ENTERPRISE ARCHITECT

Simulation Feature Guide

by Sparx Systems Japan

シミュレーション 機能ガイド

SPARX SYSTEMS

目次

1 はじめに

このドキュメントでは、コーポレート版以上のエディションで利用可能な、アクティビティ図とステートマシン図でのシミュレーション機能について概要と活用方法をお知らせいたします。プロフェッショナル版では、「手動実行」のシミュレーションのみが利用できます。「手動実行」のシミュレーションでは、遷移先が複数ある場合に、次に遷移する先をシミュレーションの実施者が選択・指定する方式であり、このドキュメントの対象とは異なります。

このドキュメントが対象としているシミュレーション機能は、過去のバージョンでは対応していない機能も含まれます。最新バージョンの Enterprise Architect でお試しください。

このドキュメントで説明する内容の他に、SysML のパラメトリック図などで記述した内容と OpenModelica や Simulink を連携させるシミュレーション機能もあります。この機能につきましては、PDF ドキュメント「SysML パラメトリック図のシミュレーション 機能ガイド」をご覧ください。

2 シミュレーション機能の概要

このドキュメントが対象とする Enterprise Architect のシミュレーション機能は、アクティビティ図・ステートマシン図が対象です。作成した図を実際に動作させることにより、モデルの内容が適切かどうかを検証することができます。また、シミュレーション実行中に任意のイベント(トリガ)を発行させることにより、モデル上で動作のテストを行えます。

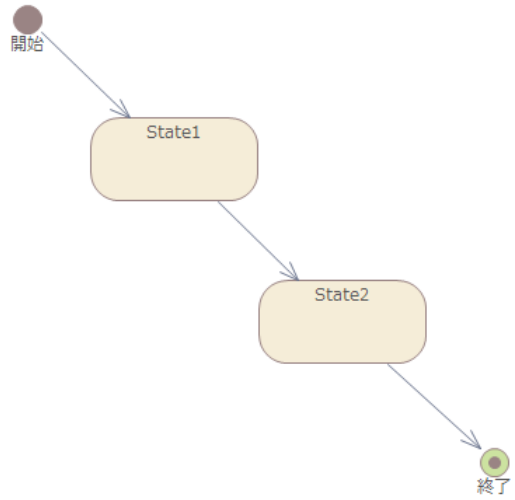
なお、このドキュメントの対象のシミュレーション機能はコーポレート版以上のエディションで利用できます。

このドキュメントでは、ステートマシン図を対象として説明します。アクティビティ図でも、ステートマシン図と同じ操作方法でシミュレーションが実行できます。シーケンス図でもシミュレーション機能の呼び出しが可能ですが、シグナルとの連携や複数の図の連携ができない等、実質的に「手動実行」のシミュレーションと変わりません。

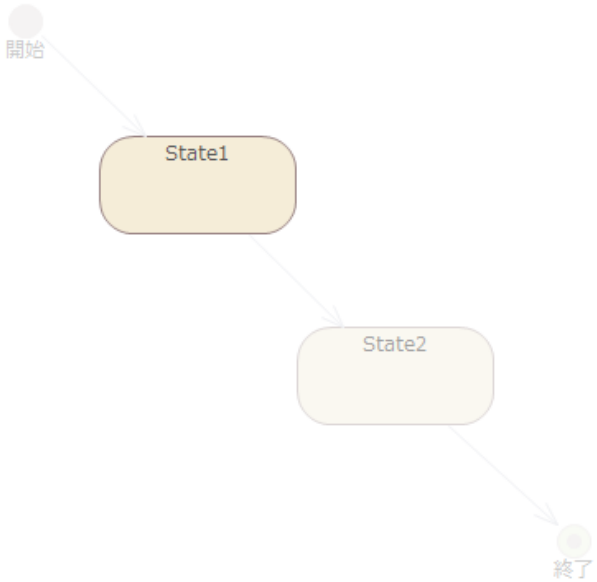
3 簡単なシミュレーション

まず、Enterprise Architect のシミュレーション機能を体験するために、簡単なステートマシン図を作成して実際に動作させます。

最初に、新規にステートマシン図を作成します。図の内容として、以下のような簡単なステートマシン図を作成します。
(基本的には、シミュレーションを実行するステートマシン図と、その図に配置する要素は同じパッケージ(あるいは親要素)に含まれていなければなりません。)

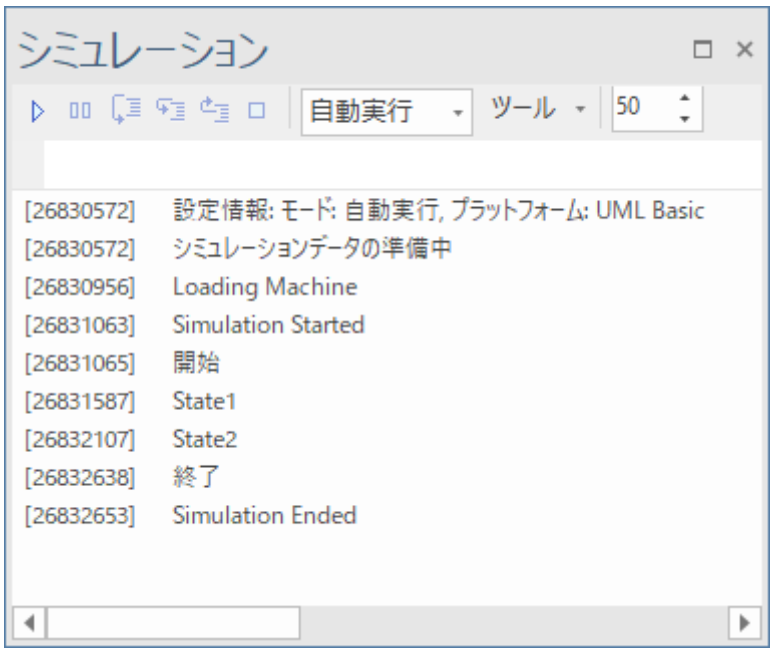


ステートマシン図を作成したら、ダイアグラムの背景で右クリックして「シミュレーションの実行」→「自動実行」を選択します。これにより、作成したモデルのシミュレーションが実行されます。この例では、State1 に移動し、その後 State2 に移動します。その次は終了状態ですので、動作が終了します。シミュレーション中は、現在実行中の位置のみが通常表示されます。また、次に遷移する可能性がある要素は薄い色で、それ以外の要素はさらに薄い色で表示されます。



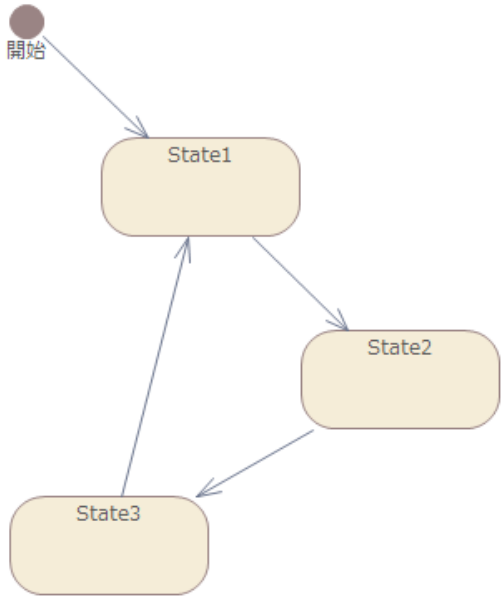
シミュレーションの実行速度は、シミュレーションサブウィンドウから変更できます。「シミュレーション」リボン内の「共通」パネルにある「ウィンドウ」ボタンを押すことで表示されます。シミュレーションの動作ログが表示

されますので、見える位置に配置しておくことをお勧めします。



シミュレーションの動作速度は、このサブウィンドウから設定できます。上の画像の例では、「50」に設定されています。「100」が最高速度で、「1」が最低速度です。「0」の場合には、このサブウィンドウのツールバーにあるボタンを押すことで、1 動作ずつ進む動作になります。

次に、ステートマシン図を変更し、以下のようにして実行します。このようにすることで、3 つの状態を無限に繰り返す事が確認できます。シミュレーションを実行中に停止させる場合には、シミュレーションサブウィンドウのボタンを押すか、「シミュレーション」リボン内の「実行」パネルにある「終了」ボタンを押してください。
(終了状態がある場合、終了状態に達すると自動的にシミュレーションの実行は終了します。)



なお、上記の図になるように変更を行った場合に、操作方法によっては、State2 で動作が停止する場合があるかもしれません。このような場合には、遷移の削除時に表示される次の画面で「接続を非表示」を選択しているか、あるいは終了要素を(モデルブラウザからではなく)ダイアグラムから削除したことが原因です。

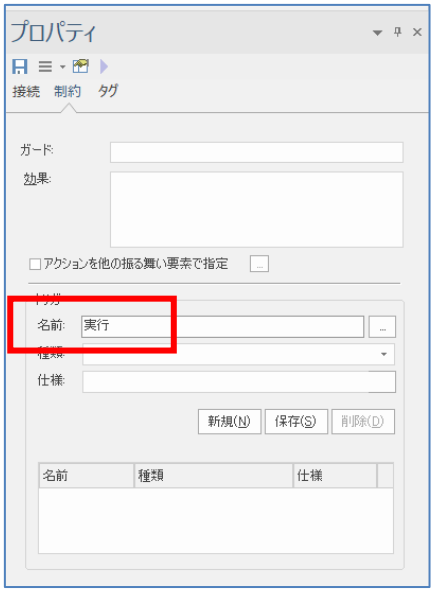
「接続を非表示」の場合には、見た目が非表示になっているだけで、モデルからは削除されていません。そのため、シミュレーション時の動作には引き続き影響を与えます。

同様に、ダイアグラム内の要素をダイアグラムから削除した場合には、モデルには該当の要素が残っています。シミュレーションはダイアグラム上の内容(表示されている内容)ではなく、モデルに対して実行されますので、ダイアグラムから要素を削除した場合には、意図した通りに動作しない場合があります。

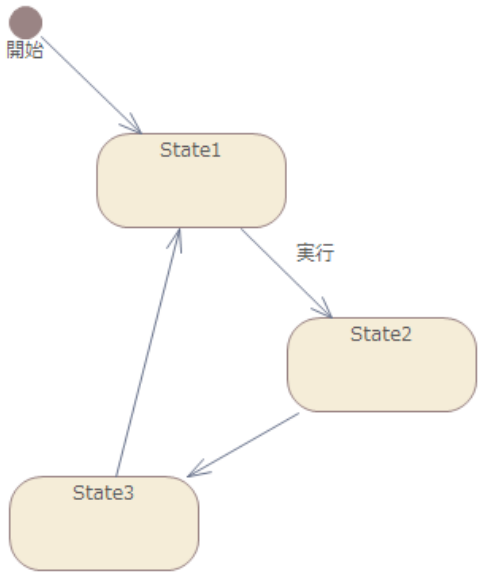
この要素や接続の削除など、基本的な操作・考え方は「Enterprise Architect 入門セミナー」で説明しています。上記の説明がわからない場合には、先にセミナーを受講してください。毎月開催しています。サポートが有効であれば無料で参加できます。

1 つの状態から複数の遷移が可能な場合には、シミュレーションは停止します。複数の遷移がある場合で、遷移のガード条件が指定されていない遷移が 1 つのみある場合には、この遷移は、いわゆる"else"文に該当します。つまり、遷移が複数存在し、1 つを除きガード条件が指定されている場合には、他の全ての遷移の条件を満たさない場合に限り、ガード条件の指定のない遷移をたどります。

上記の 3 状態の例に対して、さらに、State1 から State2 への遷移を選択して、プロパティサブウィンドウの「制約」タブからトリガ(イベント)を追加します。トリガの種類は、「シグナル」を指定した場合には後述のシグナル送信の方法でトリガを発行できますが、それ以外の種類の場合や未指定の場合には全て同じ動作になります。

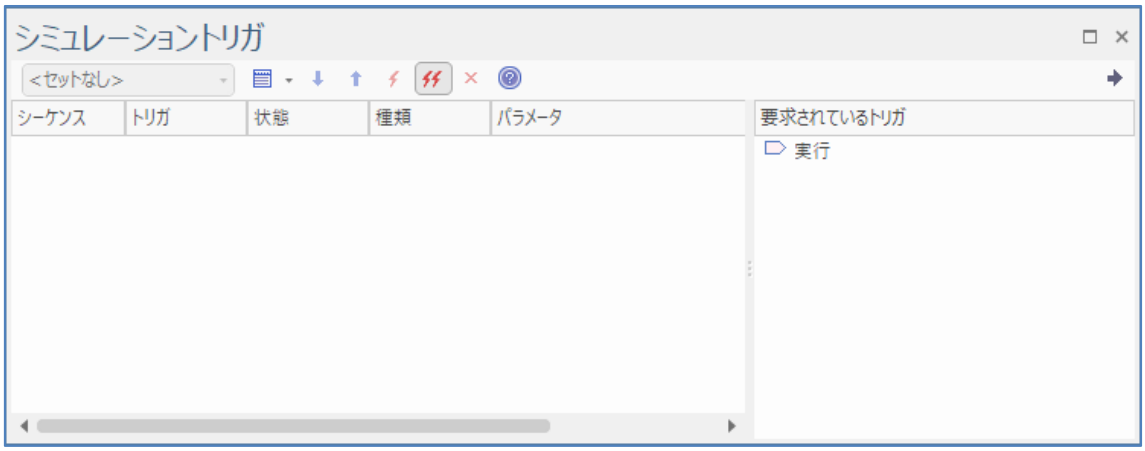


このようにすると、シミュレーション時に State1 で動作が停止し、トリガ「実行」を待機する状態になります。

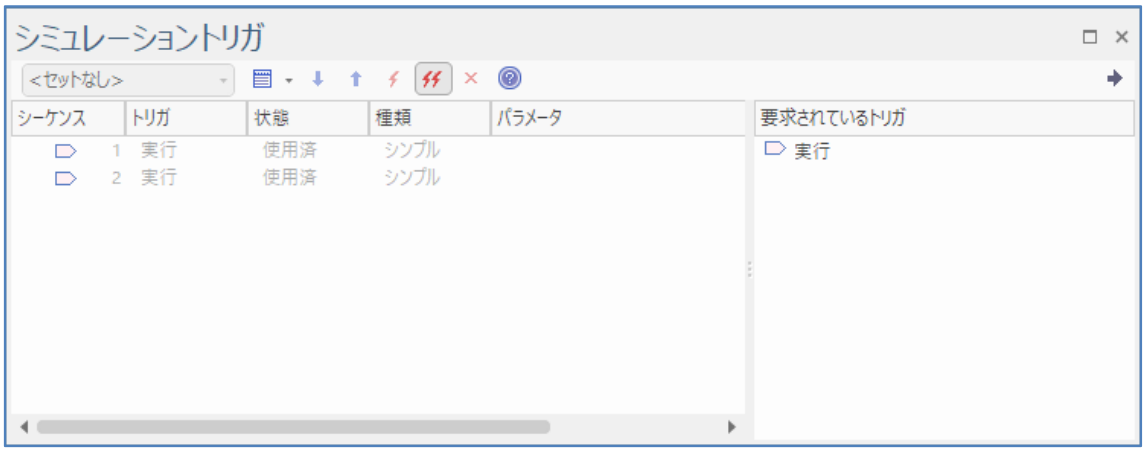


実行時にトリガを発生させるためには、「シミュレーショントリガ」サブウィンドウを利用します。「シミュレーション」リボン内の「共通」パネルにある「トリガ」ボタンを押すとサブウィンドウが表示されます。表示されるサブ

ウィンドウの「要求されているトリガ」欄に、待機しているトリガが表示されます。このトリガをダブルクリックすることで、シミュレーション中の動作に影響を与えられます。

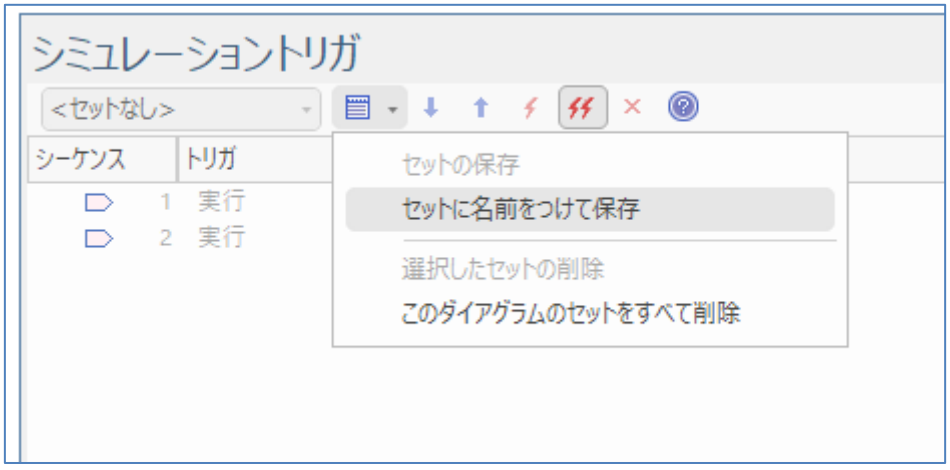


実行したトリガは、サブウィンドウの左側の一覧に追加されます。下の例は、「実行」のトリガを 2 回発行した例です。



「要求されているトリガ」に表示されていないトリガを明示的に発行したい場合には、モデルブラウザから対象のトリガ要素をドラッグし、このサブウィンドウ左側の一覧にドロップしてください。一覧に追加されます。一覧に追加されたトリガをダブルクリックすることで、そのトリガを発行できます。

このようにして、シミュレーション中に任意のトリガを任意の順序で発行できます。発行したトリガの履歴は、保存しておき再利用できます。以下のように、「セットに名前をつけて保存」することで、後から利用するための「セット」を作成できます。



この、トリガのセットの機能を利用することで、以下のように設計時のテストを効率よく行えます。

1. ステートマシン図を作成する。
2. 作成したステートマシン図のシミュレーションを実行する。トリガを(テストシナリオに沿って)明示的に発行し、動作を確認する。
3. シミュレーション終了後、トリガの発行内容・順序を「セット」として保存する。
4. 必要に応じて異なるトリガの発行内容で「セット」を作成する。
5. ステートマシン図を修正する。
6. 修正後は、「セット」を呼び出して再度シミュレーションを実行し、期待した結果になるかどうか確認する。以下、期待する結果になるまで、5 と 6 を繰り返す。

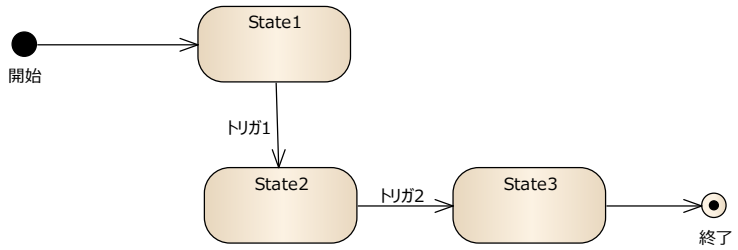
なお、「セット」は、ダイアグラムに関連づけて保存されます。作成したセットは他のダイアグラムでは利用できません。

Enterprise Architect のトリガ(イベント)には、以下の 4 つの状態があります。

- 未発行
- 発行
- 使用済
- 消失

「未発行」はトリガ要素をモデルブラウザからドラッグ&ドロップした直後や、定義済のセットを選択した直後のトリガの状態です。この状態では、トリガは有効ではありません。ダブルクリックして「発行」状態にすることで、トリガが有効になります。「発行」状態の時に、そのトリガを必要としている遷移がある場合には使用され、「使用済」になります。

なお、例えば以下の例で、State1 の状態で停止しているときに「トリガ 2」を手動で「発行」した場合には、トリガ 2 は「発行」状態のままとなります。



その後、トリガ 1 を「発行」すると、そのトリガ 1 は State1 から出る遷移ですぐに使用されて「使用済」になり、State2 に移動します。State2 では「トリガ 2」が必要ですが、これは「発行」状態にありますので、すぐに State3 に遷移します。

つまり、トリガを都度発行する(手動で発行する)場合には、トリガを発行した順序とは関係なく、要求されているものから順番に使用されていきます。発行されたトリガは、使用されるまで発行状態のままで待機します。この挙動は、想定しているものと異なるかもしれません。

一方で、「セット」の場合には、この挙動が変わります。上の例と同じく、最初に「トリガ 2」を発行し、次に「トリガ 1」を発行するセットを定義し、実行します。この場合には、次の図のような結果になります。

実行すると、State1 の時点で「トリガ 2」が発行されますが、このトリガ 2 は要求されていません。セットを利用して自動シミュレーションする場合には、このトリガ 2 は「消失」状態になり、次の「トリガ 1」が発行・使用されます。

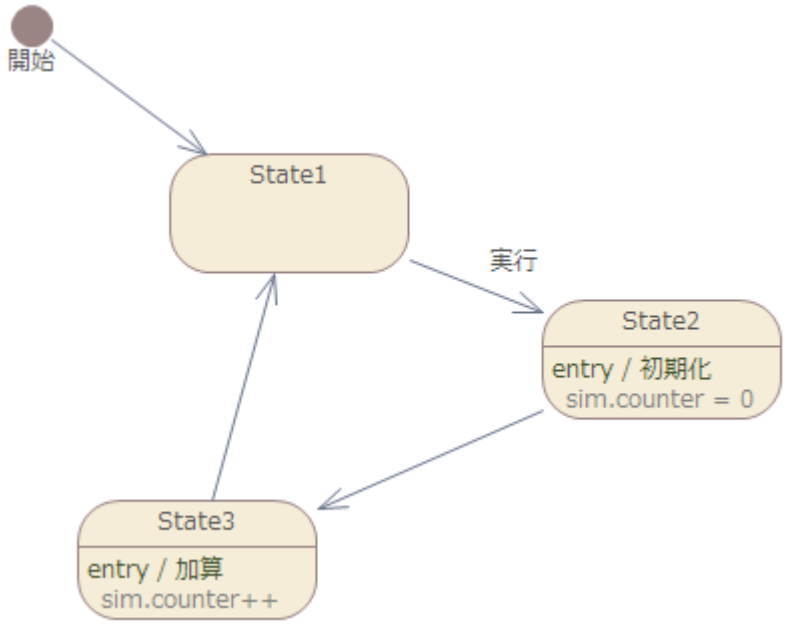
このように、不要なトリガを無視する必要がある場合には、セットを定義してシミュレーションを実行してください。この方法以外に、モデルの作成内容で対策する方法もあります。7 章のサンプルをご覧ください。

4 モデル内に処理を埋め込む

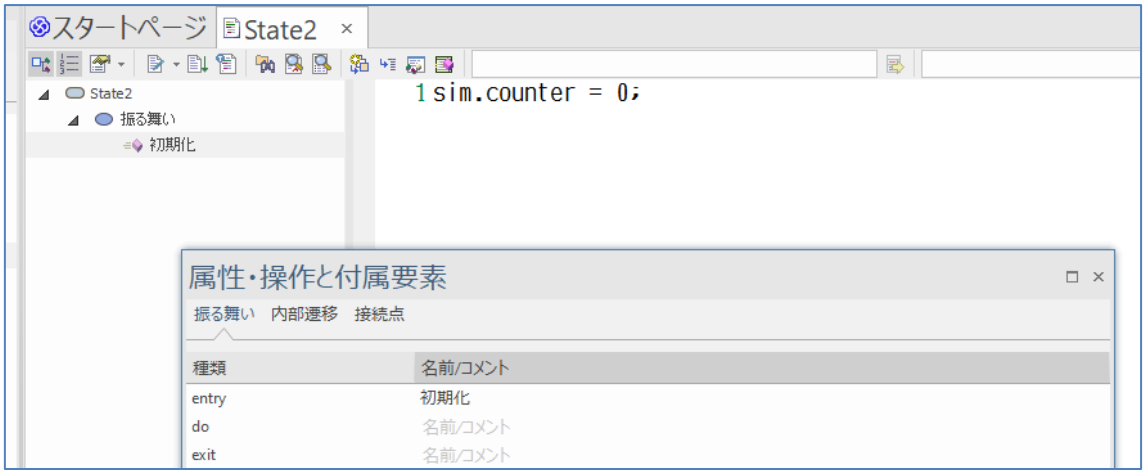
ステートマシン図のシミュレーション機能では、状態の entry/exit アクションや遷移のアクションに処理を埋め込めます。この処理は JavaScript 形式で記述し、シミュレーション実行時に Enterprise Architect が内部に持つ JavaScript エンジンで記述内容を解釈し、実行します。また、遷移のガード条件には、JavaScript 形式で条件を指定し、処理を分岐できます。

entry アクションは、ある状態に遷移してきたときに実行されます。exit アクションは、その状態から別の要素に遷移する場合に実行されます。このシミュレーションでは do アクションは利用しません。

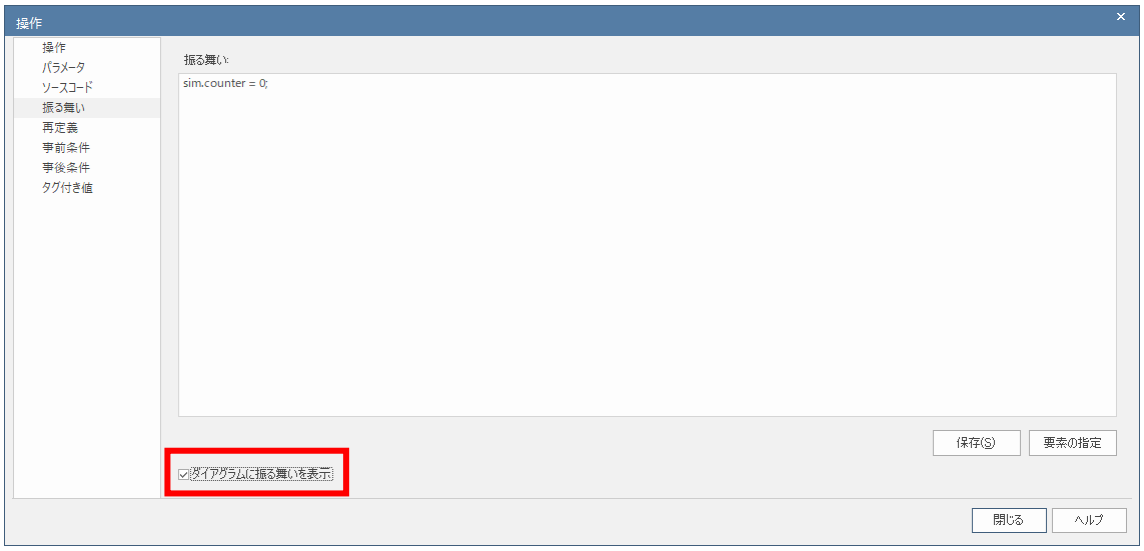
例として、先ほどの内容に、状態要素の entry アクションを追加します。なお、以下の図では、アクションの処理内容がわかりやすくなるように、処理内容をダイアグラムに表示するように設定しています。(設定方法は後述します。)



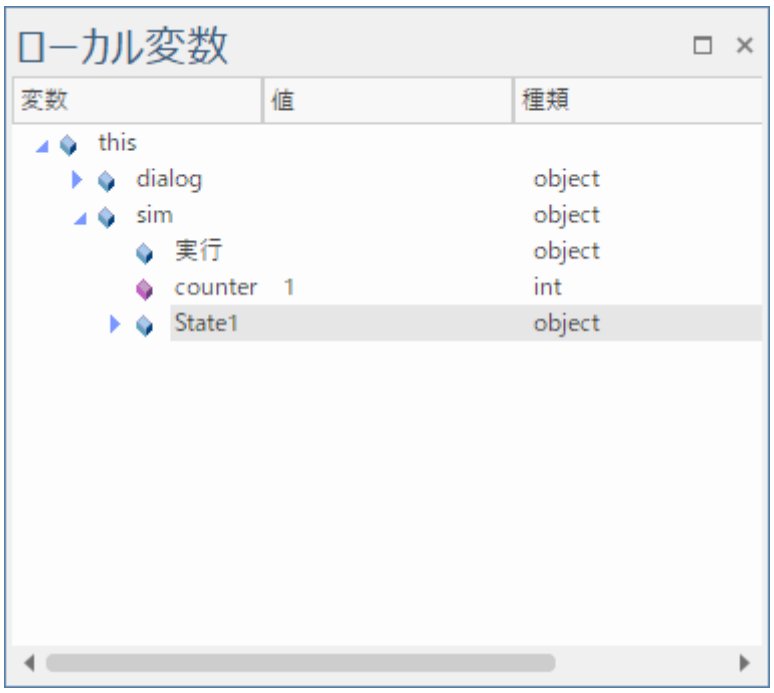
状態要素の entry/ exit のアクションは、Enterprise Architect では操作として定義されます。アクションを追加するには状態要素を右クリックして「属性・操作と付属要素」→「操作」を選択して属性・操作と付属要素サブウィンドウを表示し、追加します。実際の処理の内容は、「コードエディタ」を利用して記述します。「コード」リボン内の「ソースコード」パネルにある「コードエディタ」ボタンを押し、表示されるメニューから「振る舞いの編集」を選択してください。(あるいは、次に説明するプロパティダイアログを利用してください。)



入力した振る舞いの内容を、ダイアグラム内で表示できます。現在は、属性・操作と付属要素サブウィンドウ内の対象の振る舞いをダブルクリックすると表示されるダイアログの「振る舞い」タブにある「ダイアグラムに振る舞いを表示」のチェックボックスにチェックを入れ、「保存」ボタンを押すことで、表示できます。

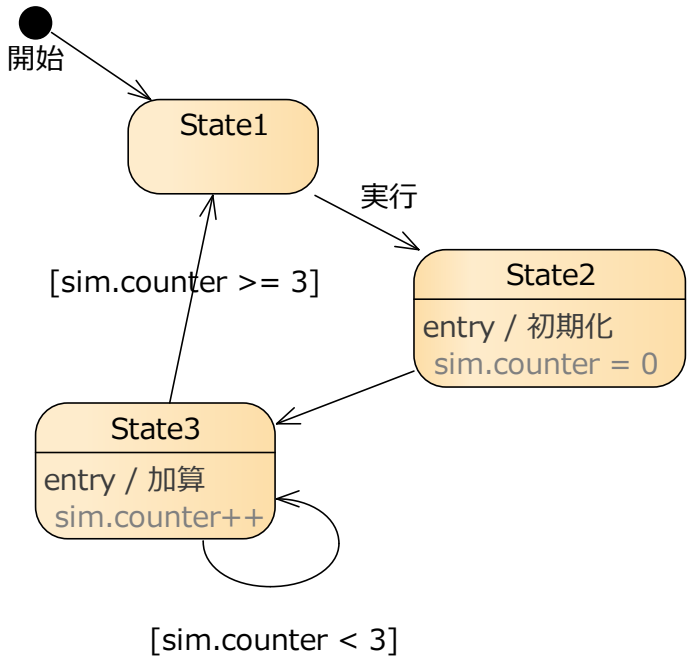


なお、シミュレーション機能で変数を利用する場合には、上の例のように、接頭辞(プレフィックス)として「sim.」(あるいは「this.」)を追加します。これらの接頭辞を設定すると、シミュレーションの動作が停止しているときに、ローカル変数サブウィンドウで変数の値を確認できます。下の例は、シミュレーションを実行後、1 回「実行」のトリガを発行した後の状態です。



変数の値は、シミュレーションサブウィンドウ内の上部にあるコンソール領域で変更できます。

また、遷移のガード条件でも変数を利用できます。次の例では、変数 sim.counter の値を見て、処理を分岐しています。ガード条件には式が入力可能で、true になる場合にはその条件を満たしたと判断されます。



sim を接頭辞に持つ変数の場合には、グローバルな変数として、現在使われているステートマシンとその子要素のなかで自由に利用できます。this を接頭辞に持つ変数の場合には、そのステートマシン図を持つクラスの属性を参照することになります。

この処理の追加機能とトリガの機能を利用することで、ステートマシン図に論理的処理を埋め込み、外部からトリガを与えてシミュレーションを実行できます。

5 シミュレーション機能の補足

下記のサブウィンドウは、いずれも「シミュレーション」リボン内の「共通」パネルにあるボタンから開けます。

コールスタックの表示

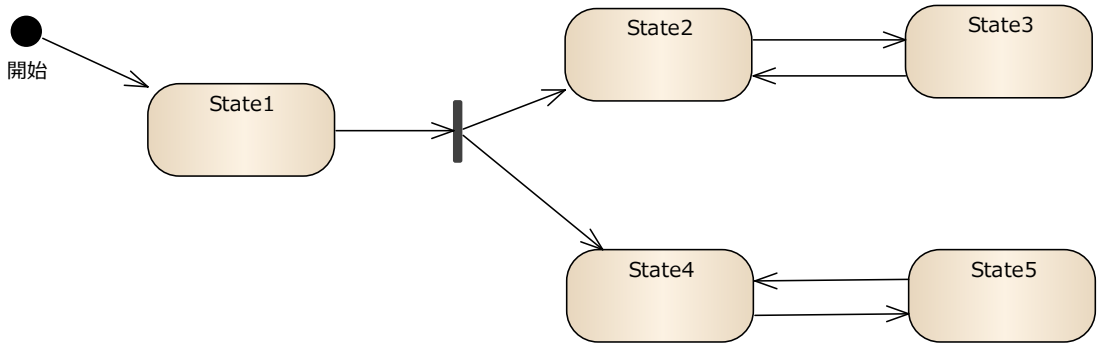
シミュレーションをデバッグするには、コールスタックの状態をシミュレーションの実行中に表示することが重要です。コールスタックサブウィンドウから状態を表示できます。コールスタックウィンドウはマルチスレッドの並行シミュレーションをチェックするのに便利です。

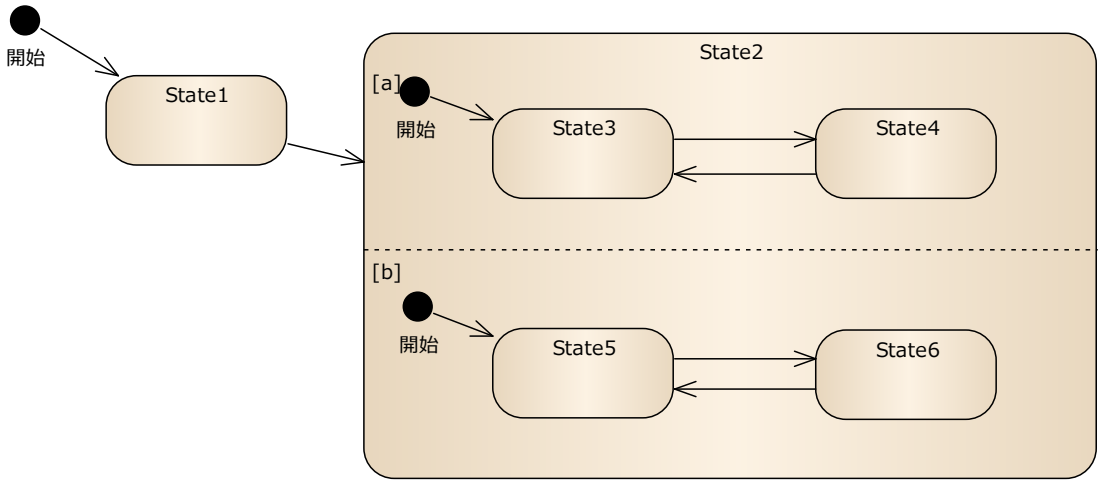
ブレークポイント

ブレークポイントを使ってシミュレーションをデバッグできます。モデルブラウザからブレークポイントウィンドウに要素をドラッグすると、その要素に対してブレークポイントを設定できます。ブレークポイントに停止すると、変数の値の確認や変更が可能になります。
(ローカル変数サブウィンドウは、シミュレーションの動作中には値は更新されません。)

6 複数の図を同時に動かす

シミュレーションでは、複数の処理を同時に実行する場合でも実行できます。ステートマシン図で 1 つの図の中で複数の処理を実行する場合には、以下のようにフォーク/ジョイン要素を利用するか、状態内に領域を定義します。

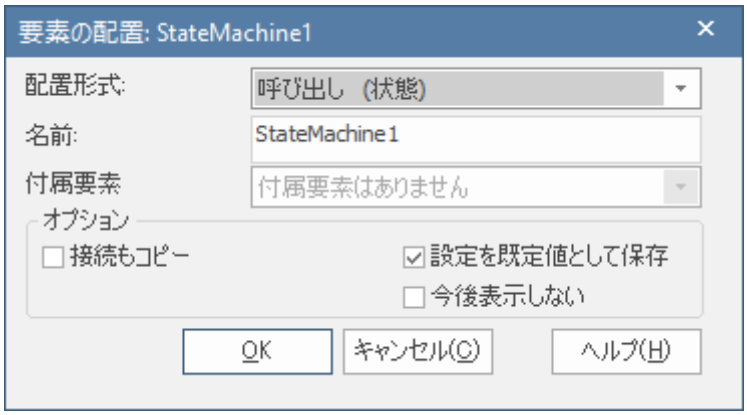




(状態内に領域を作成する場合には、それぞれの領域に異なる名前を設定してください。)

最終的にソースコードの生成を行う場合には、複数の状態マシン要素(内のステートマシン図)に対してシミュレーションを行う必要があるかもしれません。このような場合には、以下の手順で複数の図のシミュレーションを実行できます。

1. 状態マシン要素を作成し、その中に含まれるステートマシン図に、状態遷移を記述します。必要に応じて複数作成します。
2. 新規にステートマシン図を 1 つ作成します。
3. 作成したステートマシン図に、モデルブラウザから手順 1 で作成した状態マシン要素をドラッグして配置します。「要素の配置」画面では、配置形式として「呼び出し(状態)」を選択します。



並列実行させたい状態マシン要素を、同様にして全て配置します。

4. ステートマシン図に、開始状態を配置します。
5. 開始状態の先にフォーク/ジョイン要素を作成して遷移で結びます。さらに、フォーク/ジョイン要素からそれぞれの状態要素に遷移を結びます。
6. それぞれのステートマシン図を開き、ダイアグラムのタブをドラッグすると独立したウィンドウとしてステートマシン図を表示できます。この方法を利用して、複数のステートマシン図を並べて表示できます。

以上で、複数のステートマシン図の同時シミュレーションが可能になります。

なお、上記の手順のように、複数の異なるダイアグラムを連携させてシミュレーションする場合には、その対象のダイアグラムを子ダイアグラムとして保持する状態マシン要素を作成し、その要素を配置・利用する必要があります。

6.1 イベントの発行

上記のような複数のステートマシン図を同時に実行する場合には、あるステートマシン図の遷移や

entry/exit アクション内でトリガを発生させ、他のステートマシン図に影響を与えるようなことが可能です。このためには、Enterprise Architect のシミュレーションでの独自の関数 BroadcastSignal を利用します。

BroadcastSignal は、名前の通りシグナルを発生させます。シグナルが発生すると、そのシグナルに関連づけられたトリガが発行され、シミュレーション中のすべてのステートマシン図に影響を与えます。(そのトリガを要求していないステートマシン図では無視されます。)
もし、シグナルにパラメータが設定されていて、実行時にパラメータの値を指定する画面を表示したい場合には、BroadcastSignal の代わりに UIBroadcastSignal を利用します。

シグナルに関連づけるトリガを作成するには、まず、シグナル要素を作成する必要があります。クラス図のツールボックスの「クラス」グループあるいはステートマシン図のツールボックスの「シミュレーション要素」グループ内に「シグナル」要素が含まれますので、この要素を図に配置して作成します。

その後、状態要素間に遷移を追加し、その遷移のプロパティ画面からトリガを新規に作成し、その際にトリガの種類を「シグナル」に設定します。シグナルに設定すると、関係するシグナル要素を指定する画面が表示されますので、作成したシグナル要素を指定します。
あとは、BroadcastSignal 関数の引数として、シグナルの名前を指定すれば、シミュレーション中にそのシグナルが発行されます。BroadcastSignal の引数として渡す文字列は、トリガの名前ではなく、シグナルの名前である事に注意して下さい。

シグナルにパラメータがある場合には、パラメータを指定して発行できます。この場合には、UIBroadcastSignal 関数を利用します。例として、以下のようなシグナルについてのトリガを発行する場面を考えます。

この場合には、time というパラメータに、任意の値を渡して発行できます。その際には、以下のように記述します。

```
UIBroadcastSignal("START",{'time': 3 });
```

このように、第 2 引数に、{'パラメータ名': 値} の形式で入力します。パラメータが複数ある場合には、

カンマ区切りで指定し、第 2 引数全体を{と}で囲みます。

パラメータの値を参照するには、this.START.time のように、this.シグナル名.パラメータ名 として参照できます。このようにして参照できる値は、ガード条件などで活用できます。

なお、このシミュレーション機能としてはトリガの種類として「シグナル」以外も選択できますが、BroadcastSignal および UIBroadcastSignal では「シグナル」トリガのみが利用できます。

6.2 動作状況をガード条件にする

ガード条件として、シミュレーションである特定の状態であるかどうか、という判定を行うことが可能です。このための関数として IS_IN が用意されています。複数の図を同時に動作している場合に利用します。

例えば、ある遷移の条件として IS_IN(“State1”)と記述した場合、他のステートマシン図で、実行中の状態が State1 の場合に、True(真)と判断され、遷移条件を満たすことになります。

6.3 シミュレーション時間についての補足

Enterprise Architect のシミュレーションは、実時間(処理時間)に関係なく、シミュレーション内部に適用される「シミュレーション時間」が基準になります。

この 1 シミュレーション時間ごとに、1 つの処理が行われます。マルチスレッドの場合、すべてのスレッドについての 1 つの処理が終了しますと、「1 シミュレーション時間」が経過したことになり次の処理に移ります。

この「1 シミュレーション時間」は、処理が完了するまでが単位です。COM オブジェクトのメソッドの呼び出しを行う場合、その呼び出しの処理が完了して EA に戻るまでの時間が「1 シミュレーション時間」になります。

ですので、例えば、COM オブジェクトのメソッドの処理が 10 分かかる場合には、その時の「1 シミュレーション時間」は 10 分になるということです。
(その次の処理がすぐに終わる場合は、次の「1 シミュレーション時間」は 1 秒未満になることもあります。)
つまり、「1 シミュレーション時間」は、人間の時間とは対応せず、一定ではありません。

このように処理に時間がかかる場合には、COM オブジェクトでの処理は非同期にしてすぐに次のシミュレ

ーション時間に進むようにしなければなりません。また、コールバックの仕組みはありませんので、非同期で実行している処理を確認するようなメソッドの呼び出しを、別のアクション要素から行うようにする必要があるかもしれません。

なお、マルチスレッドの場合、処理が実行される実行順序は不定です。順序や優先度を指定することもできません。マルチスレッドの場合、処理の便宜上、それぞれのスレッドについて順次処理を行っていますが、理論的には「1 シミュレーション時間」で同時に行われている処理になります。ですので、実行順序に依存するような処理が記述されることは想定していません。

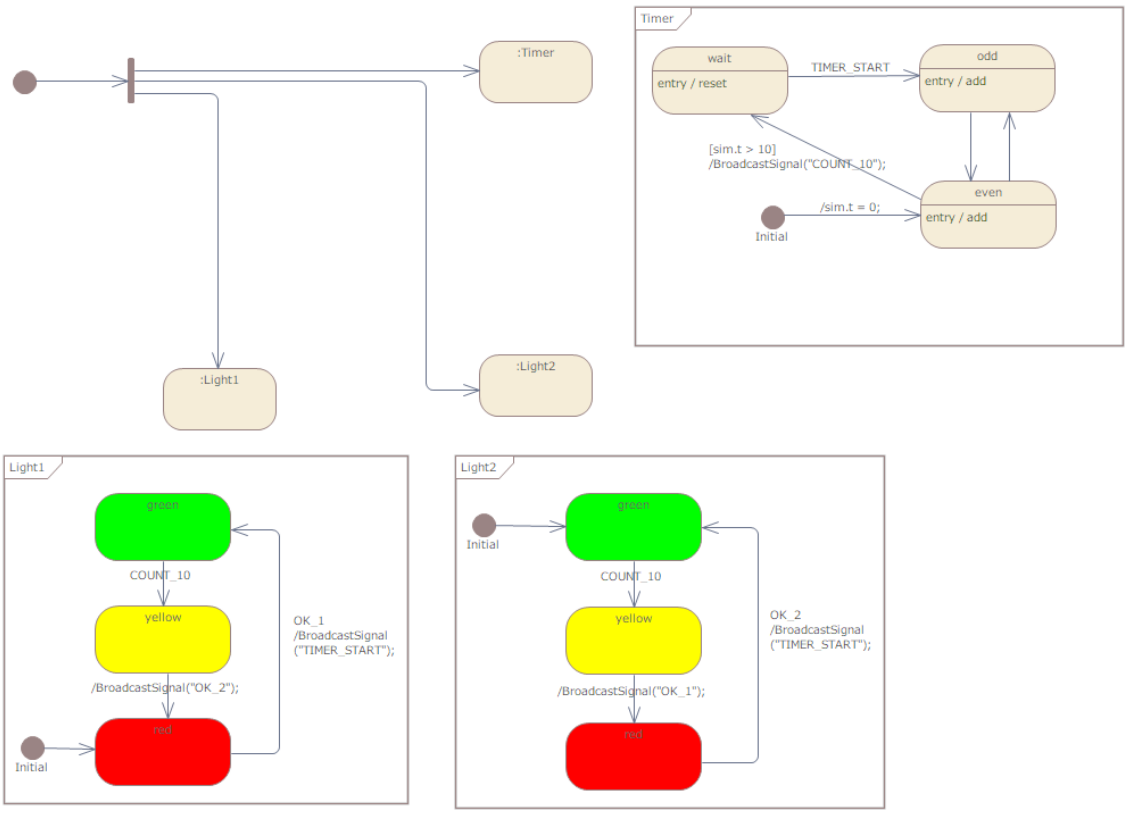
7 サンプル

Enterprise Architect のサンプルプロジェクトには、いくつかのサンプルモデルが含まれています。サンプルは、「ホーム」リボン内の「ヘルプ」パネルにある「ヘルプ」ボタンを押して表示されるメニューから、「サンプルプロジェクトを開く」を実行してください。ファイルが開くとダイアグラムが自動的に表示されますので、「シミュレーション」→「ステートマシン図のサンプル」とダブルクリックして移動してください。

以下、追加の情報を記載します。

7.1 複数のステートマシン図

この例は、サンプルプロジェクトに含まれています。3 つのステートマシン図を同時に実行していますが、それぞれのステートマシン図間でシグナル(トリガ)を投げ合っている例です。

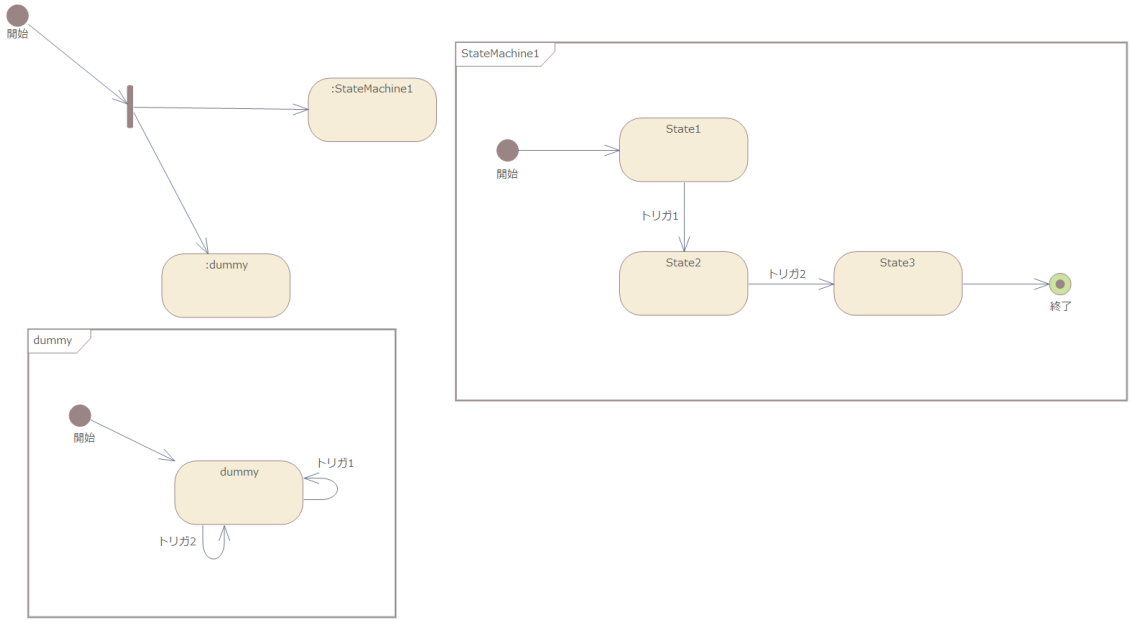


それぞれのステートマシン図は、状態マシン要素の子ダイアグラムとして作成し、その状態マシン要素をモデルブラウザからドロップして「呼び出し(状態)」として配置します。

7.2 処理しないシグナルを無視する

3 章にて前述しましたように、Enterprise Architect のイベント(トリガ)は、セットを利用せず都度トリガを発行する場合には、そのイベントを要求していない状況では「発行」状態となり、次に要求されるまで待機します。この結果、トリガの発行順序と、使用される順序が異なる場合があります。

このような状況を避けるためには、すべてのトリガを要求するダミーのステートマシン図を 1 つ作成、マルチスレッドで実行するような方法があります。3 章の例では、以下のように構成することで、発行されたトリガはすぐに使用され、トリガの発行順序と使用順序が常に一致するようになります。



8 アクティビティ図に関連する補足

アクティビティ図の場合には、以下のような追加機能が利用できます。

8.1 待機時間の設定

アクティビティ要素やアクション要素に対して、タグ付き値「duration」を追加し、その値を数値で指定します。この場合には、指定した時間(ステップ数)だけ、処理を停止します。

例えば、以下のようにタグ付きを設定した場合には、5ステップ分の間、該当のアクション要素で停止します。

8.2 時間イベント受信アクションの利用

時間イベント受信アクションを利用すると、指定した時間だけ待機し、待機後に処理を継続することができます。

時間イベント受信アクション要素を作成する場合には、ツールボックスから「アクション」をドラッグし、ドロップしたときに表示されるメニューで「時間イベント受信」を選択してください。
(ドロップしたときにメニューが表示されない場合には、Ctrl キーを押しながらドロップしてください。)

Action2

作成した要素のプロパティサブウィンドウの「トリガ」タブで、条件を指定します。トリガの種類を「時間」に、「仕様」の欄に、数値が返るような式(あるいは固定値)を指定します。

上の例の場合、変数 sim.t の値の 2 倍の時間だけ待機し、その後処理を継続します。このように、時間イベント受信アクションを利用すると、シミュレーション変数も利用して待機時間を指定できます。

8.3 イベントの発行

アクティビティ中にシグナル送信アクション要素を配置し、プロパティ画面で対象のシグナルを指定すると、シミュレーション実行中にシグナルを発行できます。この場合に、そのシグナルに関連づけられたトリガを待っ

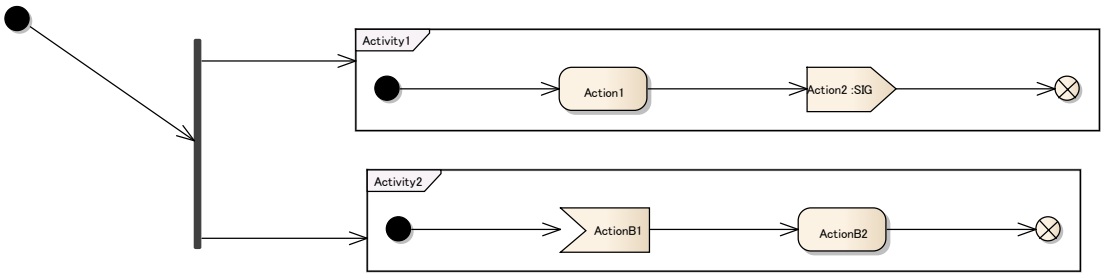
ているステートマシン図がある場合、そのステートマシン図でトリガが発行される形になります。
これにより、トリガの送信順序をアクティビティ図で定義し、アクティビティ図の内容を元にステートマシンを駆動するシミュレーションを実行できます。

具体的には、アクティビティ図のツールボックスにある「アクション」要素をダイアグラム内に配置する際に表示されるメニューで、「シグナル送信」を選択してください。すると、該当のアクション要素のプロパティサブウィンドウには「条件」タブが表示されますので、トリガと関連づけたシグナル要素を指定してください。

なお、ステートマシン図側で、シグナルに関連づけられたトリガを作成する手順については、次の通りです。

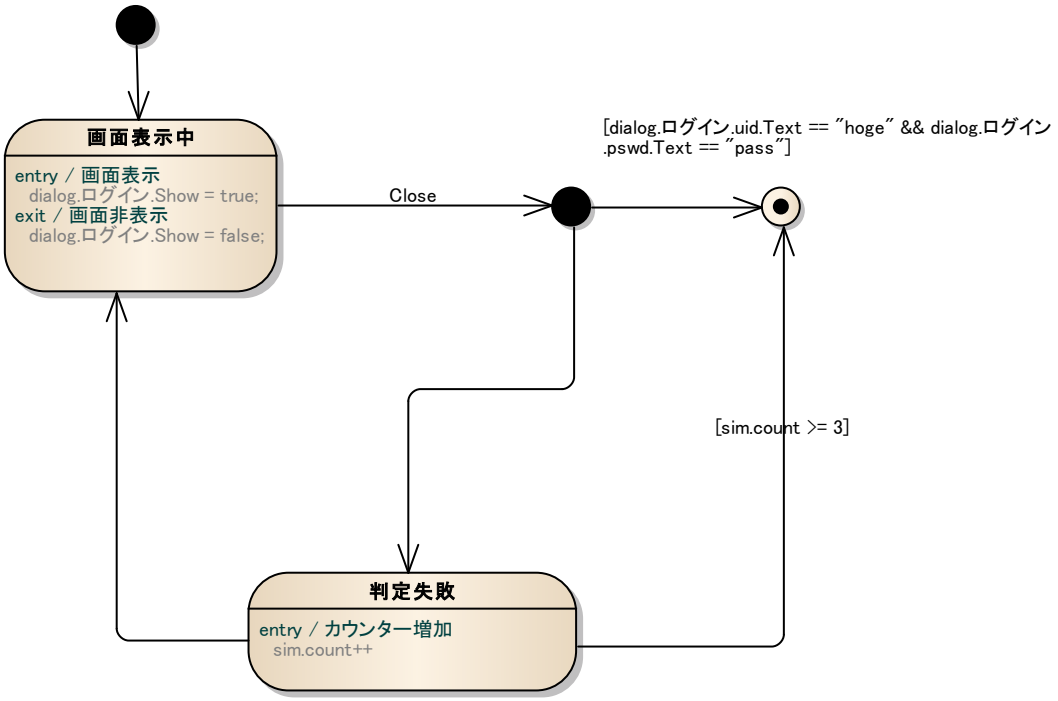
1. シグナル要素が存在しない場合、新規に作成します。
2. トリガのプロパティ画面の「トリガ」グループで、種類を「シグナル」に設定し、「仕様」の欄に既存のシグナル要素を指定します。
3. 対象のトリガを遷移に関連づけます。

アクティビティ図で、特定のイベントを受信できます。「シグナル送信」と同様の方法で、「イベント受信」のアクション要素を作成します。このアクション要素について、待機するシグナルをプロパティ画面から設定することで、イベントを受信するまで待機し、イベントを受信したら処理を継続するようなモデルを作成できます。
(イベント受信のアクション要素には、シグナルではなく、シグナルに関連づけられたトリガ要素を結びつける必要がある点にご注意下さい。)



9 画面設計との連携についての補足

Enterprise Architect のシミュレーション機能では、画面設計で作成した画面とシミュレーション機能を連動させられます。一例として、以下のようなモデルを考えます。



このようなステートマシン図と、以下のように「Win32 画面設計」で定義した画面を組み合わせてシミュレーションを実行できます。

「画面表示中」の状態の entry アクション「画面表示」の振る舞いとして、「dialog.ログイン.Show = true;」と定義しています。このように、「dialog.(モデル内の画面名).Show=true/false;」で、別途定義した画面を表示したり非表示にしたりできます。

画面のボタンを押した場合の処理を記述できますので、BroadcastSignal で Close のイベントを発行します。すると、「画面表示中」の状態から抜け、その過程で exit アクションが実行され、画面が非表示になるという流れです。

なお、画面に入力した内容は、「dialog.（モデル内の画面名）.（画面内の要素名).Text」で文字列を参照・設定できます。これにより、上記の例のように入力された内容で処理を分岐できます。

10 シミュレーション機能の拡張

この Enterprise Architect のシミュレーション機能では、任意の ActiveX COM オブジェクトを作成し、利用できます。これにより、例えば以下のような拡張を行うための独自の ActiveX COM オブジェクトを定義し、シミュレーション機能を拡張できます。

- シミュレーションの動作を、他の外部アプリケーションなどに通知し、Enterprise Architect 外のアプリケーションやシステムと連動させる
- 他の外部アプリケーションやシステムなどの情報を元に、Enterprise Architect のシミュレーション動作を変える
- 他の外部アプリケーションやシステムなどの動作を、Enterprise Architect のシミュレーション機能を利用して「見える化」する

この ActiveX COM オブジェクトとの連携は多少敷居が高いですが、シミュレーション機能を深く活用するための便利な手段です。
(注意：シミュレーション機能から COM オブジェクトを利用する場合には、対象の COM オブジェクトを

DLL 形式で作成し、デュアルインターフェースを実装する必要があります。)