B-048

ソースコード生成法を用いた利用モデルの作成
Constructing Usage Model using Source Code Generation Method

高木 智彦 †
Tomohiko Takagi

古川 善吾 †
Zengo Furukawa

1. はじめに

近年，ソフトウェアは日常生活のあらゆる場面に浸透したため，バグが社会に及ぼす影響は大きいものとなっている．しかしながら，従来のように，ソフトウェアのすべての機能を網羅的にテストするだけでは，要求される信頼性を十分に満足できない場合がある．

統計的テストは，ソフトウェアの実際の利用状況における信頼性を評価したり，信頼性に影響を与えやすいバグを検出するための効果的な手法である．この方法では，(1) ソフトウェアの仕様である状態遷移図にユーザの利用状況を関連付けて利用モデル（マルコフ連鎖）を作成し，(2) 利用モデルの確率分布に基づいてテストケース（状態遷移図上のパス）を非決定的に生成する．ただし，従来の研究において，利用モデルの具体的な作成方法は十分に論じられていない．例えば，文献 [1] では，利用モデルの確率分布を (a) 利用現場のデータに基づいて設定する，(b) 予想される利用法に基づいて設定する，(c) 一様に割り当てる，という 3 通りの方法を列挙するにとどまっている．

効果的な統計的テストを行うためには，現実の利用状況を利用モデルに反映させる必要がある．そのためには，実際のソフトウェアから直接，運用データ（ユーザの操作履歴やプログラムの各種ログなど）を取得することが望ましい．そこで本研究では，ソースコード生成法を応用することによって，自動的に利用モデルのための運用データを作成する方法を検討した．

2. 利用モデルの作成

2.1 従来方法とその問題

運用データを用いることを前提とした場合における，主要な成果物の関連を図 1 に示す．本稿の議論の対象は，ステップ 1 からステップ 4 までである：

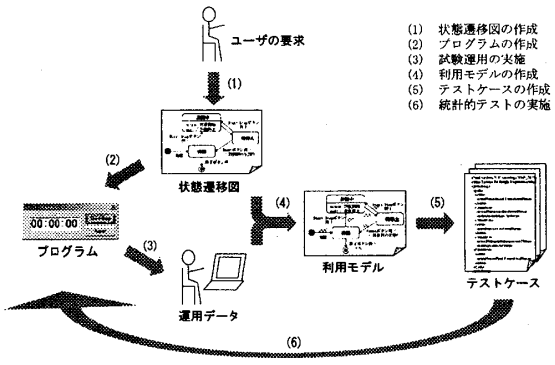


図 1: 成果物の関連

1. 状態遷移図の作成

   ソフトウェアの仕様である状態遷移図を記述する．

2. プログラムの作成

   仕様に基づいて，コーディングや基本的なテストを行う．さらに，運用データを記録するための仕組みを導入する．

3. 試験運用の実施

   運用データを取得するために，試験運用を実施する．これは，ソフトウェアがユーザの要求と合致することの確認を兼ねている．

4. 利用モデルの作成

   運用データに基づいてソフトウェアの状態遷移を分析し，状態遷移図上の確率分布を計算する．

一般的に，運用データに基づいた利用モデルの作成には困難を伴うことが多い．例えば，運用データをアプリケーションと OS の間のメッセージシーケンスとして取得する場合を想定する．メッセージは，ロギングツールによって自動的に記録される．しかし，この種のツールは，ソフトウェアの状態遷移を分析するというより，デバッグを行うことを目的として設計されている．膨大なログ中のメッセージを状態遷移図に対応付けることは，複雑な作業である．

これを回避する方法は，(メッセージではなく) 状態遷移図中のアーク，すなわち遷移を構成単位とする運用データを記録することである．しかしそのためには，ソースコードを解析して遷移に対応するステートメントを正確に抽出する必要があり [2]，多大なコストを必要とする．

2.2 ソースコード生成法を用いる方法

本研究では，前節で示した問題を解決するためにソースコード生成法に着目した．ここでのソースコード生成とは，状態遷移図からスケルトンコードを自動生成することをいう．スケルトンコードは状態遷移図と等価な状態遷移モデルとして実現され，ソフトウェアの一部として機能する．組込みシステムの分野では，短期開発・品質改善を目的として従来から広く利用されており，状態遷移表に基づく手法 [3] や OMT (Object Modeling Technique) に基づく手法 [4] などが知られている．

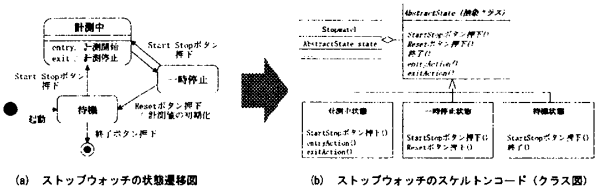
我々は，スケルトンコード生成時に探針を組み込むことにした．探針は，遷移シーケンスまたは各遷移の実行回数を運用データとして記録する機能をもつ（利用モデルの確率分布を求めるためには，少なくとも，すべての遷移の実行回数を明らかにする必要がある）．スケルトンコードは遷移に対応するプログラム要素を必ず含んでおり，スケルトンコードジェネレータによって自動的に探針を挿入できる．これによって，ソフトウェアは「状

†香川大学大学院工学研究科

態遷移図に直接マッピング可能な形式の」運用データをバックグラウンドで記録できる．

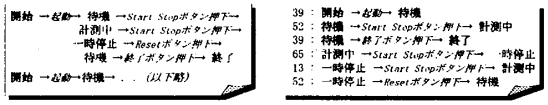
図 2 は，ストップウォッチの動作を記述した状態遷移図から，OMT に基づく手法によってスケルトンコードを生成した例である．状態遷移図中の状態はスケルトンコードのクラスに，また遷移はクラスのメソッドに直接対応していることは明らかである．したがって，探針を各メソッドに組み込むことによって図 3 に例示する形式の運用データを直接得ることができる．

運用データは (a) 遷移シーケンス，(b) 各遷移の実行回数，の 2 通りの形式で記録できることは先に述べた．(a) は，時間経過によるユーザの利用特性の変化 [1] を解析できる反面，運用データのサイズが巨大になる可能性がある．一方，(b) であれば，運用データのサイズを抑えることができる反面，遷移シーケンスとして再現できないので，利用特性の変化を解析できない．



(a) ストップウォッチの状態遷移図　(b) ストップウォッチのスケルトンコード（クラス図）

図 2: OMT に基づくソースコード生成（例）



(a) 遷移シーケンス　(b) 各遷移の実行回数

図 3: 運用データ（例）

2.3 既存のツールによる運用データ取得

運用データの記録を，(探針ではなく) ロギングツールやコードカバレッジツールに任せることも可能である．

ロギングツールは，普通，記録対象を選択するためのインタフェースを提供している．もし，設定ファイルを用いて記録対象，すなわち遷移に対応するプログラム要素を直接指定できるのであれば，スケルトンコード生成時に設定ファイルも同時に生成するとよい．この設定ファイルによって，ロギングツールは，利用モデルのために記録する情報を厳密に選択できるようになるため，運用データ（遷移シーケンス）を取得できる．この方法はスケルトンプログラムに探針を記述する必要がないので，ソースコードの可読性を損なわない．

コードカバレッジツールを用いる場合は，スケルトンコードに標識を挿入する．標識とは，図 4 に例示する形式のプログラムのコメントのことであり，スケルトンコード生成時において，遷移に対応する基本ブロックに自動挿入される．これによって，基本ブロックの累積実行回数を遷移にマッピングすることが可能となり，運用データ（各遷移の実行回数）が取得される．この方法はカバレッジ情報も同時に取得できるが，オーバヘッドが大きくなる点に注意が必要である．

```
/* event:イベント名 from:遷移元の状態名 */
```

図 4: 標識の記述形式（例）

3. 考察

本手法の特徴は以下のとおりである．

- コストの削減

  運用データの取得作業は，ソフトウェアがユーザの要求と合致することの確認も兼ねることができるため，従来の開発工程に組み込むことにコスト効果的なデメリットは少ない．また，本手法は自動化を促進するので，その分，コスト削減が期待できる．

- テスト範囲の限定

  ソースコード生成法によって，仕様を正確に反映した状態遷移のロジックが得られる．

- 動作速度の低下

  運用データを記録するための仕組みを導入することによって，ソフトウェアの動作速度が低下する．許容できない応答の遅延やシステム障害などを引き起こす場合は，本手法を適用できない．

4. おわりに

運用データから利用モデルの確率分布を自動的に作成するために，ソースコード生成法を応用する方法を提案した．スケルトンコードにおいては，遷移に対応するステートメントは明示的であるため，遷移を構成要素とする運用データを容易に記録できる．本稿では，ソースコード生成と運用データのマッピング対象を状態遷移図に限定したが，原理的には他のモデル（例えば UML 活動図）にも適用できると考えられる．

提案した内容は，試作中の支援システムにほぼ実装されている．今後の研究では，本システムを用いて実際のソフトウェア開発における有効性を評価する予定である．

参考文献

[1] G. H. Walton, J. H. Poore, and C. J. Trammell, “Statistical Testing of Software Based on a Usage Model,” Software Practice and Experience, Vol.25, No.1, pp.97-108, January 1995.

[2] 高木智彦, 古川善吾, “プログラムの実行履歴を用いた利用モデル作成の一手法,” 情報科学技術フォーラム 一般講演論文集 第 1 分冊, pp.209-210, 2003.

[3] Boris Beizer, “ソフトウェアテスト技法,” 日経 BP 出版センター, 1994.

[4] アリ ジョハル, 田中二郎, “オブジェクト指向方法論 (OMT) に基づく動的モデルからの Java コード生成,” 情報処理学会論文誌, Vol.39, No.11, pp.3084-3096, 1998.