社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2007－SE－157 (10)
2007／9／28

多重マルコフ連鎖を用いた統計的テストのための精密化利用モデルについて

高木 智彦　　古川 善吾

香川大学工学部

ソフトウェア信頼性の評価を可能とする統計的テスト法では，利用モデルに基づいてテストケースが生成される．利用モデルは，ユーザによるソフトウェアの利用方法を確率的に表したモデルである．この利用モデルが実際のユーザの利用特性を正確に反映しなければ，ソフトウェア信頼性を正確に評価することは困難である．そこで本稿では，多重マルコフ連鎖を用いることによって，利用特性のより正確な反映が可能な精密化利用モデルを構築する手法について議論する．本手法による利用モデルが，単純マルコフ連鎖や斉時マルコフ連鎖を用いる従来のものよりも正確であり，自動的に構築できることが分かった．同時に，いくつかの課題を明らかにすることができた．

The Detailed Usage Model of a High-Order Markov Chain for Statistical Testing

TOMOHIKO TAKAGI　　ZENGO FURUKAWA

Faculty of Engineering, Kagawa University

In statistical testing which enables estimation of software reliability, testcases are generated based on a usage model. The usage model represents the way users operate software. When the usage model doesn't reflect usage characteristics of actual users, it is difficult to estimate software reliability accurately. This paper shows the method of constructing a detailed usage model of a high-order Markov chain in order to represent the usage characteristics more accurately. It is illustrated that the detailed usage model is more accurate than common usage models of a simple or time-homogeneous Markov chain and its construction can be automated.

1. はじめに

近年，コンピュータシステムの品質はソフトウェアに大きく依存する傾向があり，それだけソフトウェア信頼性の重要性が増している．ソフトウェア信頼性とは，実際の利用環境下でユーザがソフトウェアの欠陥に遭遇することなく利用できる可能性のことである．これを改善するための直接的な活動としては，インスペクション，静的解析，ソフトウェアテストなどが知られているが，特にソフトウェアテストはソフトウェア開発コストの50%以上を占める主要な工程である．ソフトウェアテストを効果的に行うための技法は従来から数多く提案されており [1, 2, 3]，特に，ソフトウェアの機能やソースコードを特定の基準に基づいて網羅するための系統的テスト法は開発現場で広く用いられている．系統的テスト法は欠陥を網羅的に検出することが可能であり，また，欠陥の潜在する可能性の高い箇所に注力できるという特長がある．しかしながら，大規模で複雑なソフトウェアを網羅することは時として困難であり，仮に網羅できたとしても，一定のソフトウェア信頼性を達成したことの根拠にはならない．また，ソフトウェアに残存する欠陥の予測数をソフトウェア信頼性の尺度とするため，ソフトウェア信頼性の評価方法としては必ずしも適さない．そのような系統的テスト法の欠点を補完する技法の一つとして，統計的テスト法（statistical testing）[4, 5, 6] が提案されている．

統計的テスト法は，欠陥を網羅的に検出することよりもソフトウェア信頼性を評価することに主眼を置いた技法である．テストケースは，ユーザの振る舞いを定義したマルコフ連鎖に基づき，ランダムに生成する．このマルコフ連鎖を利用モデル（usage model）という．利用モデルが正確であるほど（すなわち，利用モデルが現実のユーザの振る舞いに近いほど），ソフトウェア信頼性を正確に評価することができる．したがって，利用モデルの正確さは統計的テスト法の有効性を左右する重要なポイントである．しかしながら，従来研究はこの問題に対して十分な解決方法を示しているとはいえない．たとえば，利用モデルとして単純マルコフ連鎖（simple Markov chain）や斉時マルコフ連鎖

(time-homogeneous Markov chain) を用いることが一般的である．単純マルコフ連鎖は，イベントの生起確率が現在状態のみに依存するモデルであり，一方の斉時マルコフ連鎖は，状態の定義を行わずにイベントの生起確率のみで構成するモデルである．このような利用モデルは容易に作成できる反面，モデルが単純なため現実のユーザの振る舞いを正確に反映すること（すなわち，ソフトウェア信頼性を正確に評価すること）が困難となる場合がある．

そこで，本稿は正確な利用モデルを構築するために，多重マルコフ連鎖 (high-order Markov chain) を用いる手法（以下，本手法）を提案する．多重マルコフ連鎖は，イベントの生起確率が現在状態だけでなくそれ以前の状態遷移にも依存するモデルである．これを用いた利用モデルが従来のものよりも正確であり，自動的に構築できることを示すのが本稿の目的である．そのために，まず 2 節で従来の統計的テスト法の概要と問題点を整理する．次に 3 節で本手法の手順を，4 節で本手法の適用例を述べる．そして 5 節では利用モデルの正確さと構築コストの観点から考察を行い，最後に 6 節で従来研究との関連を述べる．

2. 従来の統計的テスト法

2.1 概要

従来の統計的テスト法の概要を図 1 に示す．以下に各プロセスについて説明する．

プロセス 1. 問題領域からソフトウェアへの要求を分析する．そして，利用モデルの構造（以下，利用構造）として，ステートマシン図あるいは操作リストを定義する．

プロセス 2. 問題領域から仕様を導出し，これに基づいてソフトウェアを実装する．利用構造も仕様の一部であり，ソフトウェアはこれを反映しなければならない．

プロセス 3. ソフトウェアの利用現場を調査し，フィールドデータ（ソフトウェアの実行履歴や業務の過程で発生する情報など）を収集する．

プロセス 4. 収集したフィールドデータを用いて，利用構造上での確率分布を導出する．利用構造がステートマシン図の場合は単純マルコフ連鎖としての，操作リストの場合は斉時マルコフ連鎖としての利用モデルが完成する．

プロセス 5. 利用モデルをテストケース生成器として任意の方法で実装し，統計的に十分な量のテストケースを生成する．

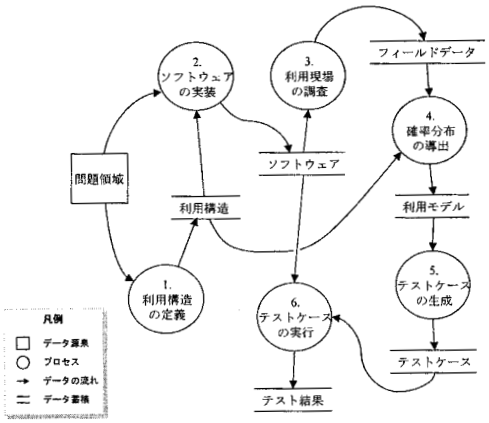


図 1: 統計的テスト法の概要（DFD による表現）

プロセス 6. 生成したテストケースを実行するためのテストドライバを用意し，ソフトウェアを自動テストする．その結果，ソフトウェア信頼性をはじめとしたテスト結果を得る．

2.2 問題点

統計的テスト法を効果的に行うためには，利用モデルが正確でなければならない．正確な利用モデルを構築するには，(a) 信頼できるフィールドデータを収集すること，(b) 利用構造を精密化すること，の 2 点が必要である．

(a) については従来研究でいくつか提案があり，すでに解決済みである．たとえば，Shukla ら [7] は，ソフトウェアコンポーネントの利用モデルを構築するために，Java API に探針（実行履歴を記録するためのプログラム片）を挿入した．また，Kallepalli ら [8] は，Web システムの利用モデルを構築するために，Web サーバのアクセスログを用いることができることを示している．

一方，(b) は十分に検討されていない．ステートマシン図は操作リストよりも精密であるため，利用構造としてはより適切であるといえる．Walton ら [6] は，ステートマシン図において，再訪可能状態をその訪問回数で区別して再定義する方法を例示している．たとえば，ある再訪可能状態 s について，1 回目に訪問した状態 s_0 と 2 回目以降に訪問した状態 s_1 に再定義する．これによって，s_0 と s_1 の確率分布が s の確率分布として平均化される，という問題を解決できる．しかしながら，この方法は利用モデルが巨大になるのでコスト的な限界があると指摘している．

s_0 と s_1 は，それぞれを起点としてまったく同一のイベント／アクションシーケンスを生成できるという意

味で等価であり，これらを等価状態（equivalent state）という [3]．以上より，正確な利用モデルを構築するためには，等価状態を現実的なコストで作成する手法が必要である．

3. 多重マルコフ連鎖を用いた精密化利用モデル

3.1 手順

2.2 節で述べた問題点を解決するために，多重マルコフ連鎖を用いた精密化利用モデル [9] の構築手法を提案する．多重マルコフ連鎖は，イベントの生起確率が現在状態だけでなくそれ以前の状態遷移にも依存するモデルである．本手法はツールによる自動化が可能であり，また，精密化（すなわち等価状態の作成）の対象を再訪可能状態のみに限定しないという特長がある．

本手法を組み込んだ統計的テスト法の概要を図 2 に示す．図 2 では，図 1 から変更のある部分を強調表示した．プロセス 1', 3', 4' は特化され，新たにプロセス 1"が追加された．以下では，本手法を構成するそれらのプロセスについて説明する．

プロセス 1'. 従来の統計的テスト法と同様に，利用構造としてステートマシン図 [10] を定義する．本手法は利用構造として操作リストを用いない．本プロセスで作成する利用構造をプロセス 1"で作成するものと区別するために，「標準利用構造」と呼ぶことにする．標準利用構造では，「ユーザが特定の操作を行う」というイベントを遷移のラベルとして，また，ユーザの振る舞いの段階を状態として表す．普通，標準利用構造は等価状態を含まない．なぜなら，ソフトウェアの仕様としては等価状態は冗長だからである．状態や遷移はアクションを持つことができるので，必要に応じて，テストデータを入力したりテスト出力を検証する方法を形式言語で定義する．

プロセス 1". マルコフ連鎖の多重度 m を決定する．そして，精密化利用構造として，m 重マルコフ連鎖のためのステートマシン図を構築する．m は，精密化利用構造の各状態が記憶する状態遷移数に 1 を加えた数である．精密化利用構造の状態は，標準利用構造における長さ $(m-1)$ の状態遷移シーケンスによって特徴付けられる．言い換えれば，精密化利用構造の状態空間は，標準利用構造における $(m-2)$ スイッチ [2] の全カバレッジ要素に相当する．そこで，まずは標準利用構造上のパスを網羅的に探索することによって全状態を識別する．次に，識別した各状態間で起こり得るすべての遷移を標準利用構造に基づいて追加すれば，精密化利用構造が完成する．

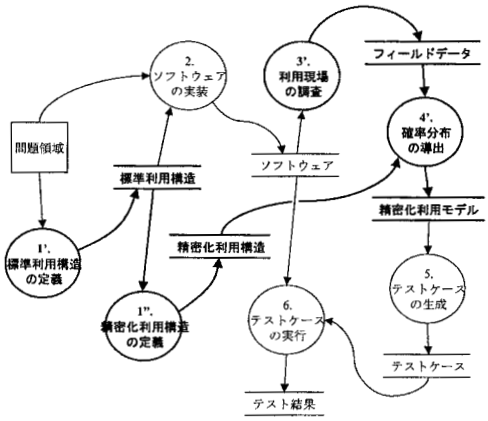


図 2: 本手法を組み込んだ統計的テスト法の概要

プロセス 3'. フィールドデータとしてイベントシーケンスを収集する．イベントシーケンスは，テスト対象ソフトウェアのプロトタイプや前バージョン，類似する他のソフトウェアの利用現場での実行履歴であり，精密化利用構造に直接対応付け可能なものである（精密化利用構造に対応付け可能であれば，標準利用構造にも対応付け可能である）．イベントシーケンスを収集するためには，ソフトウェアへの探針挿入が必要となる可能性がある．その場合は，標準利用構造がソフトウェアの仕様でもあることを利用して，標準利用構造からプログラムを生成する際に探針を自動挿入するとよい [11]．

プロセス 4'. 収集したイベントシーケンスで精密化利用構造をトレースし，確率分布を導出する．以上により，m 重マルコフ連鎖としての精密化利用モデルが完成する．

3.2 定義

以下は，標準／精密化利用構造としてのステートマシン，およびイベントシーケンスについての形式的定義である．

定義 1 標準利用構造としてのステートマシン SM

$$SM = \{S, i, F, E, T\}.$$

ここで，S は SM の状態集合，i は SM の開始疑似状態，F は SM の終了疑似状態 f の集合，E は SM のイベント集合，T は SM の遷移集合である．T の要素

は一意のリスト＜遷移元の状態,イベント,遷移先の状態＞として表され，$< x, y, z > \in T \rightarrow x \in \{S, i\} \wedge y \in E \wedge z \in \{S, F\} \wedge \neg[x = i \wedge z \in F]$ を満たす．

定義 2　SM に基づいて多重度 m で構築される，精密化利用構造としてのステートマシン SM_m

$$SM_m = \{S_m, i, F_m, E, T_m\}.$$

ここで，S_m は SM_m の状態集合であり，以下の式で表される．

$$\begin{aligned} S_m \;=\; & \{< t_1, t_2, \cdots, t_n > | \\ & \forall j[1 \le j \le n \rightarrow t_j \in T] \wedge \\ & [t_1[1] = i \rightarrow 1 \le n \le m-1] \wedge \\ & [t_1[1] \in S \rightarrow n = m-1] \wedge \\ & \forall k[[n \ge 2 \wedge 1 \le k \le n-1] \rightarrow \\ & t_k[3] = t_{k+1}[1]] \wedge \\ & t_n[3] \notin F\}. \end{aligned}$$

$t[x]$ はリスト t の x 番目の要素を表す．また，F_m は SM_m の終了疑似状態集合であり，以下の式で表される．

$$\begin{aligned} F_m \;=\; & \{< t_1, t_2, \cdots, t_n > | \\ & \forall j[1 \le j \le n \rightarrow t_j \in T] \wedge \\ & [t_1[1] = i \rightarrow 2 \le n \le m-1] \wedge \\ & [t_1[1] \in S \rightarrow n = m-1] \wedge \\ & \forall k[[n \ge 2 \wedge 1 \le k \le n-1] \rightarrow \\ & t_k[3] = t_{k+1}[1]] \wedge \\ & t_n[3] \in F\}. \end{aligned}$$

また，T_m は SM_m の遷移集合であり，以下の式で表される．

$$\begin{aligned} T_m \;=\; & \{< x, y, z > | \\ & x \in \{S_m, i\} \wedge z \in \{S_m, F_m\} \wedge \\ & [z_{first}[1] = i \rightarrow z = x \,\hat{}\, z_{last}] \wedge \\ & [z_{first}[1] \in S \rightarrow x_{first} \,\hat{}\, z = x \,\hat{}\, z_{last}] \wedge \\ & z_{last}[2] = y\}. \end{aligned}$$

x_{first}，x_{last} は，それぞれリスト x の最初の要素と最後の要素を表す．$x \,\hat{}\, z$ は x の終端に z を結合したリストを表す．

定義 3　SM，SM_m におけるイベントシーケンス ES

$$\begin{aligned} ES \;=\; & \{< t_1[2], t_2[2], \cdots, t_n[2] > | \\ & \forall j[1 \le j \le n \rightarrow t_j \in T] \wedge \\ & t_1[1] = i \wedge t_n[3] \in F \wedge \\ & \forall k[1 \le k \le n-1 \rightarrow t_k[3] = t_{k+1}[1]]\}. \end{aligned}$$

4. 適用例

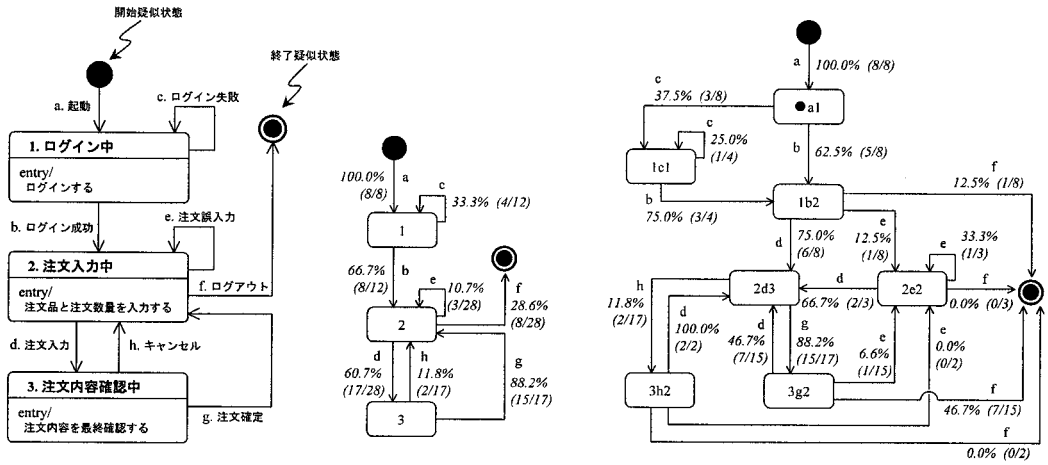
本節では，筆者らが独自に仕様を想定した「インターネット通販の注文受付システム」への適用例を示す．ここでは，注文受付システムは以下のような方法で操作するものと仮定する．

- ユーザは端末のソフトウェアを起動し，注文受付システムにログインを試みる．
- ログインに成功すれば，注文を入力できる．もし，ログインに失敗すれば，再度ログインを試みる．
- 注文の入力が完了すれば，その内容が自動的にチェックされる．もし誤入力が検出されれば，再度入力を試みることができる．また，用件が済めばログアウトする．
- 最終的にユーザが注文内容を再確認して問題がなければ確定する．もし問題があれば，キャンセルして再度注文入力を試みることができる．

以上を標準利用構造として定義したものが，図 3 (a) である．そして，利用モデルの作成に先立って，図 4 に示すイベントシーケンスが得られたと仮定する．

従来方法では，標準利用構造をイベントシーケンスでトレースするだけであったため，結果として得られる利用モデルは単純マルコフ連鎖となる．図 3 (b) は，図 3 (a) の標準利用構造と図 4 のイベントシーケンスから作成した単純マルコフ連鎖としての利用モデルである．状態名や遷移名はそれぞれ数字やアルファベットで略記している．一方，本手法を用いて 2 重マルコフ連鎖として構築した精密化利用モデルが図 3 (c) である．図 3 (c) の各状態は，現在状態の直前の状態遷移を 1 つ記憶している．たとえば，状態「1b2」は状態「1」においてイベント「b」が発生した結果，現在は状態「2」であることを表している．

図 3 (c) は，図 3 (b) と同じイベントシーケンス（図 4）から構築したにも関わらず，図 3 (b) より正確である．たとえば，図 3 (c) における等価状態「1b2」「2e2」「3g2」「3h2」，およびそれらの終了疑似状態への出力遷移を参照されたい．これらの等価状態から終了疑似状態への遷移は合計 8 回行われており，その内訳は「1b2」が 1 回,「3g2」が 7 回である．このことは，ユーザが注文確定直後にログアウトする傾向にあることを明らかにしている．これに対して，図 3 (b) には等価状態が存在せず，ログアウトを 1 本の遷移としてしか表していないので，状態「2」から終了疑似状態への 8 回の遷移を前述のように区別できない．したがって，前述の利用特性を明らかにできない．



(a) 標準利用構造　　(b) 単純マルコフ連鎖　　(c) 2重マルコフ連鎖

図 3: インターネット通販の注文受付システムの利用モデル

abdgf acbdhdgf abeedgedgf abdgdgdgf accbdgdhdgf abf abdgdgdgdgdgf acbdgf

図 4: インターネット通販の注文受付システムに対するイベントシーケンス

本研究では，多重マルコフ連鎖による精密化利用モデルを自動構築するツールを作成した．図 3 (a) と図 4 から多重度 1〜10 で構築した結果を，表 1 にまとめる．

5. 考察

本節では，利用モデルの正確さと構築コストの観点から考察を行う．そして，本手法に関して現場の技術者と議論を行ったので，その内容をまとめる．

5.1 利用モデルの正確さ

前節で例示したように，本手法は従来の利用モデルでは明らかにできなかった利用特性を抽出できることが分かった．本手法による利用モデルは，このような利用特性を統計的に反映したテストケースを生成する．したがって，統計的テスト法に本手法を導入することによって，従来よりも正確なソフトウェア信頼性の評価が可能となり，統計的テスト法の有効性を高めることが期待できる．たとえば，前節の例題において，状態「3g2」から終了疑似状態への状態遷移で欠陥が発生すると仮定する．単純マルコフ連鎖では，この状態遷移が現実の利用より低い頻度でテストされるので，この欠陥のソフトウェア信頼性への影響を過小評価してしまうが，多重マルコフ連鎖を用いればこの問題を解決できる．

ここで，マルコフ連鎖の多重度をどのように決定するかは重要な問題である．多重度が低すぎれば，有益な利用特性が明らかにならず，逆に多重度が高すぎれば，テストケースの多様性が損なわれる．個々のソフトウェアの状況に応じた適切な多重度をみつける必要があるが，そのための方法は 2 つ考えられる．1 つは，エントロピーや状態／遷移網羅率（表 1 参照）をはじめとした利用モデルの属性値が一定の条件を満足するか確認することである．他方は，利用特性に関して有意な依存関係のあるイベントを抽出し，そのようなイベントを始点，終点とする状態遷移シーケンスの長さに基づいて多重度を設定することである．後者については今後の検討課題とする．

5.2 利用モデルの構築コスト

本手法による利用モデルは，標準利用構造とイベントシーケンスから自動構築できることを示した．構築時間は，前節の適用例においては取るに足らない（表 1 では，最大でも約 10 分である）．しかしながら，マルコフ連鎖の多重度が大きくなると状態や遷移の数は急速に増加する．そのため，標準利用構造が大規模であったり，指定される多重度が大きい場合には，多くの構築時間を要する可能性がある．

なお，本手法はテストドライバの生成にも反映できる．いくつかの統計的テスト支援ツールは，利用モデ

表 1: 多重度 1〜10 で自動構築した精密化利用モデルの概要

多重度	構築時間 [1] (ミリ秒)	状態数	遷移数	サイクロマチック数 [2]	状態網羅率 [3]	遷移網羅率 [3]	エントロピー [4]
1	60	5	8	5	1.000	1.000	——
2	71	9	19	12	1.000	0.842	0.689
3	160	20	39	21	0.850	0.590	0.611
4	270	40	82	44	0.600	0.378	0.500
5	580	83	166	85	0.386	0.217	0.442
6	1132	167	337	172	0.222	0.113	0.356
7	3385	338	677	341	0.115	0.058	0.356
8	13579	678	1360	684	0.059	0.030	0.356
9	85613	1361	2724	1365	0.031	0.015	0.356
10	610187	2725	5455	2732	0.016	0.008	0.329

1) 実行環境は，OS : Microsoft Windows 2000, CPU : PentiumIII 700MHz, RAM : 320MByte である．
2) サイクロマチック数は，グラフ構造の複雑さを表すメトリクスである [2]．
3) 状態／遷移網羅率は，確率分布導出に用いたイベントシーケンス（図 4）が精密化利用構造をどれだけ網羅したかを表す．
4) エントロピーは，情報源としてのマルコフ連鎖の不確実さを表す尺度である [4]．

ルのアクションをプログラムコードとして定義することによって，テストドライバを生成する．このようなツールに本手法を組み込むことが可能である．等価状態やその出力遷移には標準利用構造で既定義のアクションが再利用可能であり，しかも再利用の自動化は単純になされる．たとえば，図 3 (c) の等価状態「1b2」のアクションは，図 3 (a) の状態「2」のアクションを再利用する．したがって，本手法によって利用モデルが巨大になろうとも人間とツールの負担はほとんど増加しない．

5.3 現場の技術者の意見

本手法に関して，品質管理部に所属するソフトウェア技術者と議論を行った結果，以下のような意見が得られた．

- 統計的テスト法は欠陥を網羅的に検出することには向かないかもしれないが，製品の信頼性を評価するには適している．本手法によってその有効性は増すであろう．
- 多重マルコフ連鎖による精密化利用モデルの状態／遷移網羅率（表 1 参照）は，過去の操作履歴に着目した一種の組合せテスト基準として有益であろう．
- 標準利用構造を短時間で正確に構築するためのガイドラインが必要である．特に，標準利用構造の粒度や階層化の方法に関して明確にしなければならない．

6. 関連研究

本節では，本研究と関連のある研究についてまとめる．

統計的テスト法によってソフトウェア信頼性を正確に評価するためには，正確な利用モデルを構築する必要がある．そのため，Whittaker ら [4] はフィールドデータから確率分布を導出するのが最善であると述べている．本研究はこの考えに基づき，フィールドデータとしてイベントシーケンスを用いる方法を採用した．正確な利用モデルを構築するための従来研究に関しては，2.2 節を参照されたい．

ソフトウェアには並行して動作するべき複数の利用モデルが存在することがある．この場合，個々の利用モデルに対して本手法を適用するか，それとも，すべての利用モデルを 1 つのプロダクトマシン（product machine）[3] に統合した上で本手法を適用するか，選択しなければならない．もし，利用特性が利用モデル間で独立でないなら，前者よりも後者の方が正確である．しかしながら，文献 [12] はプロダクトマシンが状態爆発を起こすことを問題視しているため，後者は利用モデルが巨大になりすぎる可能性が考えられる．

5.2 節において，本手法を従来の統計的テスト支援ツールに組み込むことが可能であると述べた．よく知られた統計的テスト支援ツールとしては，たとえば，Prowell [12, 13, 14] が作成している JUMBL（J Usage Model Builder Library）がある．JUMBL では，単純マルコフ連鎖としての利用モデルを定義するために TML（The Modeling Language）を用いる．確率分布の導出方法に関して複数のオプションが用意されており，アクションをプログラムコードとして記述しておくと最終的にテストドライバを生成することができる．さらに，TGL（Test Generation Language）を用いることによって，複数の利用モデルからのテストケース生成を総合的に管理できる．このツールの特長は，適用分野や開発環境に依存しない柔軟な利用方法を提供している点である．

7. おわりに

多重マルコフ連鎖を用いて正確な利用モデルを構築する手法を提案した．本手法によれば，標準利用構造とイベントシーケンスに基づいて任意の多重度の精密化利用モデルを自動生成することができる．標準利用構造は仕様の一部として定義されるステートマシン図のことであり，また，イベントシーケンスは利用環境下で得られるソフトウェアの実行履歴のことである．本稿では，そのような精密化利用モデルが，従来では明らかにできなかったユーザの利用特性を抽出すること，そしてツールによって自動的に構築できることを例示した．また，現場の技術者との議論においては，本手法の意義や課題について有益な意見を得た．

今後の研究では，マルコフ連鎖の最適な多重度を決定することを支援する分析手法を検討する．さらに，本手法の有効性を検証するために，実際のソフトウェア開発現場での適用実験を行う予定である．

謝辞

株式会社ジャストシステム 西町和也氏ならびに三橋尊志氏には，本研究に関して有益なご意見をいただきました．感謝いたします．

参考文献

[1] Myers, G. J.: *The Art of Software Testing*, John Wiley & Sons (1979).

[2] Beizer, B.: *Software Testing Techniques*, Van Nostrand Reinhold, 2nd edition (1990).

[3] Binder, R. V.: *Testing Object-Oriented Systems: Models, Patterns, and Tools*, Addison-Wesley (1999).

[4] Whittaker, J. A. and Poore, J. H.: Markov Analysis of Software Specifications, *ACM Transactions on Software Engineering and Methodology*, Vol. 2, No. 1, pp. 93–106 (1993).

[5] Whittaker, J. A. and Thomason, M. G.: A Markov Chain Model for Statistical Software Testing, *IEEE Transactions on Software Engineering*, Vol. 20, No. 10, pp. 812–824 (1994).

[6] Walton, G. H., Poore, J. H. and Trammell, C. J.: Statistical Testing of Software Based on a Usage Model, *Software Practice and Experience*, Vol. 25, No. 1, pp. 97–108 (1995).

[7] Shukla, R., Carrington, D. and Strooper, P.: Systematic Operational Profile Development for Software Components, *Proceedings of 11th Asia-Pacific Software Engineering Conference*, pp. 528–537 (2004).

[8] Kallepalli, C. and Tian, J.: Usage measurement for statistical web testing and reliability analysis, *Proceedings of 7th International Software Metrics Symposium*, London, UK, pp. 148–158 (2001).

[9] Takagi, T., Furukawa, Z. and Yamasaki, T.: Accurate Usage Model Construction Using High-Order Markov Chains, *Supplementary Proceedings of 17th International Symposium on Software Reliability Engineering* (2006).

[10] Object Management Group: *Unified Modeling Language*, http://www.uml.org/.

[11] Takagi, T. and Furukawa, Z.: Constructing a Usage Model for Statistical Testing with Source Code Generation Methods, *Proceedings of 11th Asia-Pacific Software Engineering Conference*, pp. 448–454 (2004).

[12] Prowell, S. J.: Using Markov Chain Usage Models to Test Complex Systems, *Proceedings of 38th Hawaii International Conference on System Sciences*, USA, p. 318c (2005).

[13] Prowell, S. J.: TML: a description language for Markov chain usage models, *Information and Software Technology*, Vol. 42, No. 12, pp. 835–844 (2000).

[14] Prowell, S. J.: JUMBL: a tool for model-based statistical testing, *Proceedings of 36th Hawaii International Conference on System Sciences*, USA, p. 337c (2003).