モデル検査とテストを組み合わせたソフトウェア検証手法に関する研究

M2011MM052 中村 信太
指導教員　野呂 昌満

1　はじめに

近年，組込みシステムのソフトウェア開発の事例が増加してるが，開発が困難である．組込みシステムにおいて，異常系を含めた検証をすべて網羅することはテストでは不可能である．

モデル検査を用いた検証手法が有益であることが報告されている．モデル検査では，システムを網羅的に検査することで，設計者が意図していない振舞い等を検証できる．しかし，検証できる規模に制限がある．

本研究の目的は，モデル検査とテストを連携したソフトウェア検証手法を提案することである．本研究の課題は，モデル検査とテストの役割を明確にすることである．

本研究の基本的なアイディアは，アーキテクチャの振舞いをモデル検査技術とモデル指向形式言語を用いて段階を踏んで検証する．検証されたモデルと同様な振舞いを実現するコードであることをテスト技術を用いて検証することである．モデル検査技術では，抽象度の高いシステムと段階を踏むことで詳細化された箇所に対して適用する．モデル検査技術で表現が困難な仕様をモデル指向形式言語を用いて形式的に記述する．形式的に記述された仕様から自動的にテストケースを作成する技術 [2] を用いて，システムの振舞いおよび実現コードを検証する．

本稿では，自動販売機システムを事例に，アーキテクチャのモデル検査とコードのテストについて説明する．

2　背景技術

2.1　モデル検査

モデル検査では，システムの振舞いを表現したシステム記述とシステムの入出力を表現した環境と検査仕様がある．これらの情報をもとにシステムを網羅的に走査することで，デットロック等が発生しないことを保証できる．本研究では，モデル検査器を FDR[3] とし，その入力言にプロセス代数である CSP[1] を用いて検証する..

2.2　モデル指向形式言語

モデル指向形式言語とは，形式仕様記述言語であり，インタプリタ等を用いて記述仕様を実行することが可能な言語のことである．この言語を用いて，仕様を記述することで，仕様の曖昧さを取り除き，仕様を満足していることを既存のテスト技法を用いることで検証可能である．本研究においては，この言語として VDM++[5] を用いて事例検証をおこなった．

3　基本的アイディア

本節では，本研究で取り扱う計算モデルおよび検証手法について説明する．

3.1　対象とするシステム

本研究が扱うシステムは，システムの振舞いをコンポーネントがイベント通信を用いた協調動作により表現する．各コンポーネントの振舞いは，状態遷移機械で定義し，他のコンポーネントへの通信は各コンポーネントが保持する Queue を介した非同期通信で実現する．コンポーネントの振舞いを図 3.1 に示す．コンポーネントは，イベント受理した後に，アクションの実行および状態遷移を実行する．アクションでは，イベント送信と状態遷移では表現できない処理を定義する．これらは，コンポーネントの振舞いを状態遷移図とイベント送信関係をシーケンンス図を用いて記述する．なお，本研究においてコンポーネントは Model-View-Controller(以下，MVC) を用いて分類できることを前提する．

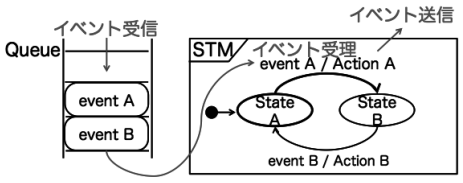

図 1　コンポーネントの振舞い

3.2　検証手法

本節では，モデル検査とテストの両方を用いた検証手法について説明する．提案するプロセスでは，状態遷移図とシーケンス図，アクションの振舞いの形式仕様記述をまとめて形式仕様記述として扱う．なお，提案する検証手法では，簡易的なシステムから仕様に定義された機能を実装されたシステムへ段階的に詳細化することで検証する．

3.2.1　モデル検査

本節では，モデル検査を用いた検証手法について説明する．

抽象化方法　システムを振舞いをイベントの送信と受理して抽象化する．アクションの振舞いは，イベント送信の有無に限定する．このように抽象化することで，アクションの実行は特定のイベントが受理されることで表現できる．なお，常に実行されないイベント送信については，非決定性を用いて記述する．

検証の分割　コンポーネントにおける入出力関係をインターフェースと定義し，インターフェースの検証を用いることで検証の分割する．各コンポーネントにおける協調動作は，各コンポーネントにおけるインターフェース同士が整合した際に，仕様を満足する振舞いを実現する．インターフェースをモデル検査における検査仕様と定義し，検証することでインターフェースが整合することを検証する．コンポーネントの振舞いをインターフェースに着目することで，コンポーネントの内部処理を隠蔽しての検証基準となる．

段階的詳細化　本研究が提案する検証プロセスでは，MVC 分割における Model 要素のみで構成されたシステム (以下，基本システム) から段階的に仕様に基づいたシステムに詳細化することで，検証範囲の局所化する．基本システムがもつ機能は，仕様に基づいたシステムを実現する上で必要とする最小限の機能のみとする．検証プロセスにおいて，基本システムを検証することで，システムを構成する全コンポーネントのインターフェースを検証する．検証されたインターフェースを基に，詳細化された各コンポーネントを検証することで，検証範囲を小さくすることが可能である．

3.2.2　テスト

本研究におけるテストでは，仕様を検証するテストと実現コードが仕様を満足していることを検証することで，仕様を満足することを保証する．

形式仕様の検証　この検証においては，形式仕様記述が仕様を満足しているかを検証する必要がある．モデル検査により，コンポーネント同士の協調動作した結果に目的とするアクションを実行することを検証できるが，そのアクションの振舞いは保証できない．その振舞いが仕様を満足することを保証することが目的である．この振舞いを検証する手法として，C.Meudec が提案する自動的テストケースを作成する手法 [2] に基づき作成する．作成されたテストケースと仕様から振舞いを満足していることを検証する．

実現コードの検証　この検証では，形式仕様記述と同様な振舞いであることを保証する．検証手法は，コンポーネントならびにアクションの振舞いを状態遷移テストと形式仕様記述から作成されたテストケースを用いることで検証する．

3.3　アーキテクチャの詳細化

段階的に詳細化する過程でよくあられた詳細化を分類することで，詳細化方法とその検証について説明する．

3.3.1　振舞いの詳細化

この詳細化は，コンポーネントの振舞いそのものを再設計することである．主に変更される振舞いは，非決定な処理順序である処理を決定的な処理順序に定義することやイベントの競合を対応させることである．前者の振舞いは主にアーキテクチャ記述の情報をコードに変換するときに必要である．後者の振舞いは，並行処理を実現する上で考慮すべき項目である．本研究では，状態遷移そのものを部品と考え，競合に対処した状態遷移に置換することで対処する (図 2 を参照)．

この詳細化は，モデル検査を用いたインターフェースを検証することでシステムに問題ないことを検証する．インターフェースを検証するとは，対象とするコンポーネントに対しての入出力順序が設計者が意図した順序で振舞われることを検証することである．

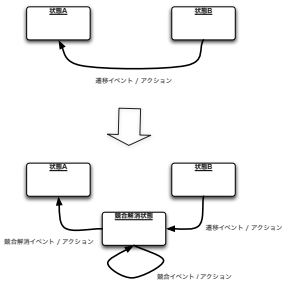

図 2　状態遷移の部品化

3.3.2　状態の詳細化

この詳細化では，コンポーネントの状態の意味を考慮することである．状態の意味が詳細化されることで，状態遷移に伴って実行されるアクションの処理も詳細化されたことを同義である．アクションの処理を詳細化することで，イベントに付加されている情報も詳細化する．

この詳細化は，アクションの振舞いの形式仕様を作成することで，状態の意味を明示する．作成された形式仕様は，インタプリタを用いたテストすることで，仕様に従った振舞いを実現していることを検証する．また，形式仕様記述における条件分岐の振舞いを非決定な振舞いとしてモデル検査で検証する．

4　事例：自動販売機システム

本節では，前節で挙げた詳細化を自動販売機システムに適用して説明する．本節の構成は，基本システムを説明した後に，詳細化について説明する．詳細化の事例は，競合を考慮した詳細化と異なる価格の商品に対する詳細化とする．なお，本節に記述されているステートマシン図では，アクション名のかわりに，「コンポーネント < –送信イベント」形式でイベント送信を記述している．

4.1　基本システム

自動販売機システムにおける基本システムは，全体の振舞いを管理する販売ポリシー，投入金額を管理する投入金貯蔵庫，商品の購入状態および在庫数を管理する商品ボタン，商品貯蔵庫と時間を管理するタイマーで構成とする

(図 3 を参照)．基本システムにおける販売ポリシーの振舞いは図 4 である．

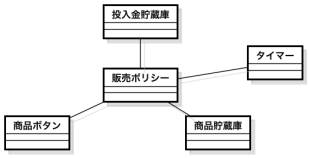

図 3 基本システムの構成

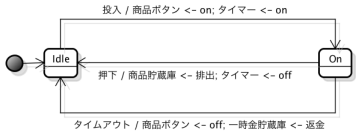

図 4 基本システムにおける販売ポリシーの振舞い

ここにおける仕様は，投入した後に，ボタンを押下，レバーを引くまたは規定時間以内ボタンを押下しないときに，商品を排出または投入物の排出することを検証する．

4.2 競合を考慮した詳細化

販売ポリシー内で発生し得るイベント競合に対する詳細化をおこなう．販売ポリシー内では，On 状態で受信するイベントは，押下とタイムアウトイベントであり，これらのイベントが競合する．これらのイベントに対して，振舞いの詳細化することで図 5 と詳細化できる．

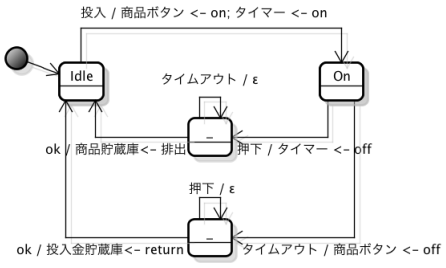

図 5 競合を考慮したシステムにおける販売ポリシーの振舞い

このシステムの仕様は，基本システムに加え，それぞれの入力操作におけるイベント競合とそれらが非決定的選択し続けられてもシステムが停止しないことを検証する．この詳細化においては，販売ポリシーの振舞いとその振舞いに対する応答処理を追加することで実現している．変更されていない部分が同一のインターフェースであることを保証することで，詳細化に伴い発生した設計ミスを早期に発覚させることができる．

4.3 異なる価格の商品に対する詳細化

商品ボタンで実行されるアクションを詳細化する．詳細化するアクションとは，商品ボタンが管理しているボタンの有効を管理するアクションである．このアクションは，販売ポリシーを経由して投入金貯蔵庫に投入金額が通知されることを前提に VDM++ 記述は図 6 に示す．

```
instance variables
価格表 : map ボタン to 価格;
operations
ボタンの有効化 : nat ==> ()
ボタンの有効化( 投入金額 ) == (
 for all 参照ボタン in set dom 価格表
 do if 参照ボタン.商品価格 <= 投入金額 then
  参照ボタン.有効();
 else skip );
```

図 6 商品ボタンに関する詳細化の形式仕様記述

この詳細化において，形式仕様記述における if 文に着目する．モデル検査では，この部分を非決定的に実行される処理とし，テストではこの部分が仕様を満足することを検証する．この形式仕様記述から同値クラスは 参照ボタン.商品価格 $\leq$ 投入金額 が真であるときに参照ボタンが有効になることを保証するテストケースを作成する必要がある．この事例は，状態の詳細化となる．

5 考察

5.1 検証の分割

提案する検証手法における検証分割について自動販売機システムを事例に検証規模が小さくなっていることを考察する．本稿における基本システム (図 3 を参照) に，View と Controller となるコンポーネントを含めたシステムと比較することで，検証規模の比較をおこなう．なお, 文献 [4] をもとに考える．View と Controller コンポーネントとして，硬貨の投入口と排出口，返金レバー，購入ボタン，購入ランプ，売切ランプと商品排出口の少なくとも計 7 個のコンポーネントを追加が考えられる．Model 部分も含めたコンポーネントの総数は，11 個となる．そして，n 個のボタンを取り扱うシステムを仮定した場合，$8+3n$ 個の実体を用いて検証する必要がある．提案する検証手法で検証する範囲は，5 個のコンポーネントでシステム全体とシステムにおける View，Controller となる各コンポーネントのインターフェースを検証する．これらのことより，自動販売機システムにおいて，モデル検査の実施範囲を小さくすることができた考える．

検証の分割方法について，一般化に適用できるかを考察する．検証の分割において，Model の集合とそれ以外に分割している．組込みシステムにおいて，2 個以上あるボタン等の機器はユーザに対する入力または出力機器であると考える．2 個以上ある機器を制御するコンポーネントは，Controller または View に分類される可能性が大き

い．これにより，Model 部分では同一のコンポーネントにおける同一イベントにおける競合を考慮することで，２個以上の機器を扱うことが可能であり，検証するコンポーネントの数は変化しないと考える．それに対して，View と Controller のコンポーネントはそれぞれ独立してすることにより，Model とのインターフェースのみで検証規模はコンポーネント 1 個のみであると考えられる．

これらにより，検証の分割方法は有用であると考える．

5.2 検証コスト

本研究が提案する検証手法における検証コストについて考察する．提案する検証手法における検証方法は，モデル検査，状態遷移テストとアクションの振舞いのテストであり，それぞれについて考察する．

まずはじめに，モデル検査について考察する．モデル検査で，コンポーネントの振舞いに対して網羅的に検証することが目的である．基本システムにおいて，入力に対する出力関係が仕様を満足していることを網羅的に検証する必要がある．そして，検証された基本システムで保証された振舞いを保つために，各コンポーネントのインターフェースが詳細化しても変更ないことを網羅的に検証する必要がある．これらについて，検証コストを削減することはできないと考える．

次に，状態遷移テストについて考察する．状態遷移テストの目的は，モデル検査により検証されたモデルと同様な振舞いをコード上で実現していることを検証である．検証されたモデルと同様な振舞いであることを保証する方法として，状態遷移図に記述された状態遷移であることを保証できるが，多くのテストケースを必要とする．

本研究が提案する検証手法の中のモデル検査のインターフェースを用いることで，状態遷移テストに伴い検証コスト削減をできると考える．状態遷移テストを通じて，検証されたモデルと同様な振舞いであることを検証する．つまり，イベント送受信関係が同等となることを検証すると考える．イベント送受信関係は，本研究においてはコンポーネントにおけるインターフェースで定義されている．インターフェースの情報と形式仕様記述から生成されたテストケース等の情報を組み合わせることで状態遷移テストにおけるテストケースが作成が可能である．このようにテストケース作成を考えることで，既存の状態遷移テストよりも少ないテストケースの数でコンポーネントの振舞いを検証することが可能であると考察する．

最後に，アクションの振舞いについて考察する．アクションの振舞いにおいては, テストケースを自動的に生成する研究 [2] を利用している．この研究においては，陽定義で記述された形式仕様記述を解析し，同値クラスを作成した後にテストケースを作成する．しかし，形式仕様記述における繰り返し文に対して検討する必要があると考える..

本研究が取り扱う計算モデルにおけるアクションでは，イベント送信また状態遷移では表現することが困難な処理を記述される．

イベント送信に対して，繰り返し処理を用いて形式仕様記述が記述されるときには，同一のコンポーネントの型である複数の実体にイベント送信を必要となるシステムであると考える．同一の型から生成された実体の振舞いは，実体における属性値により変化するがコンポーネント単位で検証が可能である．同一のコンポーネントの型から生成された必要数の実体があることが検証されていることが前提になるが，それらの中から選択し，仕様を満足していることを検証することでテストケースの作成の労力削減がきると考える．

状態遷移で表現できない処理とは，システムが取り扱う情報に対する操作処理が大半を占め，残りは組込みシステムを構成する機器の操作処理であると考える．システムが取り扱う情報に対する操作に対しては，情報処理に対して繰り返し文が必要な処理であると考える．しかし，機器の操作処理に対しては，システムを動作させるのに必要とするアプリケーションインターフェース (以下，API) として定義されていると考え，形式仕様記述では表現されないと考える．これらにより，この処理においては検証コストを削減することができないと考える．

本研究が提案する検証プロセスにおいて，状態遷移テストにおけるテストケースと同一コンポーネントの型から生成される実体にイベント送信を実行しているテストケースについては，削減するできると考える．

6 おわりに

本研究の成果は，自動販売機システムを事例に 2 つの詳細化を定義した．詳細化を定義することで，モデル検査とテストの明確な基準を与え，検証方法を提示した．

今後の課題は，多種多様な事例に適用可能であることの検証，他の詳細化がないか検討と形式仕様記述からのテストケースの生成手順の検討が考えられる．

参考文献

[1] C. A. R. Hoare, Communicating Sequential Processes, Prentice-Hall, 1985.

[2] C. Meudec, *Automatic Generation of Software Test Cases From Formal Specifications*, Queen's University, 1988

[3] Formal Systems (Europe), “Formal Systems (Europe) Ltd,” http://www.fsel.com/, 2010.

[4] 鯵坂恒夫，池田健次郎，中谷多哉子，野呂昌満，“OO’97 オブジェクト指向モデリングワークショップ報告, ” 情報処理学会研究報告，ソフトウェア工学研究会報告, pp.33-35，1997.

[5] 佐原伸, 荒木啓次郎, ”オブジェクト指向形式仕様記述言語 VDM++ 支援ツール VDMTools,”コンピュータソフトウェア, vol.24, no.2, pp.14-20, 2007.