ソフトウェアテストシンポジウム 2003
JaSST '03: Japan Symposium on Software Testing 2003

組み込みソフトウェアの上流設計における試験表現に関する研究
―状態遷移モデルを基にしたモデルベース試験―

山本 修二†

†キャッツ株式会社 ソフトウェア事業部 〒222-0033 神奈川県横浜市港北区新横浜 2-11-5
E-mail: †shuji@zipc.com

あらまし 組み込みシステムは、従来から状態遷移機械と捉え、状態遷移を基に設計されている。状態遷移は、状態遷移図や状態遷移表でモデル化されているが、状態遷移の正当性確認はレビューで行われている。この状態遷移を試験することにより、早期に上流設計における不具合が抽出できる。しかし、従来は試験用状態遷移シーケンスを人手で抽出しており、モレヌケが発生する可能性が極めて高い。さらに、試験用状態遷移シーケンスの表示・表記方法は作成者によって異なっており、再利用性が極めて低くなっている。ついては、試験用状態遷移シーケンスの表記方法の標準案を提案すると共に、状態遷移シーケンス抽出ツールプロトタイプの開発について報告する。

キーワード 組み込みソフトウェア、試験、状態遷移、ステートチャート

Research of test expression for embedded software on high-level design
―Model base testing based on state transition model―

Shuji Yamamoto†

†Software Division, CATS CO.,LTD 2-11-5 Shinyokohama, Kouhoku-ku, Yokohama-city, Kanagwa, 222-0033 Japan
E-mail: †shuji@zipc.com

Abstract Embedded system is designed based on state transition since it is concerned in state transition machine. State transition is modeled by state transition diagram or state transition matrix but rightness of state transition is done in review. By examining this state transition, defect is extracted in upper design level. However, state transition sequence to test is normally extracted by hand thus; there is high possibility of finding defect or human error. Moreover, expression and way of writing are depending on individuals so it is hard to reuse. Consequently, standard way of writing for state transition sequence to test is provided. Also, prototype of state transition sequence extraction tool is developed and the result is reported.

Keyword embedded software, testing, state transition, state chart

1. はじめに

現在組込みシステムは、携帯端末、情報家電、車載機器といった幅広い分野で利用されており、近年の高度情報化社会においてはなくてはならない基盤技術の一つである。しかし、ひとたび障害が発生すれば、利用者が単に被害を被るだけでなく、社会や開発企業にも多大な損害を与えることになりかねない。にもかかわらず、年々大規模化する組込みシステムに対し、開発環境、特に試験環境の整備が遅れているのが実情である。このため、開発工程の重要な部分である試験においては、試験項目の抽出が人手で行われているため、設計段階におけるモレヌケを見落とし、組込みシステムのバグの発生原因となっている。開発企業は、設計段階のモレヌケをなくし如何に高品質の組込みシステムを開発するかが重要な課題であることを認識している。[1]

一方、組込みシステムの分野では状態遷移表やステートチャート図を使った設計は一般的であるが、試験のための状態遷移シーケンスの表示及び表記の標準化がされていないため、各企業はそれぞれ独自のフォーマットで設計や試験を行っている。もし、組込みシステム分野における試験用状態遷移シーケンスの表示及び表記の標準化が進めば、ツールベンダーは標準規格の試験ツールを提供することが可能となり、組込みシステム試験環境の整備が促進される。この結果、組込みシステム開発者の教育も容易になり、我が国の組込みシステム産業の発展に大いに貢献するものと期待される。更に、試験環境の整備により人手で行っていた試験用状態遷移シーケンスの抽出が自動化されれば、設計段階で状態遷移のモレヌケが低減し、開発企業は高品質の組込みシステムを利用者に提供することが可能となる。

2. 状態遷移設計[2]

そもそも組み込みシステム開発における状態遷移による設計はどんなものであろうか。例えば、電源S

Wとモータオン・オフＳＷが付いたモータ制御機器のモデルを設計すると、図１の状態遷移図になる。

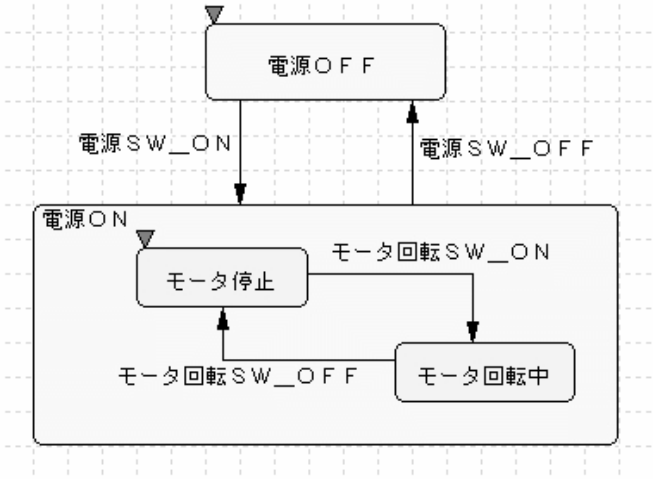

図１　モータ制御状態遷移図

Fig.1. state transition diagram of motor control

図１の状態遷移図では、状態と入力（以後イベントと呼びます。）の全ての関係を網羅して記述しにくい。図１に、状態「電源ＯＦＦ」においてイベント「モータ回転ＳＷ＿ＯＮ」が発生した場合などの状態とイベントの関係を追加記述すると、図２の状態遷移図になる。

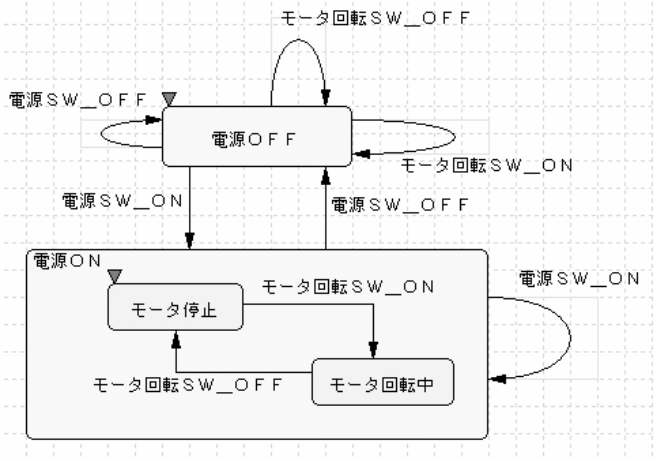

図２　モータ制御状態遷移図（追記）

Fig.2. state transition diagram of motor control (additional)

例えば、図１において状態「電源ＯＦＦ」においてイベント「モータ回転ＳＷ＿ＯＮ」は無視されるが、状態遷移図で記述すると、自己遷移（今の状態に遷移することを言う。）になるのか、無視されるのかが判別できない。追加したイベントが無視されることを明確に記述した状態遷移表を図３に示します。

□0 モータ制御	S	電源ＯＦＦ	電源ＯＮ		
				モータ停止	モータ回転中
E		0	1	2	3
電源ＳＷ＿ＯＮ	0	=>電源ＯＮ	／	／	／
電源ＳＷ＿ＯＦＦ	1	／	=>電源ＯＦＦ	／	／
モータ回転ＳＷ＿ＯＮ	2	／	／	=>電源ＯＮ:モータ回転中	／
モータ回転ＳＷ＿ＯＦＦ	3	／	／	／	=>電源ＯＮ:モータ停止

図３　モータ制御状態遷移表

Fig.3. state transition matrix of motor control

状態遷移表による設計では､状態とイベントの全ての関係が網羅して記述できるため、設計時に状態遷移のモレヌケが防止できる。

状態遷移設計は､状態とイベントが一意的に決定されることを前提としている設計手法です。「一意的に決定される」とは、一つの状態またはイベントは状態またはイベントを決める全ての内容が固定であることを言います。例えば、状態については変数の値が異なること、イベントについてはイベントに含まれる内容が異なることは想定していません。状態は変数の値が異なる毎に状態を分ける必要がありますし、イベントも内容が異なるごとにイベントを分ける必要があります。

しかし、そのような状態遷移設計を行った場合、状態数とイベント数は爆発的に増加し、本来見通しが良いはずの状態遷移設計は見通しの悪い設計手法になってしまいます。そこで現実的には、状態はある程度見える違いを基に状態を分け、イベントは状態遷移に影響を与えるイベント毎に分離して設計を行っています。

3. 状態遷移試験

状態遷移設計は､組み込みソフトウェアを開発工程で、実装モデルを明確にするためにどうしても必要な作業になります。組み込みソフトウェアを開発する上で、上流設計の最後の工程となります。しかし、従来の開発工程では、この実装モデルをレビューという形で試験（正確には試験とは呼べない。）を行い、実装を行ってきています。この状態遷移設計結果のモデルを真の意味で試験し、状態遷移モデルの問題が無いことを結果として残すことで、その後の不具合発生を抑えることが出来ます。

また､状態遷移モデルを試験するために作成された試験用状態遷移シーケンスは、コード実装後に試験を行う上で、再利用することも出来ます。

しかし、ここで考えなければならないのは、試験用状態遷移シーケンスはどのようにして作成されたものかです。図３の状態遷移表において、状態遷移シーケンスはどのくらいあるでしょうか｡抽出条件を下記の点にした場合、20 通りぐらいの状態遷移シーケン

スが抽出できるでしょうか。

1．開始状態を電源ＯＦＦ、修了状態を電源ＯＦＦとする。
2．遷移する前の状態と遷移後の状態と遷移のきっかけとなるイベントが同じ組み合わせは、１つの遷移シーケンス内に１回しか現れない。
3．イベントが発生した時、イベントが無視される場合は、遷移したと見做す。

状態遷移シーケンス抽出条件

この条件は状態遷移設計において、状態やイベントが一意的に決定されていないことを前提に設定しています。なぜ、一意的に決定されていない場合、組み合わせで抽出した状態遷移シーケンスで不十分かというと、1つの状態遷移はその前に発生した状態遷移に依存している可能性があるからです。状態が一意的に決定されていない場合、イベントが発生した状態における処理や状態遷移は、その状態にのみ依存しないで、そのイベントが発生する前の処理や状態遷移の結果で変数の内容が変わり、その変数の内容によって処理や状態遷移が分岐する場合があるためです。

このような条件で抽出した場合、状態遷移シーケンスは、4129 通りになります。下に、１例を示します。

電源 OFF - モータ OFF -> 電源 OFF - 電源 OFF -> 電源 OFF - モータ ON -> 電源 OFF - 電源 ON -> モータ停止 - 電源 ON -> モータ停止 - モータ OFF -> モータ停止 - モータ OFF

状態遷移シーケンス例

この状態遷移シーケンス例では、”電源 OFF - モータ OFF -> 電源 OFF”は状態「電源 OFF」からイベント「モータ OFF」により状態「電源 OFF」に遷移したことを示しています。

4. 状態遷移シーケンス標準の提案

従来の試験ツールにおいては、特定のプログラム言語（Ｃ言語やＪａｖａなど）をベースにツールベンダー独自のスクリプト言語により試験を記述する方法が大勢を占めている。しかし、今後状態遷移をベースにした試験を行う場合、状態遷移シーケンスの表記方法や表示方法は、ツールを使用するユーザーの利便性を図るためにも、標準的な表記方法や表示方法が必要である。

本論文では、まず状態遷移シーケンス表記方法の標準案を提案する。

4.1. 状態遷移シーケンス表記方法の提案

現在、OMG(Object Modeling Group)にて状態遷移の表記方法として、XML(eXtended Markup Language)による表記方法が検討中である。この検討結果は、UML2.0(unified Modeling Language)として発表される予定です。

しかし、開発者の利便性を考えると、状態遷移シーケンスの表記方法を試験のための表記方法として早期に作成する必要があります。表記方法としては、今後データ変換や各社ツール間での再利用性を図るために、XML による表記方法とテキストによる表記方法の２案を提案します。

まず、状態図(SD : State Diagram)、タスク状態図(TD : Task Diagram)、遷移列集合(TS : Transition Sequence)、並列遷移列(CTS : Concurrent Transition Sequence)を数学的に定義する。

4.1.1. 状態図(SD : State Diagram)

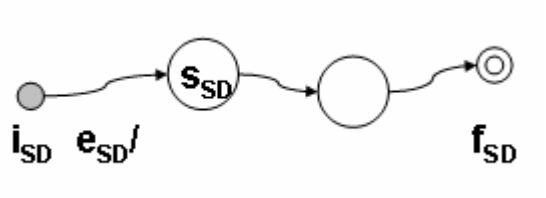

図４　状態図例

Fig.4. state transition diagram

g　…… guard
a　……. assertion
l　……. label
n　…… note
g？ (exp, TASK)
a？ (exp, TASK)
e？ (l, m, time, pri)
s_{SD}？ (g, a, l, n, PROC)
t_{SD}？ (e, g, a, l, n, PROC, s_s,s_d)
SD？ (I_{SD}, F_{SD},S_{SD}, T_{SD}, f, f-1,E_{SD})

4.1.2. タスク状態図(TD : Task Diagram)

s_{TD}？ (g, a, l_t, n, PROC, prev)

- l_t：タスク名、prev は PROC で表す。
- Temporal operator は、prev, g, a で使用する。
- Prev は、g,a で記述する。

TD？ (I_{TD}, F_{TD}, S_{TD}, T_{TD}, f, f^{-1}, l_t, pri, time, E_{TD}, TYPE)

タスク間通信　∈　E_{TD}

4.1.3. 遷移列集合(TS：Transition Sequence)

ts ∈ TS

ts ? $(i_{ts}, f_{ts}, T_{ts}, g)$

t ∈ T_{ts} , t_1 -> t_2 -> …… -> t_n

$t_{i+1} = g(t_i)$

4.1.4. 平行遷移列(CTS：Concurrent Transition Sequence)

ts ∈ TS , tg ⊂ TS

cts ? (t_g, h)

4.1.5. XML による表記方法

数学的定義により、XMLによる状態遷移シーケンスの表記方法を下記の様なDTDで定義する。

```
<?xml version='1.0' encoding='UTF-8' ?>
<!ELEMENT pathGroup (title, path+) >
<!ELEMENT title (#PCDATA) >
<!ELEMENT path (state ,transition+,state?) >
<!ELEMENT state
(activity?,stateName ,guard? ,assert? ,activity?,path?,activity?)>
<!ELEMENT transition (eventName ,guard? ,op?,state?)>
<!ELEMENT stateName (#PCDATA) >
<!ELEMENT eventName (#PCDATA) >
<!ELEMENT op (#PCDATA) >
<!ELEMENT activity (#PCDATA) >
<!ELEMENT guard (#PCDATA) >
<!ELEMENT assert (#PCDATA) >
```

状態遷移シーケンス(DTD)

本論文では、DTDの表記方法に関する詳細な説明を避けるが、各々のタグは下記の意味を示している。

- pathGroup：状態遷移シーケンス群
- title：状態遷移シーケンス群名称
- path：状態遷移シーケンス
- state：状態
- transition：遷移
- stateName：状態名
- eventName：イベント名
- op：処理内容
- activity：付随処理内容
- guard：遷移条件
- assert：検証条件

上記DTDに従った図3の状態遷移シーケンス例は下記の通りである。

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE pathGroup SYSTEM 'file: pathGroup.dtd'>
<pathGroup>
  <title>モータサンプル</title>
  <path>
    <state>
      <stateName>電源 OFF</stateName>
    </state>
    <transition>
      <eventName>電源 SW_ON</eventName>
      <state>
        <stateName>電源 ON：モータ停止</stateName>
      </state>
    </transition>
    <transition>
      <eventName>モータ回転 SW_ON</eventName>
      <state>
        <stateName>電源 ON：モータ回転中</stateName>
      </state>
    </transition>
    <transition>
      <eventName>モータ SW_OFF</eventName>
      <state>
        <stateName>電源 ON：モータ停止</stateName>
      </state>
    </transition>
    <transition>
      <eventName>電源 SW_OFF</eventName>
      <state>
        <stateName>電源 OFF</stateName>
      </state>
    </transition>
  </path>
</pathGroup>
```

状態遷移シーケンス標準案表記方法例

5. 状態遷移シーケンス抽出ツールプロトタイプ

状態遷移シーケンス抽出プロトタイプを作成し、評価を行った。プロトタイプの画面イメージを図5に示す。

図5右上枠の状態遷移表は、図3と同じである。この状態遷移表から、開始状態・終了状態共に電源OFFである条件の下で抽出した状態遷移シーケンスが右中枠に示されている。

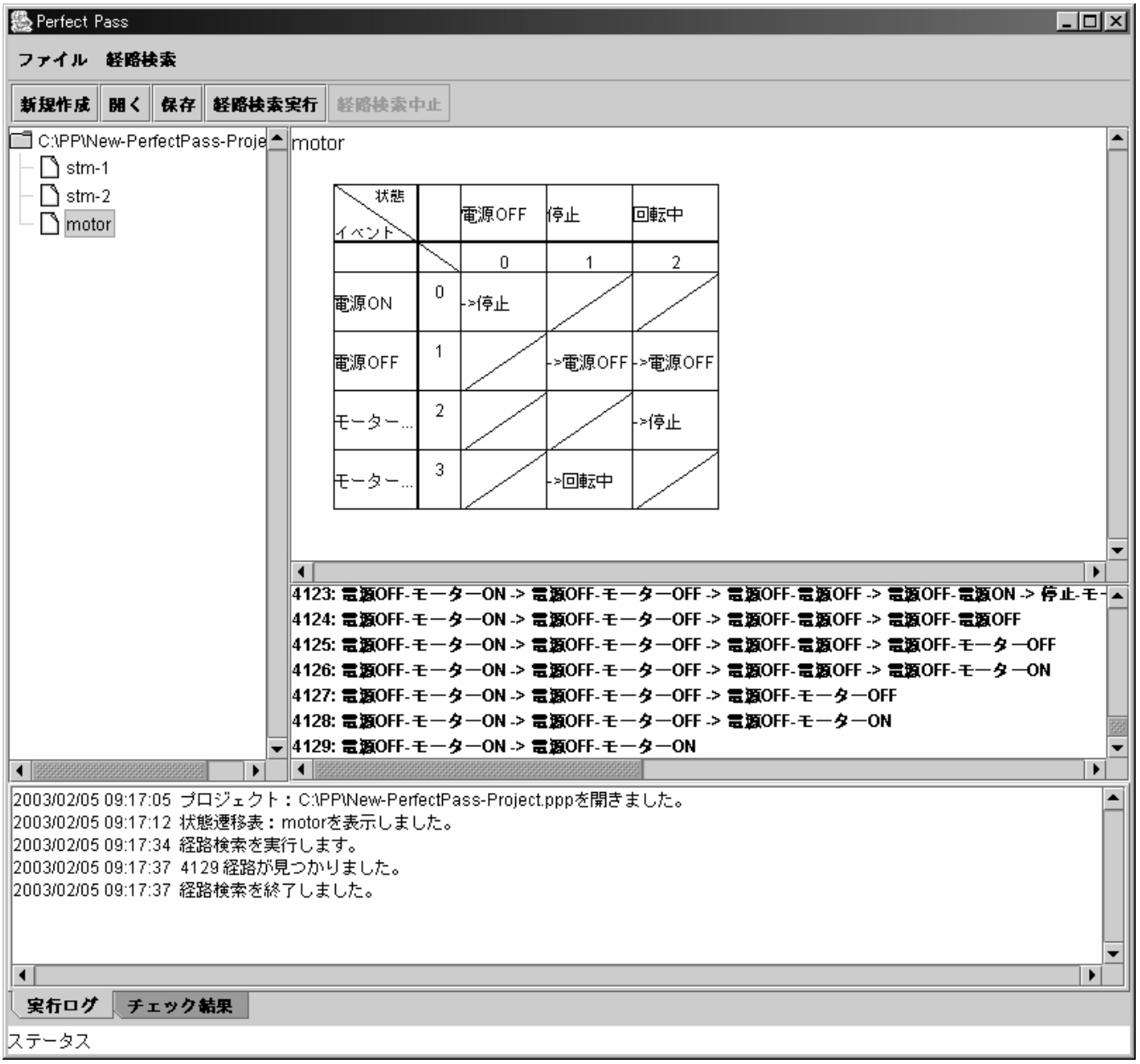

図５．状態遷移シーケンス抽出ツールプロトタイプ画面

Fig.5. proto-type tool image of extracting state transition sequence

本ツールのプロトタイプにおいて、図３の状態遷移表から 4129 通りの状態遷移シーケンスが抽出されることが判る。

本ツールでは、状態遷移シーケンスの抽出だけを目的にしたが、図３の簡単な状態遷移表でさえ、4129 通りの試験すべき状態遷移シーケンスがあり、とても人手により抽出できないし、全ての抽出された状態遷移シーケンスの試験を行うことも現実的に不可能であることが、改めて確認された。

6. 今後の課題

今後の課題として、大きく３つのことを考えていく必要がある。

第１は、状態遷移設計手法に関するものである。状態遷移設計では、いかにイベントを無視するまたはイベントが発生すべきでない組み合わせを少なく設計するかが重要になってくる。これには、状態とイベントの階層化の最適化が必要と推測される。状態遷移設計では、見通し易さが大変重要であるため、階層化を計った際に階層間の関連を以下に見易くツールによってサポートするかが重要になる。このツールによるサポートをどのように行うか検討する必要がある。

第２は、状態遷移シーケンスの分類を定義する必要がある。今回のプロトタイプでは、分類を行わず全て

の状態遷移シーケンスを抽出したが、今後は開始条件、終了条件、経由条件などを設定し抽出する予定である。しかし、試験者が最初に必要な状態遷移シーケンスは正常に動作している場合のみで、イベントが無視されるか禁止されている部分を含むシーケンスはコストや納期により行うかどうかを決めている。このようなことから、状態遷移シーケンスの抽出条件に分類を加えることで、抽出された状態遷移シーケンスの重要度を明確にする必要がある。

第３は、ツールによる自動試験を行う必要があることです。試験するための状態遷移シーケンスは、図３の単純な状態遷移でさえ、4129 通りの状態遷移シーケンスがありますので、全てを人手で試験することは不可能です。ツールにより抽出された状態遷移シーケンスにより自動試験することで、更なる効率化が期待できる。

文　献

[1] JEIDA,“日本国内における組み込みシステムにおける HW/SW Co-Design の現状と課題調査”, Dec.,1998.

[2] 渡辺政彦,”拡張階層化状態遷移表設計手法 Ver.2.0～Embedded SE のための設計手法”,東銀座出版社, 東京, 1998

[3] 渡辺政彦, 計測と制御,”状態遷移ベースのソフトウェア開発環境の現状と動向”, 第 41 巻,第 2 号, （社）計測自動制御学会, 2002.

[4] 渡辺博之,渡辺政彦,堀松和人,渡守武和記,”組み込み UML”,翔泳社,東京,2000.

[5] 渡辺政彦,情報処理,“組込みソフトウェア上流開発環境を日本で”, Vol.41,No.2,通巻 420 号,(社)情報処理学会,Feb.,2000.