B-34 並行プログラムにおけるアクティビティ図駆動型テストケースの自動生成手法

Automatic Construction of Testcases driven by Activity-diagram in Concurrent Program

小川 浩司† 鈴木 寿郎† 五味 弘†
Hiroshi Ogawa Suzuki Hisao Hiroshi Gomi

1．はじめに

ソフトウェア開発におけるテスト工程は，ソフトウェア品質を確保するための重要な工程である．しかしながら，並行プログラムの動作は非決定性を持つため，逐次動作のシステムと比べてテストが困難であり，またテストケースの網羅性に問題が生じやすい．

一方，ソフトウェアの開発において，UML は分析や設計に広く使用されている．特にアクティビティ図はシステムの並行動作の記述に適しているため，並行プログラムのモデリングに多く用いられている．

そこで本稿では，並行動作をより明示的に記述するために UML のアクティビティ図を拡張し，この図からテストケース・テンプレートを自動生成する方法を示す．

2．アクティビティ図の拡張

2.1 タスク

タスクをアクティビティ図の一部分を実行する主体と定義する．この定義から，アクティビティ図は1個または複数のタスクによって逐次または並行実行されるものとなる．アクティビティ図に実行するタスク名及び最大同時実行タスク数を記述することにより，このタスクを表す（図 1）．Ti (i = α,β,γ) はタスク名であり，括弧内の数字が最大同時タスク数を表す(タスク数が 1 の場合は省略する)．Actn(n=1, 2, ...)はアクティビティを表す．

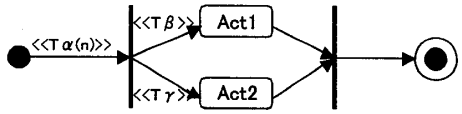

図 1．タスク概念の導入

2.2 モデル要素

アクティビティ図において，並行動作を表すモデル要素はフォークとジョインのみであり，明示的に並行動作を記述しない．そこで並行プログラムに現れる意味論を明示的に示し，テストケース作成の補助的な入力として用いられるように拡張した．

(1) クリティカル・セクション
あるアクティビティが同時に複数のタスクから実行されないことを表す（図 2 (1)）．

(2) セマフォ
2つ以上のアクティビティが同時に実行されないことを表す．排他制御が必要なアクティビティに番号を付与することにより識別する（図 2(2)）．

(3) 例外アクティビティ
例外的なケースに対するテストは重要であるため，例外発生により実行されるアクティビティを例外アクティビティと定義する．但し，図が煩雑になるため，例外発生元アクティビティにステレオタイプを付与することにより，例外アクティビティへ遷移する可能性があることを表す（図 2(3)）．

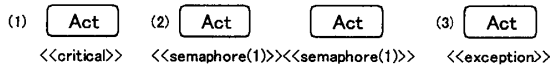

図 2．モデル要素

新たに定義したタスクとモデル要素は UML の表記法に準拠したステレオタイプで表す．これにより一般的な UML ツールで記述できる．

2.3 アクティビティ図の合成

アクティビティ図はシステムの動的な一側面を表すため，1個のユースケースに対してアクティビティ図は複数個存在する．これらのアクティビティ図間には(1) 並行関係，(2) 逐次関係と(3) 包含関係が存在する．

並行関係にある図間は，フォーク及びジョインにより合成し，逐次関係にある図は一方の図の終了状態からもう一方の図の初期状態への遷移で合成する．包含関係にある図間は複合アクティビティを導入することにより合成を行う．アクティビティ図を合成することにより，同時に動作するタスクやアクティビティが明確になるという利点がある．以下に例を示す．ここで合成後のアクティビティ図を合成アクティビティ図と定義する．

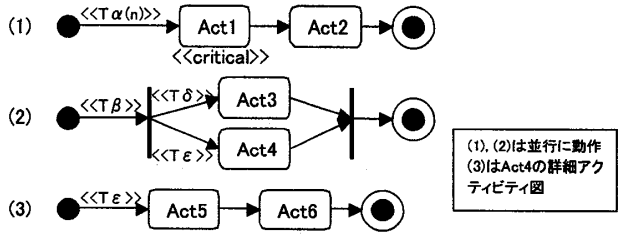

図 3．合成前アクティビティ図

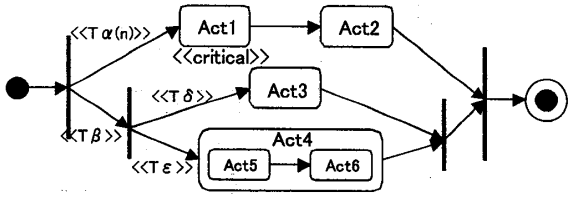

図 4．合成後アクティビティ図

†沖ソフトウェア株式会社 OKI Software Co.,Ltd.

3．テストケース生成

ここでは合成アクティビティ図からテストケース・テンプレートの生成について述べる．

3.1 用語の定義

以下の用語を定義する．

【テスト対象アクティビティ(T,t)】

合成アクティビティ図における，タスク T とアクティビティ t の任意の組(T,t)とする

【同時実行アクティビティ集合 ST(T,t)】

テスト対象アクティビティ(T,t)実行時において，他のタスクが同時に実行する可能性のあるアクティビティの組の集合とする．

$ST(T, t) = \{ PA(Ti, act(Ti, T, t)) \mid Ti \in T(T,t) \}$

$PA(T,t) = \{ (job(Ai), Ai) \mid Ai \in par(T,t) \}$

但し，par(T,t)はテスト対象アクティビティ(T,t) を実行中に同時に実行される可能性のあるアクティビティの集合であり（自分自身も含む），job(Ai) はアクティビティ Ai を実行するタスクである．また T(T,t) は テスト対象アクティビティ(T,t) の実行時において，同時実行される別のタスクの集合とする．また act(Ti,T,t) はテスト対象アクティビティ(T,t) の実行時における，タスク Ti の実行中のアクティビティを表す．

3.2 テストケーステンプレート作成アルゴリズム

Step1. 合成アクティビティ図から，テスト対象アクティビティ(T,t)をひとつ取り出す．

Step2. テスト対象アクティビティ(T,t)実行時に並行動作する可能性のあるタスクの集合 T(T,t)を取り出す．最大同時実行タスク数が２以上ならば別のタスクとしてすべて取り出す．

Step3. T(T,t)のそれぞれのタスクに対し，実行する可能性のあるアクティビティ集合 PA(T,t)を取り出す．

Step4. PA(T,t)の組み合わせをとり，同時実行アクティビティ集合 ST(T,t)を取り出す．

Step5. テストケースのテンプレート Temp(T,t)を以下のように定義し生成する．

Temp(T,t) = {s において，T が t を実行するときに正しい結果が返る | s∈ST(T,t)}

Step6. テストケーステンプレート Temp(T,t) の未定項目である「結果」を代入することにより，テストケースを作成する．

Step7. 残りのテスト対象アクティビティがあれば Step1 に戻る．

3.3 テストケース生成例

図 5 に食事をする哲学者問題のアクティビティ図を示す．

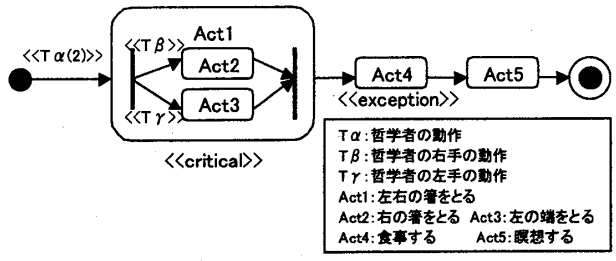

図 5．食事をする哲学者問題のアクティビティ図

尚，哲学者の数は二人とし，またデッドロック発生を避けるため，左右の箸を取る操作を複合アクティビティ「左右の箸をとる」で包含し，そのアクティビティをクリティカル・セクションとした．このアクティビティ図に対するテストケース生成結果を表 1示す．表中の下線付きアクティビティは例外アクティビティを表し，$T_{\underline{\alpha}}$は $T\alpha(2)$におけるタスク $T\alpha$のもう一方のタスクを表す．

表 1．同時実行アクティビティ集合の生成結果

テスト対象アクティビティ(T,t)	同時実行アクティビティ集合 ST(T,t)	個数
(Tα, Act1)	{{($T_{\underline{\alpha}}$, Act 1)}, {($T_{\underline{\alpha}}$, Act 4)}, {($T_{\underline{\alpha}}$, Act 4)}, {($T_{\underline{\alpha}}$, Act 5)}}	4
(Tβ, Act 2)	{{(Tγ, Act 3)}}	1
(Tγ, Act 3)	{{(Tβ, Act 2)}}	1
(Tα, Act 4)	{{($T_{\underline{\alpha}}$, Act 1)}, {($T_{\underline{\alpha}}$, Act 4)}, {($T_{\underline{\alpha}}$, Act 4)}, {($T_{\underline{\alpha}}$, Act 5)}}	4
(Tα, Act 5)	{{($T_{\underline{\alpha}}$, Act 1)}, {($T_{\underline{\alpha}}$, Act 4)}, {($T_{\underline{\alpha}}$, Act 4)}, {($T_{\underline{\alpha}}$, Act 5)}}	4

表 2．テストケース・テンプレートの例

No	テストケース・テンプレート	テスト対象アクティビティ(T,t)とST(T,t)の要素
1	哲学者$\underline{\alpha}$が左右の箸を取る時点において，哲学者αが箸を取ることに対して正しい結果が返る．	(Tα, Act1) {($T_{\underline{\alpha}}$, Act 1)}
2	哲学者$\underline{\alpha}$が左右の箸を取る時点において，哲学者αが食事することに対して正しい結果が返る．	(Tα, Act4) {($T_{\underline{\alpha}}$, Act 1)}

3.4 テストケースの絞込み

生成されるテストケースはタスク数やアクティビティ数の増加に対して組合せ的に増加する．従って実際にテストを行うためには，テストケースの絞込みが必要になる．この絞込みのために 2.2 で定義したモデル要素を指標として用いる．これらのモデル要素は一般的にエラーが発生しやすい個所であるため，これらを含むテストケースに注目することにより，エラーの発見に有効なテストを作成できる．

4．おわりに

本稿では，並行動作のシステムの動作を記述するために，アクティビティを拡張し，それを元にテストケーステンプレートを自動生成する方法について述べ，その例として食事をする哲学者問題を挙げた．

しかしながら，タスク数やアクティビティ数が多い場合に，テストケースの個数が爆発する問題については今後の課題である．

5．参考文献

(1) 片山徹郎，他: 並行処理プログラムにおけるテストケースの定義と生成ツールの試作，情報処理学会論文誌，vol.34, No.11(1993).

(2) グラディ・ブーチ: UML ユーザーガイド，ピアソン・エデュケーション(2001)

(3) Geri Schneider, etc: ユースケースの適用：実践ガイド，ピアソン・エデュケーション(2000).