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

2008－ＳＥ－160 (12)
2008－EMB－ 9 (12)
2008／6／20

複数シーケンス図を使ったシナリオ検証とタスク設計

藤倉 俊幸

イーソル株式会社リサーチ＆コンサルテーションサービス部

アブストラクト

複数のシーケンス図によって表現された動作仕様からモデル検査用のモデルを生成し，シナリオ検証とタスク設計に利用する手法を提案する．非同期メッセージの扱いを導入したことにより従来手法では検出できなかった状態不整合を検出することができる．

Using a set of Sequence Diagrams for Implied Scenario Detection and Task Design Verification

Toshiyuki FUJIKURA

Research & Consultation Services Group, eSOL Co., Ltd.

Abstract

A model for model checking is generated from a set of sequence diagrams and a state diagram. The model is able to detect implied scenarios of requirements and synchronization failures of a multitask system which were not detected by a previous method. Our method treats asynchronous messages strictly, so that the model describes an embedded software behavior correctly.

1. はじめに

シーケンス図はユースケースシナリオの記述やソフトウェアの動作の記述，テストケースの記述など，要求定義段階からテスト工程までソフトウェア開発において幅広く利用されている．シーケンス図あるいは同等なメッセージシーケンスチャート(MSC)を利用したシナリオ検証技術が多数報告されている([1][2]およびこれらのRef.)．これらの技術を要求工程で利用すればシナリオ検証を，設計工程で利用すればタスク設計等の動的構造検証を，おこなえる．しかし，実際に適用してみると検査対象に対する制約が必ずしもソフトウェア実装方法と整合性が取れているとは言えず，利用しづらい面がある．組み込みソフトの要求分析では，外界のオブジェクトが並列で動くか否かは重要であり，このことは環境をモデリングする際のコミュニケーションが同期か非同期かを区別することの重要さにつながる．設計以降では，非同期メッセージや同期メッセージ，関数呼び出しを組み合わせて動的構造を構築するが，この組合せの自由が制限される検証手法では利用に限界がある．また，RTOSを使用することも考慮されていないため要求段階から設計段階への移行の際に障害となることもある．

本研究では，同期・非同期の区別を早い段階から導入しシナリオ検証およびタスク設計に利用することを検討した．その結果，従来の手法では検出できないシナリオの不足や設計上の問題点を検出できるようになった．また，成果物として生成される動作モデルはRTOSスケジューラモデル[3]と合成することで，マルチコア

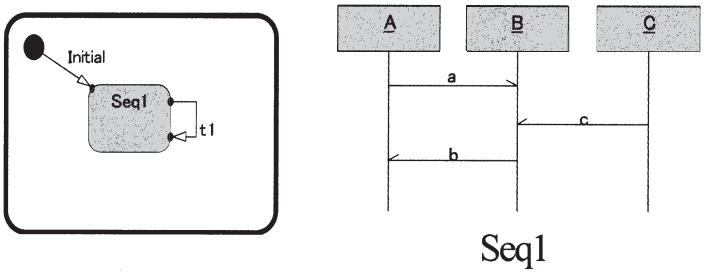

図 1. シーケンス図と繰り返し構造の例

環境も含めたプラットフォーム上でのタスク設計[4]に応用可能である.

UMLや状態遷移表からSPIN等のモデルを生成し, モデル検査導入の敷居を下げ利用工数を減らすための研究が多数報告されている([5]等). 本研究で開発した手法はUML図を入力として検査用モデルを自動生成する. 複数のシナリオをそれぞれシーケンス図で記述し, モデル検査用のモデルを生成することができる. プログラムは, Enterprise Architect[6] や Rational Rose RealTime[7]等の UMLツールのアドインとして実装した.

シーケンス図は各オブジェクトの動作順序を記述するために利用されるが, 二次元的な図であるため表現しづらい事象や, 実際には順序を規定できない事象などがあり, これらが仕様の曖昧さにつながる. このため, シーケンス図から生成されるモデルは後から開発者が追加・変更しやすいことが必要である. 本研究では, 動作順を追加・削除しやすい LTSA[8]をツールとして使用した. LTSA を使用することで, シーケンス図に含まれる動作パターンを直接目で確認できる. この段階で, 仕様の追加および意図しない動作を排除するなどの変更を生成されたモデルに対しておこなうことができる.

組込みシステムの動作仕様を単一のシーケンス図で表現できることはほとんど無く, 複数のシーケンス図とその間の関係として記述される. 本研究では, シーケンス図間の関係を状態図またはアクティビティ図で記述し関連するシーケンス図と共にモデル化をおこなう. この場合, モデルサイズが大きくなるため, 検出された問題点の効率的対応方法の開発が今後の課題である. 特にシナリオ検証において発見された隠れたシナリオ(implied scenario)を仕様として採用する場合の手順を確立したい. また, 状態爆発については LTSA モデルを FDR2[9]等のより大きな状態空間を扱えるツール用のモデルに変換して対応することを検討している.

2. 従来手法との比較

LTSAを使用して複数のシーケンス図から不足するシナリオを検出する手法が Uchitel 等によって提案されている[2].

図 1に比較のためのサンプルを示す. 左側の状態図は, Seq1 で示されたシーケンス図を繰り返し実行することを示している. 相互作用概要図をサポートしない UML ツールでは, 状態図またはアクティビティ図で, HMC に対応する情報を表現した. 図 1においてオブジェクトCに同期指定が無いためメッセージ c を送信し続ける可能性が有る. Uchitel 等の手法で検証をおこなうとこの点を見落としてしまう. この原因は, Uchitel 等の手法ではメッセージを同期メッセージとしているためである. 生成される動作モデルは以下のようなものである.

ArchitectureModel = Q0,
Q0 = (a.b.a1 → Q1),
Q1 = (c.b.c1 → Q2),
Q2 = (b.a.b1 →Q0).

各オブジェクトが並列に動くことを前提とする場合には, たとえばメッセージaに対応する LTSA アクションは a.b.a1 ではなく, 送信アクション a.a1 と受信アクション b.a1 に分けて扱う必要が

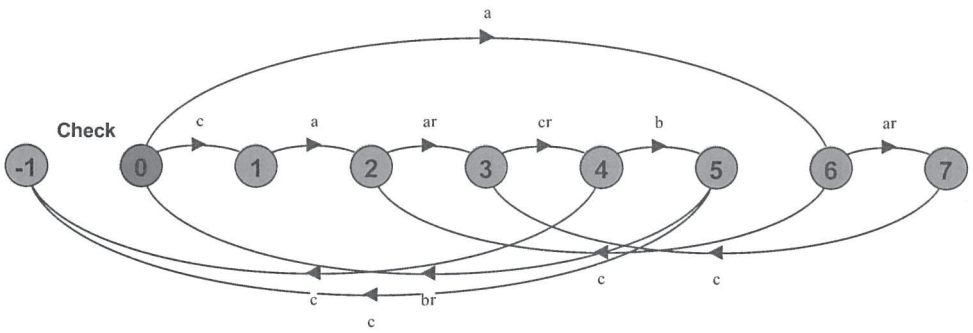

図 2. C が先行した場合の反例

ある.

我々の手法では, 送信と受信を別イベントとして扱い以下のような動作モデルを生成する.

```
ArchModel1 = Q0,
Q0      = (c → Q1
           |a → Q8),
Q1      = (a → Q2),
Q2      = (ar → Q3),
Q3      = (cr → Q4),
Q4      = (c → Q5
           |b → Q7),
Q5      = (b → Q6),
Q6      = (br → Q1),
Q7      = (br → Q0
           |c → Q6),
Q8      = (c → Q2
           |ar → Q9),
Q9      = (c → Q3).
```

このモデルでは a が送信アクション, ar が受信アクションである. そして, この動作モデルをシーケンス図単位で繰り返した場合の動きのモデルと比較することで図 2に示す反例を検出することができる. この反例は, 要求分析段階であればオブジェクト C の連続送信と言う隠れたシナリオの存在を示し, 設計段階であれば C に対する同期構造の不備を示す.

3. 単一シーケンス図の扱い

本研究の手法では図 1の Seq1 に対して以下の LTSA モデルを生成する.

```
A = (a→br→END).
B = (ar→cr→b→END).
C = (c→END).
Ma = (a→ ar→END).
Mb = (b→br→END).
Mc = (c→ cr→END).
||Seq1 =(A || B || C || Ma || Mc || Mb).
```

Seq1 モデルには 3 通りの実行パターンがある. それぞれ

```
a→ar→c→…
a→c→ar→…
c→a→ar→…
```

に対応する. 動作環境によっては, メッセージ a とcの受信順は規定できないことがある. その場合, このモデルには余計な動作制約が混じっていることになるので, 取り除かなければならない. 具体的にはプロセス式 B を以下のように変更する. この様にすると実行パターンは 5 通りになる.

```
B = ({ar,cr}→{ar,cr}→b→END).
```

RoseRT[7]の場合は Coregion を指定すること

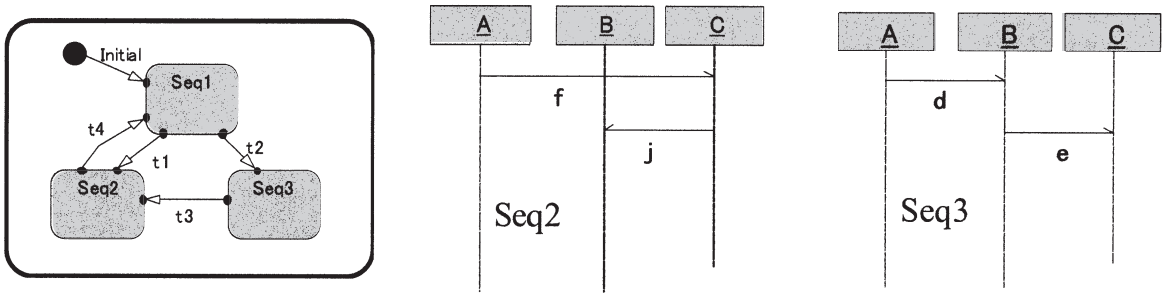

図 3. 複数シーケンス図による動作仕様

で受信順を規定しないことがある程度可能であるが，受信順を指定しない部分が図上で離れている場合には，規定できない．また，Coregion サポートしないツールも存在するので，生成されたモデルを直接修正して UML の限界を補う必要がある．

同期メッセージおよび関数呼び出しがある場合は，送信と受信の区別をしない．たとえば，メッセージ a が関数呼び出しであれば，a と ar を同一視したモデルを生成する．

4. 複数シーケンス図の扱い

図 1の様に同一のシーケンス図の実行を繰り返す場合には，以下の様なモデルを生成する．

LA = A; LA.
LB = B; LB.
LC = C; LC.
LMa = Ma; LMa.
LMc = Mc; LMc.
LMb = Mb; LMb.
||ArchModel1 = (LA ||LB ||LC ||Lma ||LMb || LMc).

各プロセス式の先頭に付けた L は loop の意味で付けた．セミコロン(;)は，LTSA のシーケンシャル結合オペレータである．ArchModel1 は，各コンポーネント A,B,C が独立に動作する場合の全体の動きを表しており，設計モデルに対応する．合成された ArchModel1 のテキスト形式は既に示した．

一方，仕様としての動きは，前記の Seq1 プロセスを繰り返すことで表現される．

LSeq1 = Seq1; LSeq1.

そして，これを property として ArchModel1 と合成することで設計モデルが仕様に違反するか否かを検証できる．

property ||Spec1 = (LSeq1).
||Check1 = (ArchModel1 || Spec1).

Check1 は，図 2に示したものになる．

シーケンス図間の関係が単純な繰り返しでない場合には，シーケンシャル結合オペレータだけでは表現できない．たとえば，図 1のシーケンス図 Seq1 に図 3の Seq2 と Seq3 を加えて，図 3の左側の状態図によってシーケンス図間の関係を定義した場合では，以下のように状態マシンテンプレートを定義して Seq1，Seq2，Seq3 の部分に，シーケンス全体の動作モデル或いはオブジェクトの動作モデルを代入することで，それぞれに対応する動作モデルを生成することができる．

STM = Seq1S,
Seq1S = Seq1; T1T2,
T1T2 = (t1→Seq2S | t2→Seq3S),
Seq2S = Seq2; T4,
T4 = (t4→Seq1S),

Seq3S = Seq3; T3,
T3 = (t3→Seq2S).
||SSTM = (STM).

そして，シーケンス図全体の動きを代入したSSTM をプロパティとして，各オブジェクトの動作モデルを代入した SSTM をさらに合成したArchModel を検査することで，隠れたシナリオの検出，あるいは同期構造の不備を検出できる.

たとえば，以下の反例を得る. これは，Seq1の終了後 Seq2 の実行を開始したが Seq2 が終了する以前にオブジェクト A が Seq1 の実行を開始してしまったことを検出している. すなわち，オブジェクト A は Seq1 を実行し，オブジェクト B と C は Seq2 を実行している. オブジェクト間で状態の不整合を起こしている.

Trace to property violation in Spec:

a
ar
c
cr
b
br
f
a

Seq2 が終了する前に Seq1 を開始することが問題であれば，Seq2 の終了をオブジェクト A に知らせるための同期機構が必要になる. あるいは，効率的な実行のために Seq2 と Seq1 を故意にオーバーラップさせて実行する場合[10]には，仕様モデルを変更しなければなせない.

5. まとめと今後の課題

非同期メッセージの扱いを厳密にすることで，従来のシナリオ検証では検出できない状態不整合を検出できることを示した. また，状態マシンテンプレートにシーケンス図やオブジェクトの動作モデルを代入する方法は，[2]のトレースモデルを生成する方法よりも状態数の増加が少なく状態爆発を起こしにくい様である.

隠れたシナリオを新たな仕様として追加する場合，[2]の様にインクリメンタルに仕様追加していくことをルーチン的におこなうことが本手法では難しい. この点は，今後の課題としたい.

参考文献

[1] Rajeev Alur, Kousha Etessami, Mihalis Yannakakis., Inference of Message Sequence Charts, IEEE Trans. of Soft Eng., pp. 623 – 633, **29**, 7, 2003.

[2] Sebastian Uchitel, Jeff Kramer, Jeff Magee, Incremental Elaboration of Scenario-Based Specifications and Behavior Models Using Implied Scenarios, ACM Trans. of Soft En. ad Method., pp. 37 – 85, **13**, 1, 2004.

[3] 藤倉俊幸，モデル検査を利用したタスク設計と検証の提案，組み込みシステム合同研究会，2006 年 1 月 23 日～24 日，大阪大学中ノ島センター.

[4] 林純也、藤倉俊幸， モデル検査を利用したマルチコアでのタスク設計と検証の提案，ESS2007 論文集.

[5] 岸知二，野田夏子，組込みソフトウェアのための UML 設計検証支援環境，SES2006 論文集.

[6] Enterprise Architect (http://www.sparxsystems.jp/)

[7] RoseRT (http://www.ibm.com/jp/software/rational/products/design/rosetech/index.html)

[8] Jeff Magee,Jeff Kramer,Concurrency: State Models & Java Programs,Wiley,2006

[9] FDR2 (http://www.fsel.com/index.html)

[10] 藤倉俊幸，オーバーラップ制御の設計と検証，第四回システム検証の科学技術シンポジウム，(http://unit.aist.go.jp/cvs/symposium/verification2007/abstract2007.html#fujikura)