ソフトウェア工学 118−23
（1998. 3. 11）

並行動作の統計的テスト法に関する一考察

古川善吾 * 川野朋彦 ** 伊東栄典 †

* 九州大学 情報処理教育センター
** 九州大学工学部情報工学科
† 九州大学大型計算機センター

ソフトウェアの利用方法に基づいてソフトウェアそのものの信頼性を計測する統計的テスト法を並行動作に適用する方法について検討する。統計的テスト法では利用方法をマルコフ連鎖として記述し、テスト実施あるいは実際の利用を通してソフトウェアの信頼性を算出する技法である。一方、並列処理計算機や計算機ネットワークの普及によって分散・並行処理システムの実用化が進んでおり、並行動作の高信頼化が必要となってきている。そこで、並行動作に統計的テスト法を適用することによって、分散・並行処理システムの高信頼化を図ると同時に、テストに要する時間が増大するという統計的テスト法の問題点を明らかにする。また、実際の利用に際して利用状況を記録することによるソフトウェアの評価および品質向上について検討する。

Statistical Testing of Concurrent Behavior

Zengo Furukawa*, Tomohiko Kawano**, and Eisuke Itoh†

* Educational Center for Information Processing, Kyushu University,
** Dept. Computer Sicence and Communication Engineering, Kyushu University,
† Computer Center, Kyushu University.
6-10-1 Hakozaki, Fukuoka 812, Japan.

This paper discusses statistical testing on concurrent processes. Reliability and MTTF (Mean Time To Failure) of software may be computed with statistical testing based on usage of software. The statistical testing models the usage of software as a usage model with Markov chain. On the other hand, distributed and/or concurrent systems are widely used because of familiarity of parallel processors and computer network. The familiarity requires the high reliable system. It is possible to develop high reliable using the statistical testing. Other applications of statistical testing on concurrent processes are discussed.

1. はじめに

ソフトウェアのテストについてこれまで多くの研究が行なわれてきている[9]。これらの多くは、開発者の立場からのテスト技法である。すなわち、プログラムや仕様に基づいてテストを行ない、作業の品質によって対象となるソフトウェアの信頼性を予測している。ハードウェアのように1つ1つの製品の品質が異なる場合には、信頼性を統計的に求めることができ、そのための手法が確立し、利用されている。ソフトウェアについては、1つ1つの製品の違いがないために、信頼性の意味さえもきちんと定義されていなかった。

ソフトウェアのテスト法の中で統計的テスト法は、利用者の観点からソフトウェアをテストし、ソフトウェアそのものの信頼性を統計的に求める手法である[8], [11]。統計的テスト法は、ソフトウェアの開発においてプログラミング終了時点でのプログラムの誤りが少ないといわれているクリーンルーム開発手法におけるテスト法として開発された[7]。クリーンルーム手法では、信頼性をプログラムの設計時点から組み込んでいるので、テスト工程では信頼性を向上することよりも、信頼性を確認することが主な目的である。

この統計的テスト法を並行動作プログラムのテストに適用する方法について検討した[1]。近年の並列プロセッサやネットワークの普及によって並行処理プログラムの高信頼化が必要になってきている[2]。並行処理プログラムには、非決定性が存在するためにテストが困難である。プログラムや仕様書に基づいてテストケースを作成したり、実行時の被覆率を測定することは、可能である。しかしながら、正当な動作の定義が困難であり、テストにおいて少なくとも1 回実行すべき測定事象数が増大する、という問題点が残されている。逆に、事象数が多いことを利用して、実際の使用方法に対応した統計的テストの方が有利とも考えられる。統計的テスト方法では、プログラムそのものではなく、利用方法を対象としてモデルを構築する。そこで、必ずしもプログラムそのものが並行処理プログラムである必要はない。ここでは、利用方法が並行であることを前提としているので、並行動作を対象とした統計的テスト法について議論する。

本報告では、第2 章で統計的テスト法について概観し、第3 章で並行動作の統計的テストのために利用法の並行状態モデルを提案する。さらに、第4 章では、並行動作の統計的テスト支援システムについて議論する。第5 章では、並行動作の統計的テスト法として提案した利用法の並行状態グラフの応用について議論する。

2. 統計的テスト法

統計的テスト法は、プログラムに誤りが存在するとして、誤りに遭遇しない確率を信頼性として算出する。もともとテストによってプログラムの正当性を証明することは現実的には不可能である。そのために、誤りが発見されると直ちに修正するという対応が取られてきた。しかしながら、流通ソフトウェアが普及するに従って、誤りが発見されても直ちに修正することが困難になり、誤りを避けた使用法が必要とされている。そのために、ソフトウェアそのものの信頼性を算出できれば、誤りがあるという前提の下で利用することが考えられる。

Poore ら[8] や Whittacker ら[11] が提案している統計的テスト法の手順を以下に示す。

1. 利用モデルの作成
   利用者がソフトウェアをどのように使用するのかを、状態遷移図 (マルコフチェーン) で表現する[3]。これが、利用モデルである (図1参照)。

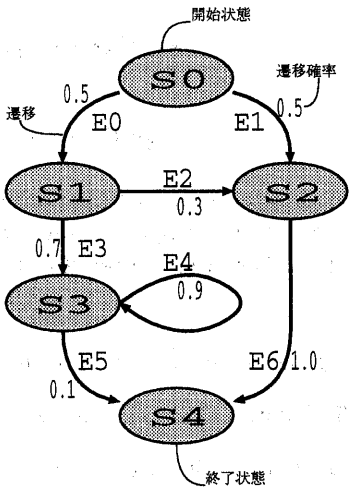

図 1: 利用モデルの例

2. テストケースの作成
   マルコフチェーンの開始節点から終了節点

に至る状態列で、各枝の遷移確率を満足する路をテストケースとして作成する。

3. テストの実施およびテストモデルの作成
テストケースを実行し、利用モデルの上に遷移回数を記入する。このようにしてできる状態遷移図がテストモデルである。テストモデルには、テストケースを実行する度に遷移回数が記入されるので、それ自身テストの履歴となる。

また、誤りが発生した時には、誤り状態を新たに追加し、その誤り状態への遷移をグラフに追加する。誤り状態からの遷移は、致命的な誤り (回復できない誤り) ならば終了状態への遷移を、また、軽微な誤り (回復できる誤り) ならば元の状態への遷移を、追加する。

4. 信頼性の計算
誤り状態に到達しない確率を求めることによって、信頼性を計算する。また、誤りが状態に到達するまでの実行回数の予測としてMTTFを求めることができる。

5. テスト終了の判定
以下のような指標に基づいてテストの十分性を判定する。

- 利用モデルとテストモデルの差が与えられた閾値以下。
- 信頼性が与えられた目標値より大きい。
- MTTFが与えられた目標値より大きい。

従来の被覆率を用いたテスト法では、利用モデルの状態あるいは遷移を少なくとも1回実行することを要求していた[4]。それに対して統計的テスト法では、利用モデルにおいて遷移確率の意味が有効になり信頼性を計算できる程度の回数遷移を実現する必要がある。そのために、従来のテスト法に比較してもテストに時間が掛かる可能性がある。並行動作の統計的テスト法と対比するために、この統計的テスト法を逐次動作の統計的テスト法と呼ぶことにする。

3. 並行動作の信頼性モデル

逐次動作の統計的テスト法では、プログラムの信頼性を誤り状態に到達しない確率として求めた。並行動作では、個々のプロセスについては逐次動作の統計的テスト法と同様にして信頼性を求めることができる。しかしながら、プロセス間での相互作用についての信頼性を定義しなければならない。そこで、並行動作の信頼性モデルを以下のように定義する。ただし、取り敢えず、並行動作を構成するプロセスは2つであると仮定する。

2つのプロセスからなる並行動作をするソフトウェアを考える。各プロセスのマルコフグラフがA、Bで与えられる場合、各々の動作の信頼性を$R(A)$, $R(B)$で表し、A、Bの並行動作における信頼性をRで表す。

AとBの間の依存関係により、M, M1, M2, M3というモデルを考える。それぞれのモデル化において信頼性Rを以下のように定義することができる。モデル化はMが基本である。AとBの相互作用による依存関係を$A*B$と表記すると、$A*B$の取り方として**M1**、**M2**、**M3**を考えることができる。

図2.に示す状態機械STMは、TCP/IPの状態遷移グラフの一部を取り出したものである。この状態機械の2つが通信して作業を進める並行動作を例にしてモデル**M1**、**M2**、**M3**を説明する。

M $R = R(A) \times R(B) \times R(A*B)$,
ただし、$R(A*B)$は、AとBの依存部分の信頼性である。

M1 $R = R(A) \times R(B)$.
AとBとが独立であるならば、AとBとが依存しないので、依存部分$A*B$の信頼性は1である。一方が故障を起こすと全体の動作そのものが失敗するので直列系のシステムの信頼性計算である。この場合、プロトコルの間での相互作用による信頼性の低下は考慮されない。

状態機械STMの場合には、信頼性は以下のようになる。
$R = R(STM)^2$

M2 $R = R(A) \times R(B) \times R(C)$,
ただし、CはAとBとの通信路であり、その信頼性が$R(C)$である。モデル**M1**においてさらに通信路Cが直列系として組み込まれたシステムである。

状態機械STMの場合には、信頼性は以下のようになる。
$R = R(STM)^2 \times R(IP)$
ただし、$R(IP)$は、ISOのOSI参照モデルで言うところのSTMの下位層のプロトコ

ルの信頼性である。この下位層の信頼性の計算も同様にして行い、最後の物理層では、回線そのもの信頼性を使用する。

M3 $R = R(A + B)$,
ただし、A+Bは、AとBの状態の組合せを並行状態とし、それぞれの動作の遷移に応じて並行状態が遷移する並行状態グラフである。これを"並行状態モデル"と呼ぶことにする。このモデル化は並行動作のモデル化としては望ましいものである。しかしながら、この並行状態は、状態数が増加することが知られているので、その対策を検討する必要がある。

状態機械STM の場合には、信頼性は以下のようになる。
[M3] $R = R(CSTM)$
ただし、CSTMは、プロトコルの状態の2つ組を並行状態とし、各プロトコルにおける遷移によって並行状態の遷移が起こる並行状態機械である。並行状態は、16個の対(S1,S1), (S1,S2), ..., (S4,S4)と仮想開始状態、仮想終了状態とからなる。

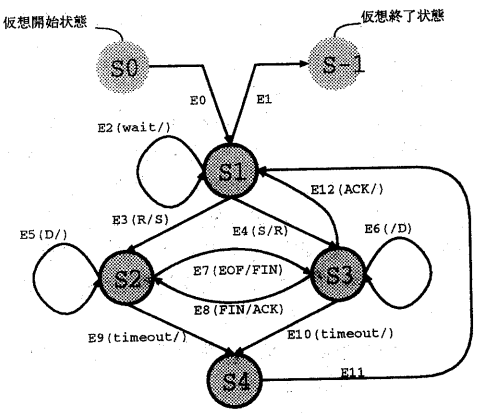

図 2: 状態機械の例題

並行動作の統計的テストのための信頼性モデルにおいて**M1**は、逐次的な利用モデルが複数存在するシステムとしてのモデル化であるので、ここでは、対象としない。また、モデル**M2**は、2つのプロセスの間の依存関係が通信路のように定義できる通信プログラムでは、有効なモデル化である。しかしながら、すべてのプログラムで依存関係を通信路として定義できる保証は存在しない。モデル**M3** (並行状態モデル) は、2つのプロセスについてそれぞれ定義できれば、適用できるモデル化である。そこで、ここでは並行状態モデルについて検討する。

並行状態モデルに基づく並行動作の統計的テスト法は、逐次動作の統計的テスト法において、利用モデルを作成した後に、並行状態グラフを作成することによって実現することができる。

並行処理プログラムにおいて並行状態グラフは、状態数や遷移数が個々のプロセスの状態数や遷移数の積で増大してしまうので実用的な方法とは、考えられない。しかしながら、利用モデルについては、利用者が興味を持った詳細さでモデルを構築すれば良いので、個々のプロセスの利用モデルは小さくなる可能性が高い。また、並行状態グラフを構築する際にも、並行動作に影響のない状態や遷移を削除することによって並行状態グラフの大規模化を制限できる可能性がある。ただし、状態数の増大は実用上問題になる可能性が高い。

4. 並行動作の統計的テスト支援システム

並行動作の統計的テスト法は、前節で述べたように、以下の手順で行なう。

1. 並行プロセス個々の利用モデルの作成
2. 並行状態グラフの作成
3. テストケースの作成
4. テストの実施とテストモデルの作成
5. 信頼性の算出

これらの作業を支援する並行動作の統計的テスト支援システム (混乱のない限り単に「支援システム」と呼ぶ) が必要である[5]。この手順の中で利用モデルの作成は、利用者自身が作成しなければならない。並行状態グラフの作成、テストケースの作成、信頼性の算出は、容易に行なうことができる。テストケースの作成は、利用モデルの上で仮想開始状態から始めて、各状態で乱数を発生させ遷移確率に応じて辿るべき遷移を決定する。このプログラムを並行状態グラフに適用することによってテストケースを作成できる。

テストの実施およびテストモデルの作成については、2つの方法が考えられる。

1. プログラム状態の記録
   並行処理プログラムにおいても、逐次処理プログラムの場合と同様に探針を挿入して

プログラムの実行状態を記録することができる[4]。しかしながら、この方法で記録できるのは、プログラムの実行状態であり、利用モデルにおける状態ではない。そのために、プログラムの状態と利用モデルの状態との間の対応をつける必要がある。

2. 利用者による操作の記録
利用モデルは、ソフトウェアの利用方法をモデル化したものである。操作画面の上での利用者の操作を記録することができれば、その操作と利用モデルとの対応をつけることによってテストモデルを自動的に作成することができる。ただし、誤りが発生した時の誤り状態の付加および誤り状態からの遷移の追加は、人手で行なう必要がある。このような操作の記録ツールとしては、X Window System の記録プロトコル[10] および Windows95 上の SPY++ プログラム[6] がある。これらは、利用者の画面上でのカーソルの動きやプログラムとのメッセージのやり取りを記録することができる。

5. 並行状態モデルの応用

3.節では、並行状態モデルを並行動作の統計的テストのための利用モデルとして考えた。ここで、ソフトウェアの利用モデルと人の操作のモデルを考えその並行状態モデルを考えることができる。例えば、図3.に示す操作モデルのように計算機画面上での操作をモデル化したとする。すなわち、操作待状態においては、マウスによるカーソルの移動あるいはマウスによるドラッグ開始、マウスによるクリック、キー入力を行なう。

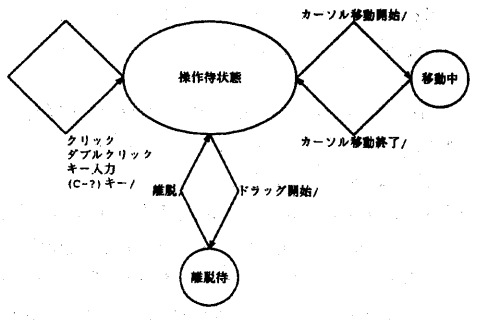

図 3: 利用者の操作モデル

ソフトウェアの利用モデルが記述されているとすると、プログラム実行中に利用者が行なうことのできるのは、図3. に示す操作である。計算機と利用者とは並行に動作しているので、利用モデルの各状態で操作モデルの操作を行なうことができるし、逆に操作モデルの各状態で計算機は、利用モデルの任意の状態を取ることができる。そのために、利用モデルと操作モデルとの並行状態グラフを作成できる。これに基づいて、4.節で述べた利用者による操作の記録を利用してテストモデルを作成できる。さらに、並行状態グラフの遷移は、個々のプロセスの遷移によって引き起こされるので、ソフトウェアの利用者モデルおよび利用者の操作モデルそれぞれにおいてテストモデルを構築することができる。

このようにして作成した並行状態グラフに対応するテストモデル (並行状態テストモデル)、ソフトウエアの利用モデルに対応したテストモデル (ソフトウェアテストモデル)、利用者の操作モデルに対応したテストモデル (利用者テストモデル) については、以下のことが考えられる。ただし、テストケースを利用モデルから作成して実施したテストとしてではなく、実際の利用そのものについて利用者による操作を記録して、テストモデルを作成する。

1. 並行状態テストモデル
並行状態テストモデルからは、利用者とプログラムとの両方に依存したソフトウェアの信頼性を求めることができる。この場合、利用者がいろいろ代わればソフトウェア自身の信頼性と考えることができるし、利用者が特定の個人であれば、その個人に依存した信頼性を求めることができる。これによって、開発時のテストによって予測した信頼性を検証できる。

2. ソフトウェアテストモデル
ソフトウェアテストモデルは、実際の利用者によるプログラムの利用状況を記録したものである。ソフトウェアの利用モデルの構築時に遷移確率を設定しなければならないが、このソフトウェアテストモデルは、次回の統計的テストの遷移確率として利用できる。

ソフトウェアテストモデルを用いてソフトウェアの操作性そのものを評価することができる可能性がある。すなわち、利用者が操作の中断や有効な作業を行なわずに終

了するのは、利用者が途中で気を変えた可能性はあるけれども、ソフトウェアそのものの操作性が悪いと考えることができる。

3. 利用者テストモデル
利用者テストモデルは、利用者自身の操作の傾向を表したものである。ソフトウェアテストモデルによるソフトウェアの操作性の評価とは逆に、利用者自身の誤りの確率(キー入力のやり直し、マウスクリックのやり直し) あるいは操作の信頼性を求めることができる。また、単に誤りの可能性だけでなく、操作の中断の傾向を知ることができる。それらが分かれば、訓練あるいは勉強すべき項目が明らかになる。

4. ソフトウェアの利用者への適応
ソフトウェアテストモデルや利用者テストモデルは、評価だけでなく、ソフトウェアの改善にも用いることができる。例えば、操作性の悪いソフトウェアの改善すべき操作を具体的に指摘することができる。

また、特定の利用者の操作の傾向に合わせたソフトウェアの条件の設定を考えることができる。例えば、日本語変換システムで学習機能を備えたシステムでは、直前に変換した漢字を候補として表示するものが多いが、利用者の変換履歴に基づいて候補を表示することも考えられる。また、繰り返し行なわれる操作の短縮形あるいはメニューを作成しておくことによって、特定の個人に適応したソフトウェアとすることができる。

このように、テストとしてではなく、実際の利用時にも記録を取ることによってソフトウェアおよび利用者の評価と使い易さの向上のために利用できる可能性がある。この場合、統計的テスト法そのものや並行状態グラフの状態数の増大は、必ずしも問題にはならない。これは、ハードウェアの統計的品質評価や統計的テストが空間的に品質を調べるので量が問題になるのに対して、利用状況の長時間に渡る記録に基づく時間的な品質の評価を行なっているためである。

6. おわりに

統計的テスト法を並行動作に適用する方法について検討した。並行動作の統計的テストにおいて信頼性を算出するモデルとして、個々のプロセスの状態を組み合わせて構築した並行状態グラフに基づいた並行状態モデルを提案し、その実施を支援するシステムについて検討した。さらに、並行状態モデルに基づいて実際の利用時に利用状況を記録することによる可能性について論じた。ソフトウェアの操作性や利用者自身の信頼性などこれまで定量化が困難であった指標の定量化の可能性がある。また、ソフトウェアの使い易さの向上についても可能性がある。今後は、これらの点についてさらに、検討する予定である。

参考文献

[1] 古川善吾, 川野朋彦, 伊東栄典, 牛島和夫: "並行動作の統計的テスト法における信頼性算出方法の提案", 電気関係学会九州支部第 50 回記念連合大会講演論文集, pp.209, 1997.

[2] 古川善吾, 伊東栄典, 片山徹郎: "並行処理プログラムの試験", 情報処理学会学会誌 Vol.39, No.1, pp.7-12, 1998.

[3] 市川昌弘: "信頼性工学", 機械工学選書, 裳華房, 1990.

[4] Eisuke Itoh, Zengo Furukawa and Kazuo Ushijima: "A prototype of a concurrent behavior monitoring tool for testing of concurrent programs", Proc. of APSEC'96, pp.345-354, 1996.

[5] 川野朋彦: "並行動作における統計的テスト法実現方式の研究," 九州大学工学部卒業論文, 1998.

[6] "Microsoft Spy++," Microsoft Co., 1997.

[7] 西崎幹俊 : ソフトウェア開発クリーンルーム手法, 電子情報通信学会誌, Vol. 80, No.5, pp.470-478, 1997.

[8] Poore, J. H., H. D. Mills and D.Mutchler. "Planning and Certifying Software System Reliability." IEEE Software, January 1993, pp 88-99.

[9] Ian Sommerville 著 / 佐野美知夫, 佐藤匡正 監訳: "ソフトウェアエンジニアリング", フジ・テクノシステム, 1993.

[10] Stephen Gildea: "Record Extension Protocol Specification," X Consotium, 1995.

[11] Whittaker, J. A. and M. G. Thomason. "A Markov Chain Model for Statistical Software Testing." IEEE Transactions on Software Engineering, October 1994, pp. 812-824.