ソフトウェア工学　136－11
（２００２．３．７）

UML 状態図を用いたテストケース作成支援システムの試作

高木　智彦 / T. Takagi　　古川　善吾 / Z. Furukawa
s98t240@stmail.eng.kagawa-u.ac.jp　　zengo@eng.kagawa-u.ac.jp

(香川大学)

概要

ソフトウェアの信頼性や開発時の生産性が低下する原因として，テスト工程と仕様化工程が十分に連携していないことが挙げられる．そこで本報告では，仕様化工程で作成される UML 状態図上のパス (初期状態から終了状態に至る状態の列) を用いて統計的テストのためのテストケースを作成する方法を確立する．すなわち，ユーザの使用状況を遷移確率付き状態図 (利用モデル) として記述する．そして利用モデル上の遷移確率を満たすパスをテストケースとして作成する．また，本手法に基づいてテストケースを作成するシステムを試作した．利用モデルの作成は，統計的テストのためのテストケース作成だけでなく，開発計画の判断材料としての可能性がある．

Test Case Generation System for Functional Testing using UML State Diagram

Tomohiko Takagi　　Zengo Furukawa
s98t240@stmail.eng.kagawa-u.ac.jp　　zengo@eng.kagawa-u.ac.jp

(Kagawa University)

abstract

Weak relationships between specifications and testing causes lower reliability of software and lower productivity in software development. This paper proposes a testcase generation method for statistical testing using a UML state diagram, which specifies usage of software and is added probability of transition by the users as usage for it. The testcases are sequences of states in the diagram from a beginning one to an end one according to probability of transition from a state. The prototype system of testcases generation is constructed.

1 はじめに

ソフトウェアの検証やテストでは様々な課題があり，ソフトウェアの信頼性や開発時の生産性が低下する原因になっている [1][2]．特に，テスト工程と仕様化工程間の効果的な連携方法に対する検討が十分ではない．

本研究では，仕様化工程で作成される UML(Unified Modeling Language) の状態図 [3] 上のパス (初期状態から終了状態に至る状態の列) を用いて統計的テスト [1][4] のためのテストケースを作成する方法を確立する．

統計的テストでは，ユーザの使用特性を統計的に分析した結果に基づいてテストを行う．すなわち，ユーザは，状態図の検証を行う過程において使用状況に応じたパスセット (実行系列) を入力する．この実行系列を用いて状態図上の遷移確率を求め，状態図から統計的テストのための利用モデルを作成する．その利用モデルから自動的にテストのためのパスセット (テストケース) を必要数作成する．

UML を利用することによって，ユーザや開発者が共通の理解のもとに仕様を検証できる．また，仕様の記述方法として状態図を用いる理由は次のとおりである．

- 直感的に分かりやすい．
- 動的特性を記述できる．
- 階層的に表現できる．

また，本手法を用いてテストケースを作成することには，次の特徴がある．

- 一般的に利用されている状態図を仕様記述に用いることによりユーザが仕様の確認に参加することが容易になる．
- 実行系列やテストケースをユーザや開発者が確認することにより仕様の誤りを削減できる．
- テストケースがユーザの利用方法に則しているために使用頻度の高いシステムの機能を重点的にテストできる．

本報告では，まず 2節で統計的テストについて概観する．そして 3節でテストケースの作成方法を，4節で試作したテストケース作成支援システムの機能や利用方法を説明する．最後に 6章で本手法および試作システムに対する考察を行う．本報告の中では，UML の用語を断りなく用いることがある．詳しくは，参考文献 [3] を参照されたい．

2 統計的テスト

2.1 概要

統計的テストは，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 [1][4]．クリーンルーム開発手法によって作成されたコードはプログラミング終了時点での品質が高いため，バグを発見するためではなく，品質を測定するためにテストを行う．ただし本研究では，クリーンルーム開発手法によって作成されたコードのみを対象にしているのではない．むしろ，従来の開発手法によって作成されたコードへの適用を想定している．

ソフトウェアの品質とは，ユーザにとっての外部的性質である．例えば，ソースコード 1000 行当たりのバグの数は，ユーザの視点からは全く意味がない．つまり，コード中にバグを含んでいても，1 回も実行されることがないのであれば問題はない．逆に，ユーザがよく使う機能を記述しているコードにはバグを含んではならない．

そこで統計的テストでは，ユーザの使用特性を外部仕様の観点から統計的に分析し，テストケースに反映させる．使用頻度の高い機能にテストの重心を置くことによって，ユーザがよく使う機能に含まれるバグを重点的に検出することが期待できる．

2.2 従来の手法との比較

統計的テストは，ユーザの立場から行い，ソフトウェアそのものの信頼性を求める [1][4]．テスト十分性は MTTF (Mean Time To Failure) の測定結果による．MTTF とは，ある欠陥が発生してから，次の欠陥が発生するまでの平均時間である．出荷基準に達したかどうかは，MTTF がある一定以上になったかどうかで判断する．統計的テストでは，原理的に出荷した後の使用環境において，障害発生確率の高いものが欠陥として起こる．ゆえに，効果的なテストを行うために，利用モデルがユーザの使用特性を反映していることを要求する．

一方，従来のテストは，開発者の立場から行う．すなわち，プログラムや仕様に基づいてテストを行い，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．この手法は，状態図上のすべてのパスを少なくとも 1 回実行することを要求する．しかし，状態図に繰り返し処理を含む場合はパスの数が無限に存在するため，すべてのパスを実行できない．そこで，実際にはすべてのパスをテストする代わりにいくつかの簡易化された被覆基準を用いる．被覆基準の種類としては，C_0 (全状態被覆) や C_1 (全遷移被覆) などがある．C_0 は全状態を 1 回以上実行するパスセットを選ぶ基準であり，C_1 は全遷移を 1 回以上実行するパスセットを選ぶ基準である．テスト十分性は被覆基準を満たしたかどうかで判断する．

2.3 テスト手順

これまでに提案されている統計的テストの一般的な手順を以下に示す [1][4]．

1. 利用モデルの作成

   (1) ソフトウェアの仕様として作成される状態図，および (2) ユーザのソフトウェアに対する使用特性を用いてマルコフモデルを作成する．マルコフモデルとは，状態図において遷移先が

確率的に決定されるものをいう．これが利用モデルである．利用モデルにおいては，状態図の状態をノード，遷移をエッジと呼ぶことにする．利用モデルでは，ソフトウェアに対する，ユーザ，利用の仕方，環境などを定義できる．利用の仕方は，ユーザの動作によって定義される．環境は，ソフトウェアが動作するプラットフォームや外部データベースなどが考慮すべき要因である．利用の状況についても，夜間と昼間，緊急時と平常時などに分類する．

2. テストケースの作成

   利用モデルから，各エッジの遷移確率に従い，初期状態から終了状態に至る状態の列としてパスを作成する．作成したパスは，ソフトウェアに対する入力条件であり，かつ期待出力でもあるため，テストケースと見なすことができる．

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，得られた出力を状態図に記入する．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状態図に追加する．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．これがテストモデルであり，テストの履歴として利用する．

4. 信頼性評価

   誤り状態に達しない確率を求めることによって，信頼性を評価する．また，誤り状態に達するまでの実行回数から， MTTF を求めることができる．

5. テスト終了判定

   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．

テスト中にバグが発見された場合は，発見毎にプログラムの修正は行わずに，テストを一通り終わらせることを優先する．修正後において，どのレベルのテストからやり直すかは場合による．その際，データフロー解析に代表されるプログラム・スライシング技術によって影響範囲を抽出することが多い．

3 テストケースの作成方法

本研究において考察した，統計的テストに基づくテストケースの作成方法を以下に示す．

1. 状態図の作成

   仕様化工程において，開発者は要求分析の結果に基づき，仕様として UML 状態図を作成する．状態図は外部仕様上のもので，ユーザが理解できる範囲において詳細化を行う．

2. 状態図の検証

   仕様化工程のレビューにおいて，ユーザは状態図上で使用状況に応じたパス (実行系列) を入力し，動的特性を検証する．その際，入力する実行系列は，多いほどより精密な利用モデルを作成することができる．客観的に十分性を判断する基準はないが，未走破のノードやエッジがなくなり，後の利用モデル作成において統計的な処理を行うのに十分な量であることが望ましい．

3. 利用モデルの作成

   開発者は，検証後の状態図とユーザの入力したパスから利用モデルとしてマルコフモデルを作成する．遷移確率が 0% のエッジが存在する場合は，実際のシステムでも利用時には通らないことが検証できない場合には，低い確率を配分する．利用モデルは後の開発計画の参考にすることができる．具体的には，使用頻度の高い機能に対するプロトタイプの作成や信頼性に重点を置いた開発管理手法の部分的導入などが考えられる．

4. テストケースの作成

   テスト工程において，利用モデルから機能テストのためのテストケースを必要数作成する．一般に，利用モデルにおいて遷移確率がエッジの遷移元のノードにだけ依存することは少ない．そのような状況を少しでも反映させるために，遷移元のノードより前のノードにも依存する条件付確率とすれば，よりユーザの使用特性に近いテストケースが得られる可能性がある．また必要に応じて被覆基準を満たすテストケースも作成する．

4 テストケース作成支援システム

4.1 概要

統計的テストでは，利用モデルを作成したり，信頼性の評価に十分足りる量のテストケースを作成したりする必要があるため，従来のテスト方法より時間がかかるという欠点がある．そこで本研究では，一連の作業を自動的に行うために，テストケース作成支援システムを試作した．システムは主に， (1) GUI を用いて状態図を編集する Graph Editor， (2) 利用モデルによりテストケースを作成する Markov Modeler， (3) 各種被覆基準によりテストケースを作成する Path Analyzer から構成される． Microsoft Windows 上で動作する．開発環境は， Borland C++ Builder 3 Standard を使用した．

次に，システムの各機能 (図 1. 参照) や利用方法について述べる．

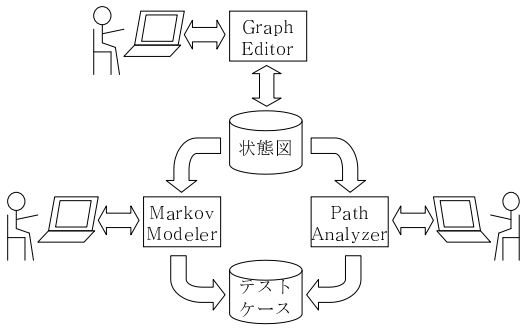

図 1: システム構成

4.2 Graph Editor

Graph Editor (図 2. 参照) は，システムのメイン・ウインドウであり，実行開始時に表示される． GUI を用いて状態図を編集する機能をはじめ， 4 つの主な機能から構成されている．

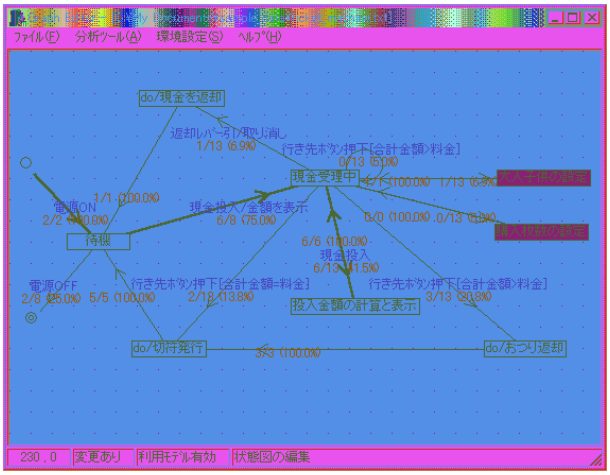

図 2: Graph Editor (利用モデルの例)

- 状態図の編集

  状態図の編集は，すべてポインティングデバイスを使って行う．メイン・ウィンドウの表示画面であるワークシート上または状態図の中のノード上で右クリックしてポップアップメニュー中の項目を選択する．ノードやエッジの作成と削除，名前の変更，コメントの挿入，サブ状態図の編集などが可能である．ノードには，以下に示す種類がある．ヒストリ指示子はサブ状態の履歴を保持するだけの特殊なノードで，自身がエッジや制御点をもたない．

  状態 ： サブ状態図の作成が可能である．

  初期状態： はじめに制御点をもつ．

  終了状態： 制御点を初期状態に戻す．

  ヒストリ指示子： 制御点の位置を記憶する．

- 利用モデルの表示

  Markov Modeler(4.3節で説明) で作成した利用モデルを視覚的に表示する．すべてのエッジの通過回数と遷移確率，および終了状態が存在しない場合は各ノードの仮終了状態 (4.3節参照) となる確率を状態図上に表す．遷移確率が 0% のエッジは色を反転表示することで，作成した利用モデルにおける問題点をユーザに示す．また，初期状態から終了状態に至るパスの中で，遷移確率が最も大きいパスを強調表示する． Markov Modeler を用いて入力された実行系列によるエッジの被覆の状況や重要なパス (ユーザが最もよく使う機能列) を把握できる．

- ファイルの読み込みと保存

  状態図や利用モデルのデータを読み込んだり，保存したりする．また，必要に応じてバックアップファイルを作成する．

- 分析ツールの呼び出し

  テストケースを作成するために， Markov Modeler や Path Analyzer を呼び出す．

4.3 Markov Modeler

Markov Modeler (図 3. 参照) は，統計的テストのためのテストケースとして状態図からパスセットを自動作成する． (1) 実行系列入力， (2) 利用モデル作成， (3) テストケース作成の 3 つの機能 (分析フェーズ) から成る．

- 実行系列入力

  Graph Editor で編集した状態図上のノードをクリックして実行系列 (ソフトウェアに対するユーザの使用特性) を入力する．終了状態が存在しない場合は，ヒストリ指示子または初期状態をクリックすることによって実行系列の終端を指定する (仮終了状態)．入力した実行系列は， Graph Editor でファイルの保存を行うと，状態図のデータと共に保存される．

- 利用モデル作成

  入力した実行系列から，利用モデルとしてマルコフモデルを作成する．遷移確率 0% エッジ群への配分確率値 (0% 以上 10% 以下) を指定できる．各エッジの遷移確率や各ノードの仮終了状態となる確率が算出され， Graph Editor の状態図上に表示される．なお， 1 回も通過しないノードを起点とするエッジの遷移確率についてはすべてを等確率とする．

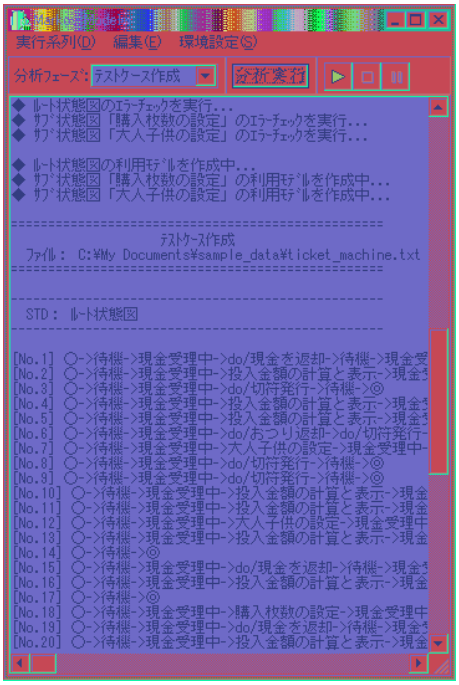

図 3: Markov Modeler 実行画面

もし，利用状況が十分に把握されていないときには，実行系列入力を行わずに利用モデルを作成することによって遷移確率が等しいエッジを持った利用モデルが作成できる．新規作成したシステムのテストでは，この方法によって，とりあえず初期のテストを行うことなどが考えられる．

- テストケース作成

  利用モデルに従い，各エッジの遷移確率を満たすパスをテストケースとして自動作成する．作成するテストケースの個数を指定できる．

なお，テストケースは以下のアルゴリズムに従って作成する．

- 親状態図で通過したノードのサブ状態を分析する．
- 初期状態と終了状態が存在する場合は，初期状態から終了状態に至るまで遷移を行う．
- 初期状態が存在し終了状態が存在しない場合は，初期状態から仮終了状態に至るまで遷移を行う．
- ヒストリ指示子が存在する場合は，あらかじめ設定した仮初期状態から，仮終了状態に至るまで遷移を行う．以降は前回の仮終了状態が仮初期状態となる．

仮終了状態については，(1) 実行系列の入力に基づいて各ノードの仮終了状態となる確率を求める方法と，(2) 全ノードに対して仮想終了状態となる確率を等しく設定する方法がある．

4.4 Path Analyzer

Path Analyzer (図 4. 参照) は，指定された始点と終点および被覆基準に基づく状態図上のパスとして，テストケースを自動作成する．従来のテストにおいてよく用いられる手法である．

分析可能な被覆基準は C_0 と C_1 である．C_0 は全ノードを 1 回以上実行するパスセットを選ぶ基準である．終点を「(任意のノード)」に設定すると，始点から深さ優先探索 [5] を行った結果を返す．また，終点を特定のノード名に設定すると，始点から深さ優先探索を行った結果に対し，終点までの最短経路をダイクストラ法 [5] によって算出した解を返す．サブ状態図については，親状態図で通過したものだけが分析対象となる．

一方，C_1 は全エッジを 1 回以上実行するパスセットを選ぶ基準である．状態図を正方行列として内積を求め，巡回路 (ループ) と順路を算出する [6].

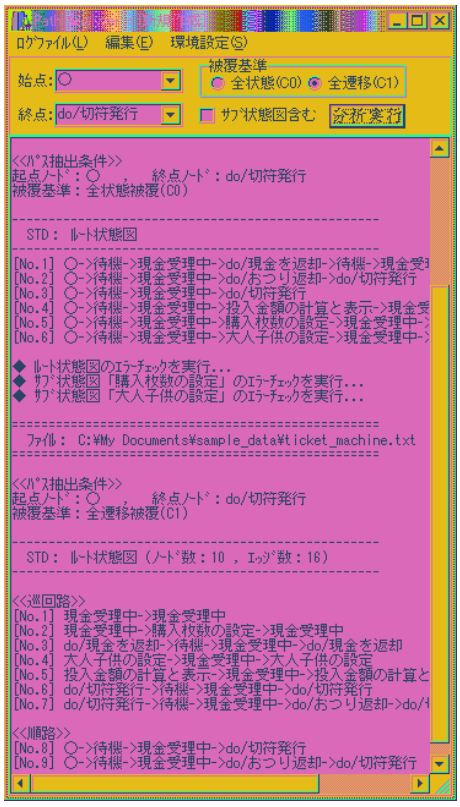

図 4: Path Analyzer 実行画面

4.5 その他の機能

4.5.1 Tree Viewer

状態図の編集，理解支援の機能として Tree Viewer (図 5. 参照) がある． Tree Viewer は，状態図間の階層的な関係をツリーを用いて表す．主な機能は，(1) Graph Editor で編集中の状態図名と全状態図における位置を示す，(2) 指定したツリー上のノードに対応する状態図へ直接アクセスする．

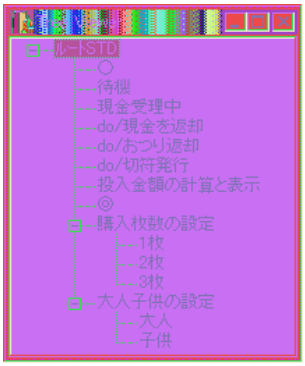

図 5: Tree Viewer 実行画面

4.5.2 実行系列のリプレイ

ユーザが入力した実行系列またはツールが自動作成したテストケースを，状態図上でノードの色の変化によって実行順序を表示するトレースを行う．リプレイは仕様の検証やレビューを行う際に役立つ．

4.5.3 状態図のエラーチェック

状態図の誤りはテストケース作成を行う上で障害となるため，作業に先立ってエラーチェックを行う．エラーが検出されるとメッセージを出力してユーザに問題点を示す．ユーザは，出力されたメッセージを手がかりにして， UML として正しい状態図を導くことができる．以下に状態図の要件やエラーチェックの方法をまとめる．

まず，階層構造中で最上位の状態図においては，次の 2 点を満たす必要がある．問題がある場合，作業を中止しエラーメッセージを出力する．

- 初期状態と終了状態が存在する
- すべてのノードが終了状態に遷移可能である

サブ状態図においては，次の 3 点を満たす必要がある．問題がある場合，作業を中止しエラーメッセージを出力する．

- 初期状態またはヒストリ指示子が存在する
- 終了状態が存在する場合は，すべてのノードが終了状態に遷移可能である
- 終了状態が存在しない場合は，すべてのノードが初期状態を除くすべてのノードに遷移可能である

また，初期状態と終了状態が存在する場合は次の点をチェックし，問題がある場合は警告メッセージを出力する．

- 初期状態がすべてのノードに遷移可能である
- 遷移確率が 0% のエッジが存在しない
- 終了状態が存在しない場合は，仮終了状態となる確率が 0% のノードが存在しない

さらに，利用モデルの作成過程に問題があると遷移確率が 0% のエッジを作ってしまい，テストケース作成時の障害となる可能性がある．そこで，利用モデル作成時においては次の 2 点をチェックする．

前者について問題がある場合の対応は，状態図の構造により異なる．すなわち，初期状態から終了状態へ遷移可能なパスが少なくとも 1 本存在すれば，警告メッセージの出力のみで，作業は続行できる．しかしそれ以外においては，初期状態 (仮初期状態) から終了状態 (仮終了状態) への遷移が不可能な場合があるため，作業を中止しエラーメッセージを出力する．

後者についてはテストケースの被覆の問題であるため，警告メッセージの出力のみで，作業は続行できる．

5 利用例

以下，試作システムの利用例として，切符の自動販売機 [3] を取り上げることにする．

まず， Graph Editor によってモデル化を行った (図 6. 図 7. 図 8. 参照)．

次に， Markov Modeler を用いて実行系列の入力を行った．

- ルート状態図において入力した実行系列は次の通りである．

  [No.1] ○ → 待機 → 現金受理中 → 投入金額の計算と表示 → 現金受理中 → 投入金額の計算と表示 → 現金受理中 → do/ おつり返却 → do/ 切符発行 → 待機 → 現金受理中 → do/ 現金を返却 → 待機 → 現金受理中 → 大人子供の設定 → 現金受理中 → do/ おつり返却 → do/ 切符発行 → 待機 → 現金受理中 → 投入金額の計算と表示 → 現金受理中 → 投入金額の計算と表示 → 現金受理中 → 投入金額の計算と表示 → 現金受理中 → do/ 切符発行 → 待機 → ◎

  [No.2] ○ → 待機 → 現金受理中 → do/ おつり返却 → do/ 切符発行 → 待機 → 現金受理中 → 投入金額の計算と表示 → 現金受理中 → do/ 切符発行 → 待機 → ◎

- サブ状態図「購入枚数の設定」においては実行系列を入力していない．
- サブ状態図「大人子供の設定」において入力した実行系列は次の通りである．

  [No.1] 大人

  [No.2] 大人

  [No.3] 大人 → 子供

実行系列の入力が終了した後，利用モデルの作成を行った (図 9. 図 10. 図 11. 参照)．遷移確率 0% のエッジが作成されないようにするため，「遷移確率 0% エッジ群への配分確率値」を 10% に設定した．

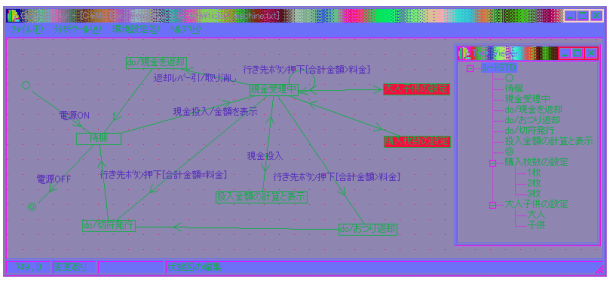

図 6: ルート状態図 (切符の自動販売機)

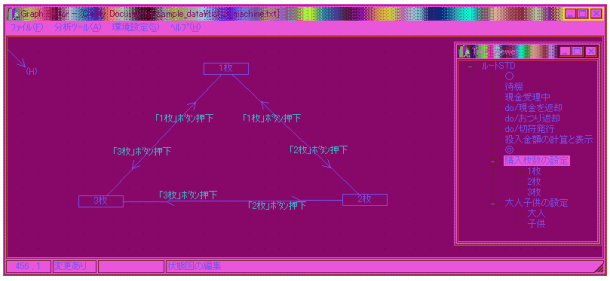

図 7: サブ状態図 (購入枚数の決定)

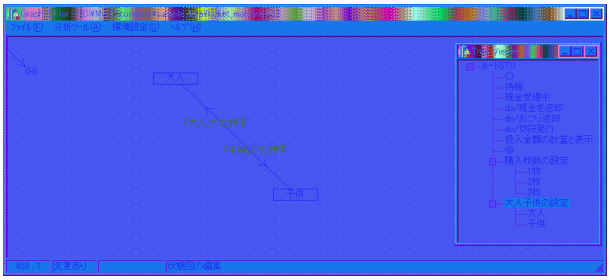

図 8: サブ状態図 (大人子供の設定)

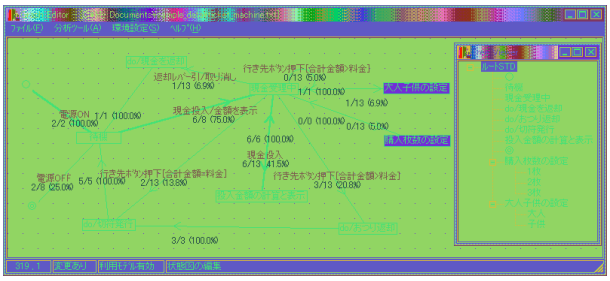

図 9: ルート状態図 (切符の自動販売機) の利用モデル

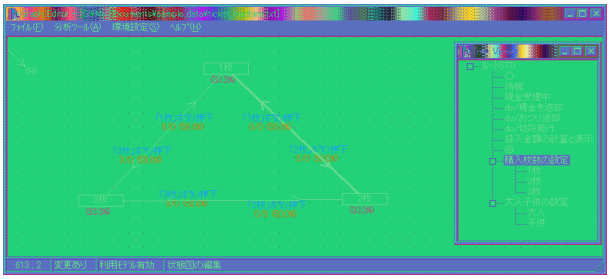

図 10: サブ状態図 (購入枚数の決定) の利用モデル

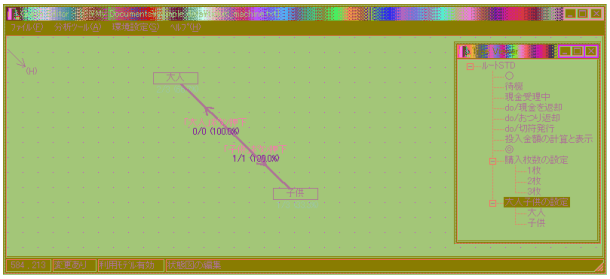

図 11: サブ状態図 (大人子供の設定) の利用モデル

最後にテストケースの作成を行った．図 9. に基づいて作成したテストケースの一部を以下に掲載する．

[No.1] ○ → 待機 → 現金受理中 → 投入金額の計算と表示 → 現金受理中 → 投入金額の計算と表示 → 現金受理中 → 投入金額の計算と表示 → 現金受理中 → 大人子供の設定 → 現金受理中 → do/ 現金を返却 → 待機 → 現金受理中 → 投入金額の計算と表示 → . . . → 購入枚数の設定 → 現金受理中 → 大人子供の設定 → 現金受理中 → 投入金額の計算と表示 → 現金受理中 → do/ 切符発行 → 待機 → 現金受理中 → do/ おつり返却 → do/ 切符発行 → 待機 → 現金受理中 → do/ おつり返却 → do/ 切符発行 → 待機 → ◎

[No.2] ○ → 待機 → 現金受理中 → 投入金額の計算と表示 → 現金受理中 → 現金受理中 → 投入金額の計算と表示 → 現金受理中 → . . .

[No.3] ○ → 待機 → ◎

[No.4] ○ → 待機 → 現金受理中 → do/ おつり返却 → do/ 切符発行 → 待機 → ◎

[No.5] ○ → 待機 → ◎

6 考察

6.1 統計的テストの課題

統計的テストに基づくテストケース作成の効果については 1節で述べた．一方，検討すべき課題としては時間がかかる点がある．これについては，以下のように考えることができる．

- ツールによって作業を完全に自動化する

  人間がソフトウェアのテストに長時間拘束されることがなくなるので，時間がかかっても構わない．

- 品質を保証するためには不可欠である

  ソフトウェアの品質を保証するためにはテストに長い時間をかける必要がある．これまで，被覆基準を用いた簡便な方法によって比較的短時間でテストを終わらせていたことに問題があった．

- テスト対象の粒度が違う

  従来の開発者の立場からのテストでは，内部仕様まで対象に含むことが多かったが，統計的テストでは，外部仕様のみを対象としている．よってテスト対象の粒度が大きいという意味では，統計的テストの方が作業量が少ないと考えることもできる．

6.2 試作システムの効果と課題

試作システムの効果として，以下のことが考えられる．

- 利用モデルの作成が容易になる．
- 統計的テストのためのテストケースが自動的に作成できる．

一方，検討するべき課題として以下のことがある．

- 表記法

  現在，UML 状態図の表記法のすべてに対応できているわけではない．特に，並行サブ状態については今後検討する．

  また，ユーザは表記法に関する知識を習得する必要がある．実用を考える上では，なるべく予備知識を必要としない方がよい．表記法がユーザにとって分かりやすければ，ユーザの使用特性をより反映した利用モデルをより簡単に得られる可能性がある．

- ツールによる自動化

  ツールによって一連の作業を自動化する必要がある一方，ユーザの判断に依存せざるを得ない部分がある．例えば，テストケースの作成過程において，遷移確率 0% エッジ群への配分確率値やテストケースの個数，条件付確率の算出におけるノードの記憶数などをユーザが決定する必要がある．これらは，判断の根拠が明確にできない．

- 他のツールとの連携

  現在，試作システムは，UML ツール (Rational ROSE, IIOSS, etc.) と連携できていない．今後は，オープンソースソフトウェアである IIOSS との連携について検討する．

- 統計的テスト支援

  ソフトウェアのテストを行う場合，実際にテストデータを入力してプログラムを実行し，結果を確認する作業は，退屈で時間が掛かる作業である．特に，統計的テスト法においては，多くのテストケースを実行しなければならないので，結果確認のためのツールが是非とも必要である．利用モデル上のテストケースを実行系列として作成しているので，プログラムの実行結果と状態系列との対応関係を自動的に確認するためのツールが不可欠である．

  さらに，テスト終了条件を判定するために，ソフトウェアの信頼性を MTTF として算出できる必要がある．

7 おわりに

UML 状態図で記述された機能仕様に基づいて統計的テストのためのテストケースを作成するテストケース作成システムを作成した．これによって，利用モデルの作成が容易になり，統計的テストのためのテストケースが自動的に作成できるので，統計的テストの品質および効率向上が期待できる．

現在，テストケース作成システムは，UML 状態図のすべての機能には対応していない．また，実際のテストにおけるテストケースの使用方法や，テスト履歴の記録および確認を行う手法についても検討している段階である．そのため，実際のシステム開発における有効性は十分に確認できていない．

しかしながら，利用モデルの作成は，統計的テストのためのテストケース作成だけでなく，開発計画を決定する際の判断材料としても活用できる可能性がある．今後は，テストの実施やテスト十分性評価までも視野に入れて，手法の検討とシステムの試作を進める．

参考文献

[1] 佐藤武久，大槻繁，金藤栄孝，ソフトウェアクリーンルーム手法，日科技連，1997.

[2] 中所武司，情報科学こんせぷつ 7 ソフトウェア工学，朝倉書店，1997.

[3] 磯田定宏，コンピュータ数学シリーズ 22　オブジェクト指向モデリング，コロナ社，1998.

[4] 古川善吾，川野朋彦，伊東栄典，並行動作の統計的テスト法に関する一考察，情報処理学会研究会報告, Vol.98, No.20, pp.173-178, 1998.

[5] 河西朝雄： C 言語によるはじめてのアルゴリズム入門，技術評論社，1992.

[6] 尾崎弘，白川功，グラフとネットワークの理論，コロナ社，1973