Simulation Feature Guide

by SparxSystems Japan

シミュレーション 機能ガイド

(2016/10/07 最終更新)

1 はじめに

このドキュメントでは、Enterprise Architect のコーポレート版で利用可能な、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。

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、バージョン 13.0 で利用できる内容になります。それ以前のバージョンでは、対応していない機能があります。また、Enterprise Architect 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。

Enterprise Architect デスクトップ版ではシミュレーション機能は利用できません。

なお、SysML パラメトリック図でのシミュレーションにつきましては、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・シーケンス図・BPMN 2.0 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うこともできます。

なお、BPMN 2.0 図のシミュレーションは、Enterprise Architect Suite ビジネスモデリング版あるいは Enterprise Architect Suite アルティメット版が必要です。アクティビティ図・ステートマシン図・シーケンス図のシミュレーションについては、Enterprise Architect コーポレート版および Enterprise Architect Suite の各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。他の図の場合でも、基本的に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。ただし、シーケンス図はシグナルとの連携や複数の図の連携はできません。

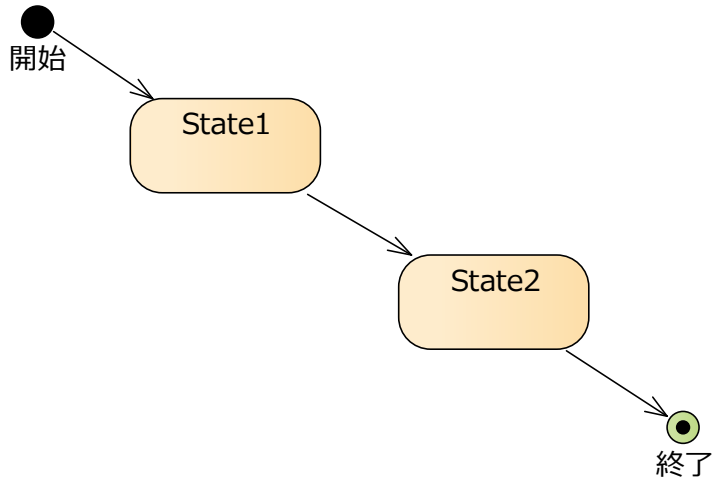
また、ステートマシン図については、状態遷移表の形式でもシミュレーションを実行す

ることができます。状態遷移表上のセルをクリックしたり、右クリックメニューを利用したりしてシミュレーションを制御することができます。

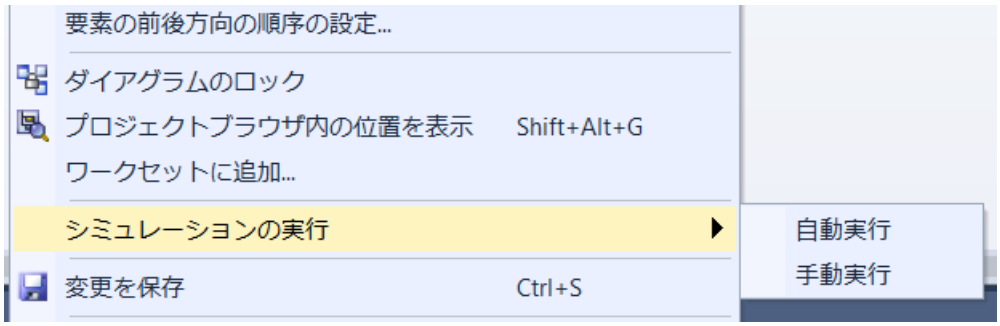
3 簡単なシミュレーション

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。

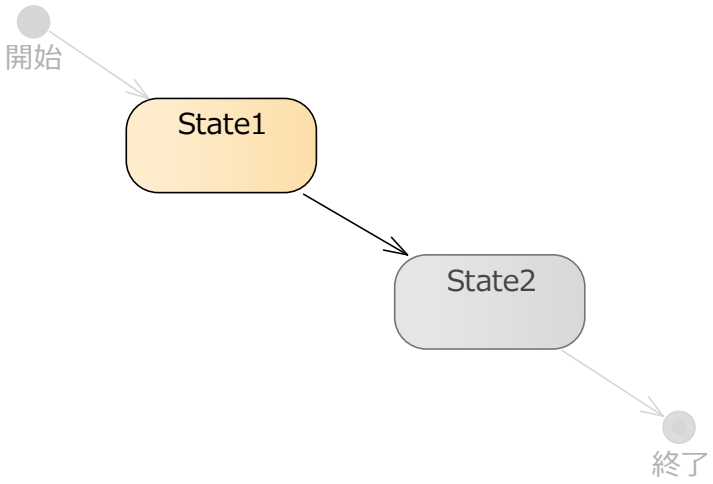
最初に、新規にステートマシン図を作成します。図の内容として、以下のような簡単なステートマシン図を作成します。
(シミュレーションを実行するステートマシン図と、その図に配置する要素は同じパッケージ(あるいは親要素)に含まれていなければなりません。)



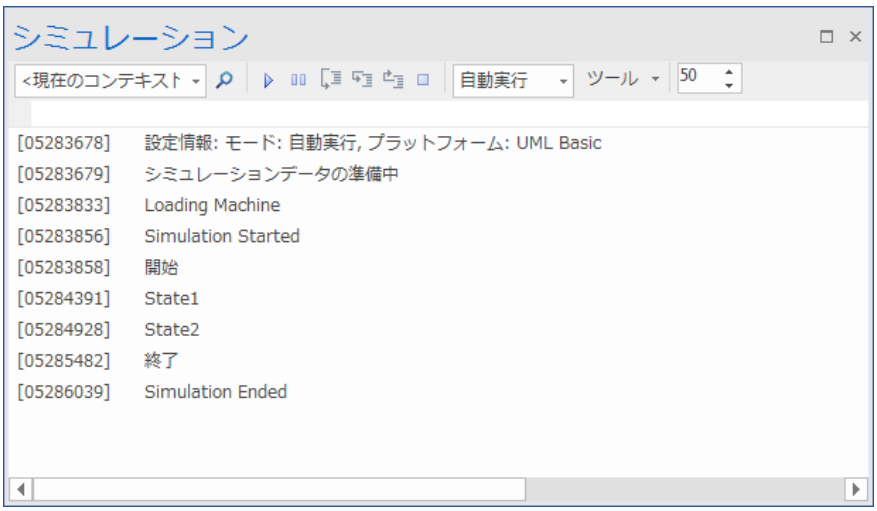
ステートマシン図を作成したら、ダイアグラムの背景で右クリックして「シミュレーションの実行」→「自動実行」を選択します。



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濃い灰色で、それ以外の要素は薄い灰色で表示されます。



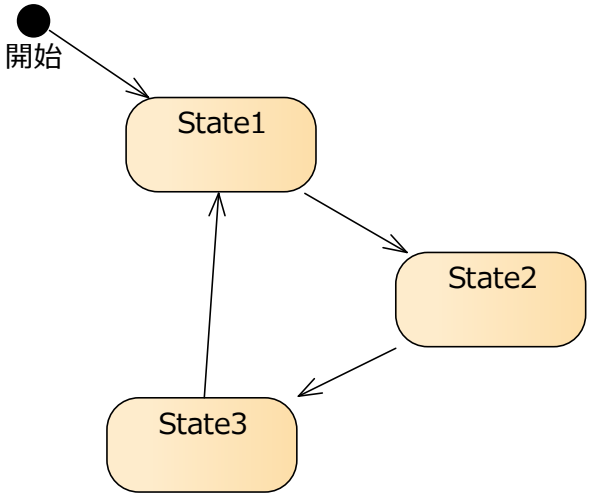
シミュレーションの実行速度は、シミュレーションサブウィンドウから変更することができます。「シミュレーション」リボン内の「シミュレーション」パネルにある「ウィンドウ」ボタンを押すことで表示されます。シミュレーションの動作ログが表示されますので、見える位置に配置しておくことをお勧めします。



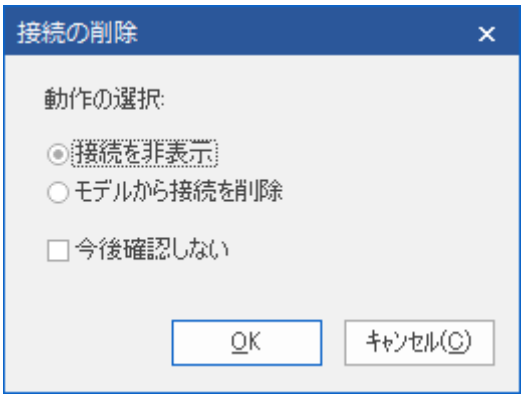
シミュレーションの動作速度は、このサブウィンドウから設定できます。上の画像の例では、「50」に設定されています。「100」が最高速度で、「1」が最低速度です。「0」の場合

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。)



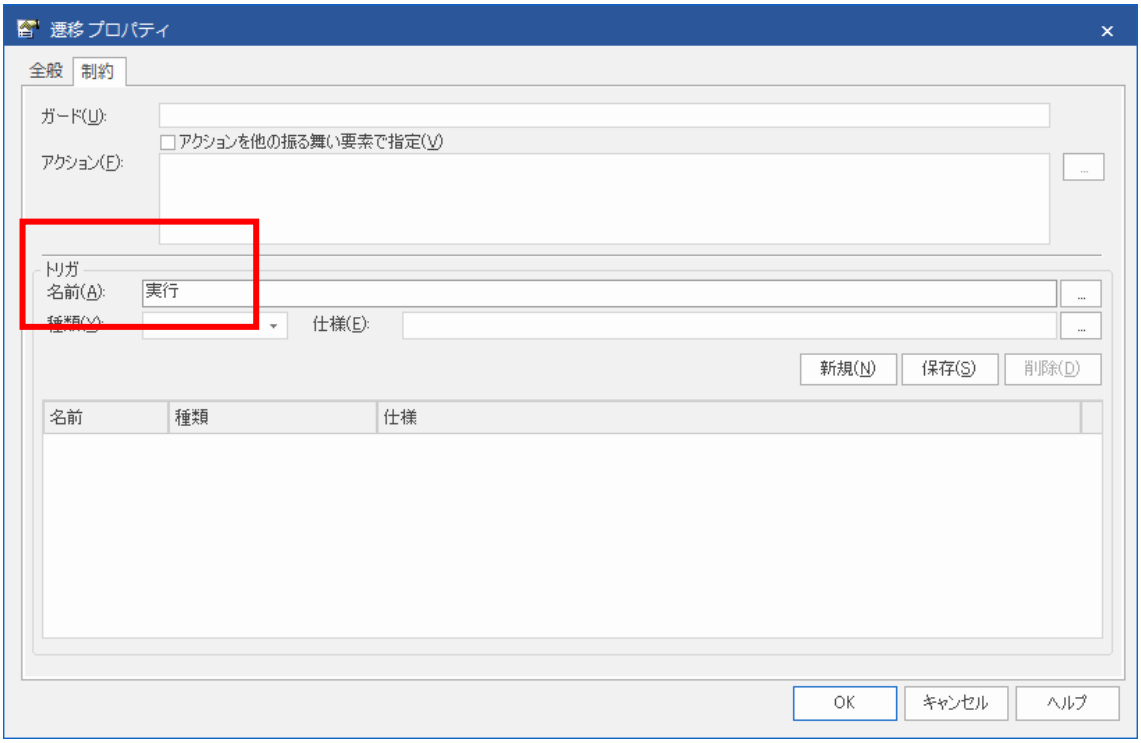
なお、上記の図になるように変更を行った場合に、操作方法や設定によっては、State2 で動作が停止する場合があるかもしれません。このような場合には、遷移の削除時に表示される次の画面で「接続を非表示」を選択していることが原因です。「接続を非表示」の場合には、見た目が非表示になっているだけで、モデルからは削除されていません。そのため、シミュレーション時の動作には引き続き影響を与えます。「モデルから接続を削除」を選択してください。



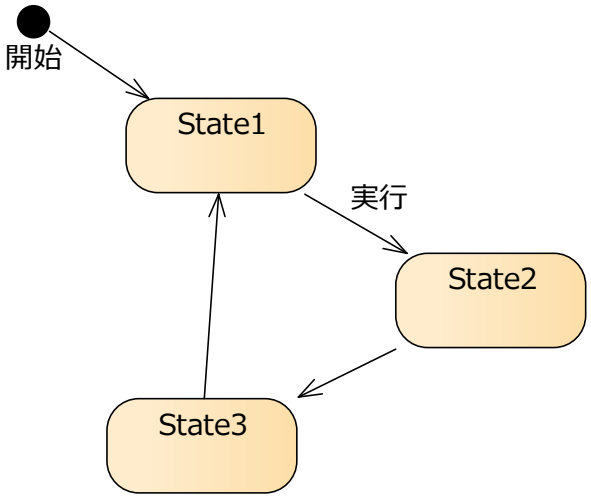
同様に、ダイアグラム内の要素を「ダイアグラムから削除」(Delete キー)した場合には、モデルには該当の要素が残っています。シミュレーションはダイアグラム上の内容(見た目の内容)ではなく、モデルに対して実行されますので、ダイアグラムから要素を削除した場合には、意図した通りに動作しない場合があります。不要な要素を削除する場合には、プロジェクトブラウザ内で要素を右クリックして「モデルから完全削除」を実行するか、要素を選択した状態で、ショートカットキー Ctrl+Delete を押して下さい。

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。ただし、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、この限りではありません。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 への遷移をダブルクリックして、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させることができますが、それ以外の種類の場合には全て同じ動作になります。

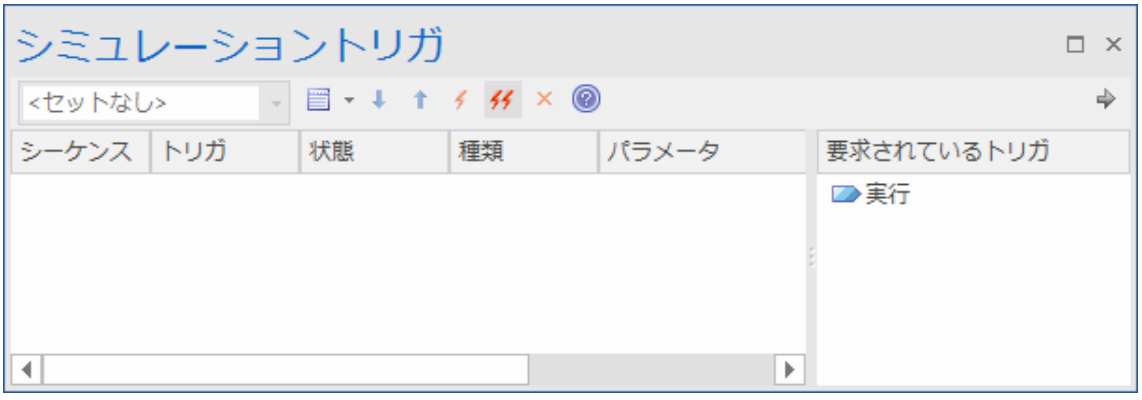


このようにすると、シミュレーション時に State1 で動作が停止し、トリガ「実行」を待機する状態になります。

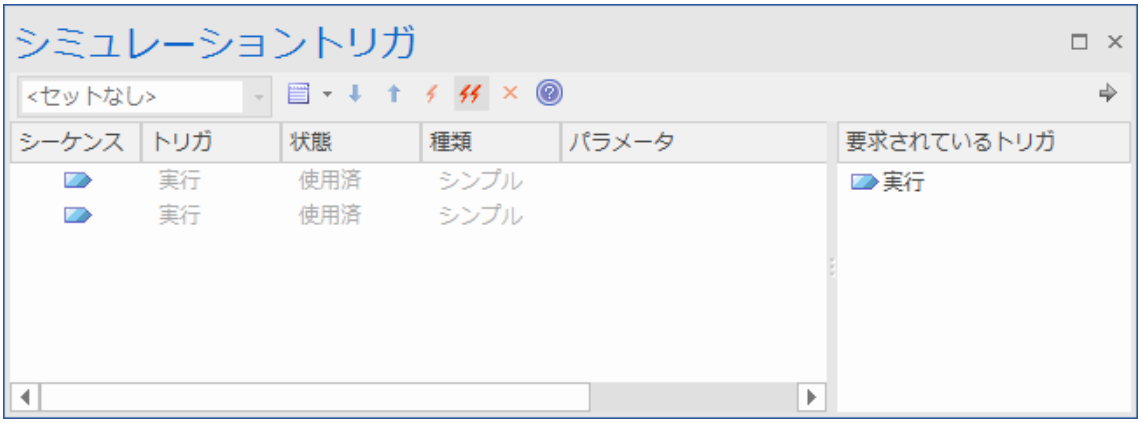


実行時にトリガを発生させるためには、「シミュレーショントリガ」サブウィンドウを利用します。「シミュレーション」リボン内の「シミュレーション」パネルにある「トリガ」ボタンを押すとサブウィンドウが表示されます。表示されるサブウィンドウの「要求されているトリガ」欄に、待機しているトリガが表示されます。このトリガをダブルクリックすることで、シミュレーション中の動作に影響を与えることができます。

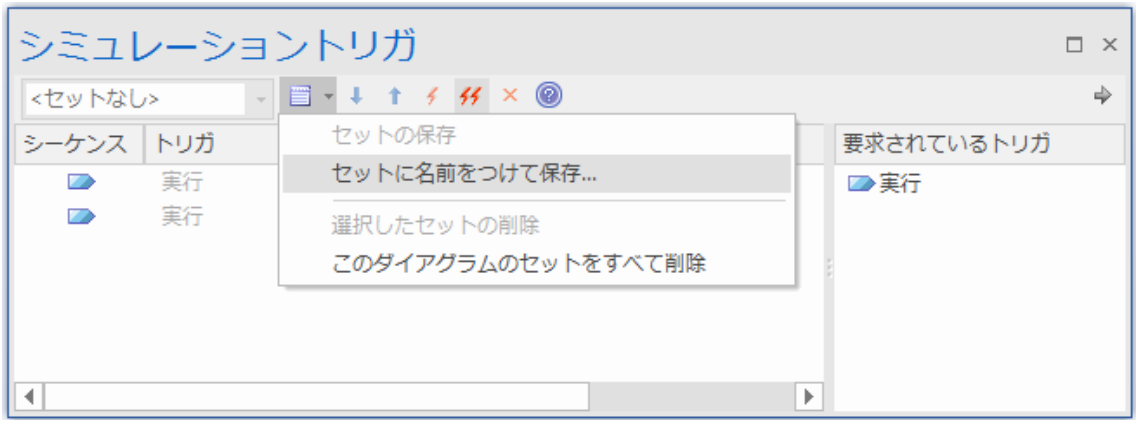
なお、「要求されているトリガ」に表示されていないトリガを明示的に発行したい場合には、プロジェクトブラウザから対象のトリガ要素をドラッグし、このサブウィンドウ左側の一覧にドロップしてください。一覧に追加されます。一覧に追加されたトリガをダブルクリックすることで、そのトリガを発行することができます。



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。



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することができます。発行したトリガの履歴は、保存しておき再利用することができます。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」を作成することができます。



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うことができます。

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. シミュレーションの結果を基に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、4 と 5 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。作成したセットを他のダイアグラムで利用することはできませんので、ご注意ください。

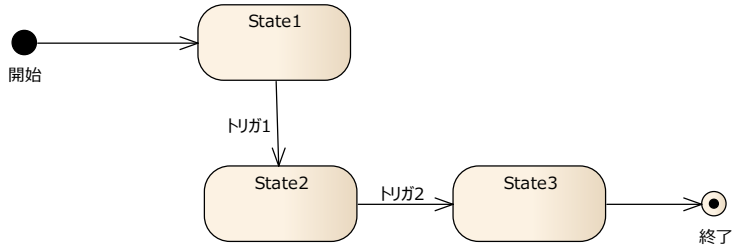
また、上記のように発行した履歴を保存する方法の他に、プロジェクトブラウザ内のトリガ要素をシミュレーショントリガサブウィンドウ内の一覧にドロップすることでも、トリガセットを定義することができます。

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」はトリガ要素をプロジェクト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。「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。

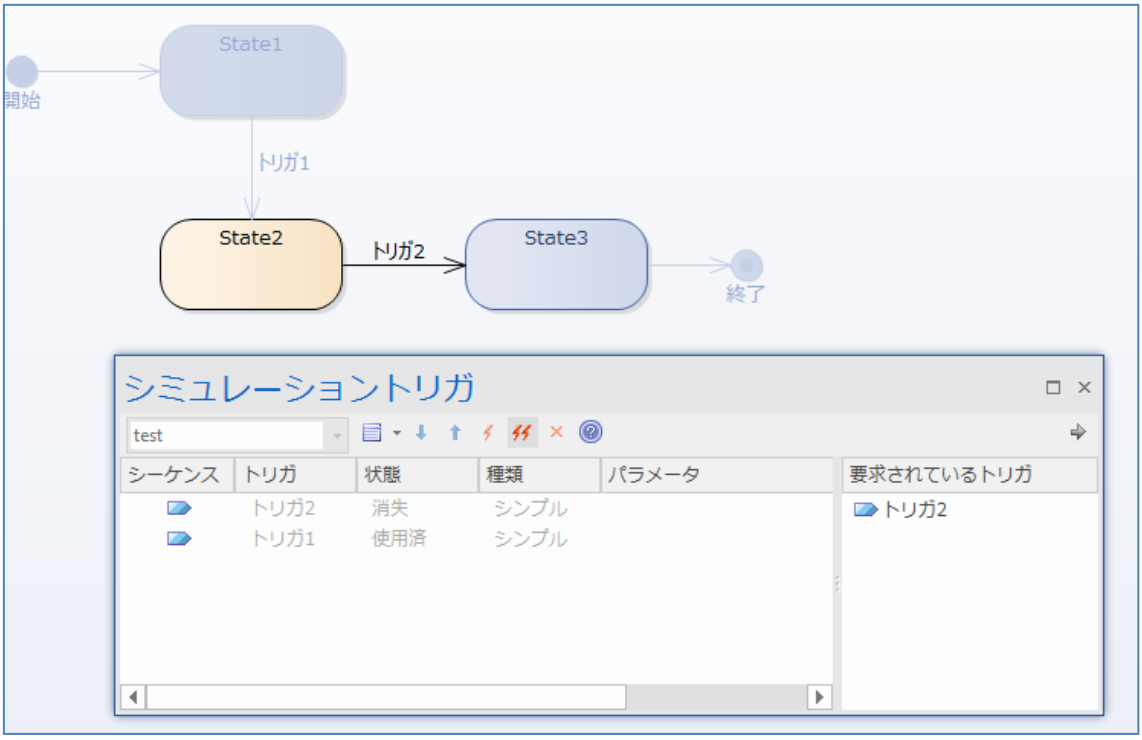
なお、例えば以下の例で、State1 の状態で停止しているときに「トリガ 2」を手動で「発行」した場合には、トリガ 2 は「発行」状態のままとなります。



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。あるいは、モデル側で対策できる方法もあります。8 章のサンプルをご覧ください。

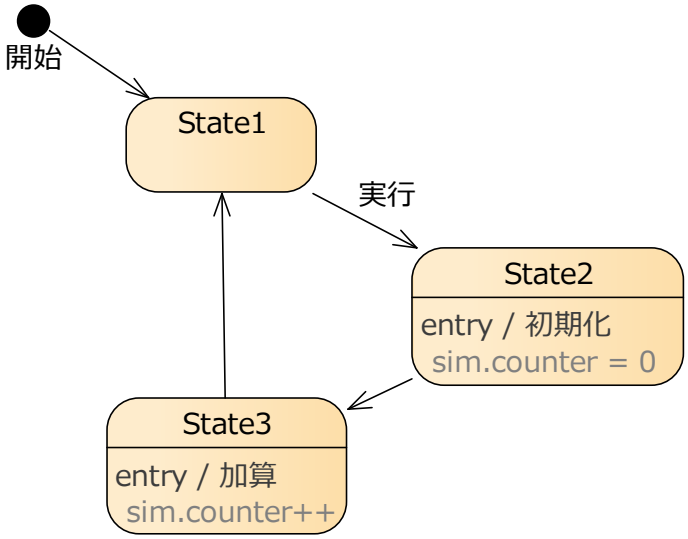
4 モデル内に処理を埋め込む

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むことができます。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、

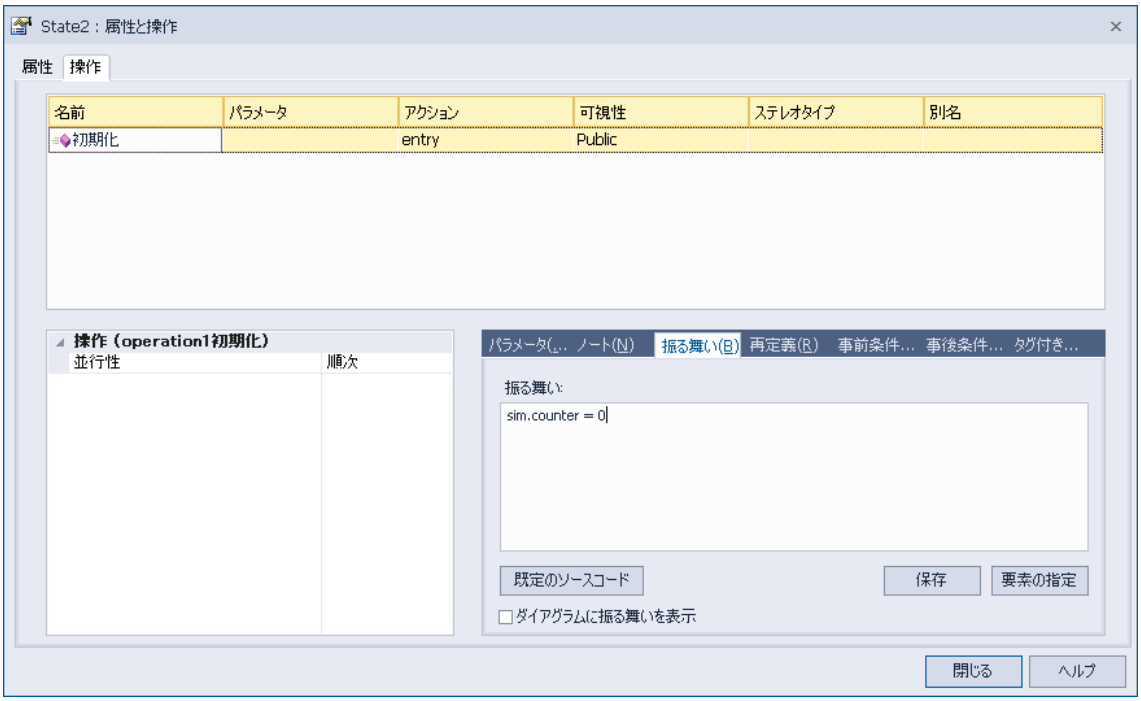
処理を分岐させることができます。

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。

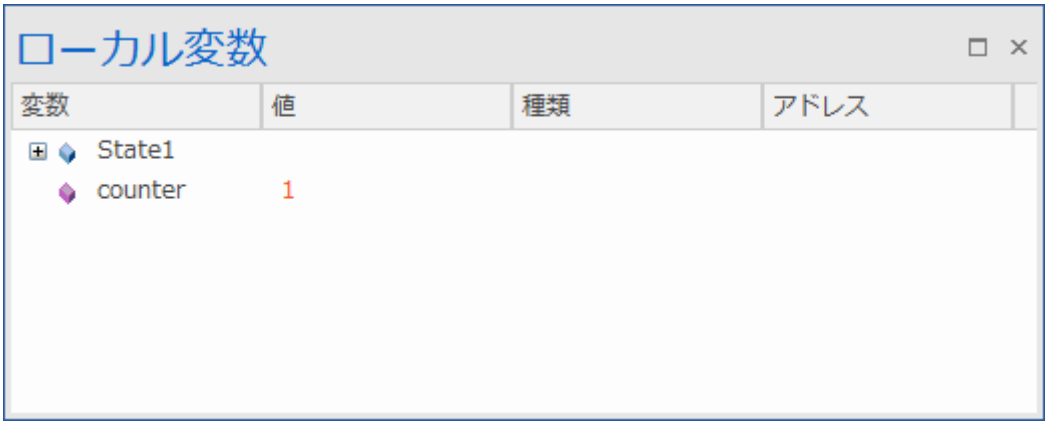
例として、先ほどの内容に、状態要素の entry アクションを追加します。



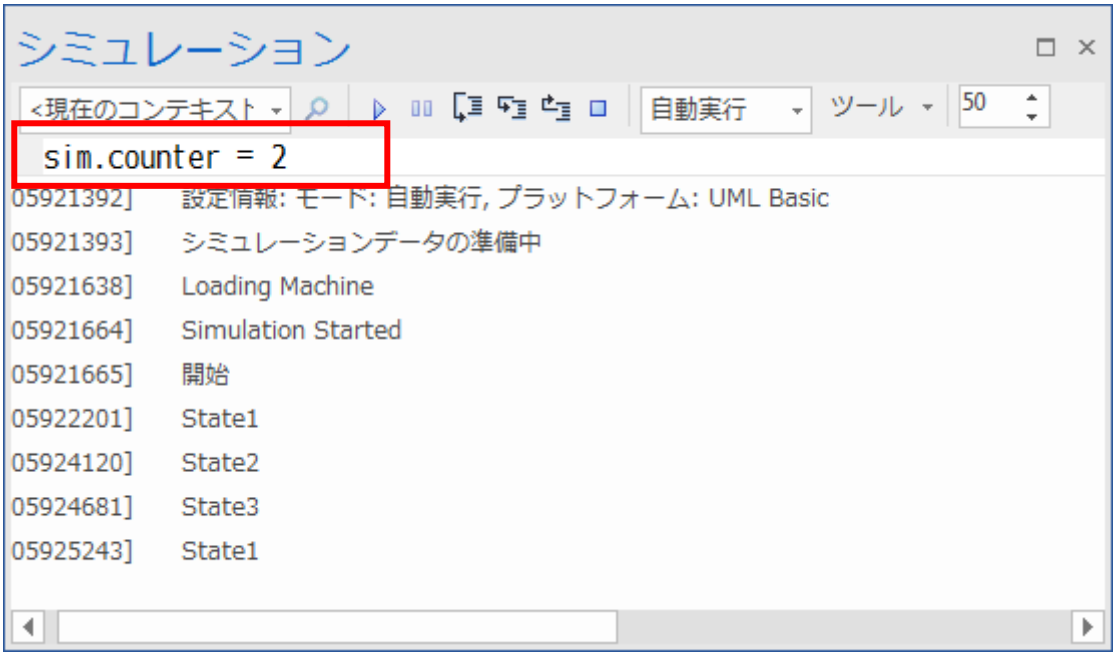
状態要素の entry/ exit のアクションは、Enterprise Architect では操作として定義されます。アクションを追加するには状態要素を右クリックして「属性・操作・インターフェース」→「操作」を選択して操作のプロパティ画面を表示し、追加します。実際の処理の内容は、「振る舞い」グループの「振る舞い」の入力欄に記述します。



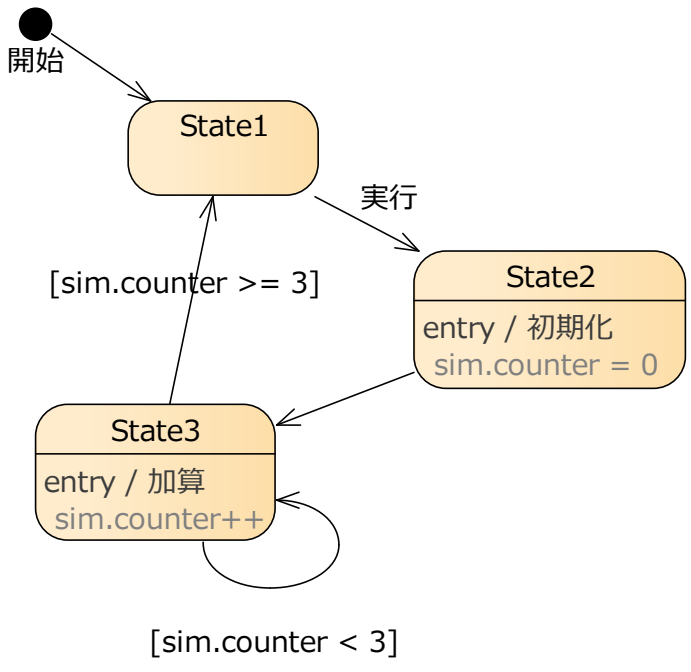
なお、シミュレーション機能で変数を利用する場合には、上の例のように、接頭辞(プレフィックス)として「sim.」(あるいは「this.」)を追加します。これらの接頭辞設定すると、シミュレーションの動作が停止しているときに、ローカル変数サブウィンドウで変数の値を確認できます。下の例は、シミュレーションを実行後、1 回「実行」のトリガを発行した後の状態です。



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、変更することもできます。下の図は、sim.counter の値に 2 をセットする場合の例です。(入力してリターンキーを押すと実行されます。)



また、遷移のガード条件でも変数を利用することができます。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することができます。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さまざまなシミュレーションを試すことができます。

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「シミュレーション」パネルにあるボタンから開くことができます。

コールスタックの表示

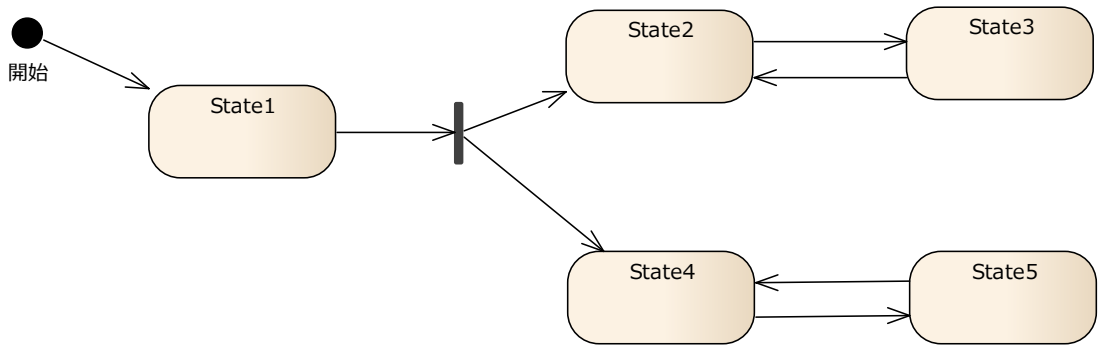
シミュレーションをデバッグするには、コールスタックの状態をシミュレーションの実行中に表示することが重要です。コールスタックサブウィンドウから状態を表示することができます。コールスタックウィンドウはマルチスレッドの並行シミュレーションをチェックするのに便利です。

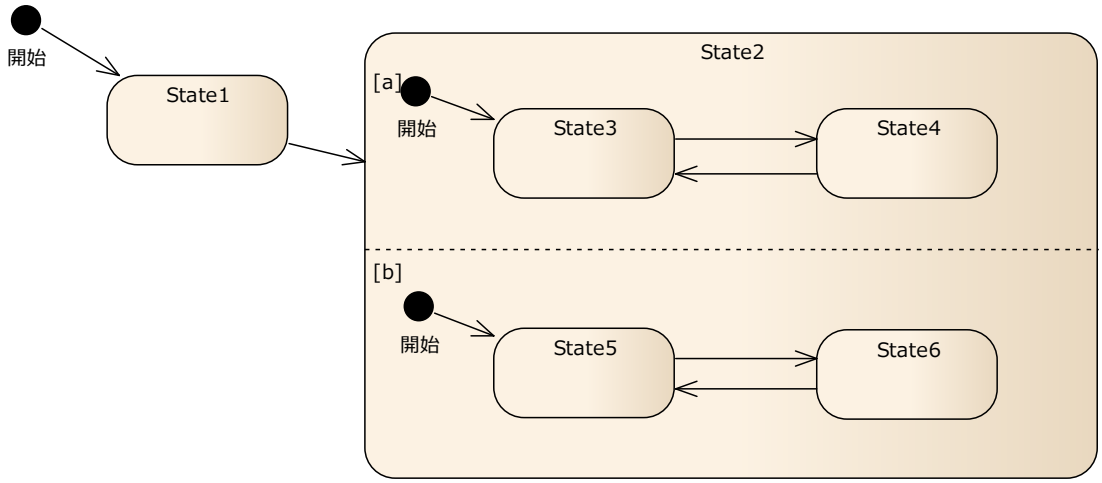
ブレークポイント

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することができます。プロジェクト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することができます。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。

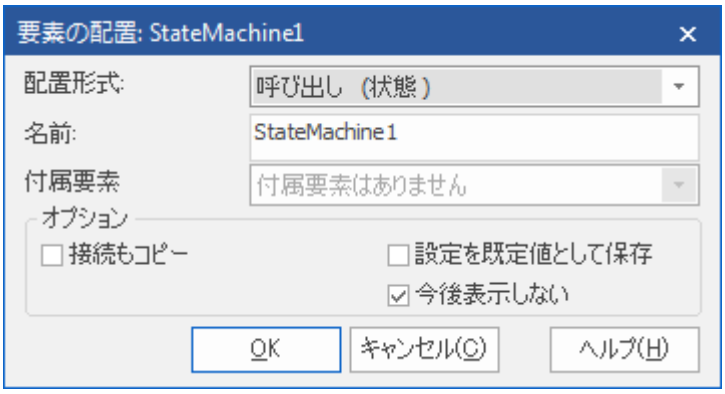




(状態内に領域を作成する場合には、それぞれの領域に異なる名前を設定してください。)

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することができます。

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、プロジェクトブラウザから手順 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。
(なお、状態マシン要素のまま配置して利用することもできます。その場合には、配置形式として「そのまま配置」を選択して下さい。)

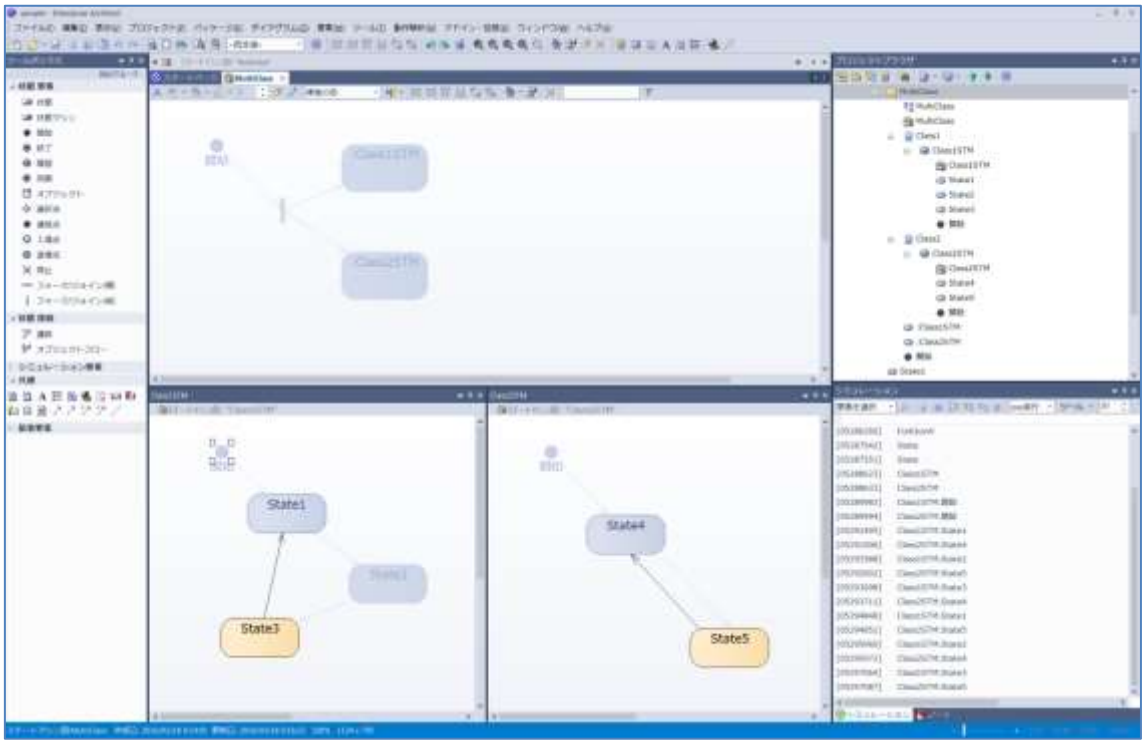
4. ステートマシン図に、開始状態を配置します。
5. 開始状態の先にフォーク/ジョイン要素を作成して遷移で結びます。さらに、フォーク/ジョイン要素からそれぞれの状態要素に遷移を結びます。
6. それぞれのステートマシン図を開き、ダイアグラムのタブをドラッグするとサブウィンドウ形式でステートマシン図を表示することができます。この方法を利用して、複数のステートマシン図を表示することができます。

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。

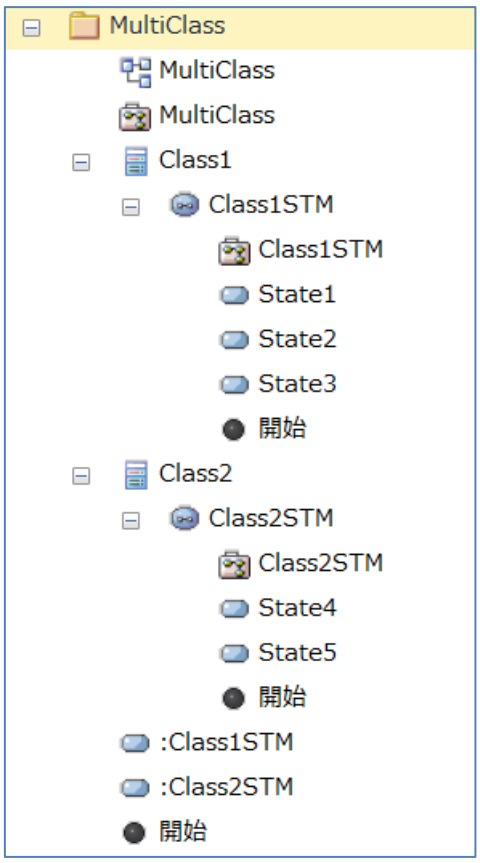
なお、上記の手順のように、複数の異なるダイアグラムを連携させてシミュレーションする場合には、その対象のダイアグラムを子ダイアグラムとして保持する状態マシン要素を作成し、その要素を配置・利用する必要があります。

6.1 サンプル

以下の画像は、上記の方法で設定した例です。



上記の例の、プロジェクトブラウザ内の構成イメージ



6.2 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や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することができます。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することができます。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 として参照することができます。このようにして参照できる値は、ガード条件などで活用することができます。

なお、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。

6.3 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。

6.4 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。11 章で説明する 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。

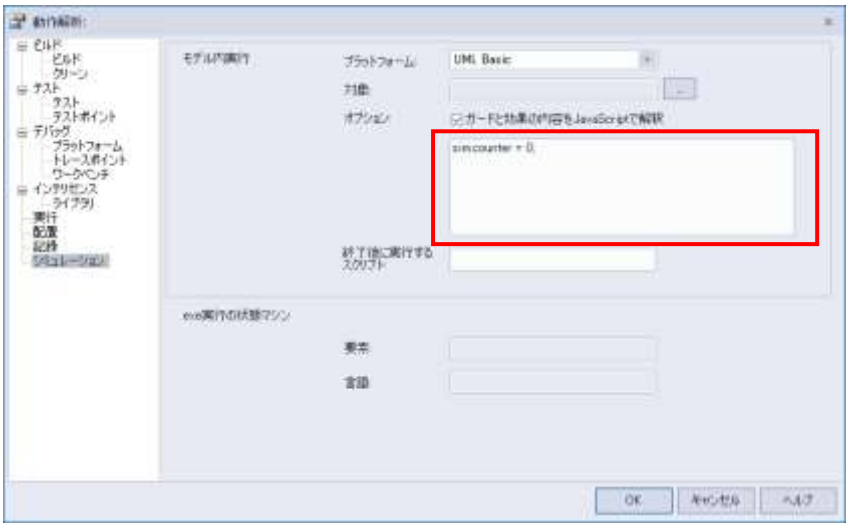
ですので、例えば、COM オブジェクトのメソッドの処理が 10 分かかる場合には、その時の「1 シミュレーション時間」は 10 分になるということです。
(その次の処理がすぐに終わる場合は、次の「1 シミュレーション時間」は 1 秒未満になることもあります。)
つまり、「1 シミュレーション時間」は、人間の時間とは対応せず一定ではありません。

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 初期処理

シミュレーションの実行前に変数の値を初期化する場合など、初期処理を定義したい場合には、「動作解析の設定」を定義する必要があります。動作解析の設定の「シミュレーション」グループ内の入力欄に入力された内容は、シミュレーションの動作開始時に実行されます。後述の ActiveX COM オブジェクトをシミュレーションに利用するような場合には、この画面でオブジェクトを新規作成することをお勧めします。



動作解析の設定を利用する場合には、状態マシン要素やアクティビティ要素を作成し、その下にステートマシン図やアクティビティ図を作成する必要があります。
(ステートマシン図やアクティビティ図からソースコード生成する場合と同じ構成が必要です。)

なお、BPMN2.0 でのシミュレーションを実行する場合には、動作解析の設定は必須です。上の画面での「プラットフォーム」を「BPMN」に設定する必要があります。

ここで記述した内容は、動作解析サブウィンドウに表示される設定内容を右クリックして「シミュレーションの実行」を選択した場合に、実行されます。

シミュレーション対象のダイアグラムを右クリックして「シミュレーションの実行」→「自動実行」を選択して動作させる場合には、この設定内容は実行されません。この方法で実行する場合には、シミュレーションを実行するモデルの最初に、内部で利用する変数の宣言と初期化を行う処理を、モデルとして追加しておく必要があります。変数を初期化しないで利用した場合、その結果は不定です。
(例えば、sim.counter=0 のような定義がなく、sim.counter++のような演算が実行された場合、sim.counter の値は参照できません。過去のバージョンでは利用する変数を自動的に初期化していた時期もあり、その際に作成したモデルのシミュレーションが新しいバージョンでは正しく動作しないという問題が発生するかもしれません。初期化処理を追加し、対応してください。)

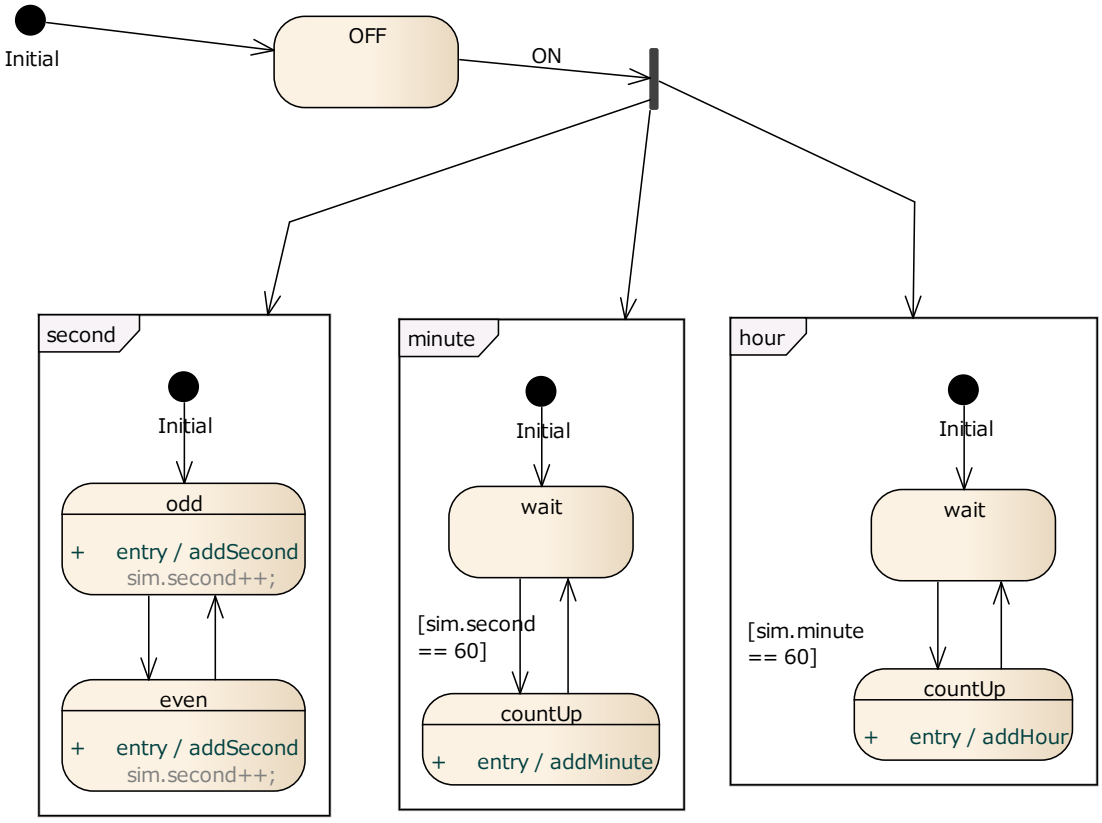
8 サンプル

Enterprise Architect のインストールディレクトリあるいはインストールしたユーザーの「マイドキュメント」フォルダに、「StateMachine_Sample.eap」という名前のサンプルファイルが含まれています。このサンプルファイルもご覧下さい。
(Enterprise Architect のインストールディレクトリ内のファイルは直接開くことはできません。ファイルを別の場所にコピーし、開いて下さい。)

これらのサンプルでは、状態マシン要素をそのまま配置しています。

8.1 3つのステートマシン図

次の例は、3 つのステートマシン図が同時に実行される例です。

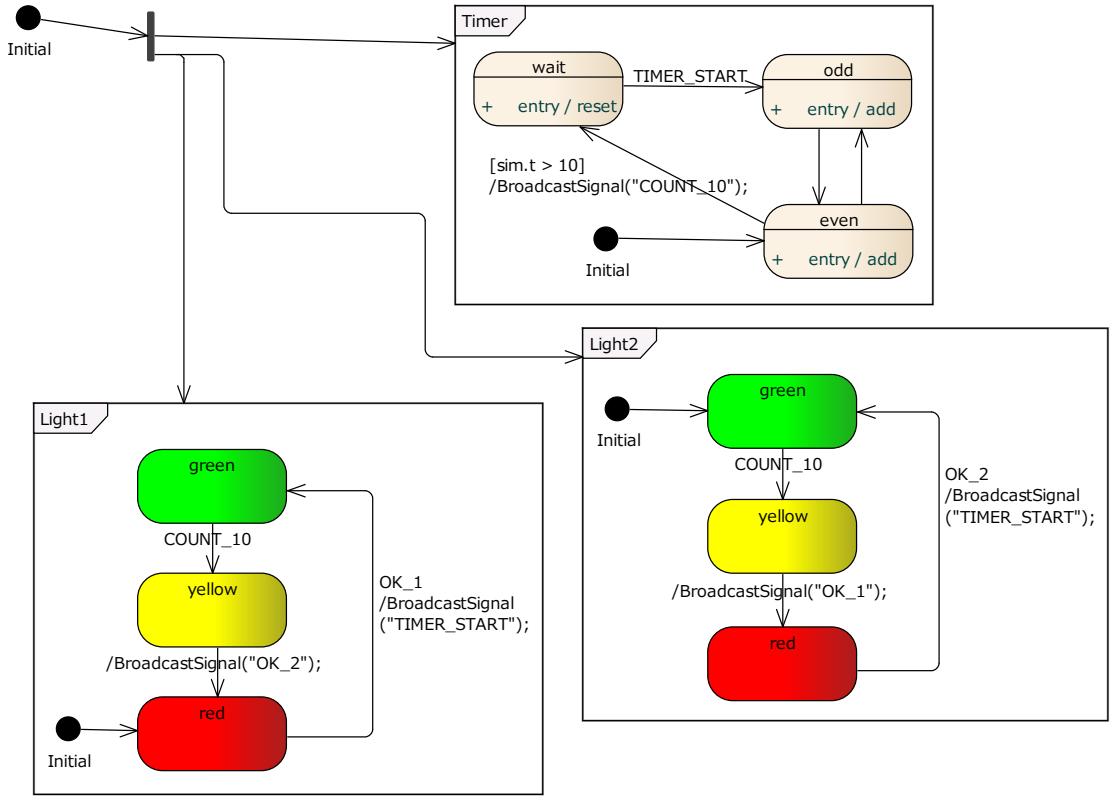


この例で、second, minute, hour の 3 つのステートマシンは、それぞれ「状態マシン」要素として作成されています。

8.2 信号機

次の例も同じく 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

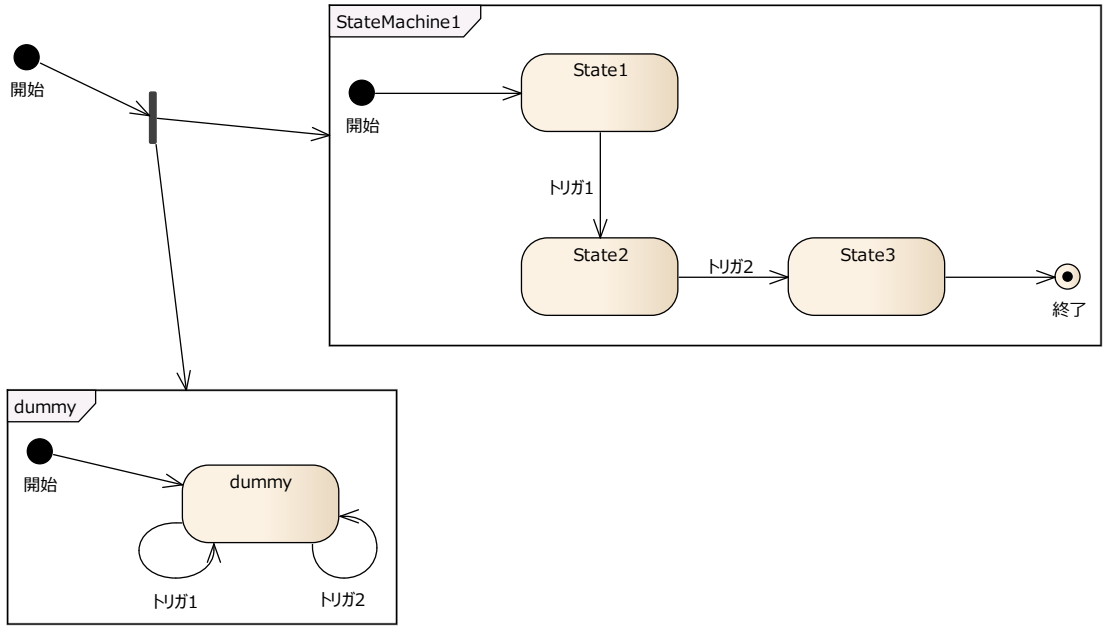
マシン図間でシグナル(トリガ)を投げ合っている例です。



8.3 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必ずすぐに使用され、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。



9 アクティビティ図・BPMN 2.0に関連する補足

アクティビティ図および BPMN2.0 の場合には､以下のような追加機能が利用できます。

9.1 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。

9.2 時間イベントアクションの利用

時間イベント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」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。)
なお、ツールボックスの「イベント受信」要素に対して時間イベントに変更したものは、このシミュレーションでは利用できませんので注意してください。

作成した要素のプロパティ画面の「トリガ」グループで、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することができます。

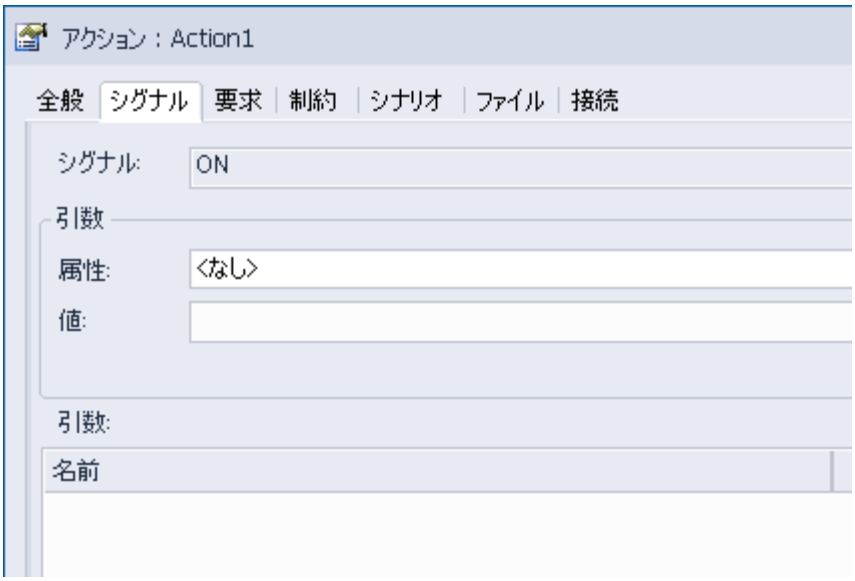
9.3 イベントの発行

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することができます。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することができます。

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。
(ツールボックスにある「シグナル送信」要素は利用しないでください。)

すると、該当のアクション要素のプロパティ画面には「シグナル」の設定項目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。



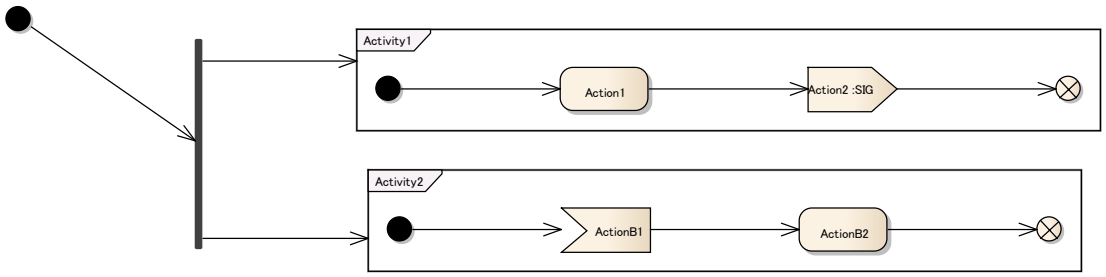
なお、ステートマシン図側で、シグナルに関連づけられたトリガを作成する手順については、次の通りです。

1. シグナル要素が存在しない場合、新規に作成します。
   (クラス図のツールボックスなどに「シグナル」要素が含まれますので、ダイアグラム上に配置して作成するか、プロジェクトブラウザのパッケージの右クリックメニューから「追加」→「要素の追加」を実行して作成してください)
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。

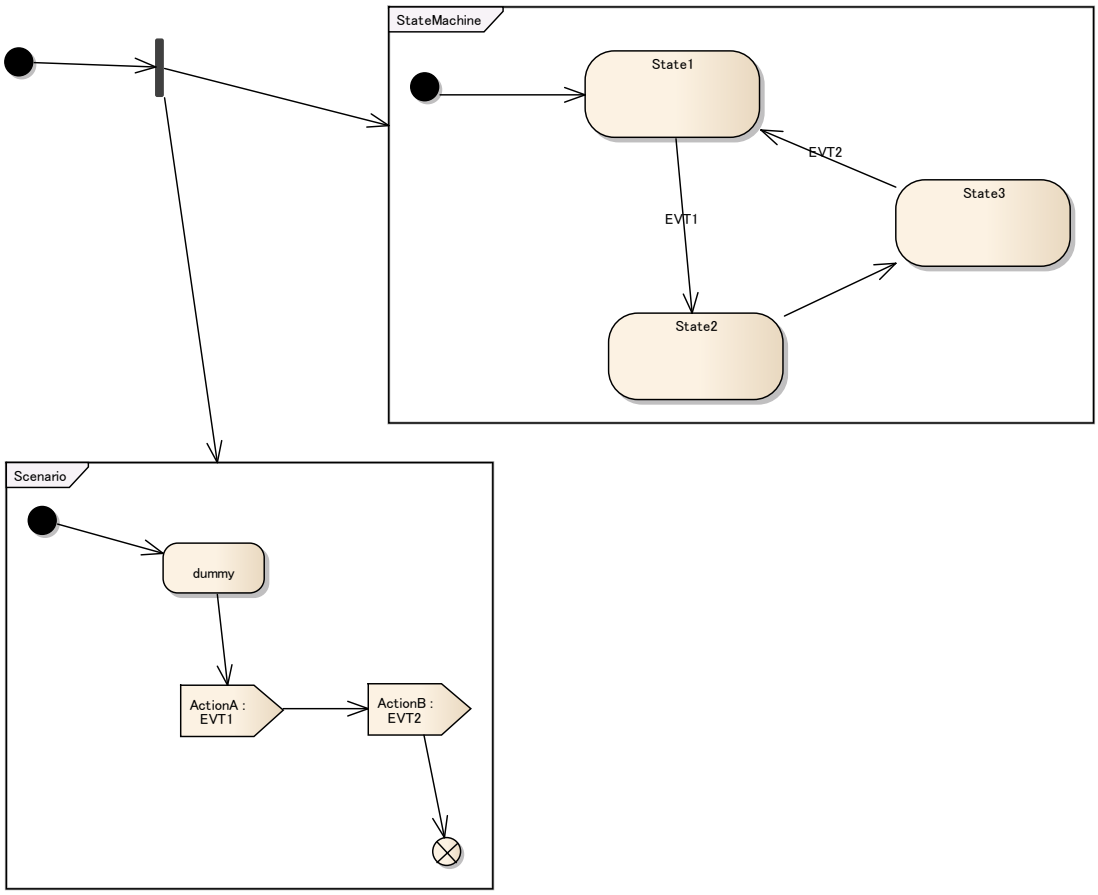
アクティビティ図で、特定のイベントを受信することもできます。「シグナル送信」と同様の方法で、「イベント受信」のアクション要素を作成します。このアクション要素について、待機するシグナルをプロパティ画面から設定することで、イベントを受信するまで待

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することができます。
(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。また、ツールボックスにある「イベント受信」要素は利用しないでください。)



9.4 複数の図をシミュレーションする

アクティビティ図の場合も、ステートマシン図と同じく複数の図を同時に動作させることができます。また、アクティビティ図とステートマシン図を組み合わせて動作させることもできますので、例えばシミュレーションのシナリオとしてアクティビティ図を作成し、アクティビティ図から 9.3 章記載の方法でシグナルを発行し、ステートマシン図を駆動させることもできます。



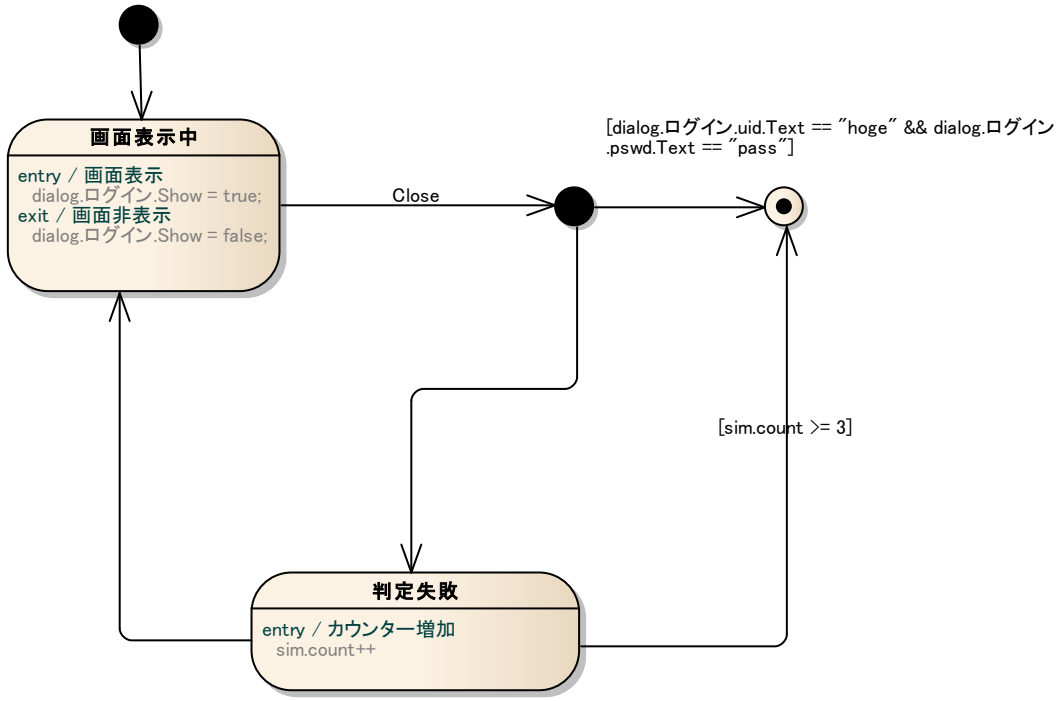
なお、この場合には、状態マシン要素の代わりにアクティビティ要素を作成する必要があります。作成したアクティビティ要素を右クリックして、「ダイアグラムの追加」→「子ダイアグラムを作成」を実行し、子ダイアグラムとしてアクティビティ図を作成します。そのアクティビティ図の中に、実行する内容を記述してください。

あとは、6 章の内容について、状態マシン要素の代わりに、上記で作成したアクティビティ要素を配置する形になります。ステートマシン図に配置する要素から、このアクティビティ要素へはクリックリンクで遷移(あるいはフロー)を作成することができません。ツールボックスから、遷移あるいはコントロールフローを選択し、フォーク/ジョイン要素とアクティビティ要素を接続して下さい。

10 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ることもできます。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することができます。

「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することができます。

画面のボタンを押した場合の処理を記述することもできますので、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・設定することができます。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することもできます。

11 コード生成に関連する補足

Enterprise Architect のステートマシン図からのコード生成は状態の定義とその遷移処理についてが出力対象であり、トリガを発行する部分は対象外です。この部分は、実際の OS などの環境に依存する部分であり、ステートマシン図には含まれません。

トリガのやりとりをコードとして表現したい場合には、コード生成テンプレートをカスタマイズし、BroadcastSignal などのシミュレーション用の関数を、OS が提供する(あるいは独自に別途構築する)トリガなどの処理を実現する仕組みを呼び出すための関数(メソッド)に置換する形になります。

また、ステートマシン図や状態遷移表の他、このドキュメントで紹介したシミュレーションなどを組み合わせる方法を、ドキュメント「ステートマシン図の整合性確保 マニュアル」で紹介しています。

https://www.sparxsystems.jp/products/EA/ea_documents.htm

これらのいずれの内容でも、ステートマシン図の内容からソースコードを生成する方法を紹介しています。

12 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、利用することができます。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することも可能です。

- 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。)

○改版履歴

2011/12/19 初版
2012/03/07　Enterprise Architect 9.3 のリリースに伴い内容を更新。
2012/12/14　Enterprise Architect 10.0 のリリースに伴い内容を更新。8 章・9 章を追加。
2013/01/18 シミュレーションについてのいくつかの補足説明を追加。
　　9 章の内容で、消えてしまっていた部分を復帰。
2013/07/18　いくつかの補足説明を追加。
2013/07/24　6.3 章および 9.4 章を追加。
2013/09/24　UIBroadcastSignal の説明を追加。
2013/11/26　アクティビティ図のシミュレーションについて、サンプルを追加。
　　全体的に、いくつかの補足説明を追加
2014/04/22　Enterprise Architect 11.0 のリリースに伴い内容を更新。
2014/11/06　細かい補足の追記。
2015/02/12　Enterprise Architect 12.0 のリリースに伴い内容を更新。
2015/12/01　Enterprise Architect 12.1 のリリースに伴い内容を更新。
2016/03/18　6 章の説明を修正。画像の差し替え。
2016/07/20　内容の誤りの修正。
　　(状態の do アクションはシミュレーションの対象外)
2016/10/07　Enterprise Architect 13.0 のリリースに伴い内容を更新。