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

信学技報
TECHNICAL REPORT OF IEICE.

# SPINを用いたウェブアプリケーションにおける階層別モデル検査支援方法

浜口　優[†]　吉村　顕[†]　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町 1-3
E-mail: †{y-hamagt,a-yosimr,okano,kusumoto}@ist.osaka-u.ac.jp

**あらまし**　ウェブアプリケーションの開発では, ユーザの誤入力である二重送信やタイムアウト発生時の動作, さらにはウェブアプリケーションの内部処理におけるデッドロックを考慮する必要がある. このため設計誤りはできる限り早期に検出することが望まれる. 近年, ソフトウェア検証手法としてモデル検査が注目を集めている. しかし, モデル検査においては, 状態爆発を考慮に入れた上での検証モデルの抽出や記述の難しさが問題点になっている. 本稿ではStrutsを用いたウェブアプリケーション仕様を, ページ遷移を検証するモデルと内部処理を検証するモデルに階層別にモデル化するモデル検査手法を提案する. 階層別に行うことにより状態爆発を回避している. また提案手法をもとにした検証用モデル生成支援ツールの試作を行った. この手法をある企業の新人研修で開発されたウェブアプリケーションをもとにした例題に適用し, 手法およびツールの実用性の評価実験を行い, 本手法の有効性を確かめた.

**キーワード**　モデル検査, Promela, SPIN, Struts, 共有資源, 状態爆発

# Hierarchical Model-checking for Web Application with SPIN

Yuu HAMAGUCHI[†], Akira YOSHIMURA[†], Kozo OKANO[†], and Shinji KUSUMOTO[†]

† Graduate School of Information Science and Technology, Osaka University
1-3 Machikaneyama, Toyonaka, Osaka, 560-8531, Japan
E-mail: †{y-hamagt,a-yosimr,okano,kusumoto}@ist.osaka-u.ac.jp

**Abstract**　In the development of the web application, double-click actions of users, time-out and deadlock should be considered. Therefore, such design faults should be found as early as possible. Recently, the model-checking attracts attention as the effective software verification technology. However, to create effective models and describing properties have become main issues of model-checking. This paper proposes a hierarchical model-checking for the web application implemented in framework Struts. The method, thus, avoids state explosion. The method is applied to a web application developed as an exercise of an enterprise's new comers' training. The result shows that the proposed method is useful.

**Key words**　Model-checking, Promela, SPIN, Struts, Shared resource, State explosion

## 1. まえがき

ウェブアプリケーション開発は効率的なシステム開発のためにMVC設計モデルに基づいたフレームワークを用いることが多い. MVC設計モデルではウェブアプリケーションの構成要素をユーザの視点 (View), システム制御部 (Controller), ビジネスロジック (Model) の三つとして独立に捉える. 開発フレームワークの1つであるStruts[1]では, 主にControllerとViewの開発を支援する.

ウェブアプリケーションの動作がページ遷移異常やデッドロックを回避するように設計されていることは重要である. モデル検査では上位仕様を基にシステムの動作を検証できるため, 設計の正しさを上位工程で保証するための手段として有効であると考えられる. 本稿ではStrutsを利用したウェブアプリケーションの動作をViewおよびControllerから成る上位層, ControllerとModelから成る下位層 (以下, ビジネスロジック層) に分けて捉え, モデルチェッカSPIN[2], [3]を用いた自動検証のためのモデルをそれぞれの層から生成する手法を提案する.

SPINのモデル記述用言語Promela[2]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. 上位層のモデルでは, サーバの振る舞いをStruts設定ファイル (struts-config.xml), クライア

ントの振る舞いを画面設計書から生成した．また下位層では処理の中心となるアクションおよびビジネスロジックの機能を仕様書から抽出し，モデルを生成した．さらにモデル化支援のために，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，その有用性を評価した．検証用言語への変換に関する研究は XML を用いたもの [5], [6] など多い．

SPIN [2], [3] は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を持つ．ただしモデル検査では検証時の状態爆発を回避することが大きな課題になっており，文献 [7] のように重要な特性のみを抽出してモデル化する手法も数多く考案されている．検証コストを抑えるために非同期通信モデルを同期通信モデルに置き換える手法 [8] もあり，本稿の提案モデルはこの手法を応用している．その他，関連研究として文献 [9], [10] はウェブ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 [4], [11] 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，モデル化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．

本稿の構成は以下の通りである．**2** では本研究に関係するフレームワーク Struts，検証器 SPIN について述べる．**3** では上位層，下位層に対してそれぞれの提案するモデルとその検証方法を述べる．**4** では各階層の検証用モデル生成支援ツールについて述べる．**5** では提案手法の有用性を評価するための例題と実験結果について述べ，**6** で本稿をまとめる．

## 2. 準　　備

本章では Struts, SPIN の概要を述べる．

### 2.1 Struts

MVC 設計モデルに基づいたウェブアプリケーション開発のためのフレームワークである．Struts を利用したウェブアプリケーションは次の要素から構成される．

- クライアント及び 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．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．サーブレットから渡されたリクエストを受け，リクエストの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．1 つの Struts アプリケーションについて 1 つ以上存在するモジュールである．
- アクションフォーム
- アクション
- コンテキスト：様々なデータを格納する．
- ビジネスロジック：アクションから呼び出され，具体的な処理を実行する．データベースへのアクセスを伴うことが多く，デッドロック等が発生しないように設計する必要がある．
- データベース：共有資源があり，処理の過程で主にビジネスロジックからのアクセスを受ける．

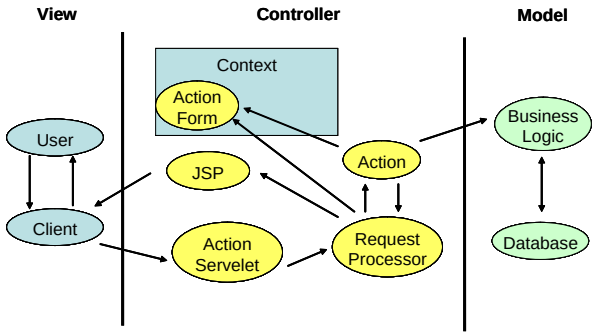


図 1　Struts の構成

```
<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）

Struts ではリクエストプロセッサが中心的な役割を果たしていて，設定ファイル (struts-config.xml) にその動作を記述する．図 2 は index.do という http リクエストを受けた場合，path=”/index”を持つ action タグで指定された Index アクションが呼び出され，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，処理が完了するとリクエストプロセッサに init という遷移情報を返すように定義されている．最後にリクエストプロセッサはその遷移情報に対応した 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であることを利用する．

### 2.2 SPIN

SPIN はモデル検査ツールである．検査したいシステムを検証言語 Promela を用いて状態遷移プロセスとしてモデル化し，そのモデルと，線形時相論理 (LTL) で表現されたシステムが決して満たしてはいけない NeverClaim を入力として与え，システムが取り得る全状態空間を自動で探索する．NeverClaim を満たす状態に到達してしまった場合は反例を出力する．NeverClaim を満たす状態がない場合，システムは NeverClaim を満たさないということが保証される．例えば式「$P -><> Q$」は「$P$ が成り立つならば，いつか $Q$ が成り立つ」という意味の LTL 式である．

## 3. 提案するモデル検査手法

Struts アプリケーションは MVC 設計モデルに基づいて構成されている．この特徴を利用し，ウェブアプリケーションの振る舞いを View と Controller から成る上位層，Controller と Model から成る下位層に分割してモデル化する手法を述べる．この手法により各層を独立してモデル検査することができ，状態数の軽減などが期待できる．

### 3.1 ページ遷移検証用モデル

上位層では基本的に，共有資源の確保は存在しない．よってシングルタスク環境を想定してモデル化しても差し支えない．またこの層では二重送信とタイムアウトが起こる可能性がある．よってこの章ではこの 2 つの外乱が非決定的に起こり得るモデルも提案している．そのモデルを使用することで外乱が起こる状況下での動作検証が可能になる [12]．

#### 3.1.1 基本モデル

外乱が起きないと想定した基本モデルは 6 種類のプロセスがオートマトンとして同期したモデルで構成される [12]．このモデルはできるだけ忠実に，かつ必要最低限の要素で Struts の振る舞いをモデル化するという考えに基づいている．

（1） ユーザ： 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： 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： 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： 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： 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：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`）．

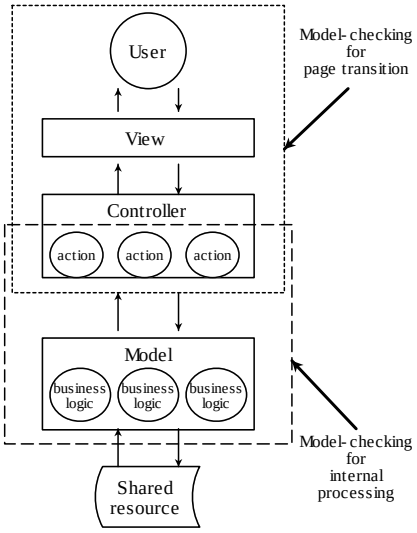


図 3 階層別モデル

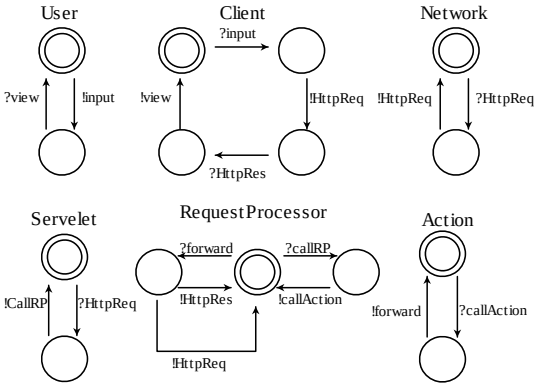


図 4 ページ遷移検証用モデルの各プロセス

#### 3.1.2 外乱を持つモデル

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 5 のような記述を加えて表現する．このネットワークプロセスはクライアントから受信したリクエストに対し以下の動作を非決定的に選ぶ．

（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

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．

a） クライアントプロセスの変更（図 6）

タイムアウトが発生した場合，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 6 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 6 でリクエスト受信（`?HttpRes`）を 2 度行う遷移が該当）．

b） 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，チャネルの長さが 0 の場合はデッドロックを発生する可能性がある．この状況に対処するため，リクエストプロセッサが持つ二つのチャネルは命令を一つ分だけ保持できるように変更した．

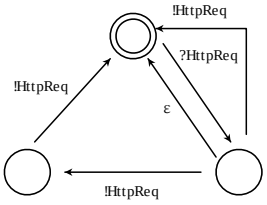


図 5　外乱を発生する Network

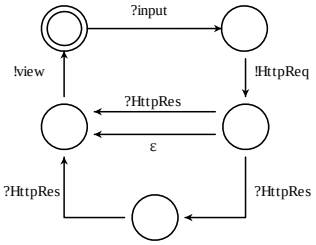


図 6　外乱を考慮した Client

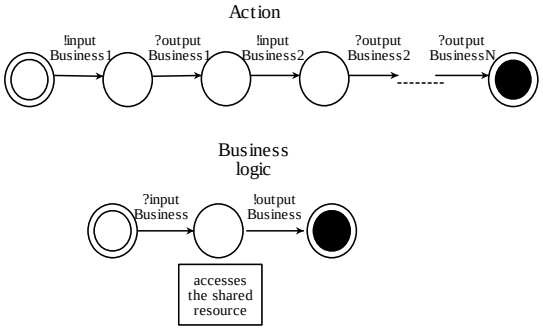


図 7　アクションプロセスとビジネスロジックプロセス

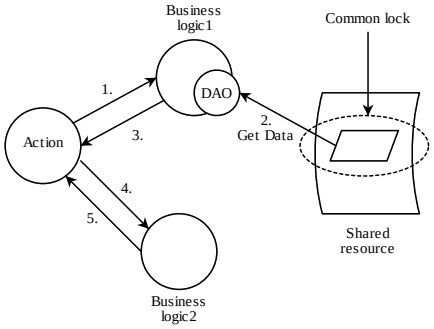


図 8　アクション単位の処理の検証用モデル例

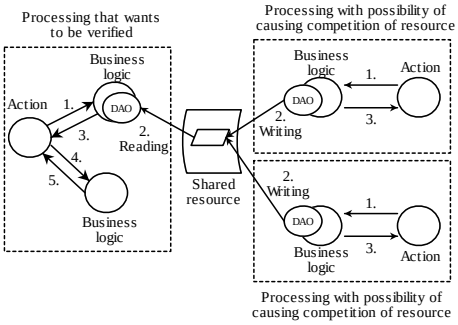


図 9　マルチタスク環境を想定した検証用モデル例

### 3.2　ページ遷移検証

このページ遷移検証用モデルと, 検証したい項目を示す LTL 式を SPIN に入力として与えることでページ遷移検証を行う. 例は **5** の適用実験で示す.

### 3.3　内部処理検証用モデル

次に下位層の内部処理検証用モデルについて述べる. 内部処理とは意味を持つまとまりある単位の処理で, Struts アプリケーションの 1 アクションが複数のビジネスロジックの呼び出し順序を制御して処理を実現している. よって以下ウェブアプリケーションの内部処理をアクション単位の処理と呼ぶことにする. アクション単位の処理は論理設計書の内容を基にモデル化する. アクション単位の処理は多くの場合, 共有資源にアクセスするので, マルチタスクに対応する設計をする必要がある. 以下のプロセスでアクション単位の処理を表現する.

（1） アクション：ビジネスロジックを呼び出すプロセス. アクションは 1 つ以上のビジネスロジックと交信を行い, ビジネスロジックを呼び出す順番を決定している. ビジネスロジックにパラメータを送信（`!inputBusiness`）すると待機状態になり, ビジネスロジックからの戻り値を受信（`?outputBusiness`）すると次に実行されるビジネスロジックを呼び出す. よってビジネスロジックを増やすごとにプロセスの状態数は増加していく.

（2） ビジネスロジック：アクションから呼び出しを受け（`?inputBusiness`）, ビジネスロジック内のデータアクセスオブジェクトが共有資源にアクセスする. 最後に適切な戻り値をアクションに返す（`!outputBusiness`）.

これら 2 種類のプロセスは, 互いのプロセスとの通信を 1 遷移のみで行うようにした. これは必要最低限の要素でアクション単位の処理を表現するためである.

#### 3.3.1　モデルと共有資源

ビジネスロジックプロセスは共有資源にアクセスすることが多い. ビジネスロジックプロセスが共有資源内のどの領域にアクセスしているのかを表現する必要がある. 複数のビジネスロジックプロセスが共有資源にアクセスすると資源の競合が起こる可能性がある. 資源の競合を防ぐ機能として共有ロックと排他ロックがある.

- 共有ロック：共有ロックされた領域は, 別のプロセスが中身を参照することはできるが書き換えを行うことはできない. また複数のプロセスが同時に同じ領域に共有ロックできる.
- 排他ロック：排他ロックされた領域は, 別のプロセスが中身を参照することも書き換えることもできない. また複数のプロセスが同時に同じ領域に排他ロックできない.
- データアクセスオブジェクト (以下 DAO)：ビジネスロジックが共有資源を扱う場合 DAO を通してアクセスする.

図 8 はプロセスやデータ全体の流れを表したアクション単位の処理のモデル例である. まずアクションがビジネスロジック 1 にパラメータを送信し (1), ビジネスロジック 1 は共有資源に共有ロックを掛けてデータを取得し (2), 戻り値をアクションに返す (3). 次にアクションはビジネスロジック 2 にパラメータを送信し (4), ビジネスロジック 2 は戻り値をアクションに返し (5), このアクション単位の処理は終了している.

### 3.4　内部処理検証

検証したいアクション単位の処理と資源の競合を起こす可能性のある別のアクション単位の処理が並列に実行されるようなモデルを作成する. このモデルと検証したい項目を表現した LTL 式を合わせて SPIN に読み込ませることで検証を行う. こうすることによりアクション単位の処理がマルチタスク環境で仕様を満たす動作を行うかどうかを検証することが可能になる.

```
<actions resource="3">
  <action name="ActionA">
    <business-objects>
    <business name="business1" order="0">
    <dao name="daoA" resource="1" order="0" type="read"/>
    <dao name="daoB" resource="2" order="1" type="write"/>
    </business>
    <business name="business2" order="1"/>
    </business-objects>
  </action>
</actions>
```

図 10　内部処理検証用モデル生成支援ツールの入力ファイル（抜粋）

## 4. 検証用モデル生成支援ツール

上位層と下位層のそれぞれについて，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．以下では各ツールについて簡単に述べる．

### 4.1 ページ遷移検証用モデルの自動生成ツール

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．

### 4.2 内部処理検証用モデル生成支援ツール

並列実行する複数のアクション単位の処理の情報を記述した入力ファイルを生成支援ツールに入力として与え, 出力される検証モデルのスケルトンにユーザが必要な情報を追加記述することでモデルは完成する．図 10 の入力用ファイルから図 7 の検証用モデルの基本形を得る．`actions` タグの `resource` 属性では共有資源の個数を定義する．内部に複数の `action` タグを持つことができ, それぞれに `name` 属性で名前が与えられる．各アクションが持つビジネスロジックの数は `action` タグ内部の `business` タグの数で決まり，名前（`name`）と呼び出される順序（`order`）を指定する．また共有資源を扱うビジネスロジックは `dao` タグを持ち, 図 10 の例では Business1 がリソース 1 に対して共有ロックした後，共有資源 2 に対して排他ロックする振る舞いを表現している．

## 5. 適 用 実 験

この章では提案手法および試作したツールを実務規模のウェブアプリケーションに適用し，検証実験を行う．また実験結果をまとめ，考察を与える．

### 5.1 適 用 例 題

提案手法の有効性を確認するため，企業の新入社員研修で実際に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(注1)．

- Web 上でレコードを購入するためのアプリケーション
- 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

#### 5.1.1 ページ遷移の正当性検査

画面遷移図には 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．支援ツールが出力したスケルトンコードを基にして，追加，修正を加えることで検証に必要なモデルを記述している．各画面には支援ツールが自動的に識別子を付加する．この識別子を LTL 式で参照し検証する．またページ遷移検証では基本モデルと外乱を持つモデルに対して検証をした．検証項目は「購入完了（complete）画面への遷移は購入確認（confirm）画面からのみである」である．

#### 5.1.2 内部処理の正当性検査

ウェブアプリケーションの個々のアクション単位の処理に対してモデル検査を行い, その設計の正当性を評価している．検証を行ったアクション単位の処理は以下の 2 つである．

- 購入確認アクション：共有資源内の注文番号を取得し, 新規注文番号の書き込みを行う．
- 検索アクション：共有資源への検索とヒットした商品群のソートを行う．

それぞれ資源の競合の可能性のあるアクション単位の処理として自分自身の処理のコピーを並列に実行させる．並列に実行させるアクション数は 2 から 6 までにした．また様々な外的要因で共有資源にロックが掛けられることも想定し, ランダムに共有資源にロック (共有ロック, 排他ロック) を掛ける外乱プロセスも並列に実行させることにした．この環境で各アクション単位の処理が仕様を満たす動作を行うかどうかを検証した．

### 5.2 実 験 結 果

検証実験の計算機はペンティアム 4, 3GHz, メモリ 2GByte, OS は WindowsXP であり, SPIN は Cygwin 上で起動した．また表中に記述してある M.O は Out of memory を意味する．

#### 5.2.1 ページ遷移の正当性検査

モデル検査の結果, 反例が出力された場合は適宜モデルを修正し, 再度モデル検査を行った．反例は例題全てにおいて一瞬で出力された．表 1, 2 で示す検証コストは全て, 検証結果が valid であった場合の数値である．また検証には次のような 5 種類のオプションを利用した．

- Default: オプションを与えない通常の検証
- Collapse: Default よりメモリを効率よく利用できる全状態探索
- MA: Collapse よりメモリを効率良く利用できるが多大な時間を要する全状態探索
- HashCompact: 迅速に検証を行う近似的探索
- BitState: HashCompact よりさらに迅速に検証を行う近似的探索

#### 5.2.2 内部処理の正当性検査

ページ遷移検証と同様に, モデル検査の結果, 反例が出力された場合は適宜モデルを修正し, 再度モデル検査を行った．反例は例題全てにおいて一瞬で出力された．表 3, 4 で示す検証コストは全て, 検証結果が valid であった場合の数値である．数値が記入されていない部分は状態爆発を起こしている．

(注1)：ただし，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．

### 5.3 考　　察

**5.3.1** ページ遷移の正当性検証について

二重送信とタイムアウトが両方発生する総合的なモデルでの検証では状態爆発が発生した．従ってシステムの様々な側面を同時に検証することは困難であり，問題となる事象を個別にモデル検査する必要があると言える．

**5.3.2** 内部処理の正当性検証について

検証の結果, 共有, 排他ロックのタイミングや外乱プロセスの影響でアクション単位の処理が正常に動作しない可能性があることが確認できた．

ウェブアプリケーションをアクション単位の処理でモデル化せず, 上位層, 下位層全てをモデル化して先で述べたような並列実行を行った場合, 簡単なウェブアプリケーションでもすぐに状態爆発を起こす. 以上よりアプリケーション全体のモデル検査を行う場合, 階層別モデル検査手法は有効であると考えられる．

表 1　外乱のないモデルの検証コスト

| | Default | Collapse | MA | HC | BitState |
|---|---|---|---|---|---|
| 探索状態数（百万） | 3.0 | 6.0 | $6.0e+06$ | 6.0 | 1.0 |
| 使用メモリ（MB） | M.O | 932 | 781 | 738 | 722 |
| 計算時間（秒） | 37 | 141 | 1701 | 37 | 5 |

表 2　外乱発生時モデルの検証コスト

| | Default | Collapse | MA | HC | BitState |
|---|---|---|---|---|---|
| 探索状態数（百万） | 3.0 | 29 | 150 | 72 | 1.0 |
| 使用メモリ（MB） | M.O | M.O | 1989 | M.O | 744 |
| 計算時間（秒） | 47 | 1064 | 55922 | 1347 | 24 |
| 探索状態数（百万） | 3.0 | 23 | 23 | 23 | 1.0 |
| 使用メモリ（MB） | M.O | 1751 | 1046 | 1011 | 723 |
| 計算時間（秒） | 38 | 1048 | 7069 | 194 | 5 |
| 探索状態数（百万） | 3.0 | 29 | 236 | 72 | 1.0 |
| 使用メモリ（MB） | M.O | M.O | M.O | M.O | 749 |
| 計算時間（秒） | 41 | 1128 | 101190 | 1339 | 25 |

上から順に 二重送信, タイムアウト, 両方の場合

表 3　Default オプションでの検証コスト

| 購入確認アクション | | | | |
|---|---|---|---|---|
| アクション数 | 時間 (秒) | メモリ (MB) | 状態数 | 結果 |
| 2 | 10 | 705 | 約 120 万 | valid |
| 3〜6 | M.O | M.O | M.O | M.O |
| 検索アクション | | | | |
| アクション数 | 時間 (秒) | メモリ (MB) | 状態数 | 結果 |
| 2 | 24 | 957 | 約 200 万 | valid |
| 3〜6 | M.O | M.O | M.O | M.O |

表 4　BitState オプションでの検証コスト

| 購入確認アクション | | | | |
|---|---|---|---|---|
| アクション数 | 時間 (秒) | メモリ (MB) | 状態数 | 結果 |
| 2 | 12 | 722 | 約 160 万 | valid |
| 3 | 22 | 732 | 約 330 万 | valid |
| 4 | 27 | 731 | 約 360 万 | valid |
| 5 | 30 | 731 | 約 380 万 | valid |
| 6 | 36 | 732 | 約 400 万 | valid |
| 検索アクション | | | | |
| アクション数 | 時間 (秒) | メモリ (MB) | 状態数 | 結果 |
| 2 | 19 | 735 | 約 220 万 | valid |
| 3 | 32 | 749 | 約 340 万 | valid |
| 4 | 36 | 750 | 約 375 万 | valid |
| 5 | 40 | 750 | 約 390 万 | valid |
| 6 | 45 | 854 | 約 405 万 | valid |

## 6. む　す　び

本稿では，Struts アプリケーションの動作検証を目的として，モデル化手法および 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した上位層のモデルでは，外乱プロセスの導入によって二重送信やタイムアウトを表現した．下位層のモデルでは，仕様書の内容から情報を抽出し，必要最小限のモデルを記述した．これによりコードの到達不能箇所を要求の漏れとみなすことができ，デッドロック発生時の状況の把握も容易になった．実用例題に適用した結果, 提案手法の有用性が確認できた．今後の課題はより多数のプロセスを用いたモデルの状態爆発の克服である．

**謝辞**　**5.** 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] Apache Struts Project: http://struts.apache.org/
[2] G. J. Holzmann: “The Model Checker Spin,” IEEE Trans. on Software Engineering, Vol.23, No.5, pp.279-295, 1997.
[3] G. J. Holzmann: “The SPIN Model Checker Primer and Refference Mannual,” Addison-Wesley Pub, 2003.
[4] 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,” 日本ソフトウェア科学会第 20 回記念大会, 6C-1, 2003.
[5] E. Mikk, Y.Lakhnech, M.Siegel, G. J. Holzmann: “Implementing Statecharts in PROMELA/SPIN,” Proc. of 2nd IEEE Workshop on Industrial Strength Formal Specifigation Techniques, pp.90-101, 1998.
[6] S. Nakajima: “Verification of Web Service Flows with Model-Checking Techniques,” First Int. Symp. on Cyber Works(CW'02), pp.0378-0385, 2002.
[7] M. Haydar, S. Boroday, A. Petrenko, H. Sahraoui: “Properties and scopes in web model checking,” Proc. of the 20th IEEE/ACM Int. Conf. on Automated software engineering, pp.400-404, 2005.
[8] X. Fu, T. Bultan, J. Su: “Analysis of Interacting BPEL Web Services,” Porc. of 13th Int. Conf. on World Wide Web(WWW '04), pp.621-630, 2004.
[9] 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.
[10] 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.
[11] A. Betin-Can, T. Bultan, X. Fu Publication Date: “Design for Verification for Asynchronously Communicating Web Services,” Proc. of 14th Int. Conf. on World Wide Web (WWW '05), pp.750-759, 2005.
[12] 藤原貴之, 岡野浩三, 楠本真二: “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,” 信学技報, vol.105, no.491, pp.73-78, 2005.