Consistency of StateMachine

by SparxSystems Japan

Enterprise Architect 日本語版

ステートマシン図の整合性確保 マニュアル

(2016/09/01 最終更新)

目次

1. はじめに

UML を利用したモデリングにおいて、ステートマシン図は振る舞いを表現する図として利用される機会が少なくありません。特に、組込み機器の設計においては、状態が強く意識されることが多く、ステートマシン図が利用される頻度が高いです。

一方で、UML のステートマシン図単独では、大きな問題が 1 点あります。この問題を解決するためには、UML には含まれない、状態遷移表を組み合わせる必要があります。

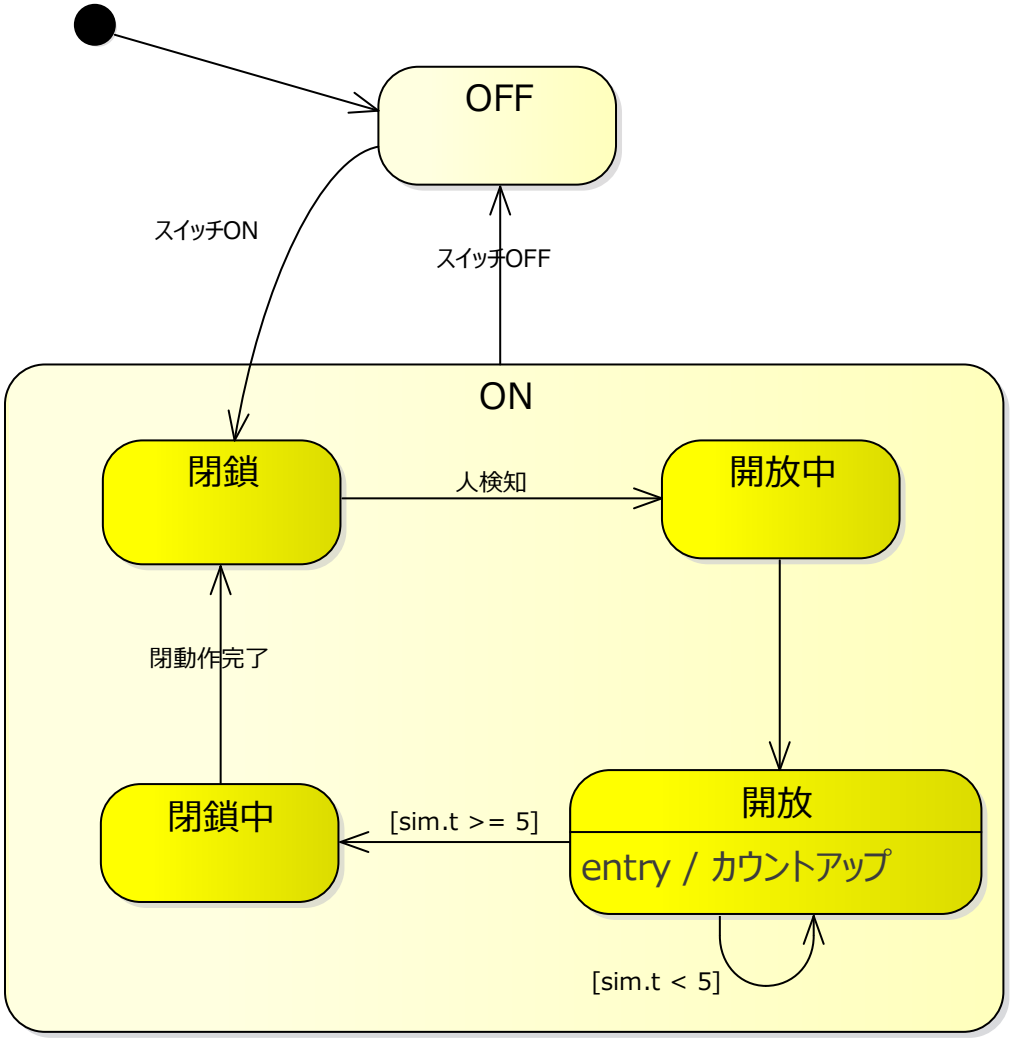
このドキュメントでは、UML のステートマシン図と状態遷移表を組み合わせることで、ステートマシン図内の考慮漏れや抜けを排除し、整合性の確保を実現する方法を紹介します。さらに、ステートマシン図を「動かす」ことで、内容に問題がないかどうかをモデル上で検証し、検証した内容をそのままソースコードとして出力する方法についても概要を説明します。

このドキュメントでは、Enterprise Architect13.0 ビルド 1305 での操作方法を説明しています。

2. ステートマシン図の長所と問題点

この章では、ステートマシン図と状態遷移表を比較することで、ステートマシン図の長所と問題点を確認します。

以下の図は、ステートマシン図の 1 つの例です。自動ドアを想定した例になります。



このステートマシン図を見ることで、以下の内容が簡単にわかります。

- ✓ 大きく分けると、ON と OFF の状態があること
- ✓ ON の状態の中には 4 つの状態があり、基本的には、この 4 状態が循環すること

このように、ステートマシン図は、全体の構成を把握したり、どのような状態や遷移があるのかを把握したりするために、効率的で有用です。上の例のように状態が入れ子になっている場合など、複雑な状態遷移を把握する場面では特に効果的です。

一方で、ステートマシン図の問題点は、網羅的な内容の検証が容易ではないと言うことが挙げられます。要素の配置や図の大きさなどによっては、全ての状態間について漏れなく確認することは困難です。

また、例えば上の例の場合、閉鎖から開放には遷移の矢印がありません。このように、状態間に遷移の矢印が存在しない場合の状況がステートマシン図から正確に判別できません。遷移がある場合には、

状態間の遷移を検証した結果存在するという結論になったことがわかりますが、遷移がない場合に、

- ✓ まだ検証していない
- ✓ 検証した結果、遷移がないという結論になった

という 2 つの違いが分かりません。これは、UML の表現として、「遷移しない」ことを示す線が存在しないことが原因です。遷移の色などで区別する(例:遷移しない場合には灰色にする)ことも考えられますが、余計な遷移が図内に表現されることで、ステートマシン図の長所である、簡単に全体構成を把握できる点が失われてしまいます。

3. 状態遷移表の長所と問題点

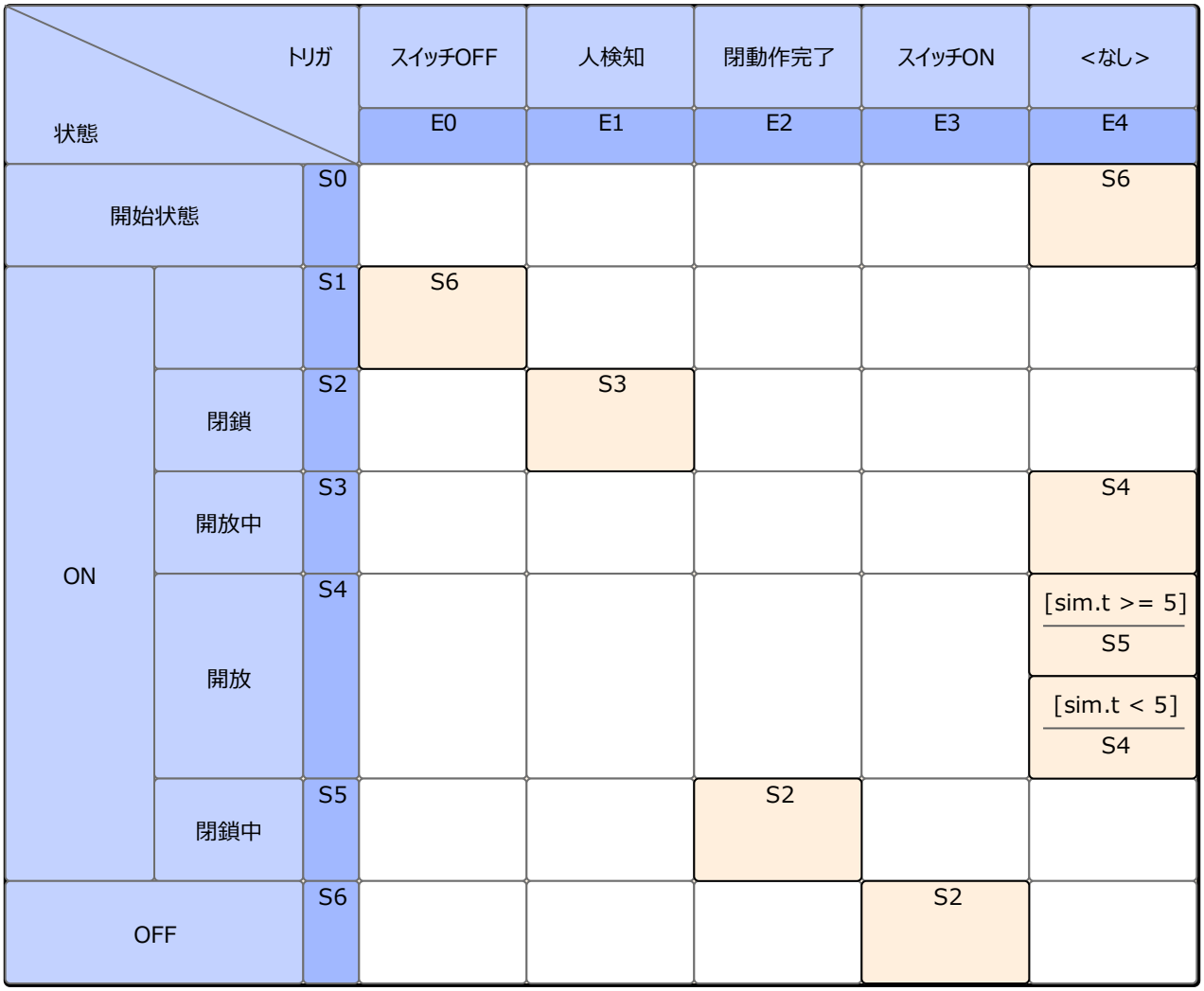
一方、同じ内容を状態遷移表(状態と状態との関係)で表現した例が、次の表になります。

次の状態 / 状態			開始状態	ON					OFF
					閉鎖	開放中	開放	閉鎖中	
			S0	S1	S2	S3	S4	S5	S6
開始状態		S0							sim.t = 0
ON		S1							スイッチOFF
	閉鎖	S2				人検知			
	開放中	S3							
	開放	S4					sim.t < 5	sim.t >= 5	
	閉鎖中	S5			閉動作完了				
OFF		S6			スイッチON				

状態遷移表の長所は、すべてのセルについて検証することで、網羅性を確保できるという点です。つまり、上記の表の空いているセル全てに対して遷移があるかどうか確認することにより、全ての可能性を検討できるのです。

また、状態遷移表には、上記の状態間の関係を示すものの他に、それぞれの状態においてトリガ(イベ

ント)が発生した時の振る舞いを示す形式もあります。



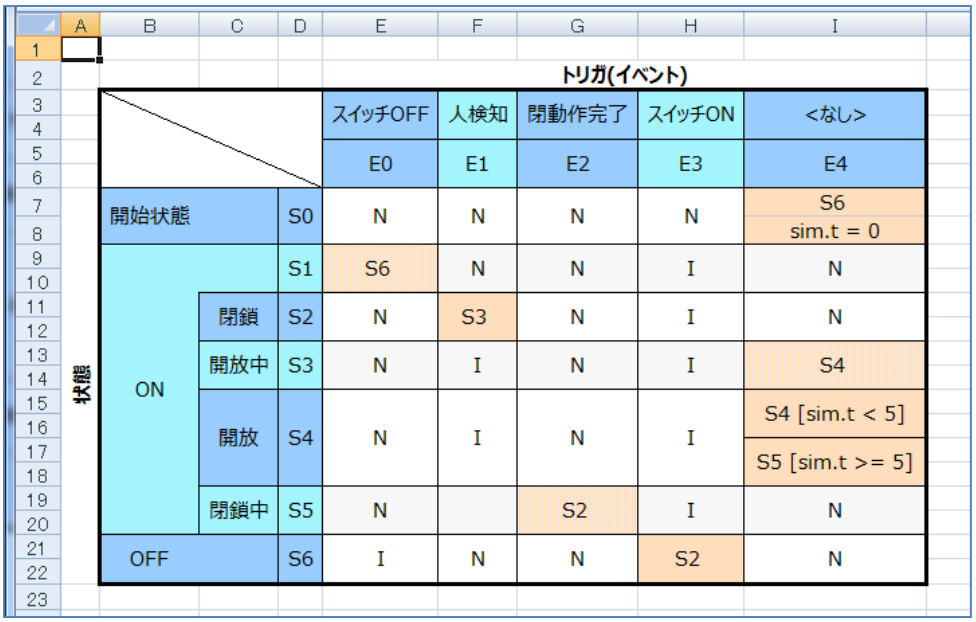
この形式にすることで、それぞれの状態において、それぞれのトリガ(イベント)が発生した場合の振る舞いを検証することができます。全てのセルに対して検証することで、トリガ(イベント)という観点からも全ての可能性を検証できます。

状態遷移表の場合には、それぞれのセルに対して検証したかどうかを、セル内に記入することで、検証済みなのか未検証なのかが判別できます。例えば、以下の例では、それぞれのセルに「N」(Never Happen/Never Allowed – 決して発生しない、発生した場合にはバグ(=assert などで対処))と「I」(Ignore – 発生しうるが何もしない)を記入しています。これにより、検証したかどうかが分かります。

さらに、未記入のセルが 1 つありますが、ここは遷移の漏れを見つけたものです。状態遷移表にして網羅的に検証することで、ステートマシン図では気づかなかった漏れや抜けを発見することもできます。

状態＼トリガ			スイッチOFF	人検知	閉動作完了	スイッチON	<なし>
			E0	E1	E2	E3	E4
開始状態		S0	N	N	N	N	S6
ON		S1	S6	N	N	I	N
ON	閉鎖	S2	N	S3	N	I	N
ON	開放中	S3	N	I	N	I	S4
ON	開放	S4	N	I	N	I	[sim.t >= 5] / S5 [sim.t < 5] / S4
ON	閉鎖中	S5	N	N	S2	I	N
OFF		S6	I	N	N	S2	N

なお、状態遷移表は CSV 形式で出力して Excel などの他のツールで加工して利用したり、画像として他のドキュメントなどで利用したりすることができます。また、無料のアドインを利用することで、下の画像のような Excel ファイルを生成することもできます。



状態			トリガ(イベント)				
			スイッチOFF	人検知	閉動作完了	スイッチON	<なし>
			E0	E1	E2	E3	E4
開始状態		S0	N	N	N	N	S6 sim.t = 0
ON		S1	S6	N	N	I	N
	閉鎖	S2	N	S3	N	I	N
	開放中	S3	N	I	N	I	S4
	開放	S4	N	I	N	I	S4 [sim.t < 5] S5 [sim.t >= 5]
	閉鎖中	S5	N		S2	I	N
OFF		S6	I	N	N	S2	N

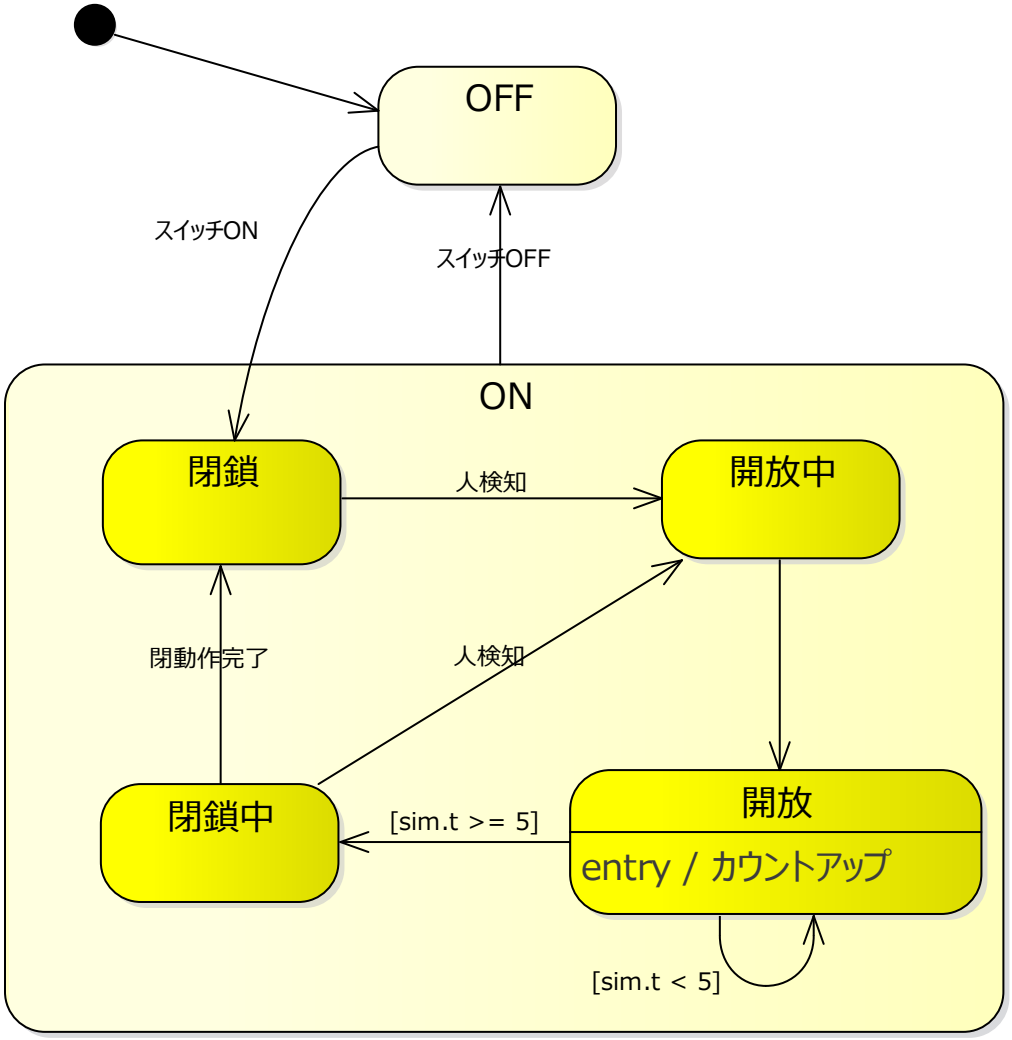
4. ステートマシン図と状態遷移表の連携

Enterprise Architect では、ステートマシン図と状態遷移表を簡単に切り替えることができます。ステートマシン図の背景で右クリックし、「表示形式の変更」から希望する形式を選択してください。

さらに、以下のような特徴があります。これにより、ステートマシン図と状態遷移表の同一性が保証され、どちらの形式で変更しても問題ないということになります。

- ✓ ステートマシン図で変更した内容は、自動的に状態遷移表に反映される
- ✓ 状態遷移表で変更した内容は、自動的にステートマシン図に反映される

今回の例では、状態「閉鎖中」からトリガ(イベント)「人検知」が発生した場合に、状態「開放中」に遷移するという遷移を追加しますが、ステートマシン図でも状態遷移表でも、どちらでも編集可能です。変更後の例は次のステートマシン図です。



さらに、状態遷移表では、以下のような操作が可能です。これにより、効率的に編集することができます。

- ✓ I および N キーで、それぞれの記号を入力可能
  (Delete キーで削除)
- ✓ 右クリックメニューで遷移の追加や削除が可能
- ✓ 既存の遷移は、ドラッグ&ドロップで他のセルに移動可能
  (遷移をトリガに関連づけることも可能)

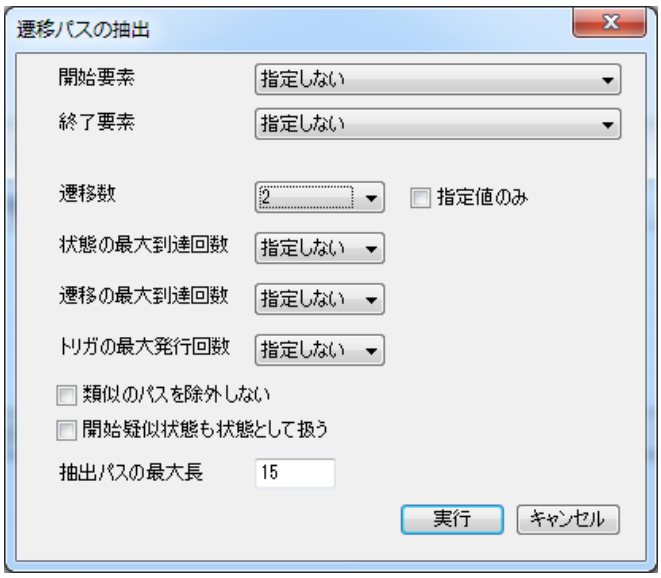
5. 遷移パスの抽出

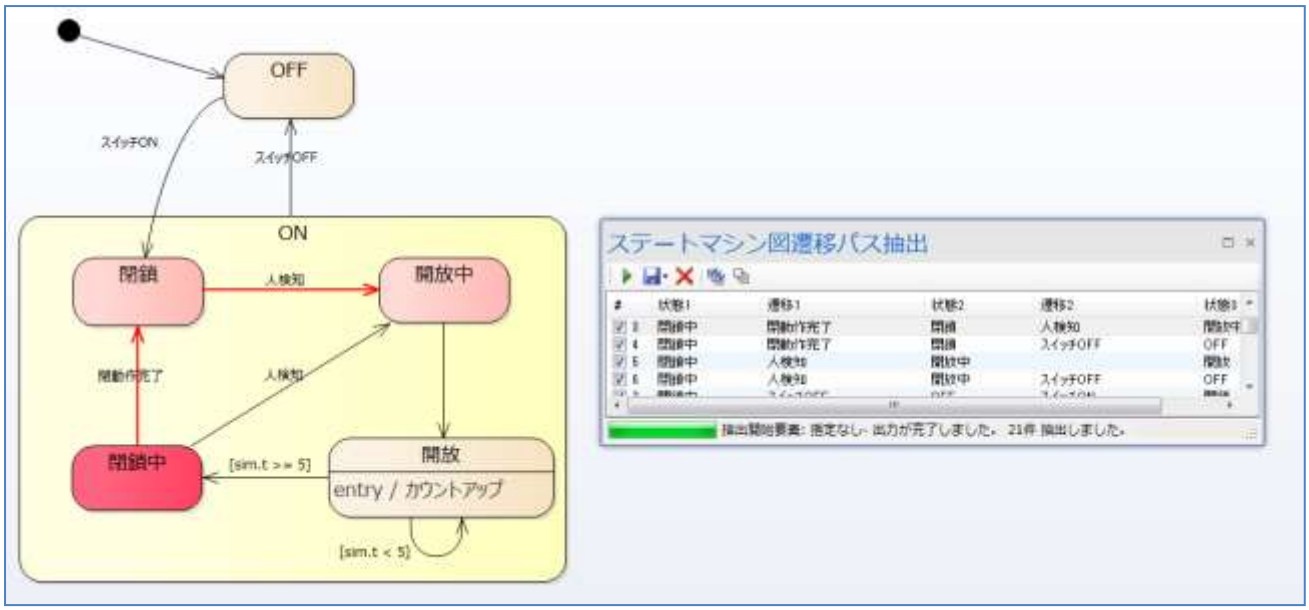
ステートマシン図と状態遷移表を組み合わせて、漏れや抜けのチェックを行うことができました。さ

らに検証するための別の方法として、実際に可能性のある遷移パスを全て抽出し、個別に確認するという方法があります。

遷移パスの抽出のためには、無料の「状態遷移パス抽出アドイン」を利用します。対象のステートマシン図を表示した状態で、アドインサブウィンドウに表示されるアドインのツールバーにある、実行ボタン(緑色の矢印ボタン)を押します。遷移数などの条件を指定後、抽出した結果が一覧で表示されます。一覧内の項目をクリックすると、その項目に関係するステートマシン図内の要素や接続のみが強調表示され、選択したパスを目で確認することができます。
(アドインサブウィンドウは、「アドイン・拡張」リボン内の「アドイン」パネルにある「ウィンドウ」ボタンを押すと開くことができます。)
パスの起点となる状態は濃い赤色で、それ以外の状態は薄いピンク色で表示されます。パスに関係する遷移は赤色で表示されます。

下の図は、遷移数 2 で抽出した場合の例です。「開放中」→「開放」→「OFF」という遷移を示しています。今回の例のように、入れ子になっているようなステートマシン図の場合には特に効果的です。





なお、抽出した内容は Excel や CSV ファイルとして保存できますので、テスト項目として利用したり他のツールの入力情報として活用したりすることができます。
(ある状態から別の状態への遷移についてはテストされることも多く、また十分考慮されていることが多いです。一方で、状態間の遷移の結果何らかの処理が行われる場合など、例えば状態 A→状態 B→状態 C のような場合で、状態 A→状態 B および状態 B→状態 C のそれぞれの単遷移では発生しない問題がある場合があります。このアドインの出力データを利用することで、このような複数遷移の場合を挙げることができます。)

「状態遷移パス抽出アドイン」についての詳細は、下記ページをご覧下さい。
http://www.sparxsystems.jp/products/EA/tech/StatePath.htm

6. ステートマシン図のシミュレーション

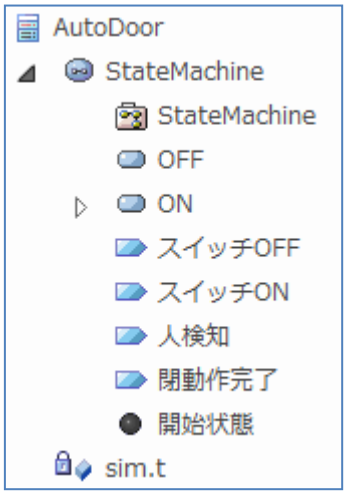
ここまで説明した方法により、ステートマシン図を網羅的に検証することを実現しました。さらに、ステートマシン図を動作させて、想定したとおりになっているかどうか確認することもできます。なお、このシミュレーション機能は、コーポレート版で利用できます。

シミュレーション機能を利用する場合には、ステートマシン図内において、JavaScript の文法でガード条件やアクション(処理)を記述することになります。遷移のガード条件とアクションのほか、状態の entry/exit アクションの振る舞いとして処理を記述することができます。

対象のステートマシン図を右クリックし、「シミュレーションの実行」→「自動実行」を選択することで、シミュレーションを実行できます。

なお、最終的にソースコード生成する場合には、プロジェクトブラウザの構成を決められた形にする必要があります。パッケージの下にそのままステートマシン図を配置した場合には、ソースコード生成は利用できません。プロジェクトブラウザの構成が、下の図左側のような構成になっている場合でもシミュレーションの実施は可能ですが、最終的にソースコード生成を行う場合には、下の図右のように、クラス要素の下に状態マシン要素が配置される形にすることが必須です。

(モデルの記述のみの構成)

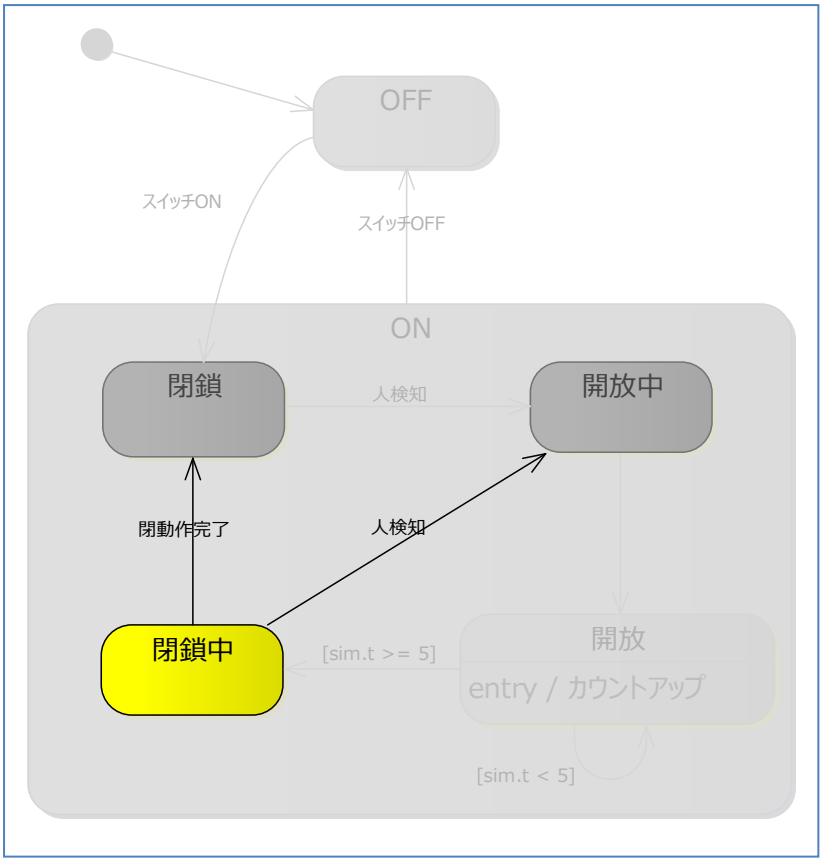


(コード生成可能な構成)

シミュレーションサブウィンドウとシミュレーショントリガサブウィンドウを利用することで、シミュレーションの制御や情報取得が可能です。シミュレーションサブウィンドウは、「シミュレーション」リボン内の「シミュレーション」パネルにある「ウィンドウ」ボタンを押すと表示できます。シミュレーショントリガサブウィンドウは、同じパネル内の「トリガ」ボタンで表示できます。

シミュレーションでは、状態にブレークポイントを設定し、ブレークポイントで停止した状態での変数の値を確認したり変更したりすることもできます。また、トリガが利用されている場合には、任意のタイミングでトリガを発行させて動作を変化させることができます。さらに、トリガの発行順序を保存して利用することもできます。
(これらの内容・操作の詳細は、ヘルプをご覧ください。)

シミュレーション機能は、実際に動作させることで問題を発見する手助けになります。例えば、今回の例では、内部で利用している、開放状態での待機タイマーのための変数の初期化処理が抜けているため、2 回目以降の「開放」ではすぐに「閉鎖中」に遷移してしまうという問題を、シミュレーション機能で発見することができます。



(実行中の一例)

なお、シミュレーション機能で利用する変数をローカル変数サブウィンドウに表示するためには、変数名が sim.あるいは this.で始まる必要があります。ステートマシン図のソースコード出力をする場合には、この接頭辞 sim.を削除するようにコード生成テンプレートをカスタマイズすると良いでしょう。

C 言語/C++言語用にスパークスシステムズ ジャパンが提供するカスタマイズテンプレートでは、この削除の処理も含まれています。このテンプレートについての詳細は、次の章をご覧ください。

シミュレーション機能のみを試したい場合には、Enterprise Architect のインストールディレクトリにある「Simulation_Sample.eap」ファイルを利用してください。
(通常は、アクセス権の関係でインストールディレクトリ内の EAP ファイルを開くことができません。このファイルを、デスクトップなどにコピーしてから開いてください。)

その他、シミュレーション機能の詳細については、Enterprise Architect のヘルプをご覧ください。

7. ステートマシン図のコード生成

Enterprise Architect Suite の各エディションでは、ステートマシン図などの振る舞い図からのコード生成も可能です。この機能を利用すると、状態遷移表やシミュレーション機能で検証した内容を、ソー

スコードとして出力することができます。

ソースコードを生成するためには、下図の構成のように、クラス要素の下に状態マシン要素を配置する必要があります。この状態で、クラス要素に対してソースコードの生成を実行することで、ステートマシン図の内容を含めた処理を出力することができます。

C 言語および C++言語の場合には、Enterprise Architect の標準の出力形式の他、スパークスシステムズ ジャパンでカスタマイズしたテンプレートを利用することもできます。いずれの場合にも、テンプレートの内容をそれぞれの部署・環境などに応じてさらにカスタマイズすることを想定しています。

参考: ステートマシン図から C 言語のソースコードを出力するサンプル
http://www.sparxsystems.jp/products/EA/tech/GenerateStateMachine.htm
参考: ステートマシン図から C++言語のソースコードを出力するサンプル
http://www.sparxsystems.jp/products/EA/tech/GenerateStateMachineCPP.htm

なお、上記のページで利用しているサンプルテンプレートでは、変数名やガード条件などで「sim.」を削除する処理を追加済みです。上記のプロジェクトブラウザの構成では、属性名 sim.t となっているものは、出力されるソースコードでは t となります。

また、下の例のように、モデル内で日本語を利用していると、その内容がそのままソースコードにも出力されます。ソースコード生成を行うことが前提であれば、モデルに記述する内容についてはそのプログラム言語で利用可能な形式・文字を利用する必要があります。

```
116 void AutoDoor::stateMachine_ON_開放(CommandType command, Event e)
117 {
118     switch(command)
119     {
120         case Do:
121         {
122             // Do Behaviors..
123             // State's Transitions
124             if ((t < 5))
125             {
126                 nextState = ST_StateMachine_ON_開放;
127             }
128             else
129             if ((t >= 5))
130             {
131                 nextState = ST_StateMachine_ON_閉鎖中;
132             }
133 
134             break;
135         }
136         case Entry:
137         {
138             //Entry Behaviors..
139             t++;
140             break;
141         }
142         default:
143         {
144             break;
145         }
146     }
147 }
```

8. まとめ

このドキュメントでは、状態遷移の設計をする際に、ステートマシン図だけでなく状態遷移表やシミュレーション機能などを利用して設計内容を検証する方法を紹介しました。さらに、ステートマシン図の内容をそのままソースコード生成することで、ステートマシン図の内容を実装する際の間違いを防ぎ、効率的に作業を行うことができます。

ステートマシン図からのコード生成の場合には、モデルの内容から自動的にソースコードを生成できる部分が大きいため、特に効果が大きいです。