xml）に記述する内容も含まれている．モデル生成の際，画面設計書を基に struts-config.xml ファイルを作成することで支援ツールを利用することも可能であるが，例題は作成済みのアプリケーションであるため，既存の struts-config.xml を利用した．またクライアント側のモデルは自動生成されたスケルトンコードに対し，画面設計書の情報を著者が追加記述した．

この WEB アプリケーションでは，全ての画面にメニューフレームが表示されている．図 12 では省略しているが，メニューフレームから可能な遷移（TOP 画面，Goods List 画面，CartDetail 画面）は全画面が持つ．

3.2　検証項目

3.2.1　正常な状況下でのページ遷移検証

画面遷移書では，購入確認（confirm）画面だけが購入完了（complete）画面への遷移を持つよう設計されている．前述のように Next 演算子を用いないため，この制約が満たされることを次のように検証した．

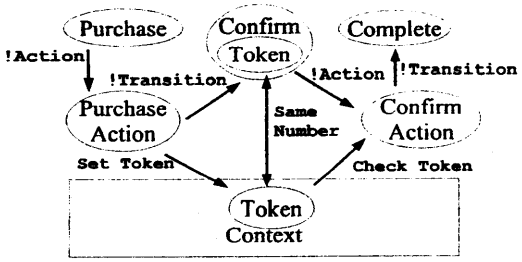

図 15　二重送信検出用の設計

（1）　各画面に付いている番号を確認する．ここでは図 12 のように番号が付いているものとする．

（2）　表示中の画面番号 viewState の他に，前画面番号 pre_view という変数を用意する．

（3）　ユーザプロセスがクライアントプロセスからメッセージを受ける直後に図 13 のような記述を加える．

図 13 は，表示中画面が購入完了画面であるとき，その前の画面が購入確認画面であることを検証するためのコードである．この性質が破られたとき，SPIN は直ちに検証を終了し，assertion 違反があったことを示す．

3.2.2　二重送信発生時の動作検証

この WEB アプリケーションにおいて，二重送信による影響が最も大きいのは購入完了確認画面でのミスクリックによるものであると考えられる．従って，購入確認画面から呼び出されるアクションは二重送信を検出し，対処する設計が望まれる．設計が適切であれば，ユーザが購入完了画面を二重に受信することはない．ここではこの性質を検証する．

まず二重送信の発生を示すフラグ wsend を利用し，ユーザプロセスに対して図 14 のような記述を与える．viewState==7，すなわち購入完了画面において，wsend が真であるとき，サーバは購入処理を二度行い，その結果をユーザに返したことになる．従って以下の手順でアクションのモデルに追加記述を行い，設計上そのような状況が発生しないことを検証する．

二重送信の検出は，呼び出されたアクションの冒頭で行われる．従って，購入確認画面から呼び出されるアクションは購入完了画面への遷移情報を返す前に，購入確認画面に埋め込まれているトークン番号を参照する必要がある．また購入確認画面にトークン番号が埋め込まれるためには，購入手続き画面から呼び出されたアクションが購入確認画面への遷移を返す前に，コンテキスト内の session スコープにトークン番号を設定し，同じ番号を購入確認画面に埋め込む処理が必要である（図 15）．以上の内容を該当するアクションモデルに記述すると共に，検出された二重送信の片方を無視するよう動作設計を行う．

4.　実験結果と考察

本章では，前章で行った実験の結果と考察について述べる．例題アプリケーションでは全画面にメニューフレームが表示されるので，画面遷移は無限ループである．そのため今回の実験

表 1 実験における検証コスト

	項目 1		項目 2	
	a	b	a	b
状態数	$5.23e+06$	18604	$6.10e+06$	207060
使用メモリ (MB)	1783	9.08	2077	70.812
計算時間 (秒)	136	0	1380	0
結果	Valid	Invalid	Valid	Invalid

では実行ステップの深さを 10000 に限定した．また各項目について a（図 13，14 による assertion 定義）と b（図 13，14 の assertion 定義において「==」を「!=」で置き換えたもの）の検証を行った．ここで，本来満たすべき a の性質を P とおく．項目 1b，2b の検証は，それぞれ P が実行パスの中で成立することを証明するために必要である．この検証結果が invalid であれば実行パス中に P が存在することを保証し，valid であった場合は満たすべき性質を違反しているか，購入完了画面を通るような実行パスが指定された探索範囲で見つからなかったことを示す．なお，検証実験に利用した計算機はペンティアム 4 プロセッサ 3GHz，搭載メモリ 2GByte，OS は Windows XP Professional Edition であり，SPIN は Cygwin 上で起動した．

4.1 実験結果

4.1.1 正常な状況下でのページ遷移について

ステップ数 10000 以下という条件の下で，「購入完了画面の前画面が購入確認画面である」という性質が満たされること（表 1 項目 1a），および「購入完了画面の前画面が購入確認画面であることはない」という性質が満たされないこと（表 1 項目 1b）を検証した．

4.1.2 二重送信時の振る舞いについて

ステップ数 10000 以下という条件の下で，「購入完了画面への遷移はただ 1 度だけ受信する」という性質が満たされること（表 1 項目 2a），「購入完了画面をただ 1 度だけ受信することはない」という性質が満たされないこと（表 1 項目 2b）を検証した．

4.2 考察

4.2.1 正常な状況下でのページ遷移検証

検証の結果，購入完了画面の前画面が購入確認画面である状況が存在することを確認した．またユーザが各画面でどのように選択しても，購入完了画面の前に購入確認画面を通らない可能性が極めて低いことを確認した．

実行パス上で購入完了画面を経由することがない場合も項目 a の検証結果は同様であるため，満たすべき性質が存在することを確認するために項目 b の検証が必要である．

例題では購入確認画面から購入完了画面への 1 遷移分のみを扱ったが，同様の手順で if 文の分岐節を追加することにより，システム全体のページ遷移が仕様を満たすことを検証できる．

4.2.2 二重送信発生時の自動検証

項目 2b の検証結果より，購入完了画面を受信し，それが二重送信ではない状況が存在することを確認した．変更前のモデルでは assertion 違反の振る舞いが存在したが，方針に従ってモデルに記述を加えた結果，assertion 違反が発生する可能性が極めて低くなった（表 1 項目 2a）．SPIN はエラーを迅速に発見できるツールであるが（表 1 項目 1b,2b），エラーが存在しない場合はステップ範囲内における全状態空間を探索するため，検証時間が長くなる（表 1 項目 1a,2a）．項目 2a は項目 1a より非常に長い検証時間が必要であった．その理由として，項目 2 の方が元のスケルトンコードに対して検証のために追記したコードが多いことや，検証性質がより複雑であることが考えられる．従って，検証用の追加記述方法にも工夫が必要であると言える．

5. むすび

本論文では，Struts アプリケーションの動作検証を目的として，モデリング手法および PROMELA コードを自動生成するための支援ツールを提案，実装した．提案手法により生成されたモデルでは，外乱プロセスの導入によって異常時の振る舞いを表現することができ，異常検出時の動作設計についても検証することが可能である．検証実験を通して，提案手法が WEB アプリケーション設計時のページ遷移異常や，誤操作を受けた場合の動作設計におけるミスの検出に効果があることを示した．

今後の課題の 1 つは実行ステップに制約をおかず，LTL で期間を指定し，より正確な検証式で証明を行えるよう対処することである．また，典型的な検査項目を表す LTL の生成支援，状態爆発への対処などについても検討していきたい．

謝辞 3. で述べた適用例題をご提供頂いた日本ユニシス・ラーニング株式会社毛利幸雄様，星野隆之様に深謝致します．

文献

[1] X. Fu, T. Bultan, J. Su: "Analysis of Interactin BPEL Web Services," Proceedings of the 13th international conference on World Wide Web (WWW '04), pp.621–630, 2004.

[2] G. J. Holzmann: "The SPIN Model Checker Primer and Refference Manual," Addison-Wesley Pub, 2003.

[3] G. J. Holzmann: "The Model Checker Spin," IEEE Trans. on Software Engineering, Vol.23, No.5, pp.279–295, 1997.

[4] 崔銀惠，河本貴則，渡邊宏: "画面遷移仕様のモデル検査"，産業技術総合研究所 システム検証研究センター テクニカルレポート, PS-2005-002, 2005.

[5] 崔銀惠，渡邊宏: "Web アプリケーションのクラス設計仕様に対するモデル化と検証"，産業技術総合研究所 システム検証研究センター テクニカルレポート, PS-2005-003, 2005.

[6] 加藤大樹，結縁祥治，阿草清滋: "高信頼 Web アプリケーションのための振舞い合成手法"，日本ソフトウェア科学会 第 2 回ディペンダブルソフトウェアワークショップ，pp.41–50, 2005.

[7] 加藤啓史，結縁祥治，阿草清滋: "Struts に基づく Web Application に対する振舞い動作記述の導入"，日本ソフトウエア科学会第 20 回記念大会, 6C-1, 2003.

[8] A. Betin-Can, T. Bultan, X. Fu Publication Date: "Design for Verification for Asynchronously Communicating Web Services," Proceedings of the 14th international conference on World Wide Web (WWW '05), pp.750–759, 2005.

[9] A. Cimatti, E. Clarke, F. Giunchiglia and M. Roveri: "NuSMV: a new symbolic model verifier," In Lecture Notes in Computer Science, Vol.1633, pp.495–499, 1999.

[10] S. Burbeck: "Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC)," http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html

[11] Apache Struts Project: http://struts.apache.org/