

DocGen ユーザーガイド

このドキュメントは、別のプロジェクトで作成および管理されている DocGen ユーザー ガイドを参照します。

目次

1 DocGen	10.1.1 DocGen の
概要.....	10.1.2 DocGen を使
用したオフライン コンテンツの作成	10.1.2.1 ドキュメントの生
成	10.1.2.2 DocGen スタイルシートの使
用	13.1.3 視点メソッドの作
成	13.1.3.1 モデル要素の収集/並べ替え/フィルタリ
ング	13.1.3.1.1 収
集.....	15.1.3.1.1.1
CollectOwnedElements	15.1.3.1.1.1.1 単一の深さの
例	18.1.3.1.1.1.2 無限の深さの
例.....	18.1.3.1.1.2 所有者を収集す
る	19.1.3.1.1.2.1 CollectOwners ビュ
ーの例	21.1.3.1.1.3
CollectThingsOnDiagram.....	21.1.3.1.1.3.1
CollectThingsOnDiagram ビューの例	21.1.3.1.1.4
CollectByStereotypeProperties.....	23.1.3.1.1.4.1 簡条書きの
例.....	27.1.3.1.1.4.2 テーブル構造
例.....	27.1.3.1.1.5
CollectByDirectedRelationshipMetaclasses.....	28.1.3.1.1.5.1 出力方向の
例	30.1.3.1.1.5.2 例の方
向	31.1.3.1.1.5.3 方向出
力	31.1.3.1.1.6 CollectByDirectedRelationship
ステレオタイプ.....	31.1.3.1.1.6.1 ビューの
例	33.1.3.1.1.7
CollectByAssociation	33.1.3.1.1.7.1 ビューの
例.....	36.1.3.1.1.8
CollectTypes....	36.1.3.1.1.8.1 ビューの
例.....	38.1.3.1.1.9
CollectClassifierAttributes.....	38.1.3.1.1.9.1 ビューの
例.....	40.1.3.1.1.10
CollectByExpression	40.1.3.1.1.10.1 ビューの
例.....	41.1.3.1.2 フィル
タ。	41.1.3.1.2.1
FilterByDiagramType.....	42.1.3.1.2.1.1 画像
例	42.1.3.1.2.2
FilterByNames	42.1.3.1.2.2.1 ビュー
の例.....	42.1.3.1.2.3
FilterByMetaclasses.....	42.1.3.1.2.3.1 ビューの
例.....	43.1.3.1.2.4
FilterByStereotypes	43.1.3.1.2.4.1 ビ
ューの例.....	43.1.3.1.2.5
FilterByExpression	43.1.3.1.2.5.1 ビュ
ーの例	44.1.3.1.3 並べ替
え	44.1.3.1.3.1 属性によ
る並べ替え.....	44.1.3.1.3.1.1 ビュ
ーの例	44.1.3.1.3.2
SortByProperty	44.1.3.1.3.2.1 例ビュ
ー	44.1.3.1.3.3
SortByExpression	44.1.3.1.3.3.1 ビューの
例.....	45.1.3.2 現在のモデルデー
タ	45.1.3.2.1
表	46.1.3.2.1.1 簡易テー
ブル	48.1.3.2.1.2 複合テーブ
ル.....	49.1.3.2.2 段
落.....	49.1.3.2.2.1 ターゲットを含む段落ア
ション	49.1.3.2.2.1.1 一般的な段落の
例.....	52

1.3.2.2.1.2 段落名の例.....	52
の例.....	52 1.3.2.2.1.4 段落の値
例.....	52 1.3.2.2.2 本文を含む段落アクション
ン.....	53 1.3.2.2.1 段落本文の評価
例.....	54 1.3.2.2.3 段落アクションの OCL の評価
ト.....	54 1.3.2.2.3.1 本文のある段落。
さ.....	56 1.3.2.2.3.2 段落ターゲット付き
ト.....	56 1.3.2.2.3.3 段落付きボディとターゲット
ト.....	56 1.3.2.3 リス
	56
1.3.2.3.1 例リスト.....	57 1.3.2.4
画像.....	58 1.3.2.4.1 画像
像例ビュー.....	59 1.3.2.5 動的セクション
グ.....	59 1.3.2.5.1 単一セクションの例
例.....	61 1.3.2.5.1.1 動物園の動物
物.....	61
1.3.2.5.2 複数のセクションの例.....	62 1.3.2.5.2.1 ダチヨウ
ヨウ.....	62
1.3.2.5.2.2 クロコダイル.....	62
1.3.2.5.2.3 ゼブラ.....	62
1.3.2.5.2.4 シール.....	62
1.3.2.5.2.5 キョクアジサシ.....	62
1.3.2.6 トムソーや図.....	62 1.3.2.6.1 タムの作成ソーやー
図.....	62 1.3.2.6.1.1
myView.....	64 1.3.2.6.2 よくある間違い
間違い.....	64
1.3.2.6.3 内部ブロック図.....	65 1.3.2.6.4 前の図からの生成
成.....	66 1.3.2.7 プロット
ト.....	67
1.3.2.7.1 プロットの作成.....	69
1.3.2.7.1.1 折れ線グラフの例.....	71 1.3.2.7.1.2 レーダーの例プロット
ダーピーの例.....	71 1.3.2.7.1.3 平行軸プロットの例
ズ.....	72 1.3.2.7.2 プロットのカスタマイズ
題.....	73 1.3.2.7.3 一般的な問題
1.3.3 その他.....	75
1.3.3.1 シミュレーション.....	75
1.3.3.1.1 構成のシミュレート.....	76 1.3.3.1.2 データのシミュレート
タ.....	79
1.3.3.2 不透明な動作.....	79 1.4 作成および OCL
制約の評価.....	81
1.4.1 OCL とは何ですか?なぜ使用するのですか?.....	81 1.4.2
OCL ブラックボックス式とは何ですか?.....	82 1.4.2.1 値
().....	83 1.4.2.1.1 ビュー
の例.....	83 1.4.2.2 関係性
「r()」.....	83 1.4.2.2.1 ビューの例
例.....	84 1.4.2.3 名前
「n()」.....	84 1.4.2.3.1 ビューの例
例.....	84 1.4.2.4 ステレオタイプ
「s()」.....	84 1.4.2.4.1 ビューの例
例.....	84 1.4.2.5 メンバ
「m()」.....	84 1.4.2.5.1 ビューの例
例.....	85 1.4.2.6 「eval()」の評価
ト.....	85 1.4.2.6.1 ビューの例
例.....	86 1.4.2.7.7 型
「t()」.....	86 1.4.2.7.1 ビューの例
例.....	87 1.4.2.8 所有者
「owners()」.....	87 1.4.2.8.1 ビューの例
例.....	87 1.4.2.9
log()	87

1.4.2.9.1 ビューの例.....	87	1.4.2.10
実行(ビュー/視点).....	87	
1.4.2.10.1 ビューの例.....	88	
1.4.3 ビューポイントで OCL 式を使用するにはどうすればよいですか?.....	88	
1.4.3.1 式による収集/フィルター/並べ替えの使用.....	88	1.4.3.2 高度なトピック.....
ク.....	88	1.4.3.2.1 反復フラグの使
用.....	88	1.4.4 OCL エバリュエータ
ーとは何ですか?また、それを使用する理由は何ですか?.....	88	1.4.5 OCL 視点
制約を使用するにはどうすればよいですか?.....	91	1.4.6 どうすればよいですか?
OCL ルールを作成しますか?.....	91	
1.4.6.1 特定のモデル要素に対するルールを作成するにはどうすればよいですか? (制約評価者).....	91	1.4.6.2 ビューポイント内で
ルールを作成するにはどうすればよいですか?	92	1.4.6.3 モデル内の OCL ルー
ルを検証するにはどうすればよいですか?	92	1.4.7 式ライブラリを作成するには
どうすればよいですか?.....	92	1.4.8 クエリで正規表現を使用するに
はどうすればよいですか?	92	1.4.9 OCL クエリでトランスクルー
ジョンを作成するにはどうすればよいですか?	92	ト

テーブルのリスト

1. DocGen ×ソッド	14
2. 表例	27
3. 表の例	36 4. 分類
子	38 5. 所
有アイテム	40
6. DocGen ×ソッド	45
7. 簡単なテーブルの例	48
8. <>	49 9. ヘッダ
一付きの表	68 10. ヘッダーのな
い表	68 11. リンゴとオレ
ンジのデータ	73 12. DocGen ×ソッ
ド	75 13. インスタンス
テーブル	79 14. 不透明
な行動データ	81 15.
<>	83 16. 人間関
係	84 17.
<>	84 18. 固定觀
念	84
19. メンバー	85
20. <>	86
21. 種類	87
22. <>	87
23. <>	88
24. <>	91 25.
<>	92 26.
<>	92 27.
<>	92

フィギュア一覧

1. ビュー図の例	16 2. 図
例	17 3. CollectOwnedElements
の単一の深さ	17 4. CollectOwnedElements の無限の深
さ	18 5. ビュー図
例	19 6. 所有者を収集す
る	20 7. BDD の
例	21 8. ビュー図の
例	22 9.
CollectThingsOnDiagram	22 10.
BDD の例	23
11. ビュー図の例	24 12.
CollectByStereotypeProperties リスト	24 13.
13. BDD の例	26
14. CollectByStereotypeProperties テーブル	27 15. ビュー図の
例	28 16. CollectByDirectedRelationshipMetaclasses
の方向外	29 17. BDD の
例	30 18.
CollectByDirectedRelationshipMetaclasses の方向	30 19.
CollectByDirectedRelationshipMetaclasses ビューポイント方法 2	31 20. ビュー図
例	32 21. BDD の
の例	32 22.
CollectByDirectedRelationshipStereotypes	33 23. ビューの例図
表	34
24. CollectByAssociation	35 25.
要素の例	36 26. ビュー図
の例	36 27. タイプの収
集	37 28. BDD の
例	37 29. ビュー図
例	38
30. CollectClassifierAttributes	39 31. BDD の
例	39 32.
32. ビュー図例	40 33.
CollectByExpression	41 34. BDD の
の例	41 35. ビュー図の
例	42 36. 図
表	42 37.
FilterByDiagramType 画像	42 38. 図
表	42 39. あるプロッ
ク	42 40. ビュー図
例	42 41. BDD の
例	42 42.
FilterByNames	42 43. ビ
ユー図の例	43 44.
FilterByMetaclasses	43 45. BDD の
の例	43 46. ビュー図
例	43 47. ステレオタイプに
よるフィルター	43 48. BDD の
例	43 49. ビュー図の
例	43 50.
FilterByExpression	43 51. BDD の
例	43 52. ビュー図の
例	44 53. 属性による並べ替
え	44 54. BDD の
例	44 55. ビュー図
例	44 56. BDD の
例	44 57.
SortByProperty	44

58. ビュー図の例.....	44 59. ソートバイ
式.....	45 60. BDD の
例.....	45 61. ビュー図の
例.....	47 62. シンプルテープ
ル.....	47 63.
ComplexTable	48 64.
一般的な段落ビュー図.....	50 65. 一般的
な段落.....	50 66. 氏名の段
落.....	51 67. ドキュメントの段
落.....	51 68. デフォルト値の
項.....	52 69. ビュー図の
例.....	53 70. 段落本
文.....	53 71. ビュー図の
例.....	54 72. ターゲットを含む段落
OCL	55 73. パラグラフ OCLターゲットな
し.....	55 74. ターゲット用パラグラフ
OCL	56 75. リストビュー図の
例	57 76. 節条書きリス
ト.....	57
77. ビュー図の例.....	58 78. 画
像.....	58 79. 動物
園.....	59
80. 動物園の動物.....	59
81. ビュー図の例.....	60 82. ダイナミックセクショ
ン.....	60 83. 複数のセクショ
ン.....	61 84. トム・ソーヤBDDビュー
図.....	63 85. トム・ソー
ヤ.....	63 86. tomSawyerDiagram
.....	64 87. よくある間違いの
図.....	65 88. トムソー
ヤ.....	65 89. トム・ソーヤ IBD ビュー
図.....	66 90. トム・ソー
ヤ.....	66 91. 前の図 図を表
示.....	67 92. トム・ソー
ヤ.....	67 93. ヘッダー付きのデモプロッ
ト.....	68
94. ヘッダーのないデモ プロット.....	68
95. ラインプロットブロック	69
96. 折れ線グラフ表示図.....	69 97. ラインプロッ
ト.....	70
98. リンゴとオレンジ.....	71 99. リンゴとオレン
ジ.....	72 100. リンゴとオレン
ジ.....	73 101. リンゴとオレン
ジ.....	74 102. リンゴとオレン
ジ.....	75 103. 力のシミュレーション
ン	76
104. 点質量.....	76
105. システムコンテキスト	77 106. 力のシミュレー
ション	77
107. シミュレートビュー図.....	77 108. シミュレート
する	78
109. OpaqueBehavior ビュー図.....	79 110.
OpaqueBehavior の視点	80 111. 動
物	83
112. value() ダイアグラムの表示.....	83 113. 値
ブラックボックスのデモ	83
114. relationship() ダイアグラムの表示	84 115. 人間関係
ブラックボックスのデモ	84 116. 名前の表示
図	84 117. 名前ブラックボックスの
デモ	84

118. ステレオタイプ図の表示	84	119. ステレオ
タイプのブラックボックスデモ	84	120. メンバーピ
ユートピア	85	121. メンバ
一ブラックボックスデモ	85	
122. ビュー図の評価	86	123. ブラックボックスを評価す
るデモ	86	
124. トランスクルージョンテストモデル	86	
125. タイプビュー図	86	126. タイプ ブラック
ボックス デモ	87	127. オーナーズビューの
図	87	128. オーナーズブラックボックスのデ
モ	87	87
129. ログビュー図	87	130. ログ ブラック ボ
クスのデモ	87	131. RunViewViewpoint ビュー
図	88	132. ブラックボックスデモを実行す
る	88	
133. 特定のモデル要素にルールを作成するにはどうすればよいですか?	92	

方程式のリスト

1 ドキュメントジェン

DocGen は、DocGen の目的と機能についてのより詳細なガイドです。

さまざまな DocGen メソッドおよび式ライブラリの詳細については、「[ビューポイント メソッドの作成](#)」を参照してください。

目次

[DocGen の概要](#)

[DocGen をインストールする](#)

[DocGen を使用してオフライン コンテンツを作成する](#)

[ビューポイントメソッドの作成](#)

[DocGen ユーザースクリプト](#)

1.1 DocGen の概要

ドキュメント ジェネレーター (DocGen) は、MagicDraw の MDK プラグインのモジュールです。 MagicDraw の UML/SysML モデルから正式なドキュメントを生成する機能を提供します。「ドキュメント」は、モデルのビュー、または階層的な方法で構造化されたモデル データの表現です。ドキュメントは段落、セクション、分析の集合であり、コンテンツの順序とレイアウトが重要です。 DocGen は MagicDraw 内で動作し、ドキュメントの「アウトライン」をたどって情報を収集し、分析を実行して、出力をファイルに書き込みます。 DocGen は DocBook XML ファイルを生成します。このファイルは変換ツールに入力され、PDF、HTML、またはその他の形式でドキュメントを生成できます。

DocGen は次のもので構成されます。

- ドキュメント フレームワークを作成するための要素を含む UML プロファイル・ドキュメント フレームワークを横断し、分析を実行し、出力を生成するためのスクリプトのセット・ユーザーがドキュメントの正確さと完全性を検証するのに役立つツールのセット。

ユーザーは時間をかけてドキュメントの内容と形式を定義する必要があります。ただし、これが完了すると、モデル データが更新されるたびにボタンをクリックするだけでドキュメントを作成できます。これらはすべて PDF、HTML などへの変換中に自動的に実行されるため、ユーザーはセクションに番号を付けたり、不本意な書式設定に格闘したりして時間を無駄にする必要はありません。DocGen は拡張することもでき、再利用可能な分析関数の形式でクエリを追加することもできます。[プロジェクトに使用できるドキュメント フレームワークがあるかどうかを確認してください。](#)

1.2 DocGen を使用したオフライン コンテンツの作成

EMS サーバーでモデルが使用できない場合は、MDK プラグインの DocGen モジュールを使用してオフライン ドキュメントを作成できます。このセクションでは、MagicDraw でオフライン の読み取り専用ドキュメントを作成する方法について説明します。これらのドキュメントは、EMS で公開されているドキュメントのように編集できないことに注意してください。すべての変更は MagicDraw モデルで直接行う必要があります。

1.2.1 ドキュメントの生成

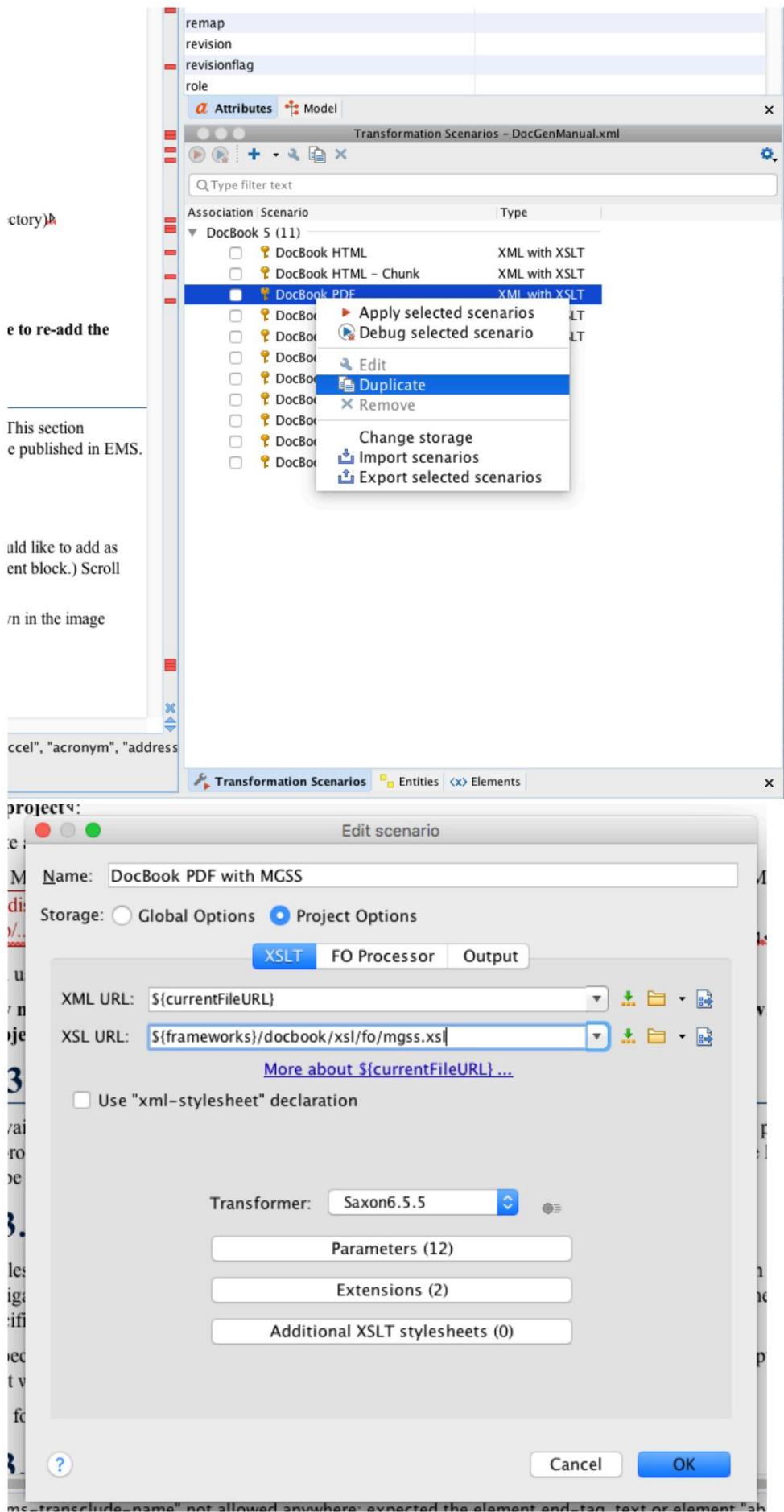
DocGen を使用すると、EMS サーバーにアクセスせずにドキュメントのローカル バージョンを生成できます。ドキュメントを選択し、右クリックメニュー「DocGen>Generate DocGen3 Document」を使用します。 DocBook XML ファイルが生成されます。[プロジェクトに、推奨される生成方法があるかどうかを確認してください。](#)ほとんどの実務者は、oxygen などの .xml ビューアを使用して、.xml ファイルを PDF ファイル（または必要に応じて他の種類のファイル）に変換します。 oxygen は SSCAE からインストールできます。ドキュメントは静的であることに注意してください。ドキュメントを変更したい場合は、MagicDraw モデルに変更を加え、ドキュメントを再生成する必要があります。

oxygen で .xml 出力を開くには、まず MBEE から mgss.xsl スタイルシートのコピーを取得します（MBEE から Crushinator のバンドル インストールをダウンロードした場合、mgss.xsl は、DocGenUserScripts -> DocGenStyleSheet の下のインストール フォルダーにあります）。

oxygen インストール ディレクトリを開き、<oxygen_dir>/frameworks/docbook/xsl/fo に移動し、mgss.xsl をそのディレクトリに保存します。このスタイルシートにより、MagicDraw からの前付入力を PDF 上で表示できるようになります。

これで、保存した .xml 出力ファイルを oxygen で開くことができます。ファイルを開いた後、以下の図に示すように、変換シナリオ内の「DocBook PDF」オプションを右クリックします。表示されるメニューで「複製」を選択します。必要に応じて、新しいシナリオの名前を変更します。選択する

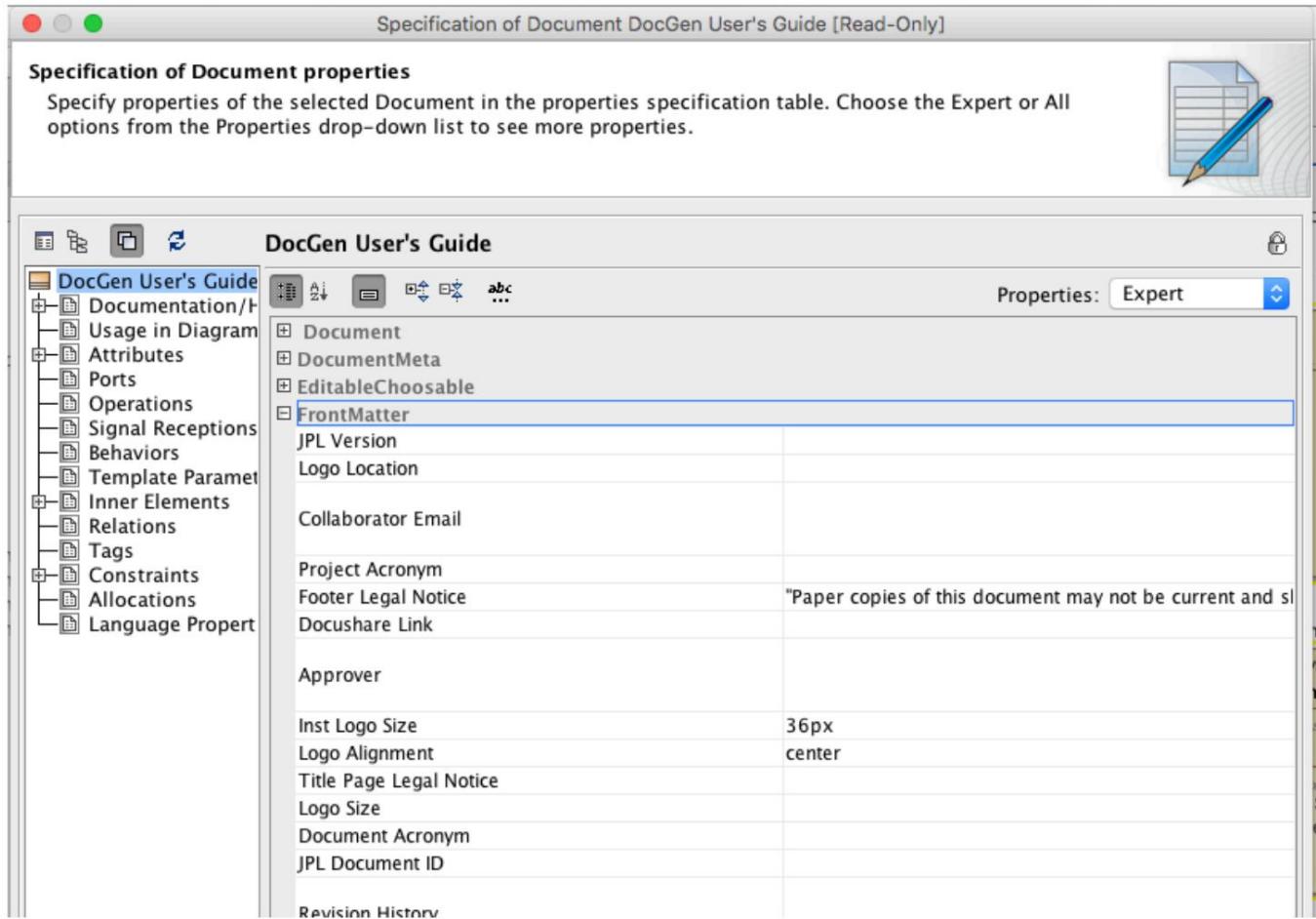
XSL URL テキスト ボックスに入力し、下図のように「\${frameworks}/docbook/xsl/fo/docbook_custom.xsl」を \${frameworks}/docbook/xsl/fo/mgss.xsl に置き換えて、「OK」をクリックします。」



新しい PDF を生成するには、「変換シナリオ」ウィンドウで新しく作成した変換をダブルクリックします。以下に示すように DocBook PDF が選択されている場合は、赤い「再生」ボタンをクリックして PDF を生成することもできます。

1.2.2 DocGen スタイルシートの使用

DocGen スタイルシートを使用すると、ユーザーは文書に前付情報を入力できます。前付として追加する情報を入力するには、ドキュメント ブロックの仕様ウィンドウに移動します。(最も簡単な方法は、ドキュメント ブロックをダブルクリックすることです。) 以下に示すように、仕様ウィンドウを「FrontMatter」セクションまで下にスクロールします。



ここでは、ロゴのサイズと位置、承認者、法的フッターなどのさまざまなフィールドを指定できます。タグのリファレンス カードを以下に示します。

1.3 視点メソッドの作成

ビューポイントの作成には 2 つの手順が必要です。まず、視点要素を作成します。次に、視点の視点メソッド図を定義します。

両方を作成して定義した後、ビューにビューを適用できます。次のサブセクションでは、Cookbook ライブラリによって提供される DocGen および式メソッドを含む、各ステップについて詳しく説明します。

1.3.1 モデル要素の収集/並べ替え/フィルタリング

ビューに公開されると、要素は 3 種類のビューポイント演算子(収集、並べ替え、フィルター)によって操作できるようになります。これらの操作を使用して、視点メソッドで使用される要素のコレクションを拡張または絞り込むことができます。

次のビューは、生成時に含まれるサブビューとなるビュー ダイアグラム、ビューポイント メソッド ダイアグラム、およびブロック定義ダイアグラムを示します。

以下は、説明されている DocGen メソッド（特に、収集/並べ替え/フィルター メソッド）の概要表です。

表 1. DocGen メソッド

メソッド名	メソッドの説明
CollectOwnedElements 無限の深さ	「CollectOwnedElements」は、ビューポイント メソッドで使用されるモデル内の公開された要素が所有するものをすべて収集します。 この特定のメソッドは無限の深さを扱います。つまり、指定された深さだけの要素（レベル 1 の子など）ではなく、所有されているすべての要素を収集します。 参照: CollectOwnedElements
CollectOwnedElements 単一の深さ	「CollectOwnedElements」は、ビューポイント メソッドで使用されるモデル内の公開された要素が所有するものをすべて収集します。 この特定のメソッドは単一の深さを処理します。つまり、すべての子ではなく、指定された深さの子を収集します。 参照: CollectOwnedElements
所有者を収集する	「CollectOwners」は本質的に「CollectOwnedElements」の逆です。ビューポイント メソッドで使用するために、公開された要素のすべての所有者（包含ツリーから）を収集します。 参考: CollectOwners
CollectThingsOnDiagram	「CollectThingsOnDiagram」は、図に描かれているすべての要素を収集します。 参照: CollectThingsOnDiagram
CollectByStereotypeProperties リスト	「CollectByStereotypeProperties」は、ステレオタイプのプロパティを収集します。 この特定のメソッドは、ステレオタイプ「List」のプロパティを収集します。 参照: CollectByStereotypeProperties
CollectByStereotypeProperties テーブル	「CollectByStereotypeProperties」は、ステレオタイプのプロパティを収集します。 この特定のメソッドは、ステレオタイプ「Table」のプロパティを収集します。 参照: CollectByStereotypeProperties
CollectByDirectedRelationshipMetaclasses この特定方向出力リファレンス: CollectByDirectedRelationshipMetaclasses	CollectByDirectedRelationshipMetaclasses アクションは、他の要素が接続するために使用する関係に基づいて要素を収集します。 このメソッドは、「方向出力」関係に基づいて要素を収集します。
CollectByDirectedRelationshipMetaclasses この特定参照の方向: CollectByDirectedRelationshipMetaclasses	CollectByDirectedRelationshipMetaclasses アクションは、他の要素が接続するために使用する関係に基づいて要素を収集します。 このメソッドは、「方向イン」関係に基づいて要素を収集します。
CollectByDirectedRelationshipStereotypes ステレオタイプ	「CollectByDirectedRelationshipStereotypes」は、要素を他の要素に接続する関係のステレオタイプに基づいて要素も収集します。 参照: CollectByDirectedRelationshipStereotypes
協会による収集	「CollectByAssociation」は、複合、共有、またはなしのいずれかの集約でブロックを収集します。 参照: CollectByAssociation
タイプの収集	CollectTypes は、すでに収集されている要素のタイプを収集します。ほとんどの場合、CollectOwnedElements または CollectOwners を使用して要素を収集します。 参照: CollectTypes

メソッド名	メソッドの説明
分類子属性の収集	「CollectClassifierAttributes」はクラスの属性を収集します。 リファレンス: CollectClassifierAttributes
CollectByExpression	「CollectByExpression」は、オブジェクト制約言語 (OCL) を使用してモデルをクエリする、よりカスタマイズされたアプローチです。 参考: CollectByExpression
FilterByDiagramType 画像	「FilterByDiagramType」アクティビティは、データ セットを調べて、図である要素を調べます。フィルター処理する要素を収集するには、最初に収集操作を使用する必要があります。 参考: FilterByDiagramType
名前によるフィルター	「FilterByNames」アクティビティは、データ セットを調べて、データ セット内の特定の名前を持つすべての要素、または特定の名前を持つ要素に接続されている要素を検索します。フィルター処理する要素を収集するには、最初に収集操作を使用する必要があります。 参考: FilterByNames
FilterByMetaclasses	「FilterByMetaclasses」アクティビティは、データ セットを調べて、指定されたメタクラスによって定義されているすべての要素を検索します。フィルター処理する要素を収集するには、最初に収集操作を使用する必要があります。 リファレンス: FilterByMetaclasses
ステレオタイプによるフィルター	「FilterByStereotypes」アクティビティは、データ セットを調べて、指定されたステレオタイプによって定義されているすべての要素を検索します。フィルター処理する要素を収集するには、最初に収集操作を使用する必要があります。 参考: FilterByStereotypes
FilterByExpression	「FilterByExpression」アクティビティは、データ セットを調べて、オブジェクト制約言語 (OCL) 式を満たす (布尔値) すべての要素を検索します。フィルター処理する要素を収集するには、最初に収集操作を使用する必要があります。 参考: FilterByExpression
属性で並べ替え	「SortByAttribute」を使用すると、ユーザーは選択した属性 (名前、ドキュメント、値) によってデータ セットを並べ替えることができます。 参考: SortByAttribute
プロパティで並べ替え	「SortByProperty」を使用すると、ユーザーは指定されたプロパティによってデータ セットを並べ替えることができます。 参考: SortByProperty
式による並べ替え	「SortByExpression」は、オブジェクト制約言語 (OCL) 式で指定されたデータ セットです。 参考: SortByExpression

1.3.1.1 収集

「Collect...」は、公開されたパッケージ内の要素を、引き続き使用される要素と無視される要素に分離するビューポイント演算子です。多数の収集メソッドがあり、それぞれのビューポイント メソッド図のサイド バーに独自の呼び出し動作アクションがあります。

1.3.1.1.1 所有要素の収集

CollectOwnedElements は、ビューポイント メソッドで使用されるモデル内の公開された要素が所有するものをすべて収集します。一般的な使用例は、パッケージ内に含まれる要素のグループをビューに公開する必要がある場合です。の代わりに

各要素を個別に公開すると、ユーザーはビューポイント内で「CollectOwnedElements」を使用して、パッケージのみを公開できます。コレクション メソッドは、公開された要素を調べ、それが所有するすべての要素を返します。

[[単一深さの例](#)] ビューと [[無限深さの例](#)] ビューには、以下に示すサンプル ビュー ダイアグラムの出力が表示されます。これらのビューはどちらも同じ Lunchbox を公開しますが、異なる視点に準拠しています。ビューポイント間の唯一の違いは、「DepthChoosable」プロパティです。CollectOwnedElements Single Depth の DepthChoosable は 1、CollectOwnedElements Infinite Depth の DepthChoosable は 0 です。

CollectOwnedElements はバーツのプロパティも収集することに注意してください。この例では、FilterByStereotypes を使用して、出力にブロック要素のみを表示します。

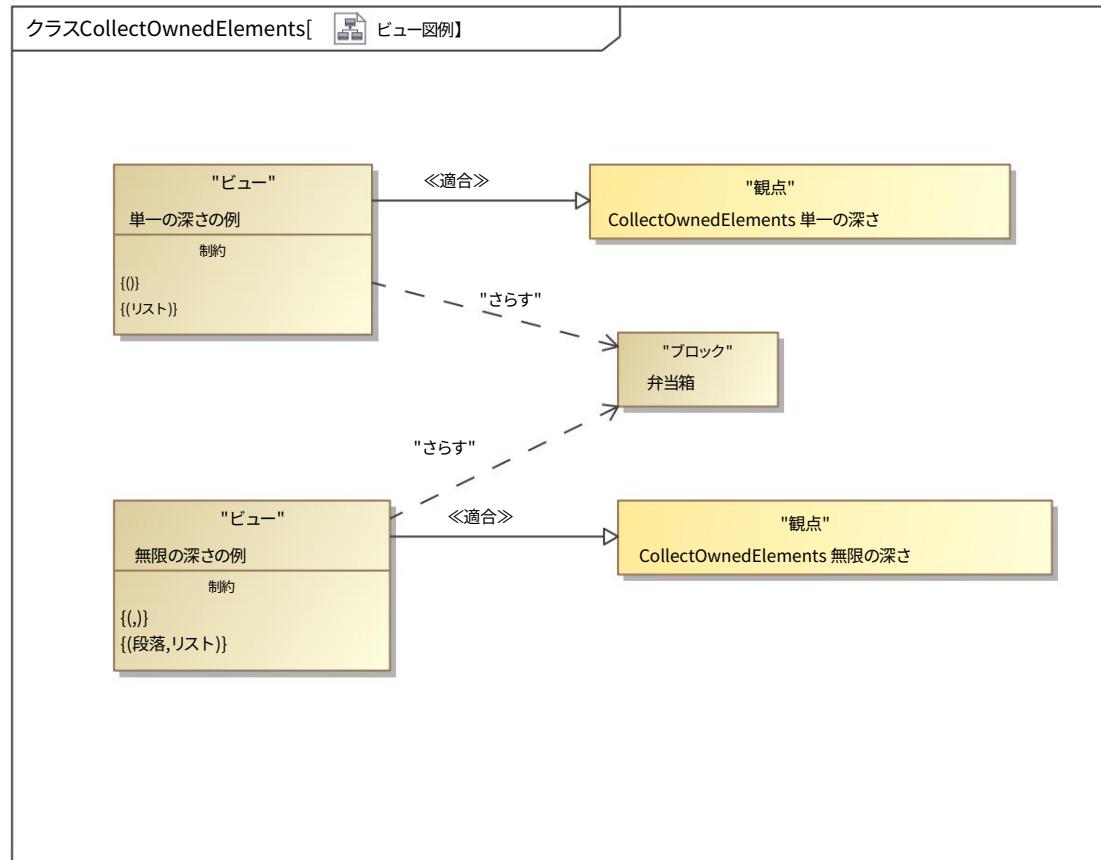


図 1. ビュー図の例

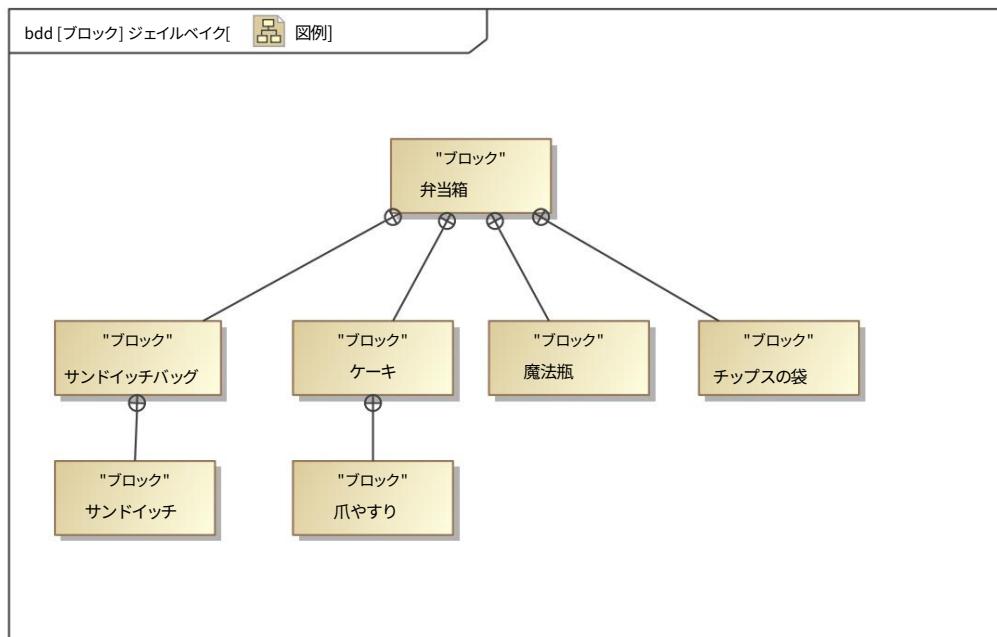


図 2. 図の例

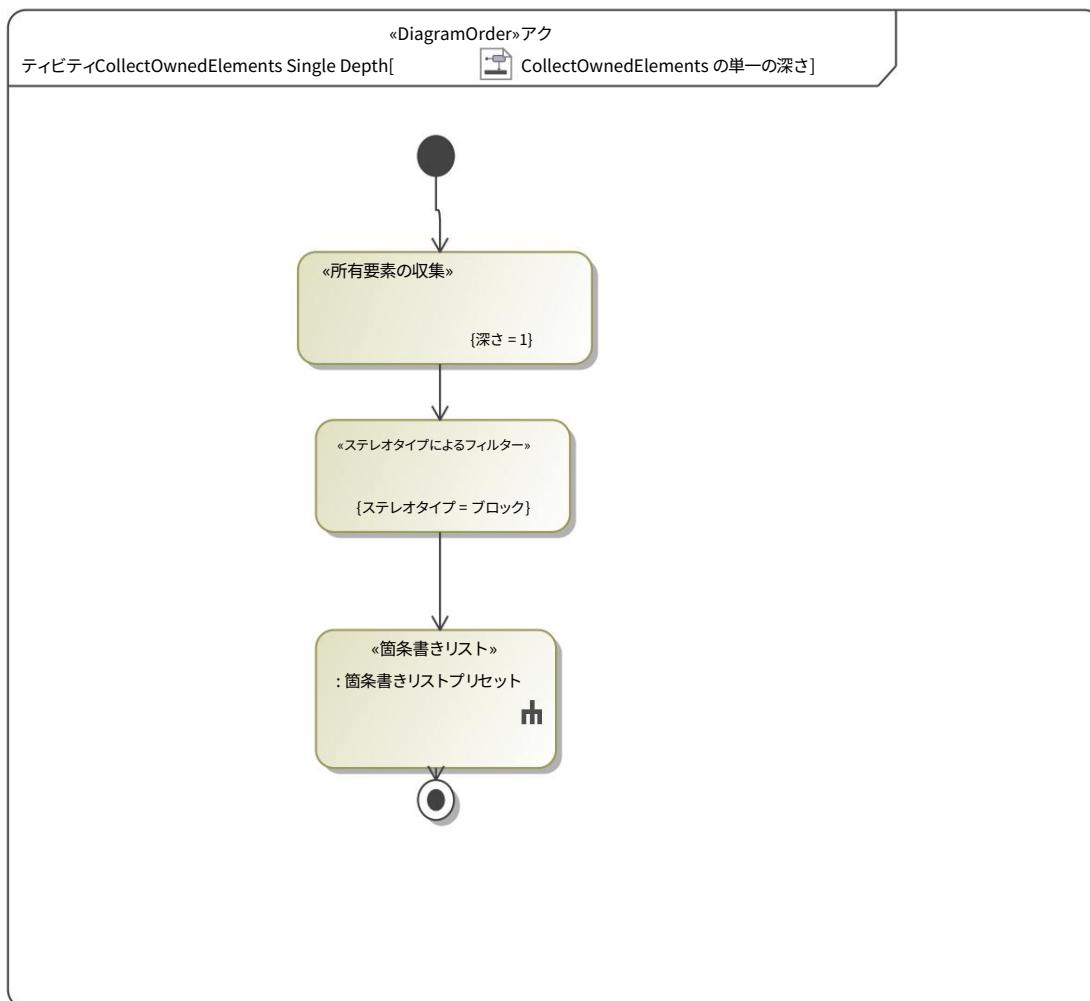


図 3. CollectOwnedElements の単一の深さ

「CollectOwnedElements」は、ビューポイント メソッドで使用されるモデル内の公開された要素が所有するものをすべて収集します。

この特定のメソッドは単一の深さを処理します。つまり、すべての子ではなく、指定された深さの子を収集します。

参照: [CollectOwnedElements](#)

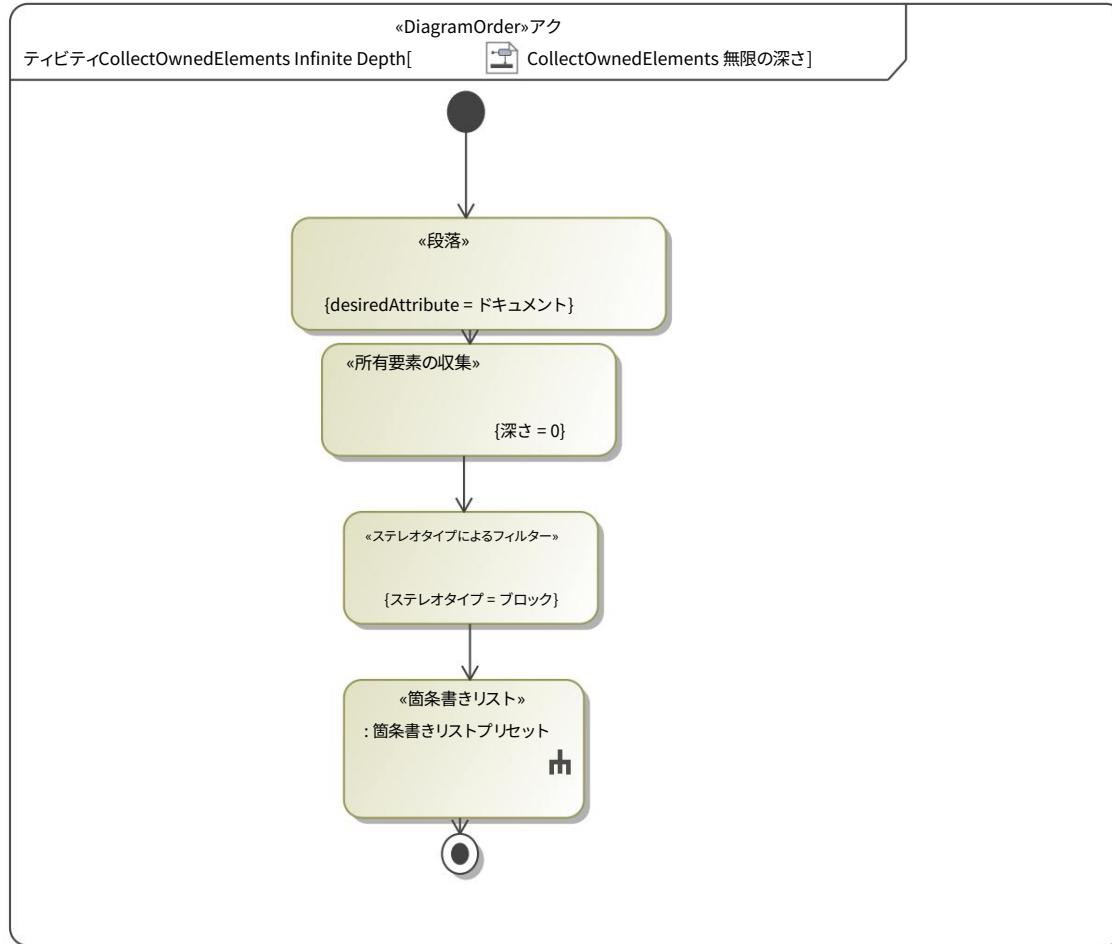


図 4. CollectOwnedElements の無限の深さ

「CollectOwnedElements」は、ビューポイント メソッドで使用されるモデル内の公開された要素が所有するものをすべて収集します。

この特定のメソッドは無限の深さを扱います。つまり、指定された深さだけの要素（レベル 1 の子など）ではなく、所有されているすべての要素を収集します。

参照: [CollectOwnedElements](#)

1.3.1.1.1 単一の深さの例

1. サンドイッチの袋 2.

魔法瓶 3. ポテト

チップスの袋 4. ケー
キ

1.3.1.1.2 無限の深さの例

1. サンドイッチの袋 2.

サンドイッチ 3.

魔法瓶 4. ポテト

チップスの袋 5. ケー
キ 6. 爪や
すり

1.3.1.1.2 所有者の収集

「CollectOwners」は本質的に「CollectOwnedElements」の逆です。ビューポイント メソッドで使用するために、公開された要素のすべての所有者 (包含ツリーから) を収集します。別のブロック「ターゲット」を使用すると、包含ツリー内の所有権のパスが返されます。「DepthChoosable」が 0 (無限) に設定されているため、これは直接所有権パッケージ (サンプル要素) から始まり、モデル全体 (データ) に戻ります。

[CollectOwners のサンプル ビュー](#)には、次の図の出力が表示されます。

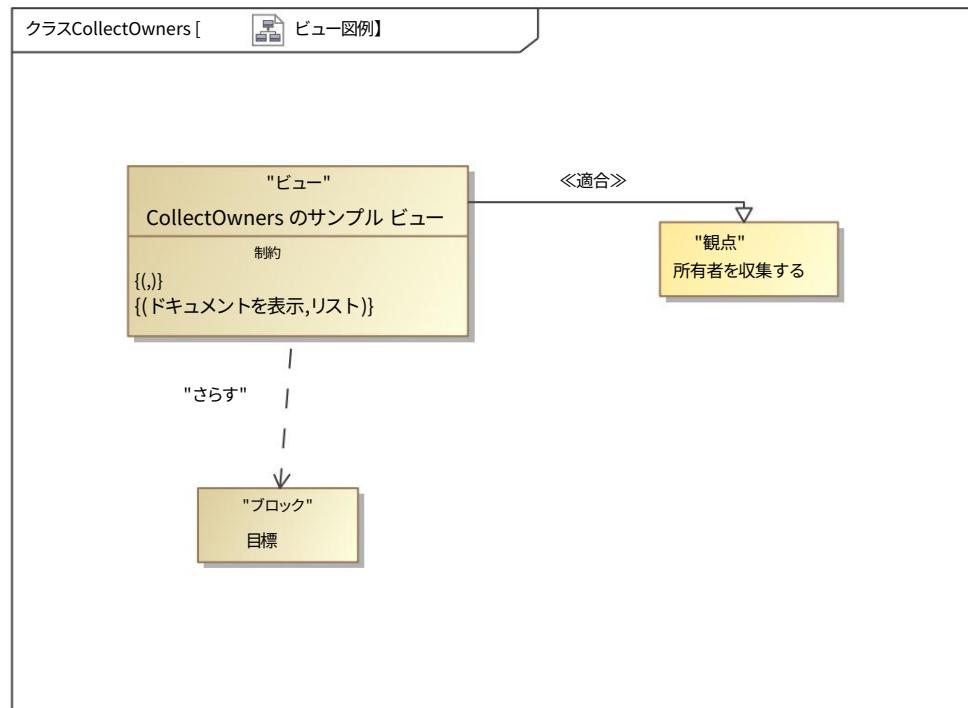


図 5. ビュー図の例

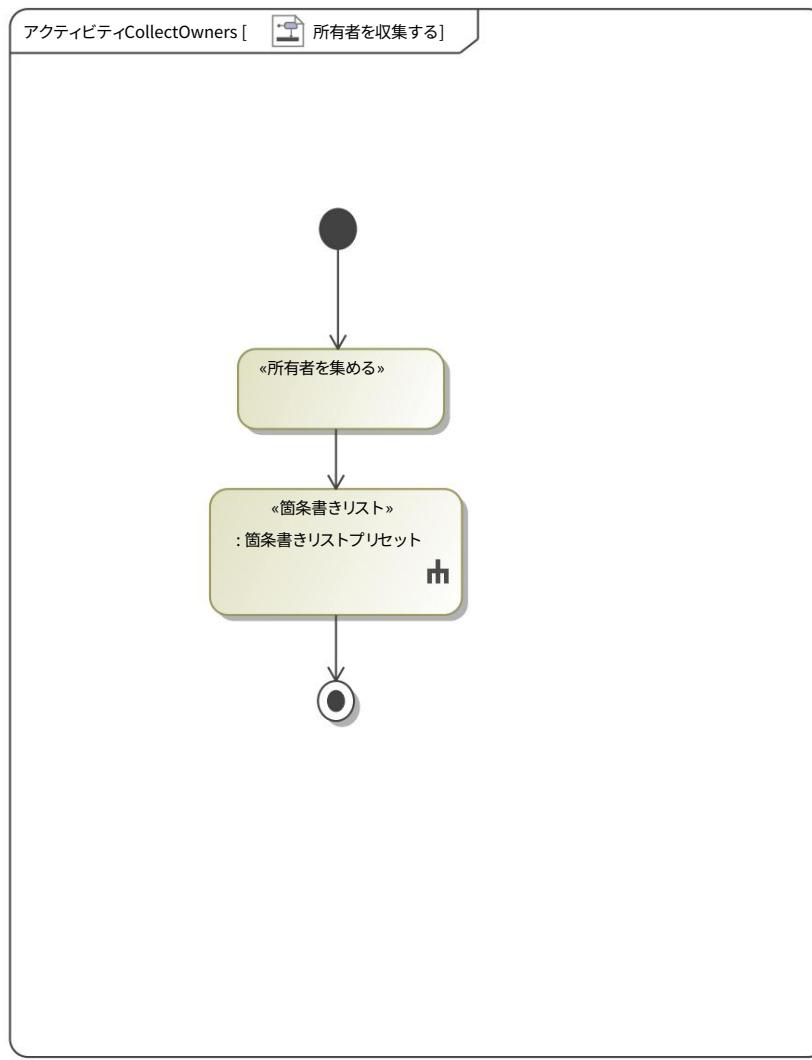


図 6. CollectOwners

「CollectOwners」は本質的に「CollectOwnedElements」の逆です。ビューポイント メソッドで使用するために、公開された要素のすべての所有者 (包含ツリーから) を収集します。

参考: [CollectOwners](#)

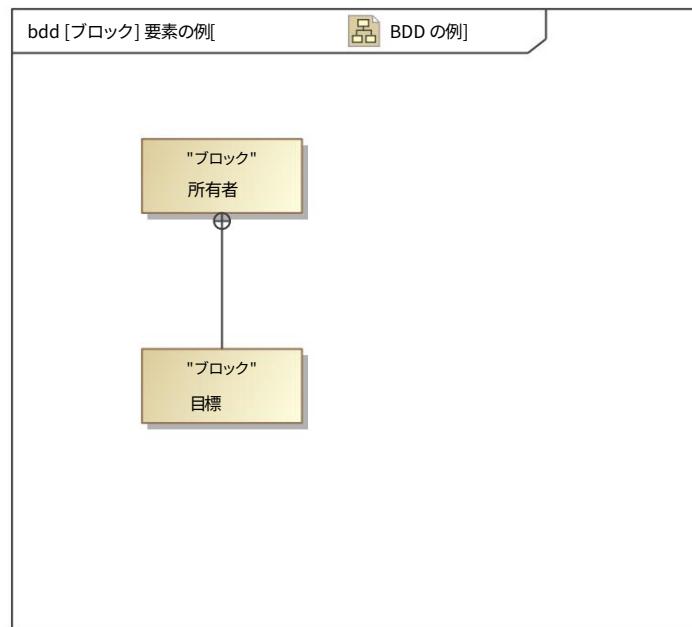


図 7. BDD の例

1.3.1.1.2.1 CollectOwners のサンプルレビュー

1. 所有者 2.
- 要素の例 3. CollectOwners
4. 収集 5. ビューポイント
- の作成メソッド
- の例 6. モデル
7. Docgen 8. データ

1.3.1.1.3 CollectThingsOnDiagram

CollectThingsOnDiagram は、図に描かれているすべての要素を収集します。これを実証するために、例の BDD が公開されました。各要素の名前と視点メソッド自体を以下に示します。多くの場合、名前は要素とともに自動的に作成されないため、これらをモデルに挿入する必要があります。そうしないと、名前の部分に「... のコンテンツがありません」というメッセージが表示されます。

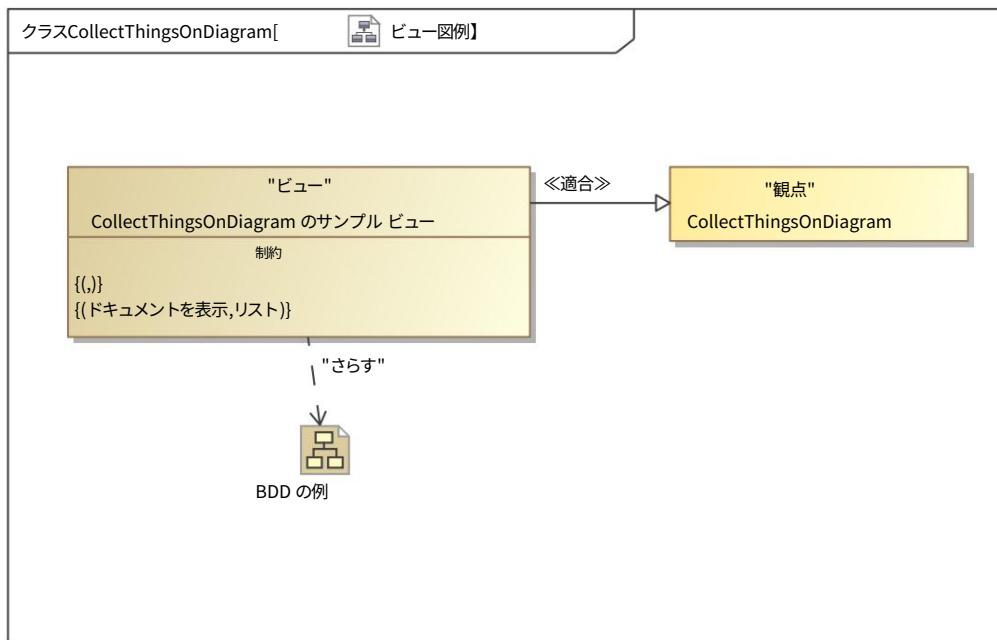


図 8. ビュー図の例

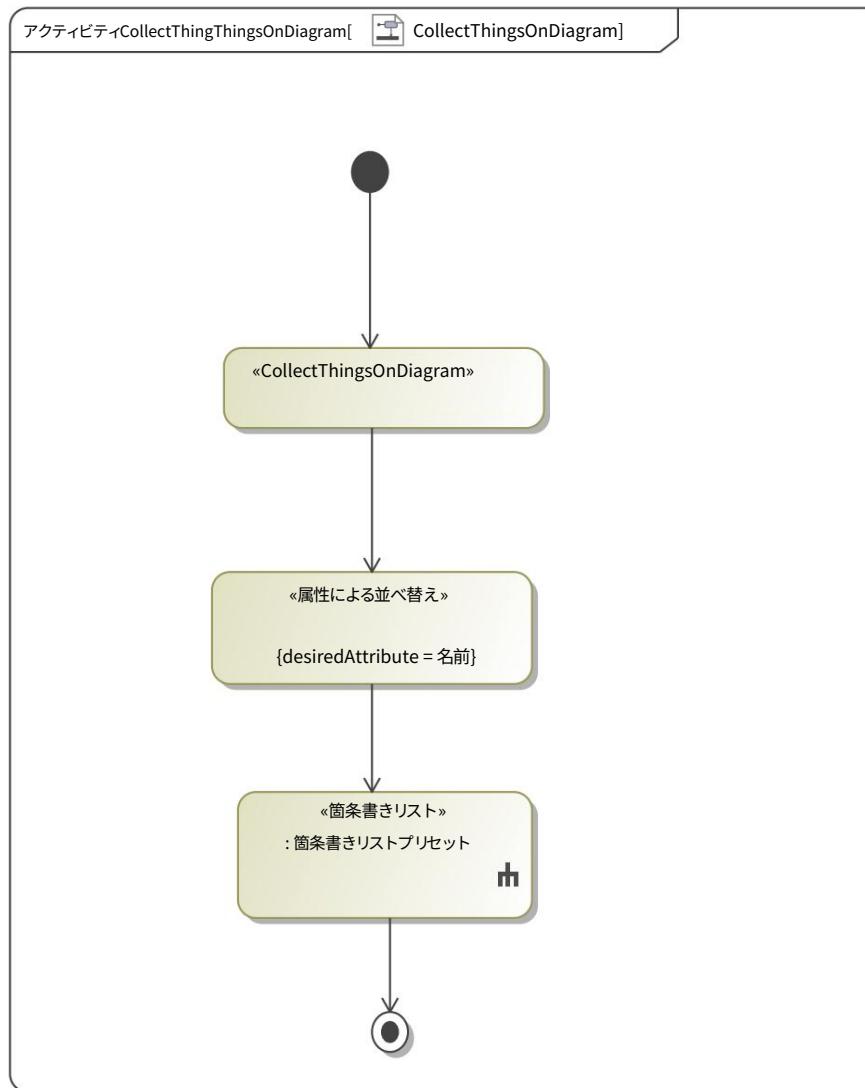


図 9. CollectThingsOnDiagram

「CollectThingsOnDiagram」は、図に描かれているすべての要素を収集します。

参照: [CollectThingsOnDiagram](#)

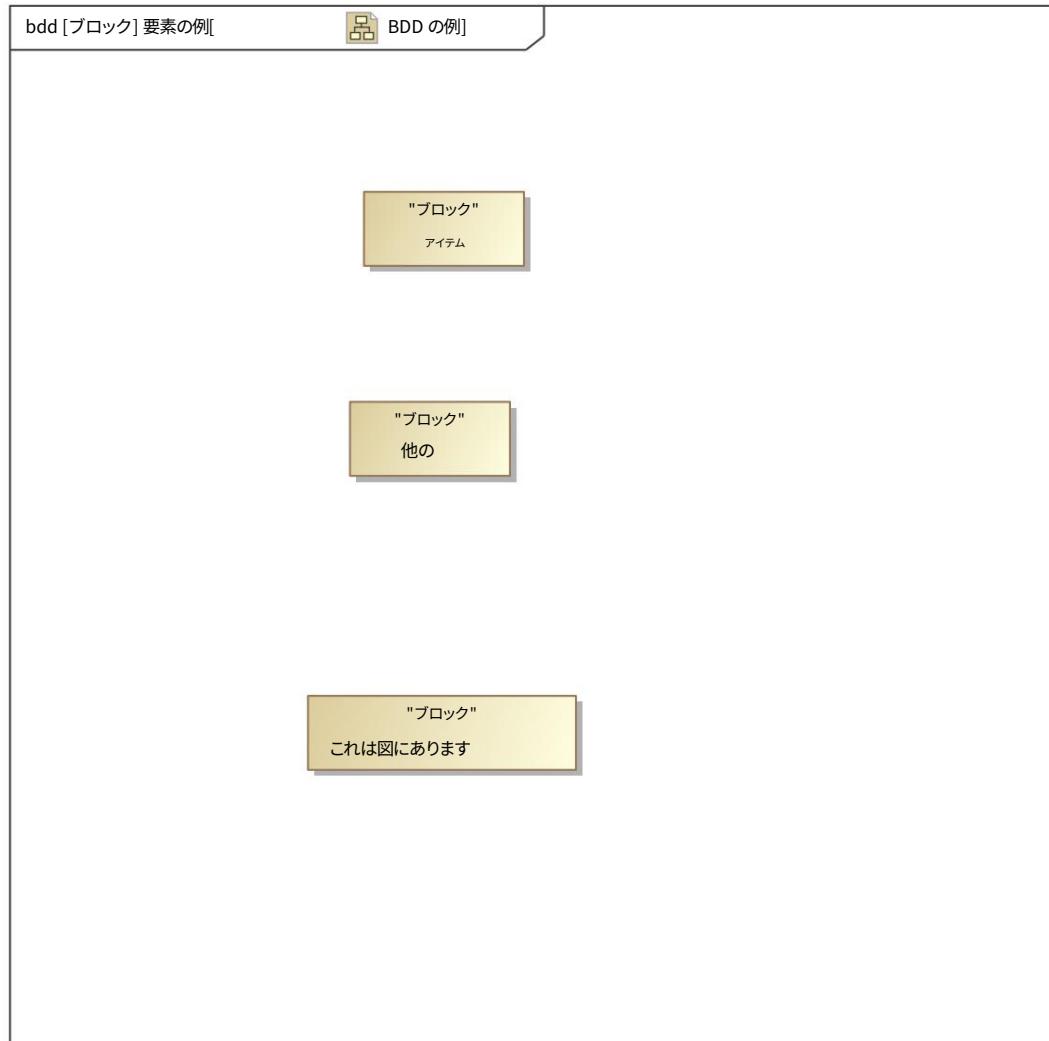


図 10. BDD の例

この図に示されていない別の要素は、このメソッドでは取得されません。

1.3.1.1.3.1 CollectThingsOnDiagram のサンプルレビュー

1. BDD の例 2. 項目 3.

その他 4.

これは図に

あります

1.3.1.1.4 CollectByStereotypeProperties

CollectByStereotypeProperties は、ステレオタイプのプロパティを収集します。次のビューの [箇条書きリストの例](#) と [テーブル構造の例](#) は、以下のビュー図の例で説明されています。これらのビュー構造の出力を示しています。

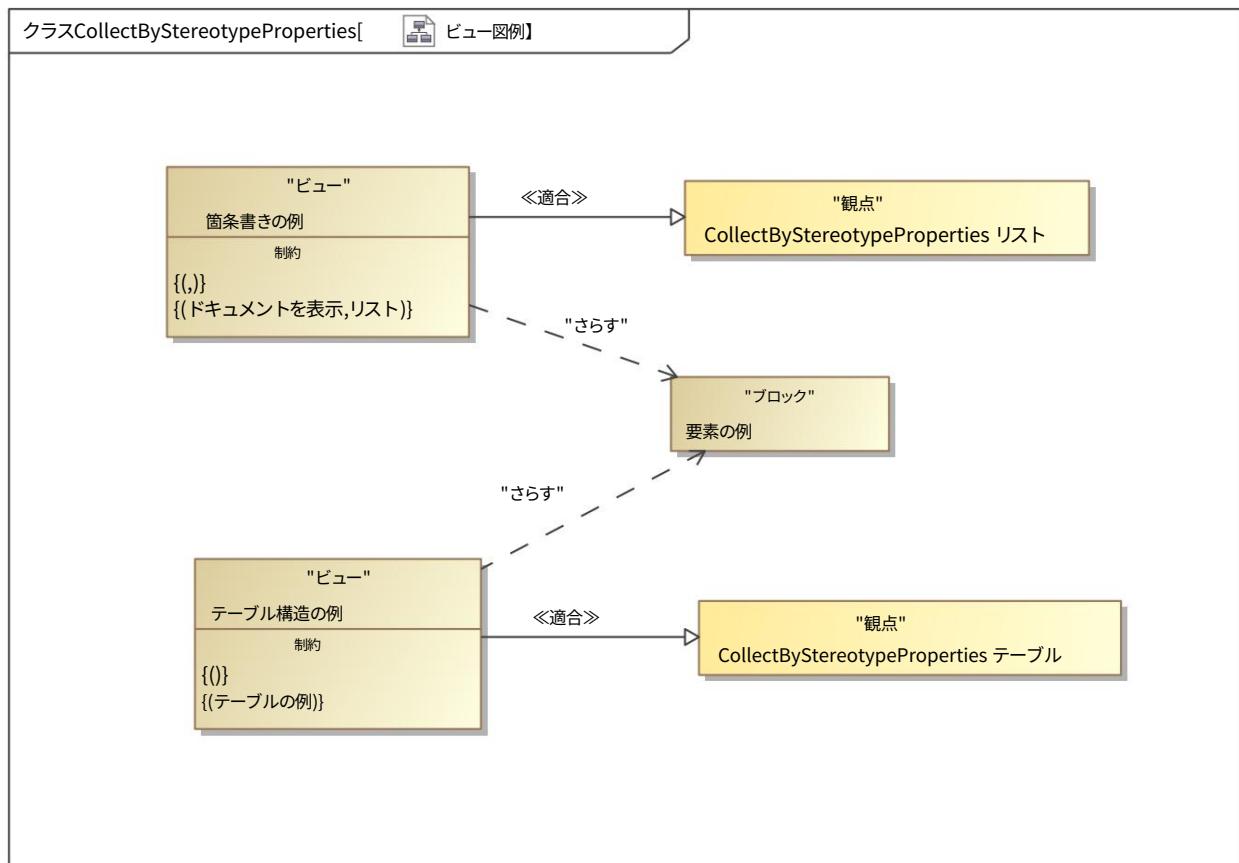


図 11. ビュー図の例

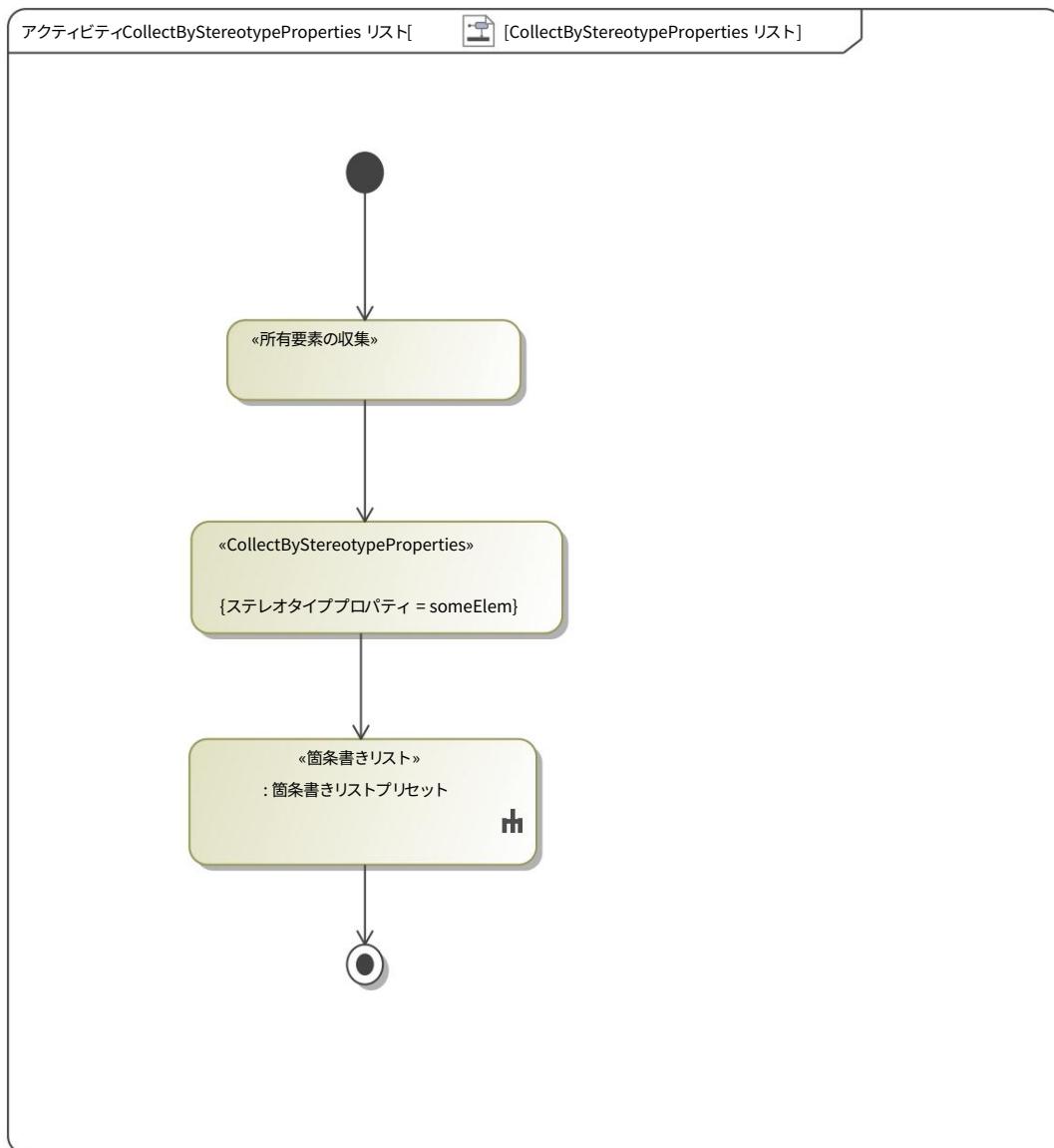


図 12. CollectByStereotypeProperties リスト

「CollectByStereotypeProperties」は、ステレオタイプのプロパティを収集します。

この特定のメソッドは、ステレオタイプ「List」のプロパティを収集します。

参照: [CollectByStereotypeProperties](#)

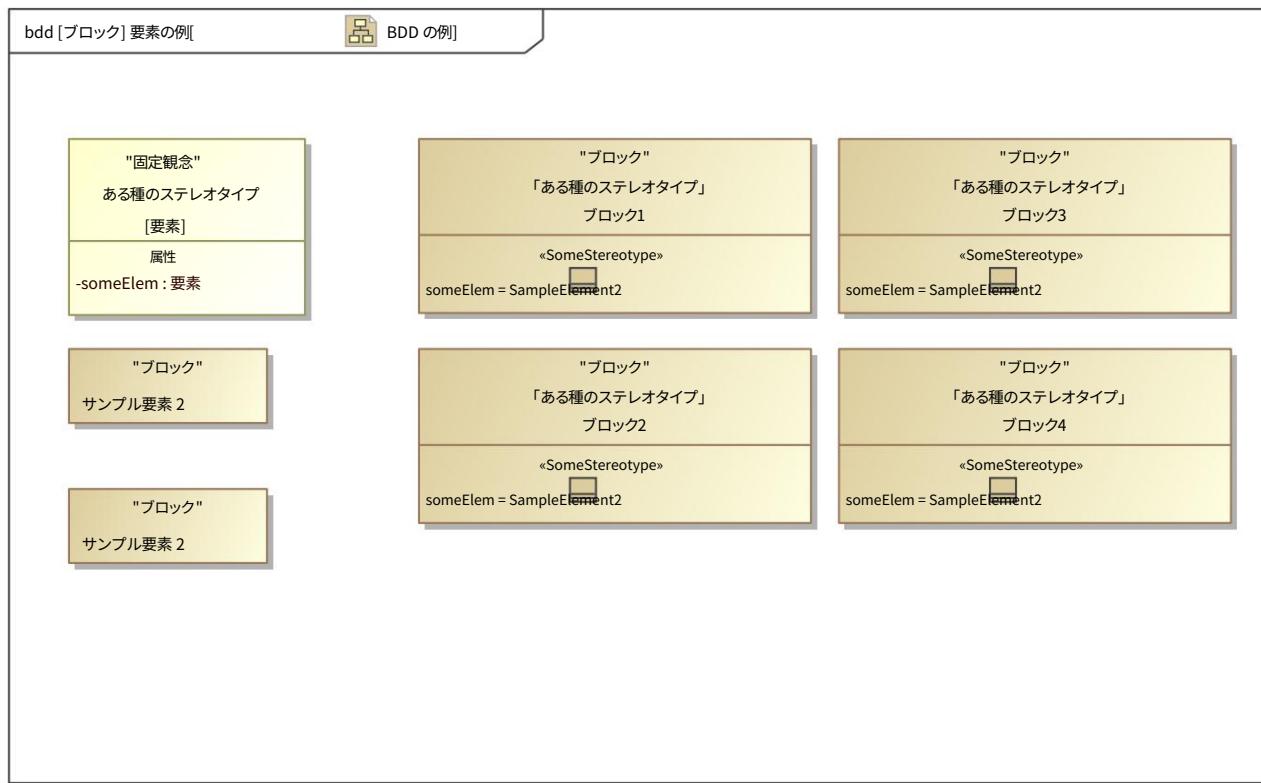


図 13. BDD の例

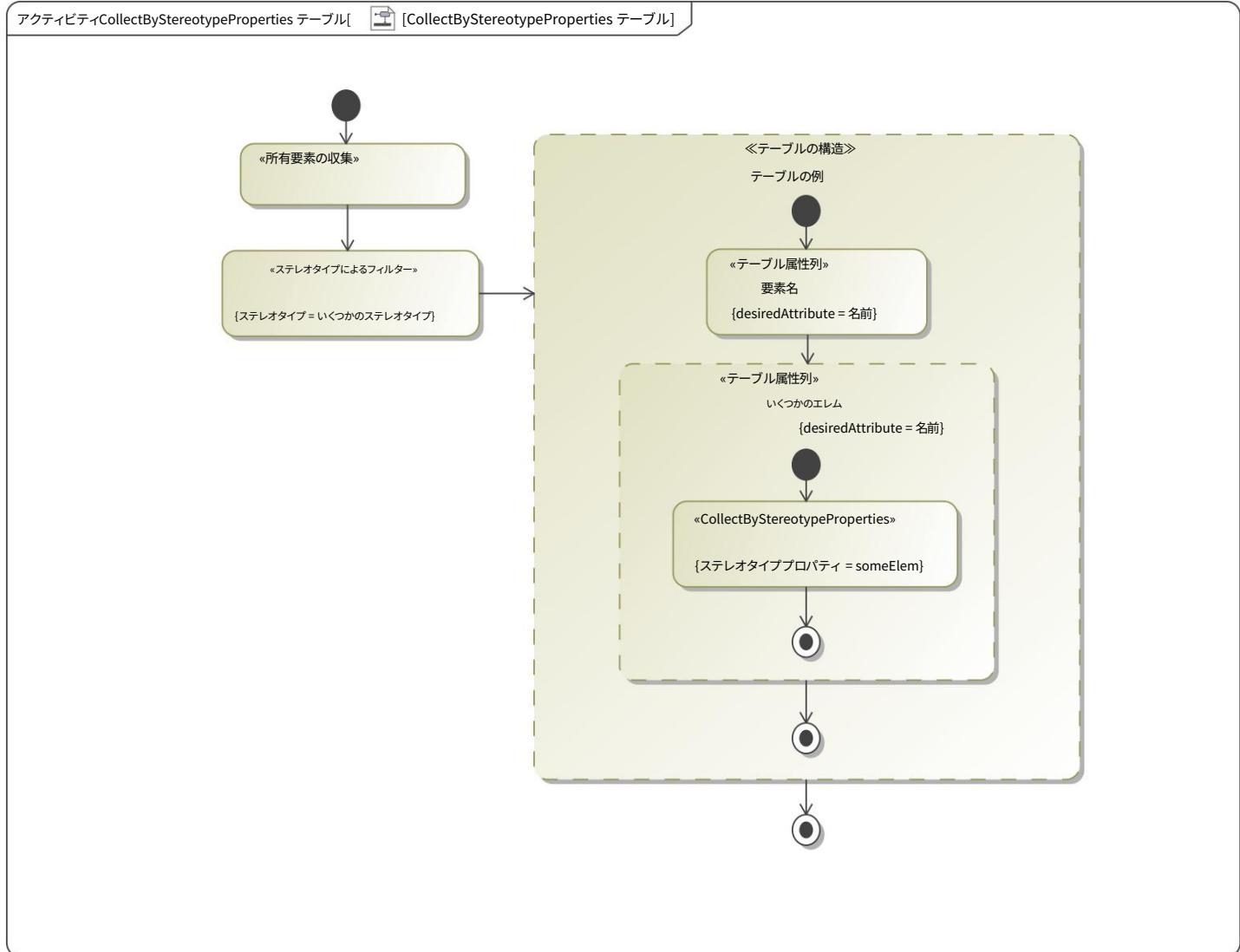


図 14. CollectByStereotypeProperties テーブル

「CollectByStereotypeProperties」は、ステレオタイプのプロパティを収集します。

この特定のメソッドは、ステレオタイプ「Table」のプロパティを収集します。

参照: [CollectByStereotypeProperties](#)

1.3.1.1.4.1 箇条書きリストの例

1. サンプル要素
2. サン
プル要素

1.3.1.1.4.2 テーブル構造の例

表 2. テーブルの例

要素名	いくつかのエレム
ブロック1	サンプル要素 2
ブロック2	サンプル要素 2
ブロック3	サンプル要素 2
ブロック4	サンプル要素 2

1.3.1.1.5 CollectByDirectedRelationshipMetaclasses

CollectByDirectedRelationshipMetaclasses アクションは、他の要素が接続するために使用する関係に基づいて要素を収集します。この例では、Hello と Athena が「依存関係」(破線の矢印) で Hera に接続されていることに注意してください。言い換えれば、ヘラはヘロとアテナに依存しています。この例ではこれらが唯一の依存関係であるため、Example Elements パッケージを公開し、所有されている要素を収集し、この演算子をメタクラス "Dependency" で使用すると、Hello と Athena のみが収集されます。依存関係の元である Hera は、この演算子では収集されないことに注意してください。

仕様の MetaclassChoosable で [...] を選択すると、要素セレクターが開き、目的のメタクラスを選択できるようになります。左下隅でメタクラス オプションが選択されていることを確認してください。選択されていない場合、メタクラスは検索に表示されません。また、Magic Draw で使用している拡張機能、プラグインなどによっては、同じ名前の複数のメタクラス (つまり、組み込みのメタクラスと追加のカスタム メタクラス) が存在する可能性があることに注意してください。モデルで使用しているメタクラスの選択には注意してください。

この演算子の最終結果を次のビューに示します。

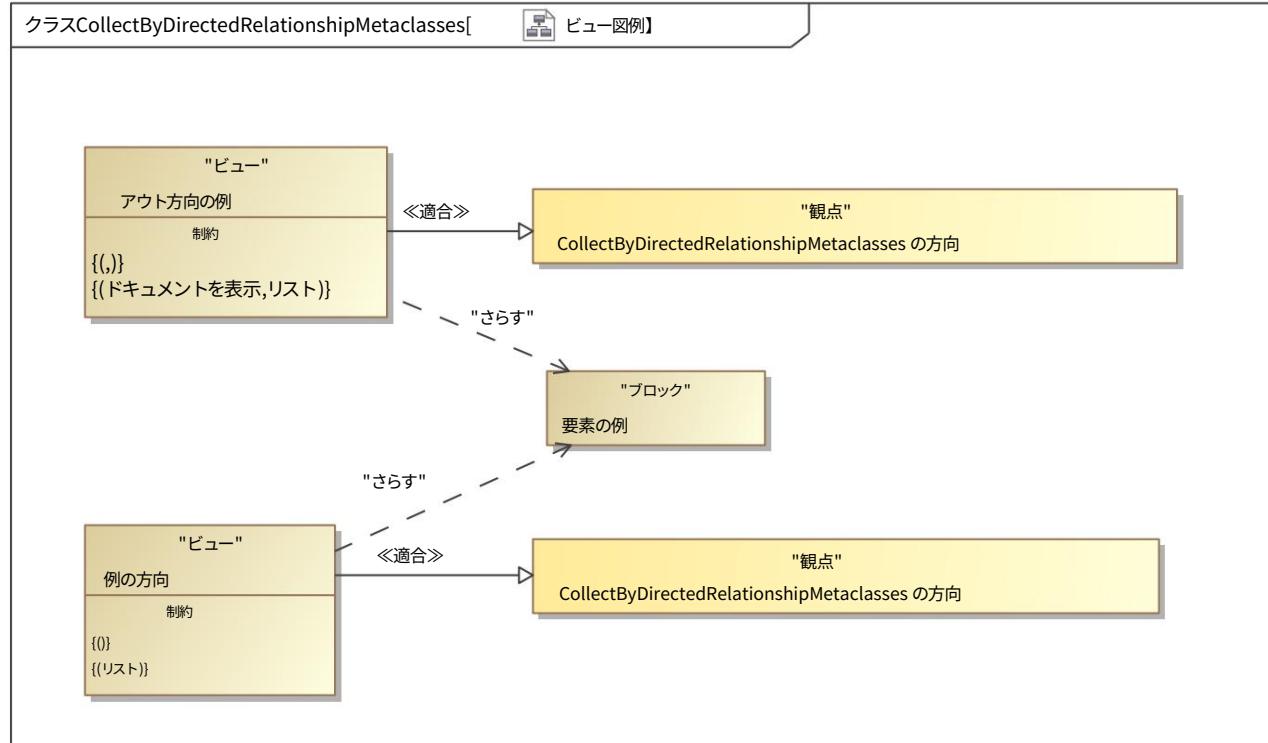


図 15. ビュー図の例

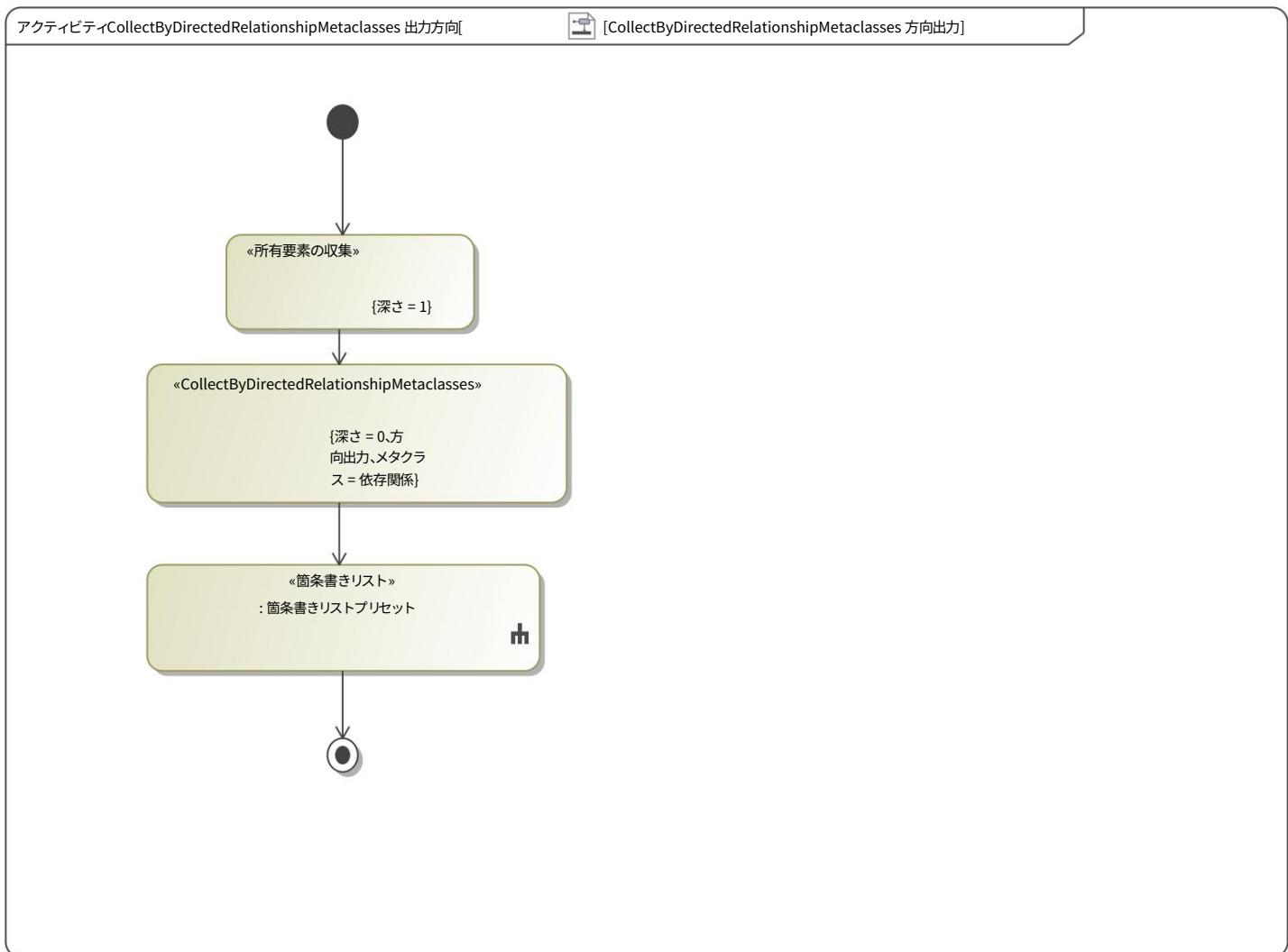


図 16. CollectByDirectedRelationshipMetaclasses の出力方向

CollectByDirectedRelationshipMetaclasses アクションは、他の要素が接続するために使用する関係に基づいて要素を収集します。

この特定のメソッドは、「方向出力」関係に基づいて要素を収集します。

リファレンス: [CollectByDirectedRelationshipMetaclasses](#)

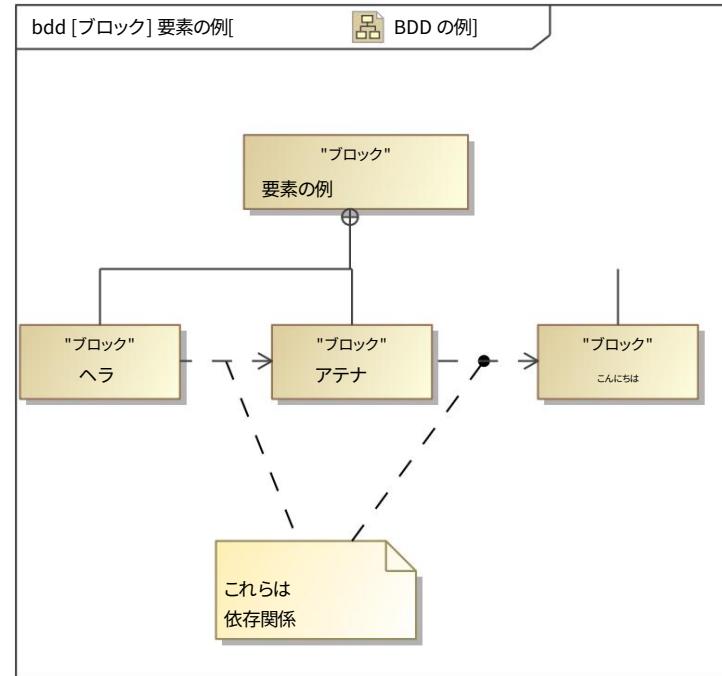


図 17. BDD の例

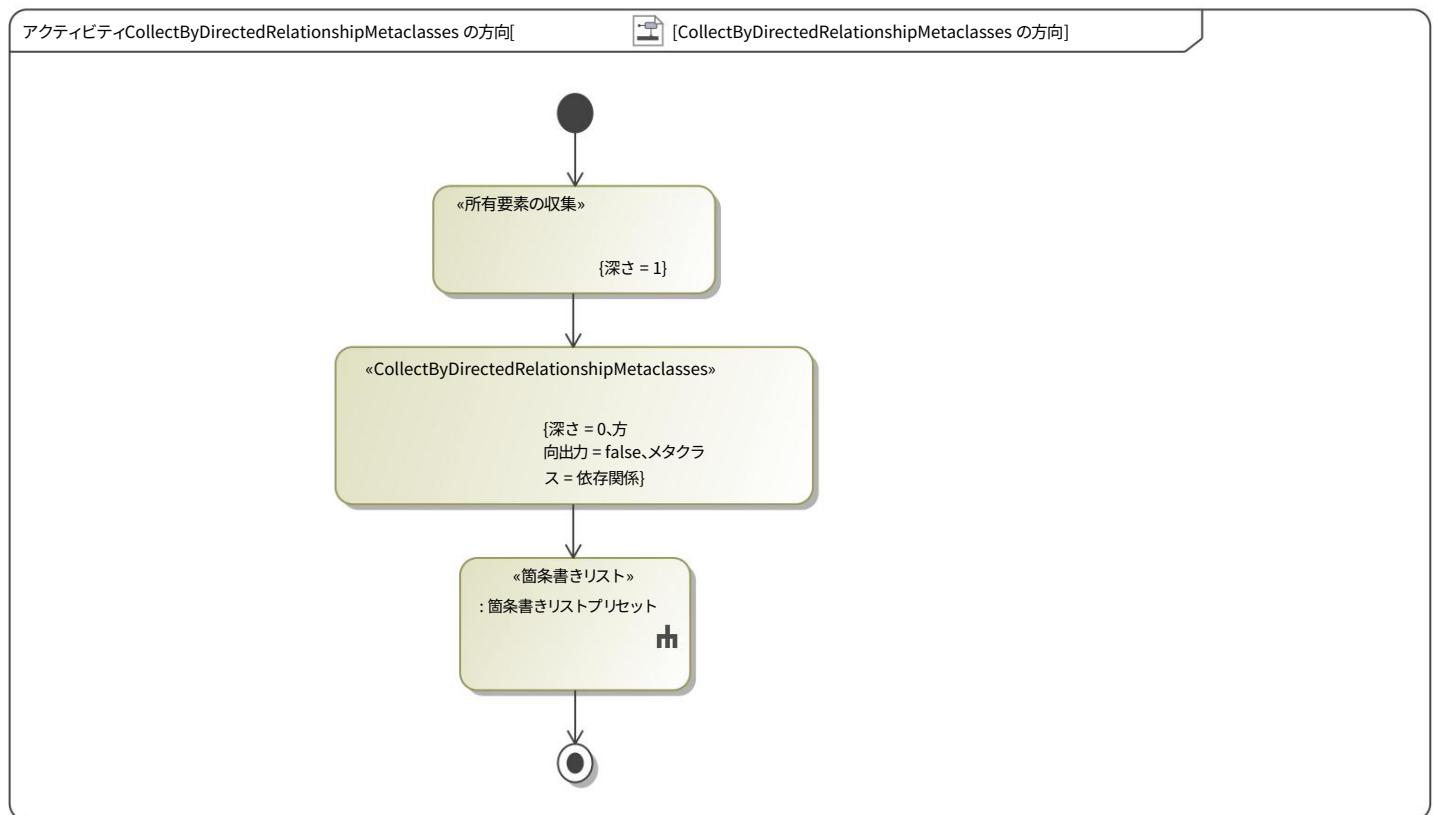


図 18. CollectByDirectedRelationshipMetaclasses の方向

CollectByDirectedRelationshipMetaclasses アクションは、他の要素が接続するために使用する関係に基づいて要素を収集します。

この特定のメソッドは、「方向イン」関係に基づいて要素を収集します。

リファレンス: [CollectByDirectedRelationshipMetaclasses](#)

1.3.1.1.5.1 出力方向の例

1. アテナ 2.
ヒーロー

1.3.1.1.5.2 例の方向

1. ヘラ 2.
アテナ

1.3.1.1.5.3 出力方向

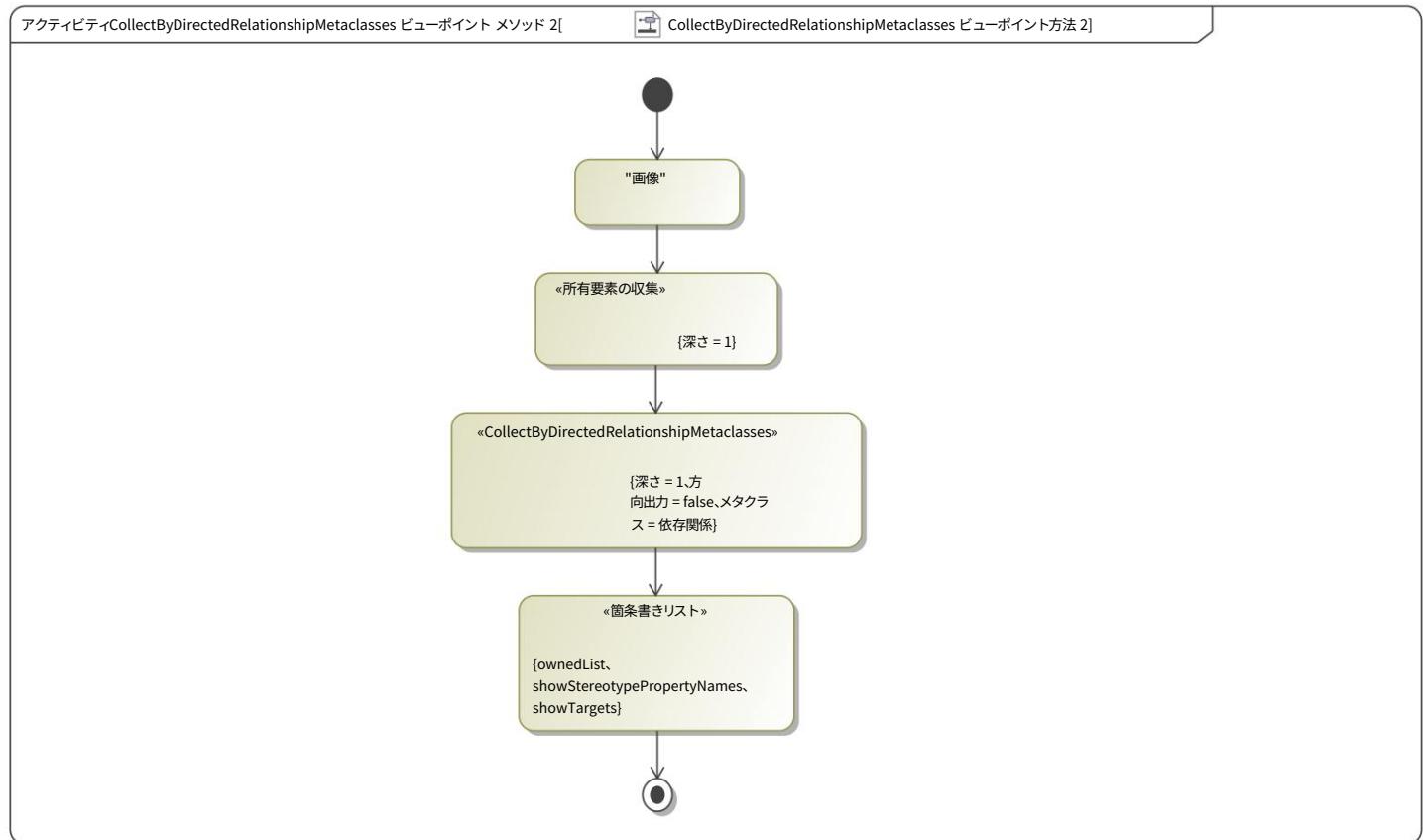


図 19. CollectByDirectedRelationshipMetaclasses ビューポイント メソッド 2

1.3.1.1.6 CollectByDirectedRelationshipStereotypes

`CollectByDirectedRelationshipStereotypes` は、要素を他の要素に接続する関係に基づいて要素も収集します。
ただし、ここでは、コレクションはメタクラスではなく関係のステレオタイプによって行われます（前のセクションと同様）。

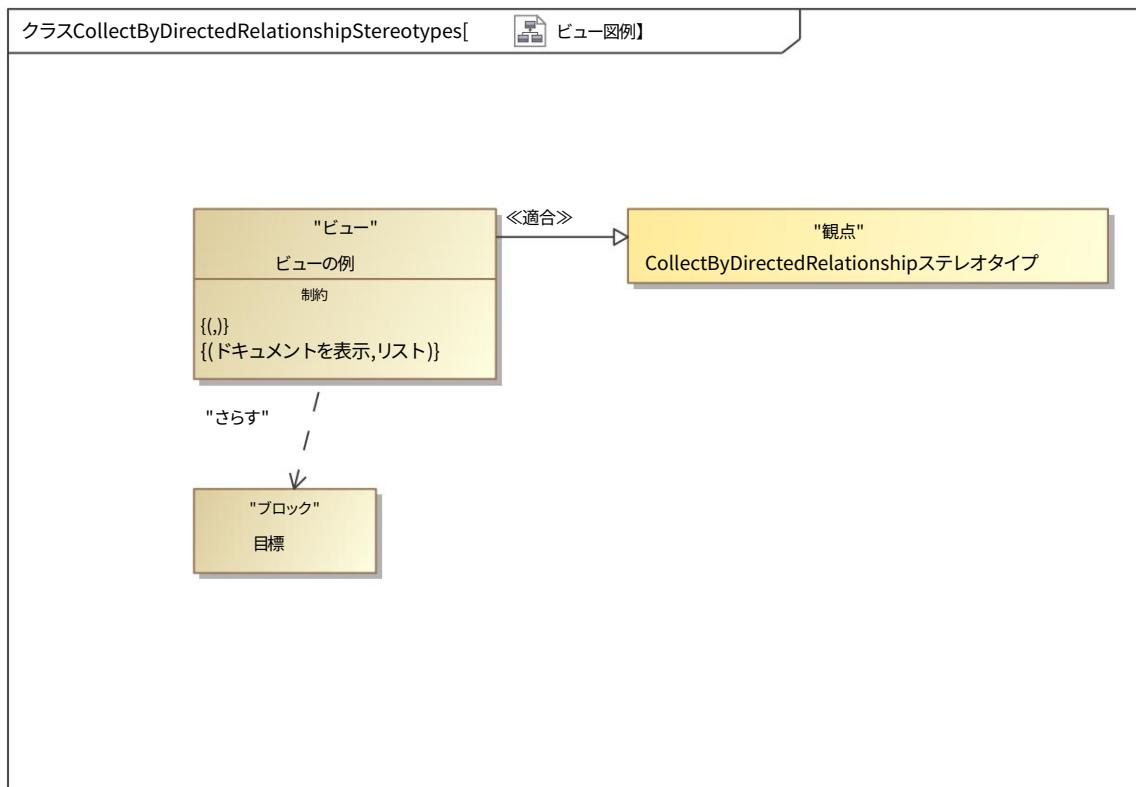


図 20. ビュー図の例

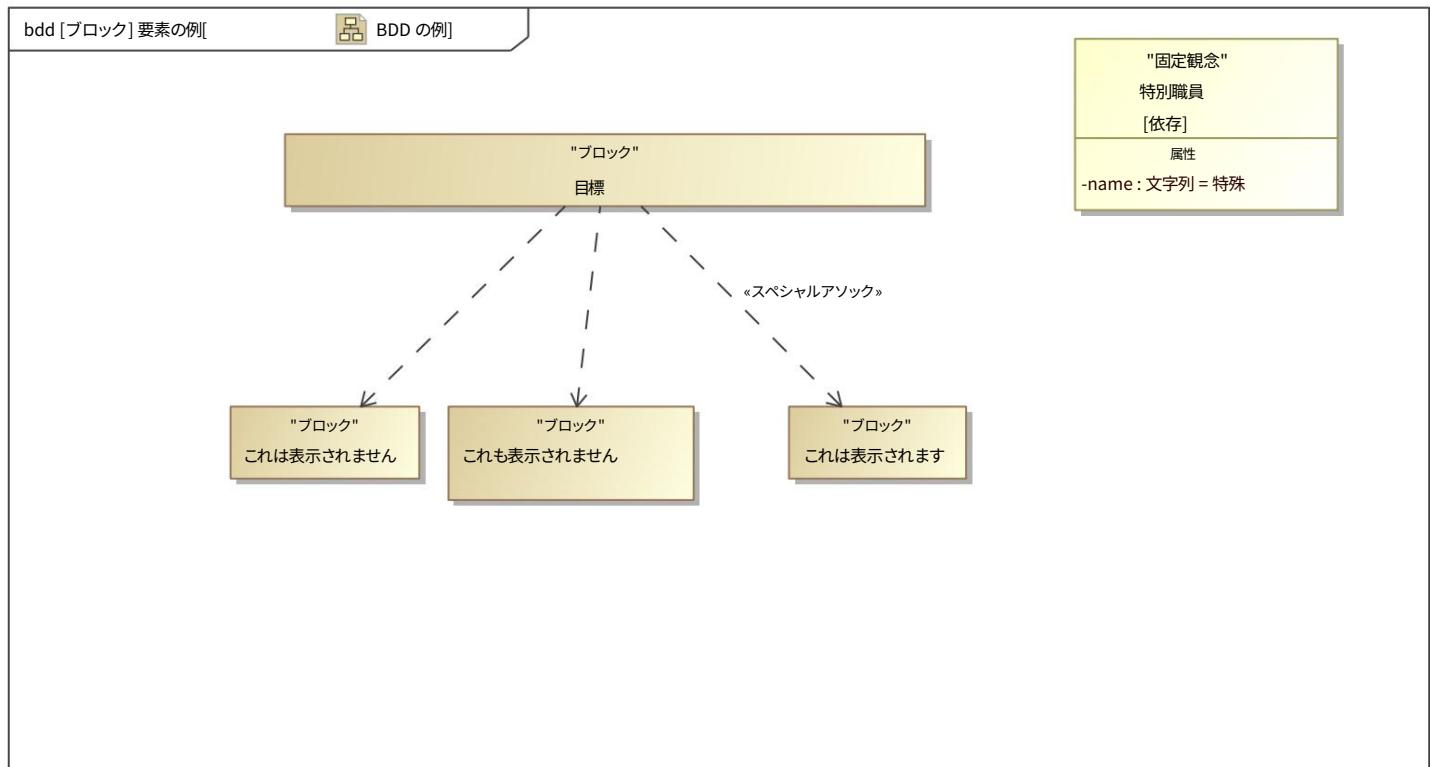


図 21. BDD の例

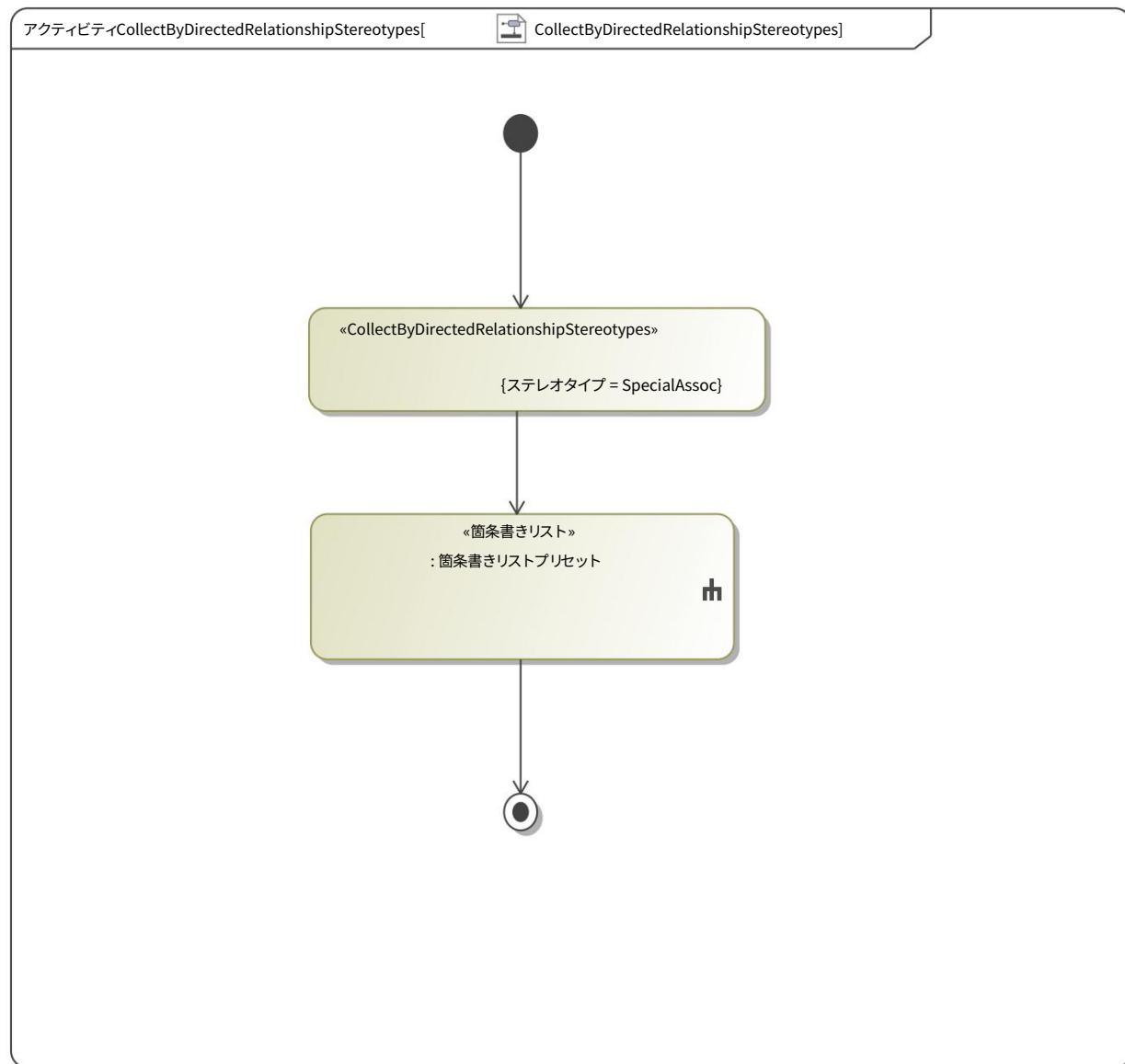


図 22. CollectByDirectedRelationshipStereotypes

「CollectByDirectedRelationshipStereotypes」は、要素を他の要素に接続する関係のステレオタイプに基づいて要素も収集します。

参照: [CollectByDirectedRelationshipStereotypes](#)

1.3.1.1.6.1 ビューの例

1. これは表示されます

1.3.1.1.7 関連付けによる収集

CollectByAssociation は、複合、共有、またはなしのいずれかの集約を使用してブロックを収集します。つまり、このアクションにより、白いひし形の関連付け（共有）と黒いひし形の関連付け（複合）の両方を収集できます。CollectByAssociation アクションは、CollectByAssociation ブロックの仕様に記載されている集計タイプを持つブロックのみを収集します。

ビュー図では、このビューはサンプル要素を公開し、CollectByAssociation ビューポイント メソッドに準拠します。この視点法の結果を次の図に示します。

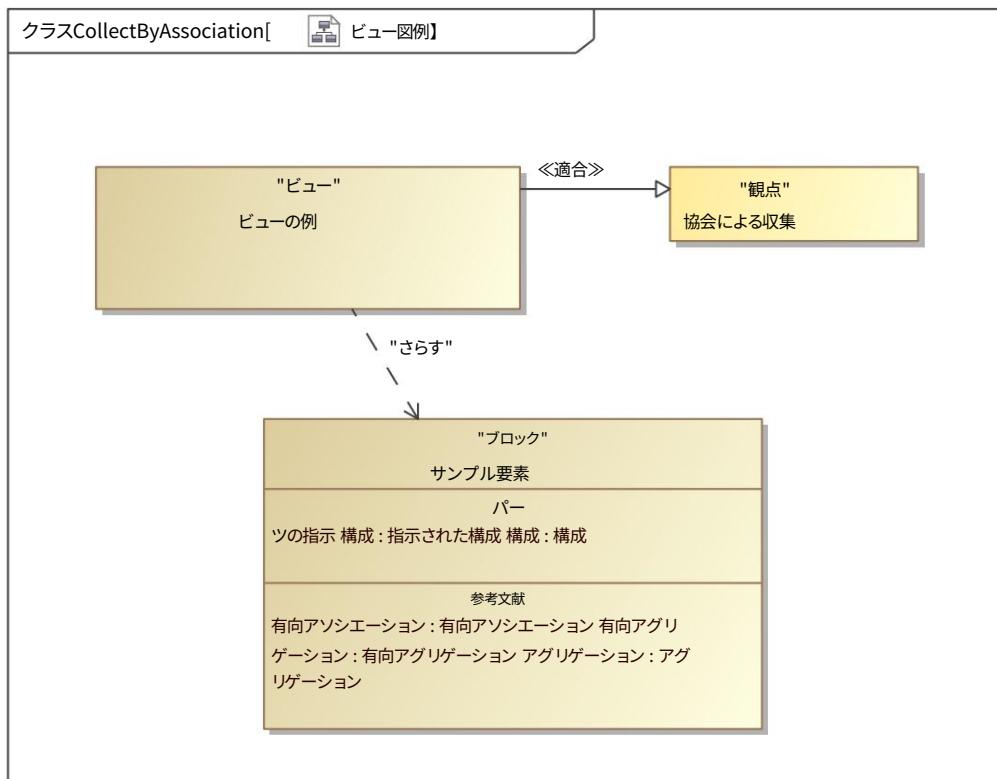


図 23. ビュー図の例

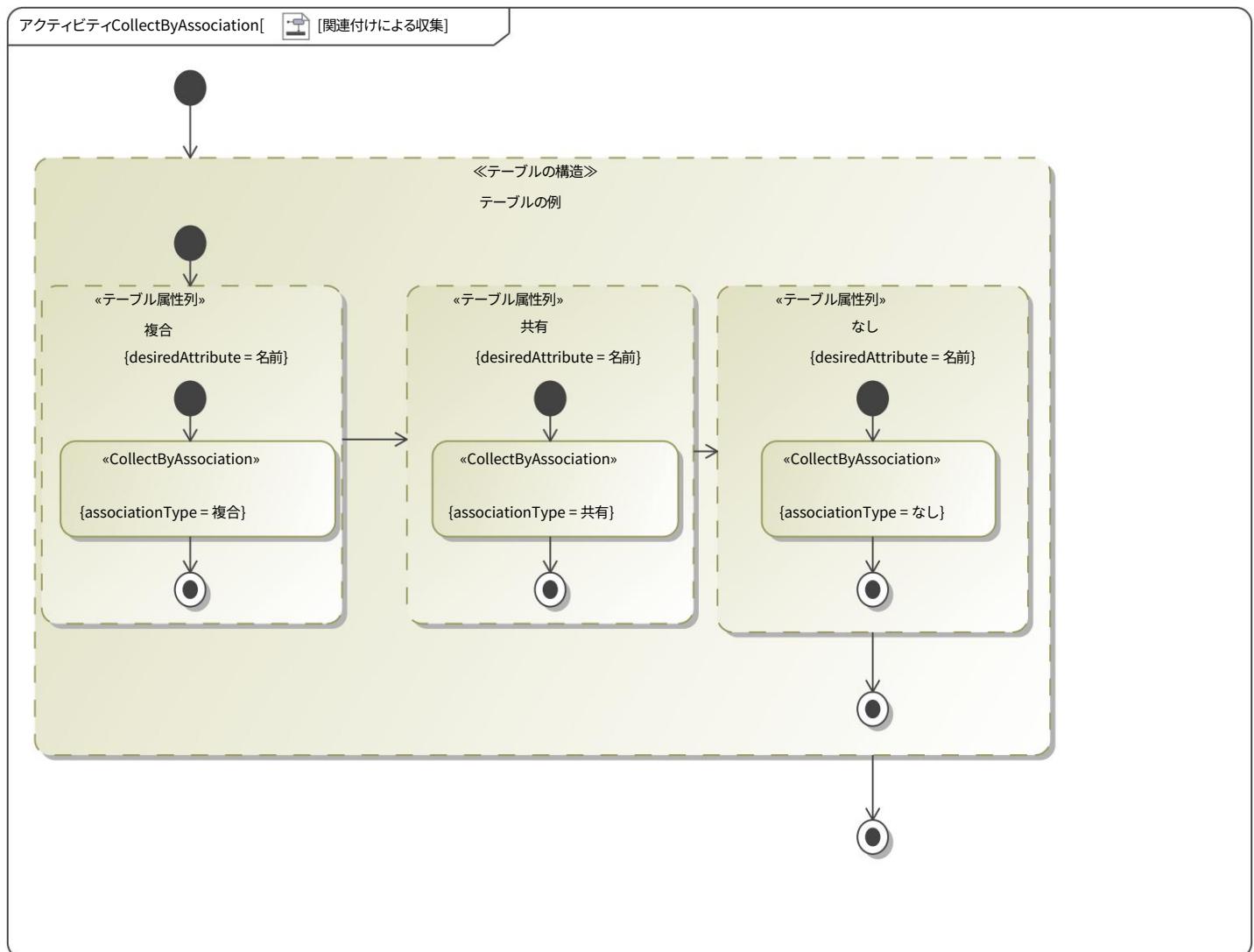


図 24.CollectByAssociation

「CollectByAssociation」は、複合、共有、またはなしのいずれかの集約でブロックを収集します。

参照: [CollectByAssociation](#)

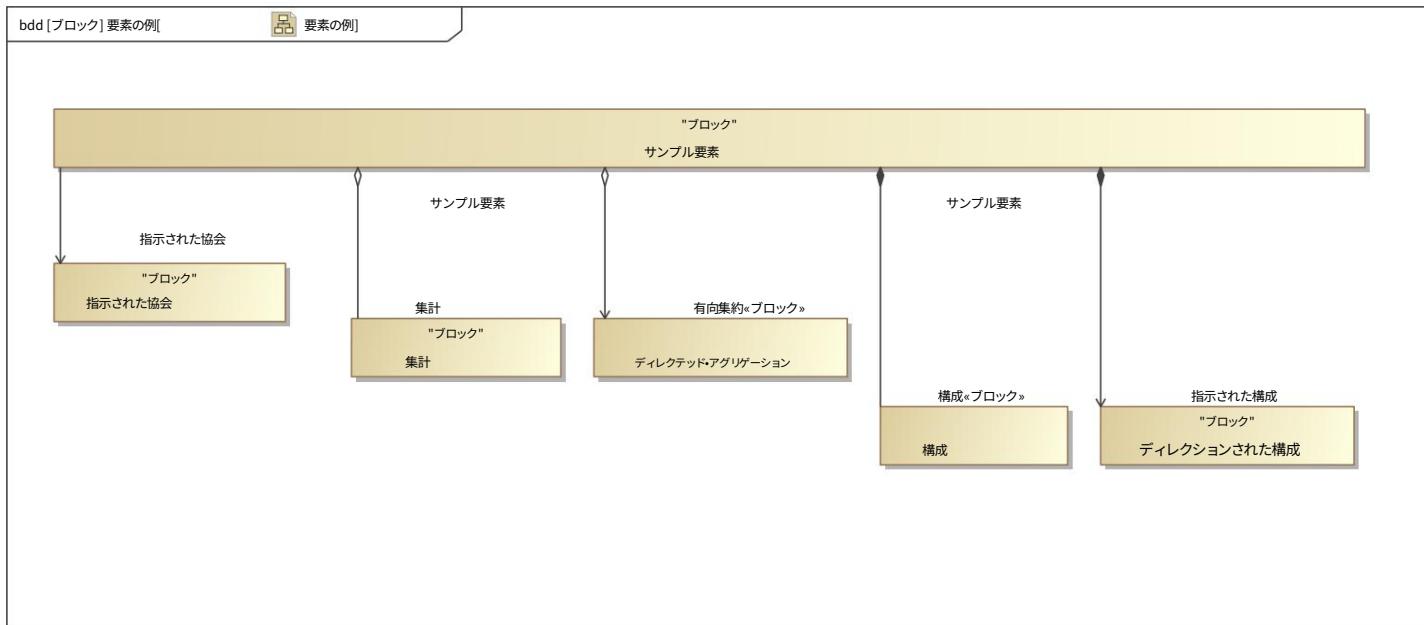


図 25. 要素の例

1.3.1.1.7.1 ビューの例

表 3. テーブルの例

複合	共有	なし
指示された構成 構成	ディレクテッド・アグリゲーション 集計	指示された協会

1.3.1.1.8 タイプの収集

CollectTypes はタイプを収集します。この場合、CollectTypes の前に、CollectOwnedElements または CollectOwners を使用して要素を収集する必要があります。次に、CollectTypes は各要素に関連付けられた型を収集します。観点法図を以下に示します。

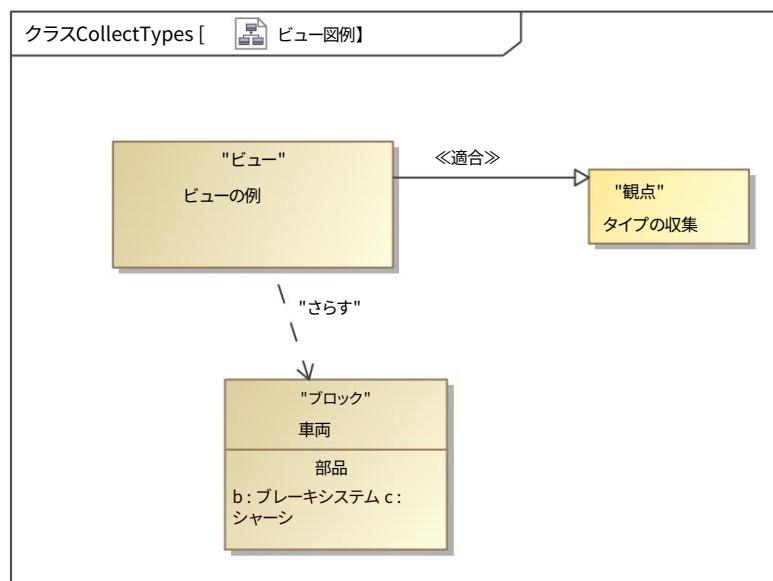


図 26. ビュー図の例

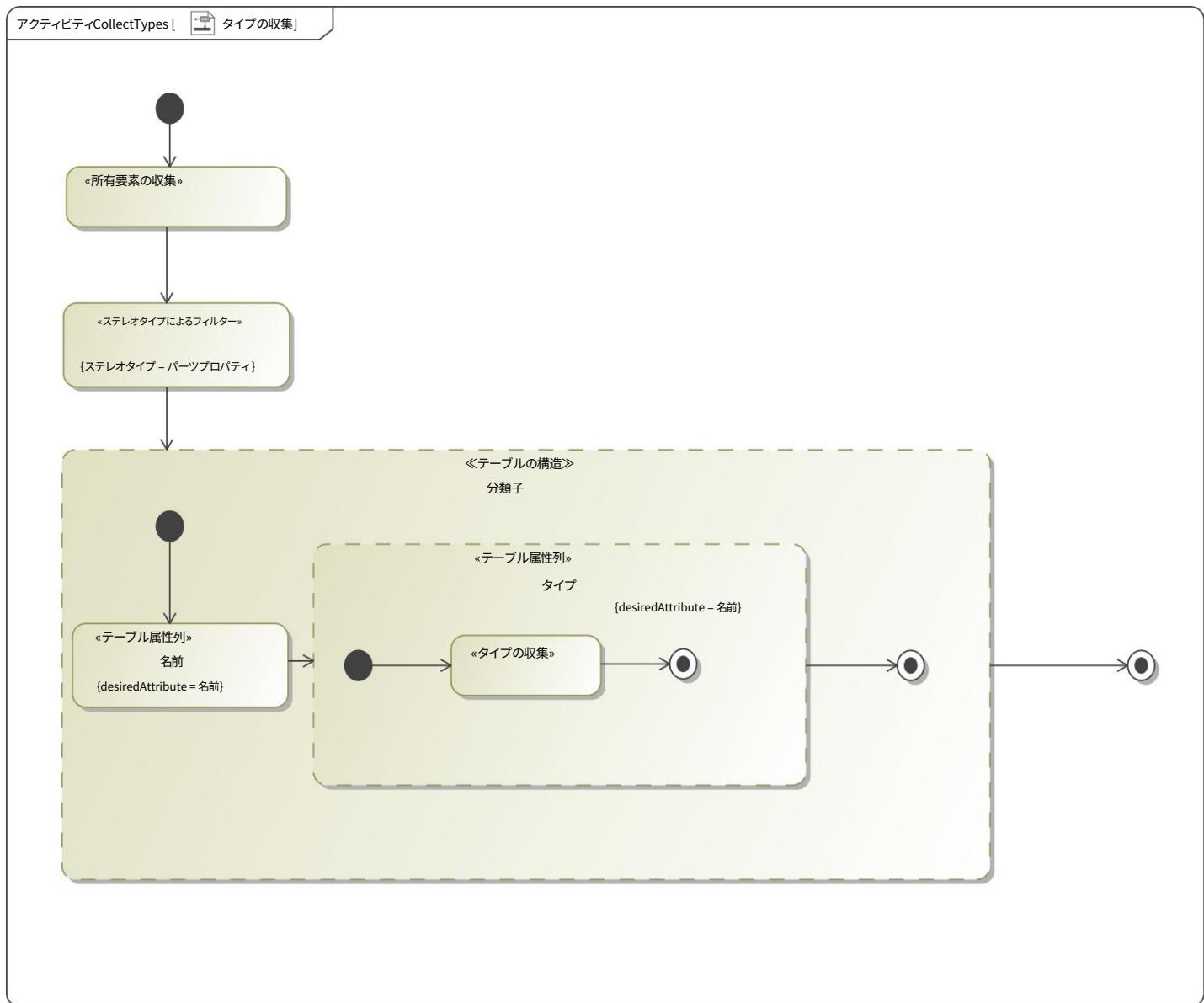


図 27. タイプの収集

CollectTypes は、すでに収集されている要素のタイプを収集します。ほとんどの場合、CollectOwned Elements または CollectOwners を使用して要素を収集します。

参考: [CollectTypes](#)

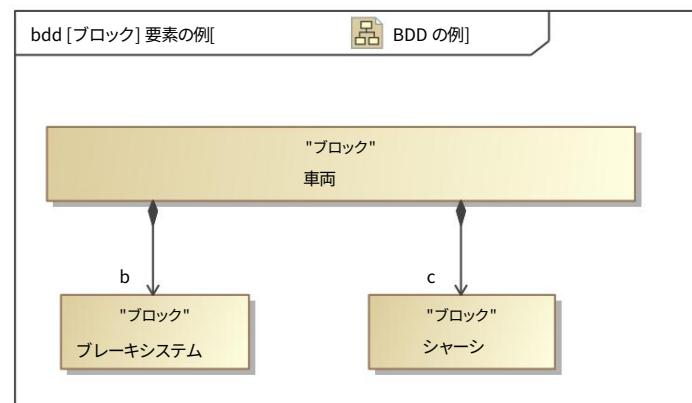


図 28. BDD の例

1.3.1.1.8.1 ビューの例

表 4. 分類子

名前	タイプ
b	ブレーキシステム
c	シャーシ

1.3.1.1.9 CollectClassifierAttributes

CollectClassifierAttributes はクラスの属性を収集します。この操作を使用するには、前のセクションとは異なり、CollectOwnedElements または CollectOwners を使用する必要はありません。結果は、次のビューの例の表に表示されます。

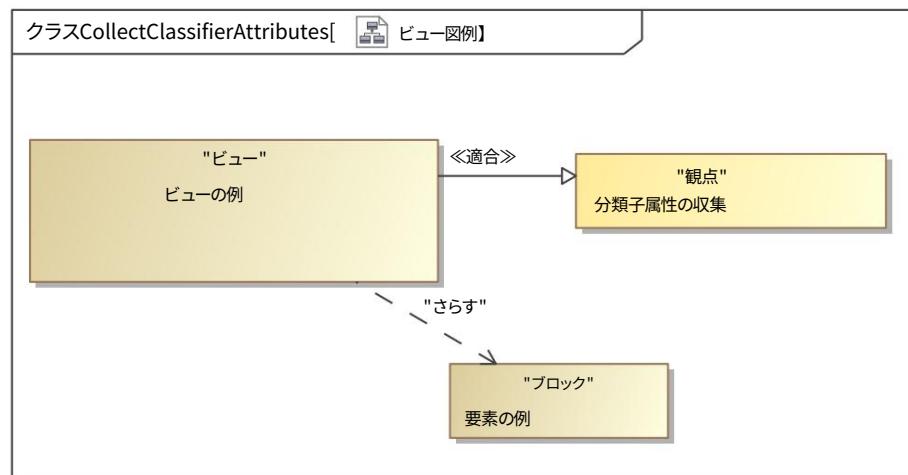


図 29. ビュー図の例

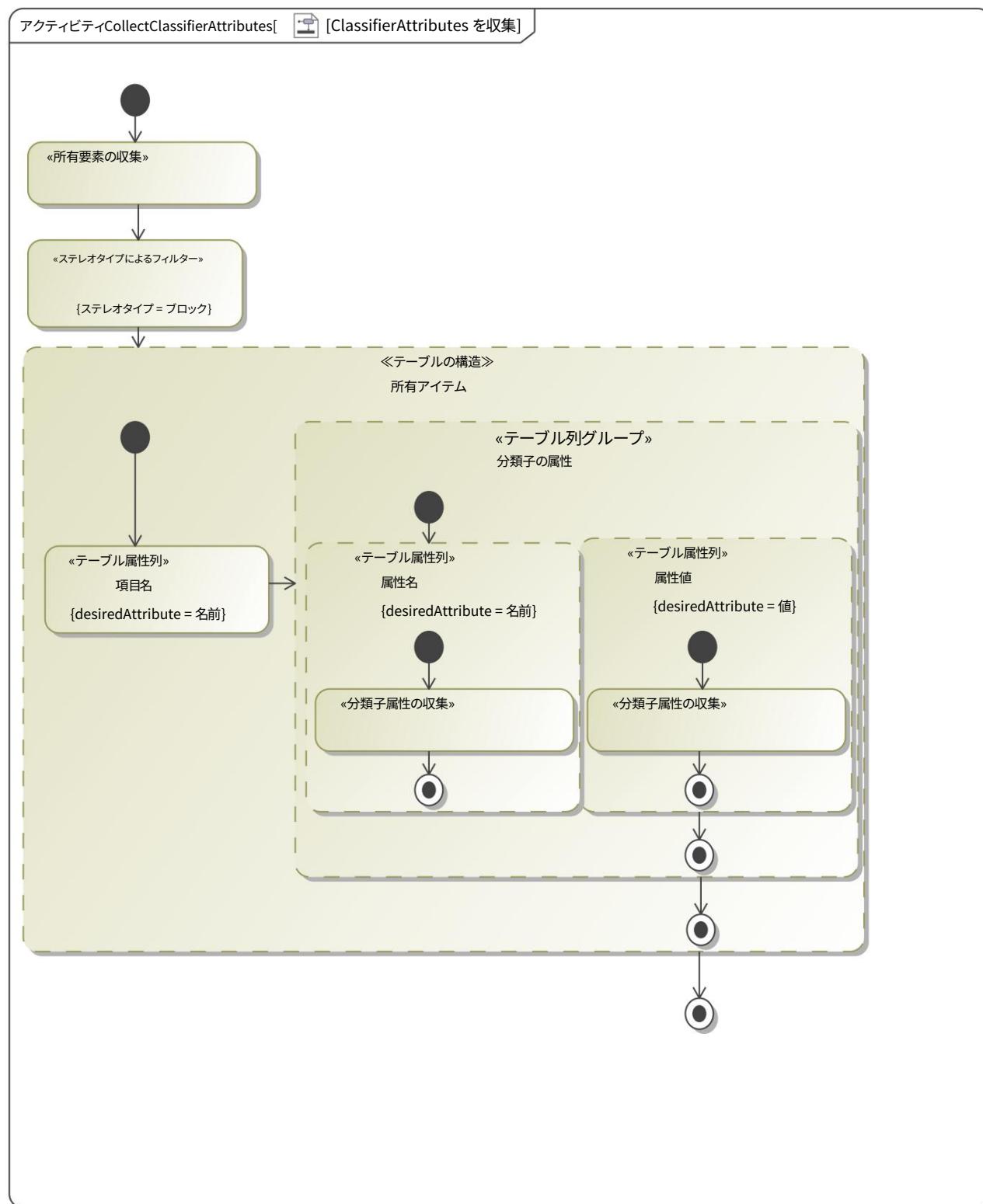


図 30. CollectClassifierAttributes

「CollectClassifierAttributes」はクラスの属性を収集します。

リファレンス: [CollectClassifierAttributes](#)



図 31. BDD の例

1.3.1.1.9.1 ビューの例

表 5. 所有アイテム

項目名	分類子の属性	
	属性名	属性値
アイテム	番号 他の	4 5
他の	文章	"これ"

1.3.1.1.10 CollectByExpression

CollectByExpression は、オブジェクト制約言語 (OCL) を使用してモデルをクエリする、よりカスタマイズされたアプローチです。

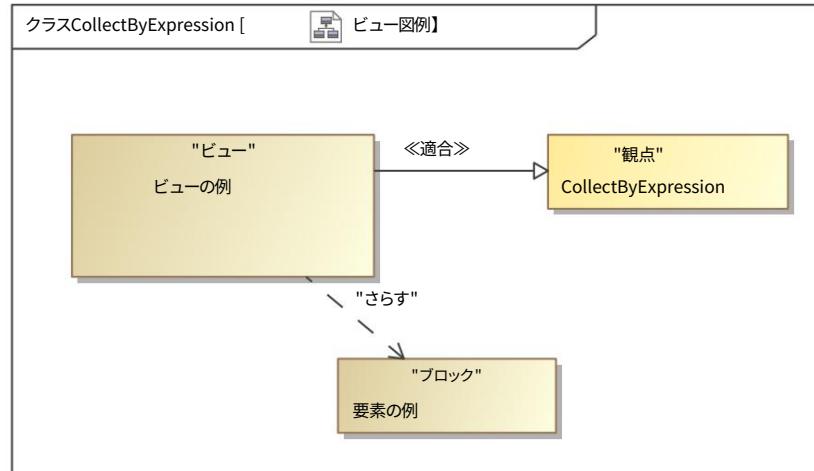


図 32. ビュー図の例

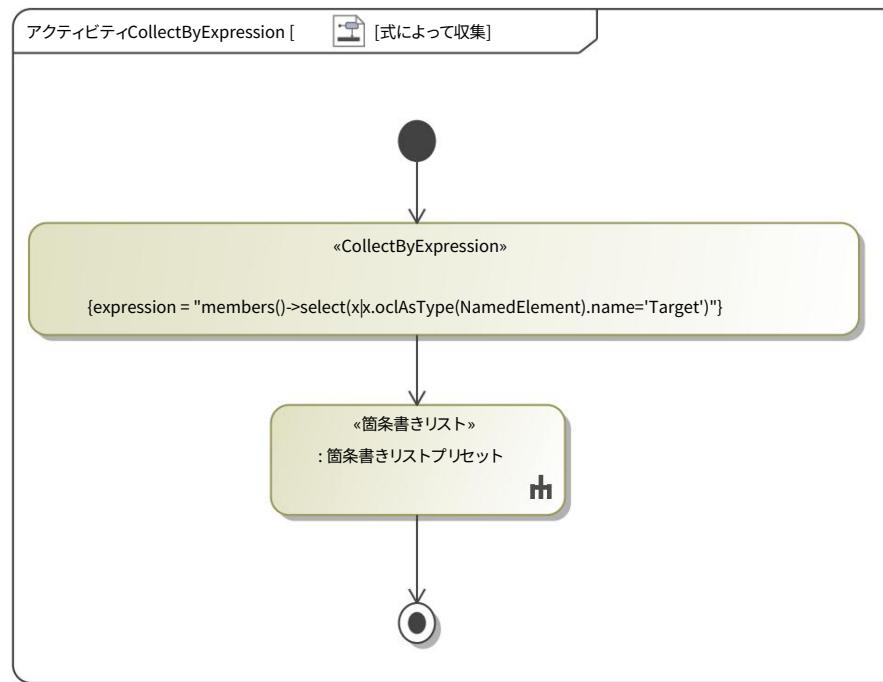


図 33. CollectByExpression

「CollectByExpression」は、オブジェクト制約言語 (OCL) を使用してモデルをクエリする、よりカスタマイズされたアプローチです。

参考: [CollectByExpression](#)

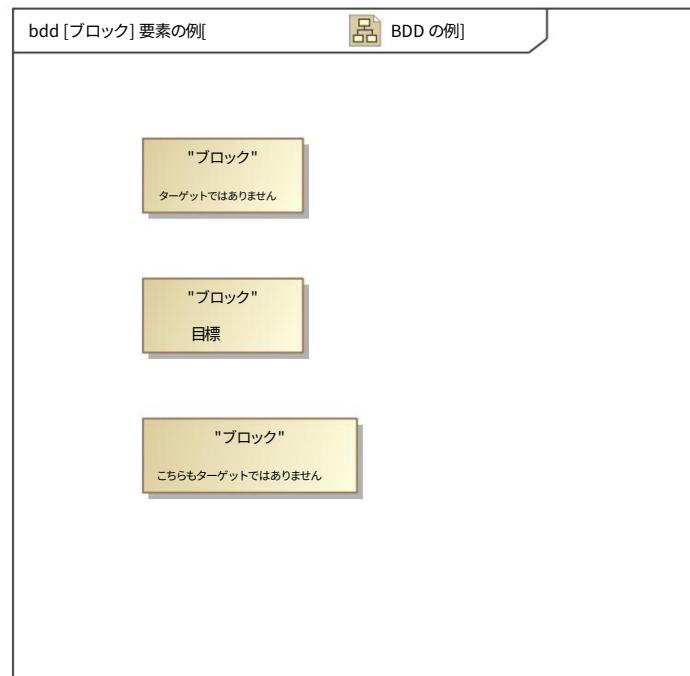


図 34. BDD の例

1.3.1.1.10.1 ビューの例

1. ターゲット

1.3.1.2 フィルター

フィルター操作を使用すると、ユーザーは、さまざまなタイプ（ステレオタイプ、名前、メタクラスなど）の1つまたは複数のフィルター基準に従って、データのフィールドを目的のデータに絞り込むことができます。ほとんどのアプリケーションでは、「FilterBy...」操作の前に「CollectBy...」操作が必要です。これにより、フィルター操作で調べてフィルターするためのデータセットが得られます。

次のセクションでは、さまざまなフィルター操作、その機能、操作の設定方法、および出力がどのようなものになるかを見ていきます。以下の例は、実際のプロジェクトで使用するものよりも単純であり、必要となるより複雑な出力を生成するために使用できる基本原則を簡単に説明することを目的としています。

1.3.1.2.1 FilterByDiagramType

FilterByDiagramType アクティビティは、データセットを調べて、図である要素を調べます。ユーザーは、どのタイプの図に興味があるかを決定し、それらの図の名前のみを文書内に表示できます。

図 35. ビュー図の例

図 36. 図

図 37. FilterByDiagramType イメージ

「FilterByDiagramType」アクティビティは、データセットを調べて、図である要素を調べます。フィルター処理する要素を収集するには、最初に収集操作を使用する必要があります。

参考: [FilterByDiagramType](#)

1.3.1.2.1.1 画像の例

図 38. 図

図 39. 一部のブロック

1.3.1.2.2 FilterByNames

次に、「FilterByNames」操作の使用法を説明します。これにより、ユーザーは、特定の名前を持つデータセット内のすべての要素、または特定の名前を持つ要素に接続されている要素を検索できます。この場合、正規表現「Bat.*」を使用して、名前が「Bat」で始まるすべての要素を収集します。

図 40. ビュー図の例

図 41. BDD の例

図 42. FilterByNames

「FilterByNames」アクティビティは、データセットを調べて、データセット内の特定の名前を持つすべての要素、または特定の名前を持つ要素に接続されている要素を検索します。フィルター処理する要素を収集するには、最初に収集操作を使用する必要があります。

参考: [FilterByNames](#)

1.3.1.2.2.1 ビューの例

1. バットマン 2.

バットガール

1.3.1.2.3 FilterByMetaclasses

FilterByMetaclasses を使用すると、メタクラスによる要素のフィルタリングが可能になります。前の Filter 操作と同様に、フィルタリングする必要がある要素を収集するには、最初に Collect 操作を使用する必要があります。以下の例では、CollectOwnedElements が再度使用されます。今回は「Example Elements」ブロックが持つ要素を集めています。次に、FilterByMetaclasses オペレーションを使用して、この要素のサブセット内のアクターを表示します。

「FilterByMetaclasses」ブロックをダイアグラムにドラッグした後、「FilterByMetaclasses」ブロックの仕様ウィンドウに移動します。このウィンドウで、「IncludeChoosable」の下の「true」ボックスをチェックします。前の例と同様に、このボックスをオンにすると、フィルタリングの対象として選択するメタクラスを含めることになります。次に、仕様ウィンドウの「MetaclassesChoosable」セクションに移動し、「...」ボタンをクリックします。

「クラスの選択」ウィンドウが表示されます。このウィンドウで、左下隅にある小さなボックスが選択されていることを確認し、選択ボックスにメタクラスが表示されるようにします。これで、「UML 標準プロファイル」->「UML2 メタモデル」->「アクター」を選択->「+」ボタンをクリックします。たとえば、「Stereotype」メタクラスを選択したり、他の多くのメタクラスを選択したりすることもできます。

図 43. ビュー図の例

図 44. FilterByMetaclasses

「FilterByMetaclasses」アクティビティは、データ セットを調べて、指定されたメタクラスによって定義されているすべての要素を検索します。フィルター処理する要素を収集するには、最初に収集操作を使用する必要があります。

リファレンス:[FilterByMetaclasses](#)

図 45. BDD の例

1.3.1.2.3.1 ビューの例

1. ある俳優
2. 別の俳優

1.3.1.2.4 ステレオタイプによるフィルター

FilterByStereotypes を使用すると、要素に適用されるステレオタイプによって要素をフィルタリングできます。

図 46. ビュー図の例

図 47. FilterByStereotypes

「FilterByStereotypes」アクティビティは、データ セットを調べて、指定されたステレオタイプによって定義されているすべての要素を検索します。フィルター処理する要素を収集するには、最初に収集操作を使用する必要があります。

参考:[FilterByStereotypes](#)

図 48. BDD の例

1.3.1.2.4.1 ビューの例

1. アイテム

1.3.1.2.5 FilterByExpression

FilterByExpression は、オブジェクト制約言語 (OCL) を使用してモデルをクエリする、よりカスタマイズされたアプローチです。式の値は、OCL で記述された単なる布尔値です。

図 49. ビュー図の例

図 50. FilterByExpression

「FilterByExpression」アクティビティは、データ セットを調べて、オブジェクト制約言語 (OCL) 式を満たす (布尔値) すべての要素を検索します。フィルター処理する要素を収集するには、最初に収集操作を使用する必要があります。

参考:[FilterByExpression](#)

図 51. BDD の例

1.3.1.2.5.1 ビューの例

1. 名前のないブロック
2. これも名前のないブロック

1.3.1.3 並べ替え

並べ替え操作を使用すると、必要なパラメータに従ってデータ セットを並べ替えることができます。データは、属性、プロパティ、または式のいずれかによって並べ替えることができます。これらの並べ替え操作を以下に示します。フィルター操作と同様に、並べ替えるためのデータ セットを作成するには、何らかの方法で要素を収集する必要があります。

1.3.1.3.1 属性による並べ替え

「SortByAttribute」を使用すると、ユーザーは選択した属性によってデータ セットを並べ替えることができます。並べ替えることができる属性のリストには、名前、ドキュメント、または値が含まれます。この操作では、並べ替えの際に一度に 1 つの属性のみを選択できます。「FilterByStereotypes」の例では、ユーザーがビューポイント × ソッドの出力をアルファベット順に並べ替えたい場合は、並べ替え操作が必要であることが強調されました。

「SortByAttribute」の仕様ウインドウで、「AttributeChoosable」セクションのドロップダウン メニューから「Name」オプションを選択します。「Reverse」が `false` であることを確認してください。

図 52. ビュー図の例

図 53. 属性による並べ替え

「SortByAttribute」を使用すると、ユーザーは選択した属性（名前、ドキュメント、値）によってデータ セットを並べ替えることができます。

参照: [SortByAttribute](#)

図 54. BDD の例

1.3.1.3.1.1 ビューの例

1. アルファ
2. ベータ
3. ガンマ

1.3.1.3.2 プロパティによる並べ替え

図 55. ビュー図の例

図 56. BDD の例

図 57. SortByProperty

「SortByProperty」を使用すると、ユーザーは指定されたプロパティによってデータ セットを並べ替えることができます。

参照: [SortByProperty](#)

1.3.1.3.2.1 ビューの例

1. 最初にする必要があります
2. 3 番目にする必要があります
3. 2 番目にする必要があります

1.3.1.3.3 式による並べ替え

`SortByExpression` は、オブジェクト制約言語 (OCL) を使用してモデルをクエリする、よりカスタマイズされたアプローチです。

図 58. ビュー図の例

図 59. 並べ替え式

「SortByExpression」は、オブジェクト制約言語 (OCL) 式で指定されたデータ セットです。

参考: [SortByExpression](#)

図 60. BDD の例

1.3.1.3.3.1 ビューの例

1. スーパー
2. スーパーではない

1.3.2 現在のモデルデータ

データが収集、フィルタリング、および/または並べ替えられた後、いくつかの演算子を使用してデータの表示方法を調整できます。プレゼンテーション要素の演算子は、テーブル、イメージ、段落、リスト、セクションです。これらはビューポイント方法図で使用され、ドキュメント ビューに表示される書式設定を決定します。

次の例はすべて、サンプルとして機能するさまざまな要素を含む共通の Zoo パッケージを使用しています。 OCL を扱う例では、さまざまな動作を示すために Robot Zoo パッケージを使用します。

表 6. DocGen メソッド

メソッド名	メソッドの説明
シンプルテーブル	このマニュアルでは、指定された DocGen Table メソッドを使用して、属性列で構成される「単純な」テーブルを構築する方法を示しています。 参考: 表
複雑なテーブル	このマニュアルでは、指定された DocGen Table メソッドを使用して、属性列とテーブル列グループで構成される「複雑な」テーブルを構築する方法を説明します。 参考: 表
ジェネリック 段落	段落は、テキストを表示するために設計されたプレゼンテーション要素です。デフォルト (別名「汎用段落」) では、公開された要素またはビュー自体のドキュメントが表示されます。 参考: 段落
の段落 名前	段落は、テキストを表示するために設計されたプレゼンテーション要素です。「名前の段落」には、公開された要素またはビュー自体の名前属性が表示されます。 参考: 段落
の段落 ドキュメンテーション	段落は、テキストを表示するために設計されたプレゼンテーション要素です。「ドキュメントの段落」には、公開された要素またはビュー自体のドキュメント属性が表示されます。これは汎用段落と同じ結果を生成しますが、単にそれを指定する別の方法です。 参考: 段落
の段落 デフォルト値	段落は、テキストを表示するために設計されたプレゼンテーション要素です。「デフォルト値の段落」には、公開された要素またはビュー自体の値属性が表示されます。 参考: 段落
段落 体	段落は、テキストを表示するために設計されたプレゼンテーション要素です。本文は段落仕様で指定されたテキスト (HTML の場合もあります) であり、「段落本文」にはその本文が表示されます。 参考: 本文を使用した段落アクション

メソッド名	メソッドの説明
段落 OCLあり ターゲット	段落は、テキストを表示するために設計されたプレゼンテーション要素です。 Object Constraint Language (OCL) を使用してターゲットを設定することにより、上記の式に従ってより具体的な段落を作成できます。 参考: 段落アクションの OCL の評価
段落 OCLなし ターゲット	段落は、テキストを表示するために設計されたプレゼンテーション要素です。オブジェクト制約言語 (OCL) を使用すると、ターゲットがなくても、前述の式に従って、より指定された段落を作成できます。 参考: 段落アクションの OCL の評価
段落 OCL用 ターゲット	段落は、テキストを表示するために設計されたプレゼンテーション要素です。 Object Constraint Language (OCL) を使用すると、上記の式に従ってより詳細な段落を作成できます。 参考: 段落アクションの OCL の評価
箇条書きリスト	「BulletedList」は、動作に公開されるモデル要素に基づいてリストを作成します。どのような情報が表示されるかは、動作の仕様で選択されたオプションと、収集されたデータに適用されるフィルターによって異なります。 参考: 一覧表
画像	画像要素は、「画像」アクションを使用してビューエディターに表示できます。画像要素の例としては、図があります。 参考: 画像
動的 セクション	「動的セクショニング」とは、視点方式で定義されたセクションを作成することです。これらは、ビューポイント メソッド図の「構造化クエリ」アクティビティを使用して作成されます。 この特定の例は、単一のセクションの作成を示しています。 参考: 動的セクショニング
複数 セクション	「動的セクショニング」とは、視点方式で定義されたセクションを作成することです。これらは、ビューポイント メソッド図の「構造化クエリ」アクティビティを使用して作成されます。 この特定の例は、複数のセクションの作成を示しています。 参考: 動的セクショニング

1.3.2.1 表

以下のセクションでは、テーブルの作成方法について説明します。 MagicDraw ではさまざまな複雑さのテーブルを作成できますが、これらのテーブルはいくつかの共通の基本コンポーネントを共有しています。左側のサイドバーには、「テーブル構造」で使用するさまざまなコンポーネントが表示されます。

最初のオプションである TableStructure は、テーブルベースを作成するためにすべての場合に使用されます。サイドバーの「テーブル構造」の下に表示される最後の 3 つのオプションを使用すると、ユーザーは各列に表示する必要がある情報に基づいて列のタイプを選択できます。 TableExpressionColumn は OCL 式を使用します。 TablePropertyColumn を使用すると、ユーザーの選択に応じて、クラスのステレオタイプ プロパティまたは値プロパティを表示できます。 TableAttributeColumn を使用すると、ユーザーは要素の名前、ドキュメント、または値を表示できます。サイドバーのこれらのオプションの右側にある小さな黒い矢印をクリックすると、一連の選択項目が開きます。2 番目の選択項目にはアイコンの外側に点線が表示されます。この外側の点線の選択は 3 つの列タイプのそれぞれに存在し、ユーザーが列内でフローを構成できるようにします。 TableColumnGroup を使用すると、複合テーブルの選択で示されるように、結合されたヘッダーを持つ列グループを作成できます。

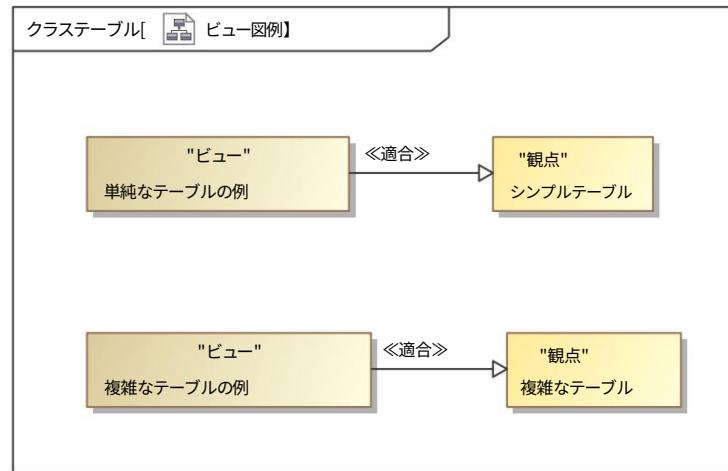


図 61. ビュー図の例

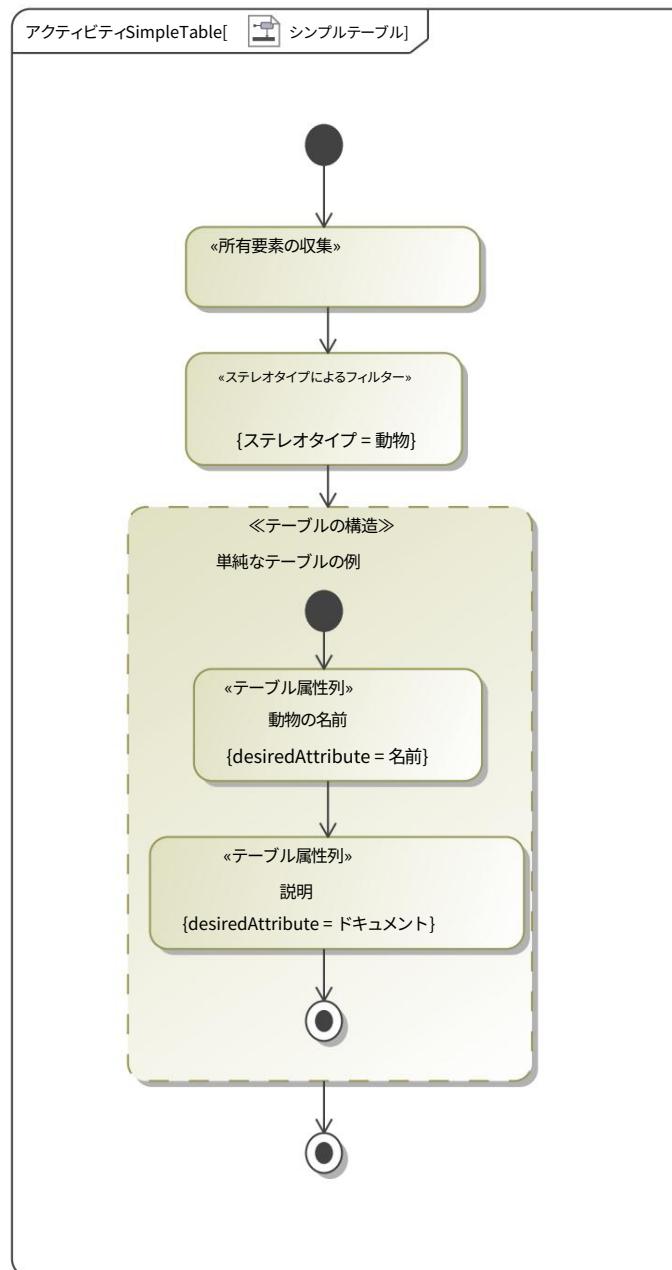


図 62. シンプルテーブル

このマニュアルでは、指定された DocGen Table メソッドを使用して、属性列で構成される「単純な」テーブルを構築する方法を示しています。

参考 :表

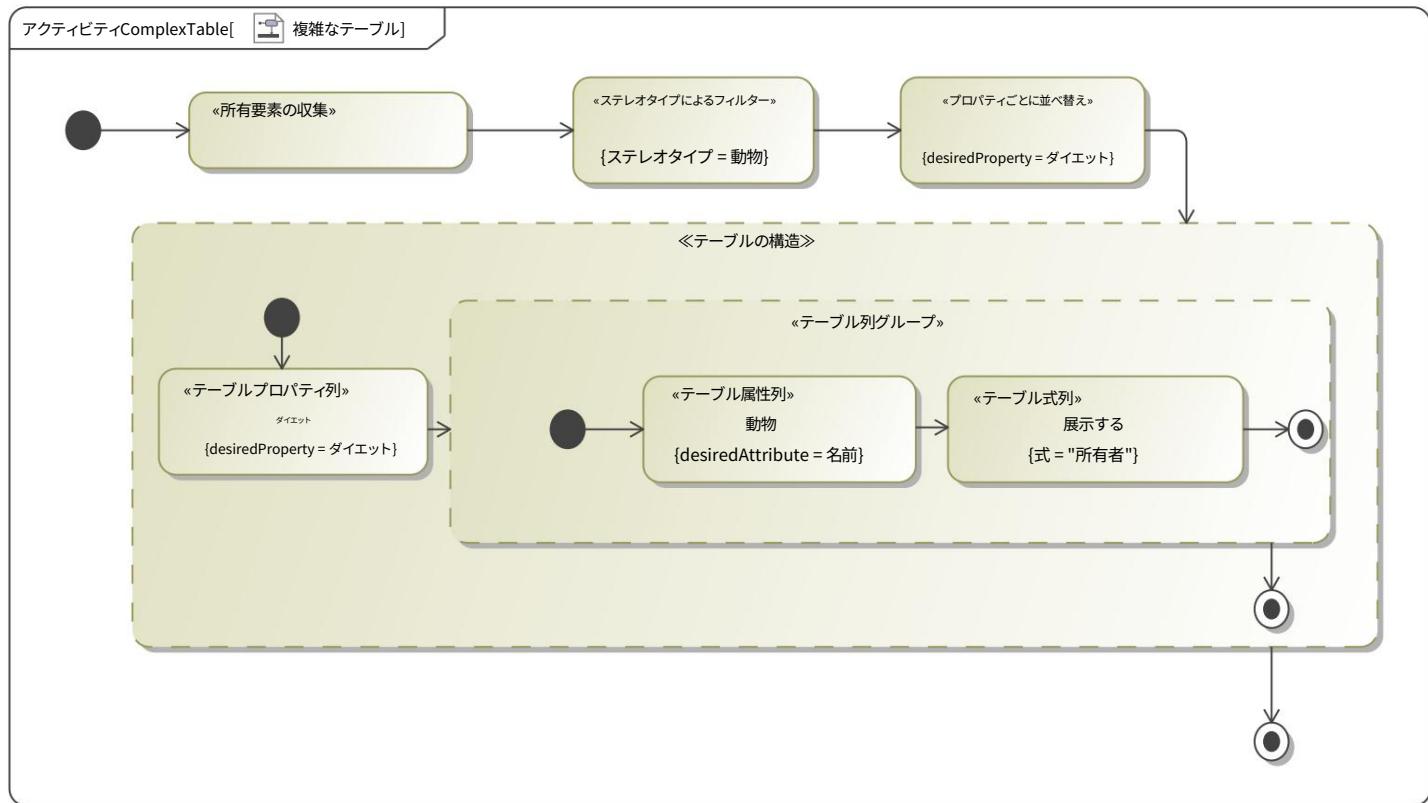


図 63. 複合テーブル

このマニュアルでは、指定された DocGEN Table メソッドを使用して、属性列とテーブル列グループで構成される「複雑な」テーブルを構築する方法を説明します。

参考 :表

1.3.2.1.1 単純なテーブル

この視点方法は、Zoo パッケージ内のすべての動物の名前とドキュメントを示す単純な表を作成するために使用されます。

まず、Zoo パッケージが所有するすべての要素を収集します。次に、<<Animal>> ではないすべての要素をフィルターで除外します。これで 5 つの動物要素が残り、TableStructure に渡します。

作成する最初の列には、動物要素の Name 属性が表示されます。この列を「動物名」と呼び、ドキュメントを生成するとき、または ViewEditor で表示するときに列のヘッダーになります。TableAttributeColumn アクションで、desiredAttribute として「Name」を選択します。

次の列「説明」に対しても同じことを行います。ただし、desiredAttribute を「Documentation」に設定して、Animal 要素の別の属性をターゲットにする点が異なります。

表 7. 単純なテーブルの例

動物の名前	説明
ダチョウ	ダチョウは時速 70 km で走ることができ、これは鳥の中で最速の地上速度です。
クロコダイル	基本的には恐竜です。
シマウマ	シマウマはほぼ完全に草を食べます。
シール	ほとんどが脂肪です。
ホッキョクアジサシ	ホッキョクアジサシはどんな鳴き声を出しますか？

1.3.2.1.2 複合テーブル

表 8.<>

ダイエット	動物	展示する
肉食動物	クロコダイル	アフリカ展
肉食動物	シール	南極展示
肉食動物	ホッキョクアシサン	南極展示
草食動物	シマウマ	アフリカ展
雑食動物	ダチョウ	アフリカ展

1.3.2.2 段落

段落は、テキストを表示するために設計されたプレゼンテーション要素です。すべてのビューには視点メソッドが必要です。視点メソッドが指定されていない場合は、デフォルトのメソッドは、ビュー要素のドキュメントを対象とした單一段落アクションです。これは何もしなくても自動的に追加されますユーザーに見えるようになります。

このセクションでは、段落を作成する、段落を他の視点メソッドのアクションと結合する、および OCL を使用して段落を生成します。

段落は、画像と表の機能を組み合わせたものです。これは、あらゆるドキュメントを表示するという点でイメージに似ています。ビューによって公開される要素。また、次のような属性を追加表示できる高度な表現機能も備えています。
ビュー (TableAttributeColumn と同様) または式の結果 (TableExpressionColumn と同様) によって公開される要素。

一般に、段落の結果では、情報よりもそのモデルからビュー エディターで編集可能なコンテンツを表示する柔軟性が高くなります。画像またはテーブルの厳格な制約内で表示されます。

次のセクションでは、段落を使用するさまざまな方法について説明します。

1.3.2.2.1 ターゲットを使用した段落アクション

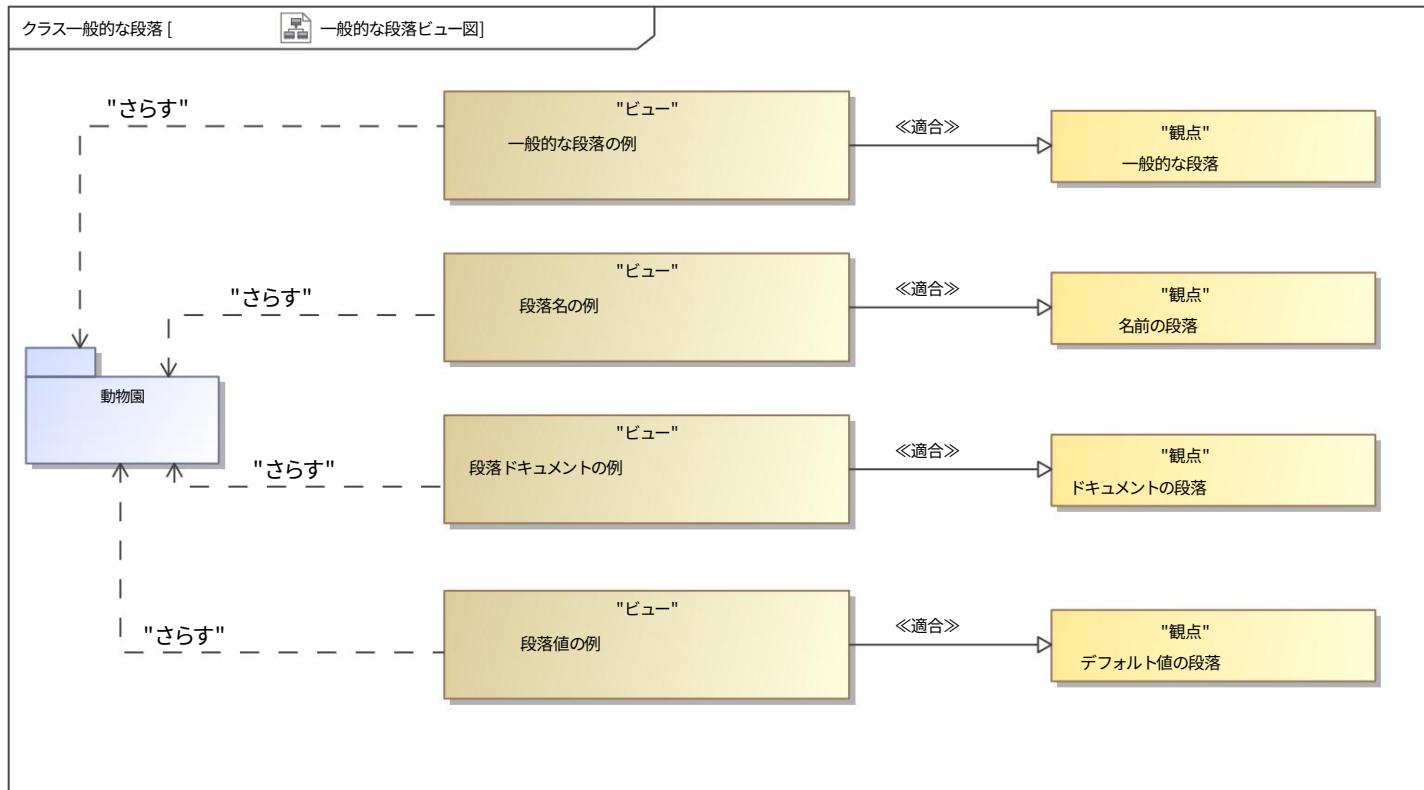


図 64. 一般的な段落ビュー図

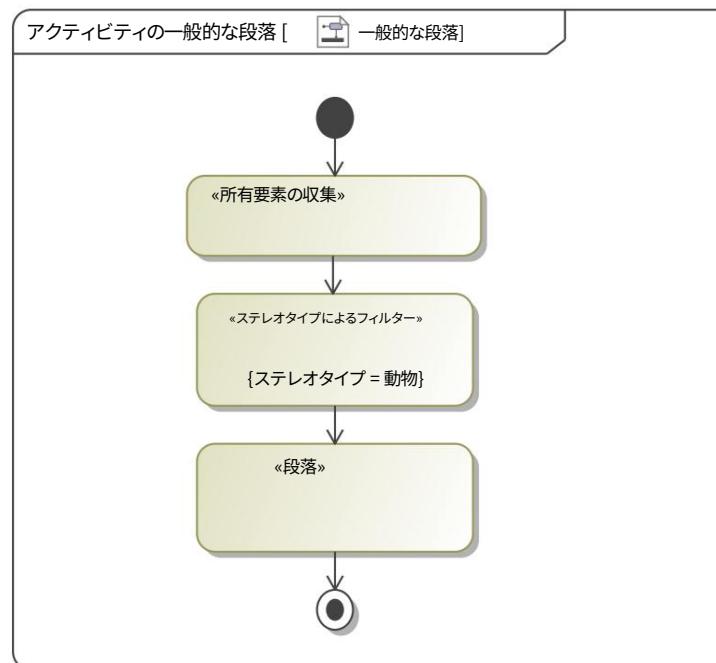


図 65. 一般的な段落

段落は、テキストを表示するために設計されたプレゼンテーション要素です。デフォルト(別名「汎用段落」)では、公開された要素またはビュー自体のドキュメントが表示されます。

参考 : [段落](#)

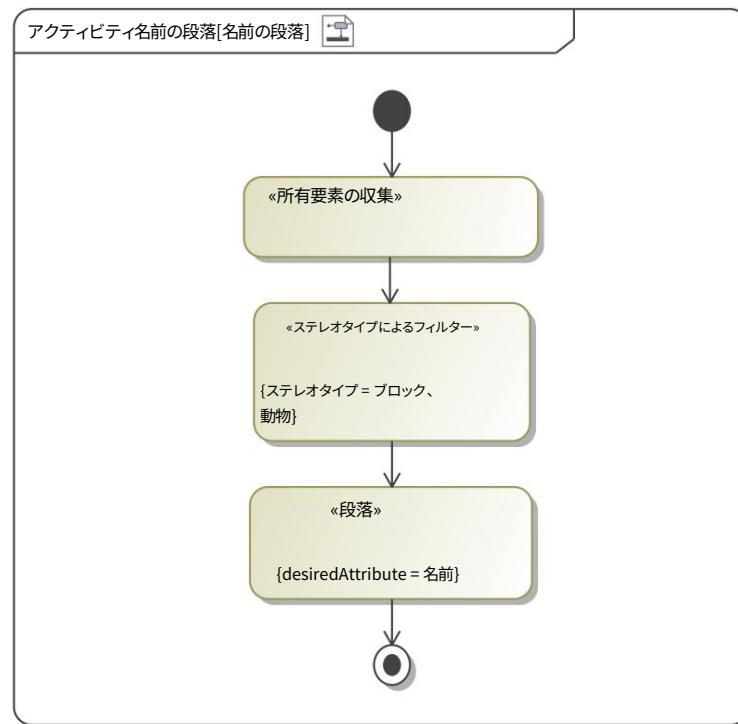


図 66. 名前の段落

段落は、テキストを表示するために設計されたプレゼンテーション要素です。「名前の段落」には、公開された要素またはビュー自体の名前属性が表示されます。

参考 :[段落](#)

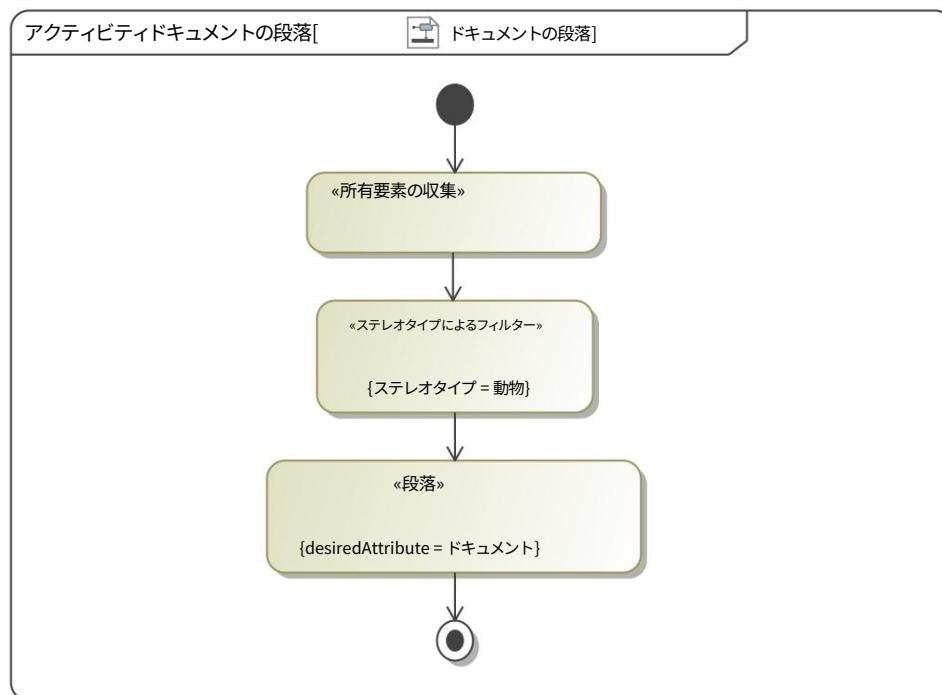


図 67. ドキュメントの段落

段落は、テキストを表示するために設計されたプレゼンテーション要素です。「ドキュメントの段落」には、公開された要素またはビュー自体のドキュメント属性が表示されます。これは汎用段落と同じ結果を生成しますが、単にそれを指定する別の方法です。

参考 :[段落](#)

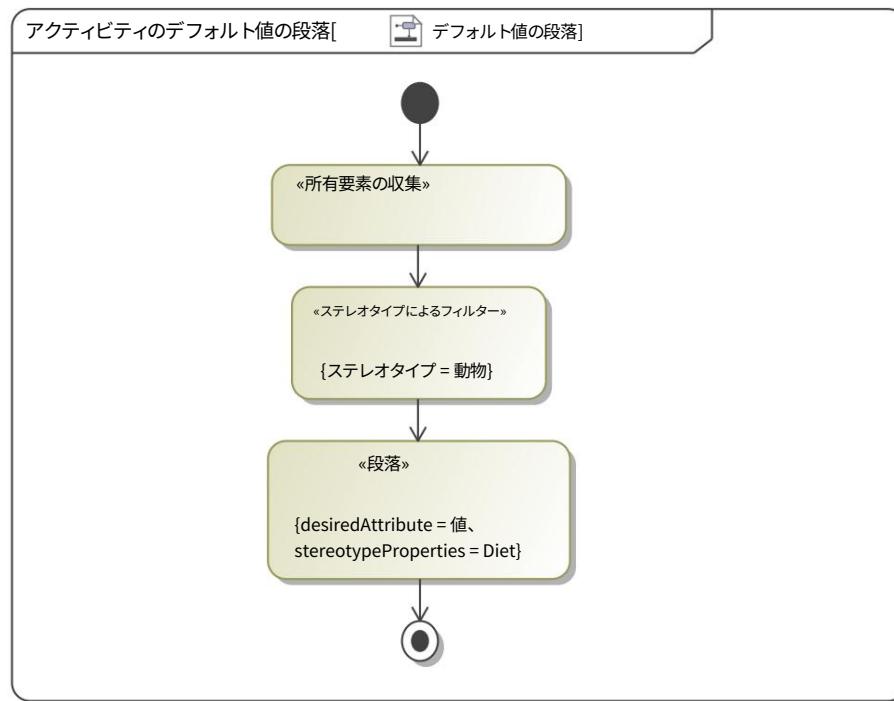


図 68. デフォルト値の段落

段落は、テキストを表示するために設計されたプレゼンテーション要素です。 「デフォルト値の段落」には、公開された要素またはビュー自身の値属性が表示されます。

参考 [段落](#)

1.3.2.2.1.1 一般的な段落の例

ダチョウは時速 70 km で走ることができ、これは鳥の中で最速の陸上速度です。

基本的には恐竜です。

シマウマはほぼ完全に草を食べます。

ほとんどが脂肪です。

ホッキョクアジサシはどんな鳴き声を出しますか？

1.3.2.2.1.2 段落名の例

アフリカ展

ダチョウ

クロコダイル

シマウマ

南極展示

シール

キョクアジサンにはタイ

トルブロックがあります

1.3.2.2.1.3 段落ドキュメントの例

ダチョウは時速 70 km で走ることができ、これは鳥の中で最速の陸上速度です。

基本的には恐竜です。

シマウマはほぼ完全に草を食べます。

ほとんどが脂肪です。

ホッキョクアジサシはどんな鳴き声を出しますか？

1.3.2.2.1.4 段落値の例

雑食動物

肉食動物
草食動物
肉食動物
肉食動物

1.3.2.2.2 本文を含む段落アクション

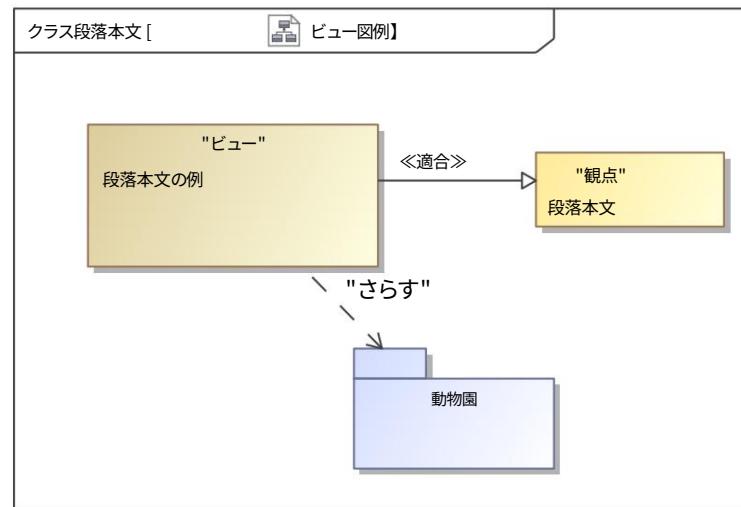


図 69. ビュー図の例

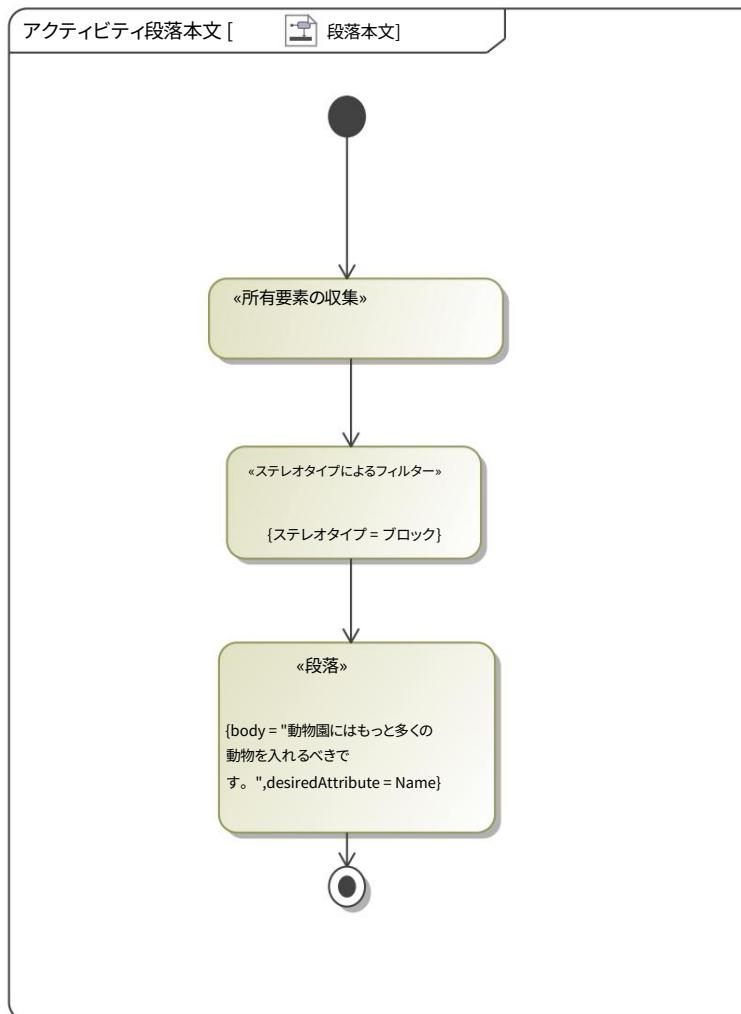


図 70. 段落本文

段落は、テキストを表示するために設計されたプレゼンテーション要素です。本文は段落仕様で指定されたテキスト (HTML の場合もあります) であり、「段落本文」にはその本文が表示されます。

参考:[本文を使用した段落アクション](#)

1.3.2.2.1 段落本文の例

私たちには動物園にもっと多くの動物を入れるべきです。

1.3.2.2.3 段落アクションの OCL の評価

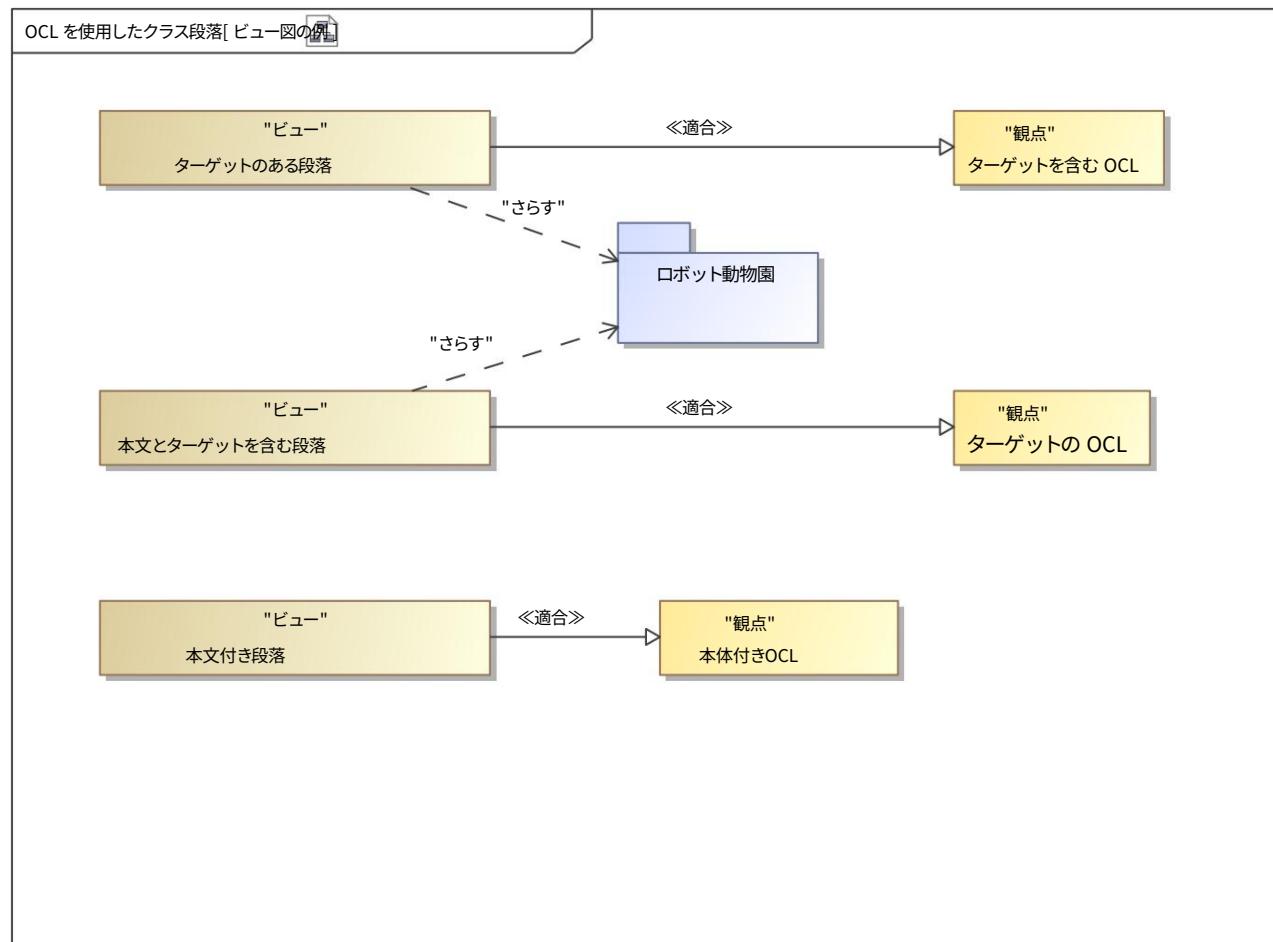


図 71. ビュー図の例

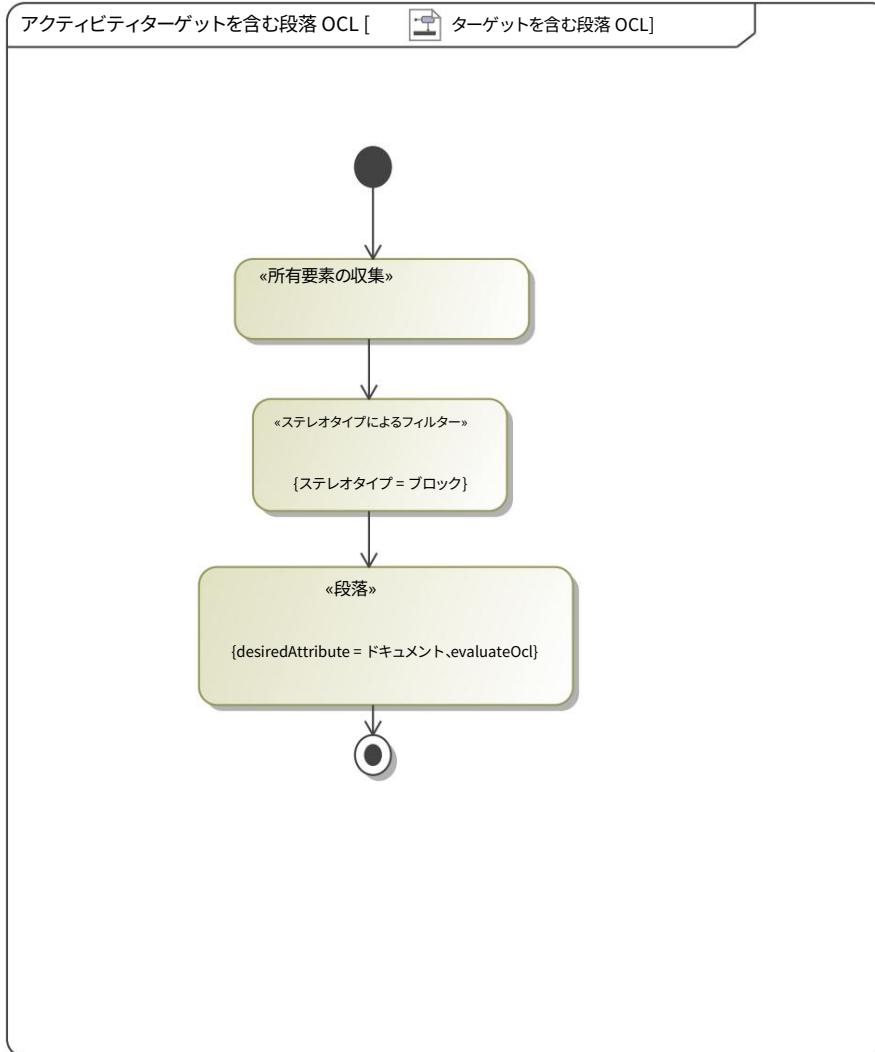


図 72. ターゲットを含む段落 OCL

段落は、テキストを表示するために設計されたプレゼンテーション要素です。 Object Constraint Language (OCL) を使用してターゲットを設定することにより、上記の式に従ってより具体的な段落を作成できます。

参考:[段落アクションの OCL の評価](#)

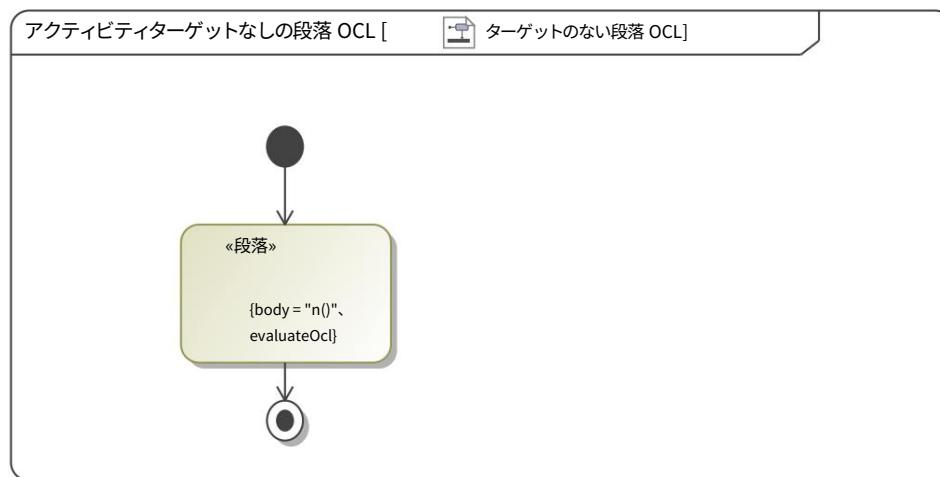


図 73. ターゲットのない段落 OCL

段落は、テキストを表示するために設計されたプレゼンテーション要素です。オブジェクト制約言語 (OCL) を使用すると、ターゲットがなくても、前述の式に従って、より指定された段落を作成できます。

参考:[段落アクションの OCL の評価](#)

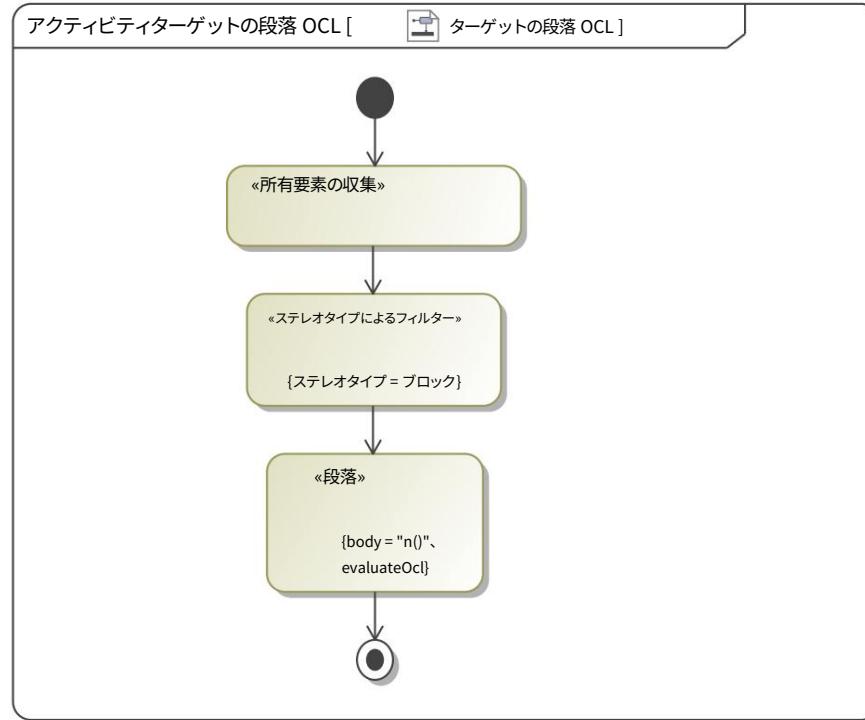


図 74. ターゲットの段落 OCL

段落は、テキストを表示するために設計されたプレゼンテーション要素です。 Object Constraint Language (OCL) を使用すると、上記の式に従ってより詳細な段落を作成できます。

参考:[段落アクションの OCL の評価](#)

1.3.2.2.3.1 本文のある段落

本文付き段落

1.3.2.2.3.2 ターゲットを含む段落

真実

間違い

1.3.2.2.3.3 本文とターゲットを含む段落

口ボットムース

口ボットリス

1.3.2.3 リスト

「BulletedList」は、動作に公開されるモデル要素に基づいてリストを作成します。この 1 つのプレゼンテーション要素は、以下に示すような順序付き（番号付き）リスト、または前の例で表示されたような箇条書きリストのいずれかを作成できます。どのような情報が表示されるか（名前、ドキュメント、ステレオタイプ プロパティ値）は、動作の仕様で選択されたオプションと、収集されたデータに適用されるフィルターによって異なります。たとえば、「Show Targets」が「true」の場合、要素の名前がリストされます。

以下の視点方法図は、以下のリスト例を作成するために使用された図を示しています。公開されたパッケージは前の例の「Zoo」であり、<<Animal>> ステレオタイプのみを識別するためにフィルターが適用されました。この例を作成するには、番号付きリストを作成するために「順序付きリスト」を「true」に選択しました。このオプションが false の場合、代わりにリストに箇条書きが表示されます。箇条書きの仕様内には、他にも多数のオプションがあります。オプションをクリックすると、下部のボックスに説明が表示されます。

注: 現在、「ステレオタイプ プロパティ名の表示」は機能しません。理論的には、値をリストする前にステレオタイプのプロパティ名を出力します。

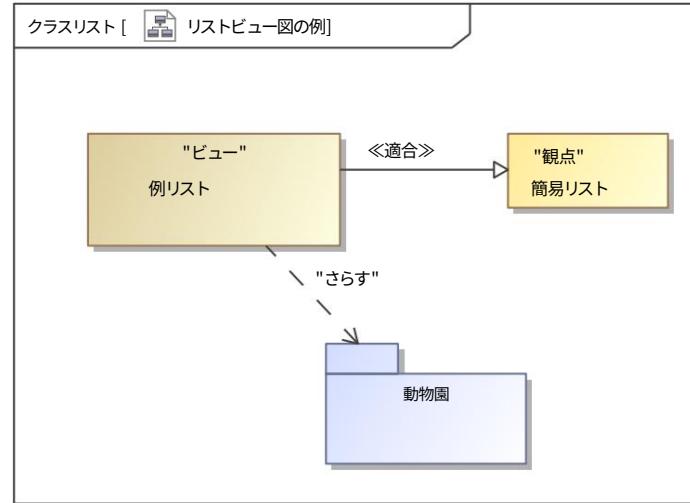


図 75. リストビュー図の例

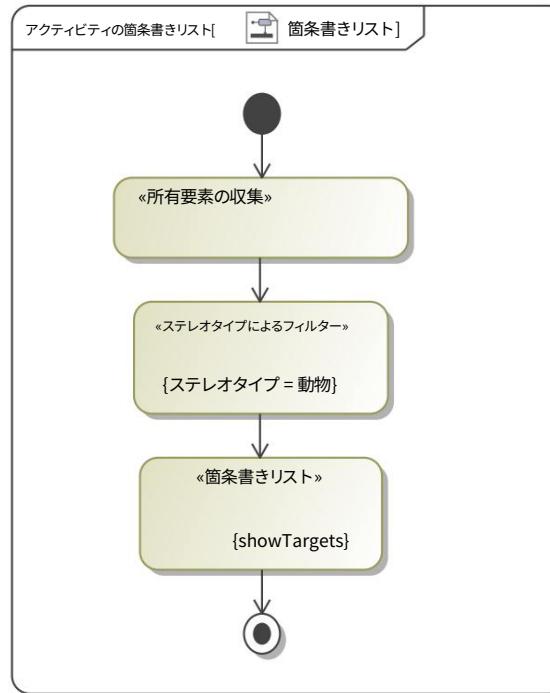


図 76. 箇条書きリスト

「BulletedList」は、動作に公開されるモデル要素に基づいてリストを作成します。どのような情報が表示されるかは、動作の仕様で選択されたオプションと、収集されたデータに適用されるフィルターによって異なります。

参考 : [一覧表](#)

1.3.2.3.1 サンプルリスト

- ダチョウ

- ・ワニ・シマウ
- マ・アザラ
- シ・キヨ
- クアジサシ

1.3.2.4 画像

画像要素は、<<画像>> アクションを使用してビュー エディターに表示できます。画像要素の例としては、図があります。

このアクションにより、任意の画像要素に関連付けられた画像が表示され、その後にそのドキュメントが表示されます。複数の図が公開または収集されている場合、単一の <<Image>> アクションですべての図が繰り返し表示されます。

モデルが更新されると、この画像も更新されます。変更を反映するために新しい画像をドキュメントに追加する必要はありません。画像にキャプションやタイトルを追加することもできますが、これらの機能は現在機能していません。

この操作のビュー図とビューポイント方法図を以下に示します。

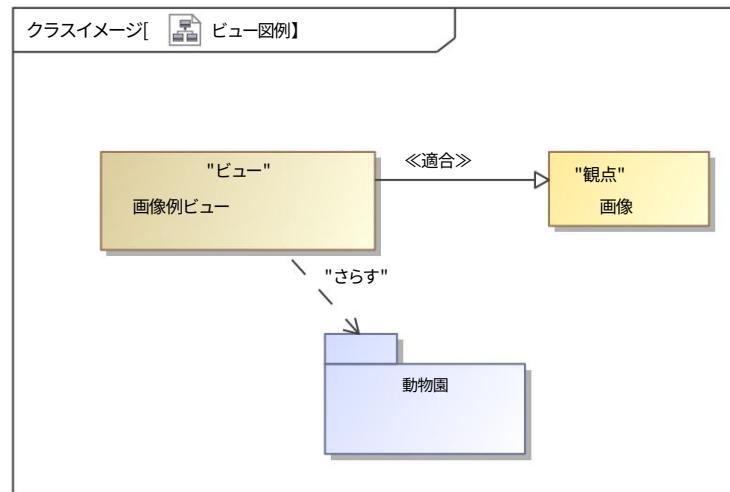


図 77. ビュー図の例

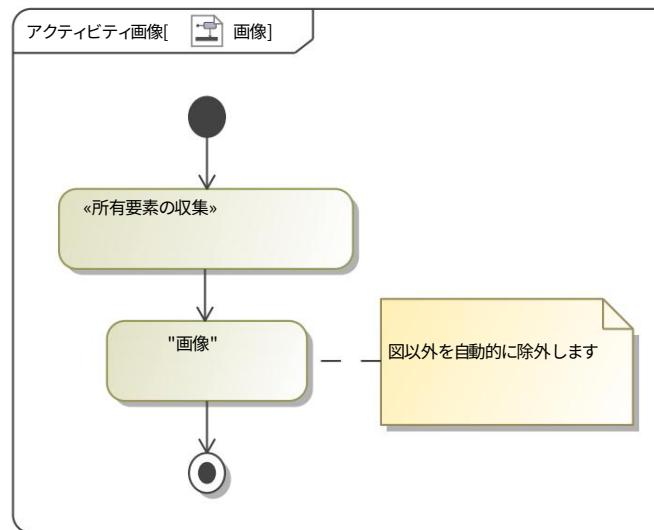


図 78. 画像

画像要素は、「画像」アクションを使用してビュー エディターに表示できます。画像要素の例としては、図があります。

参考 :[画像](#)

1.3.2.4.1 画像サンプルビュー

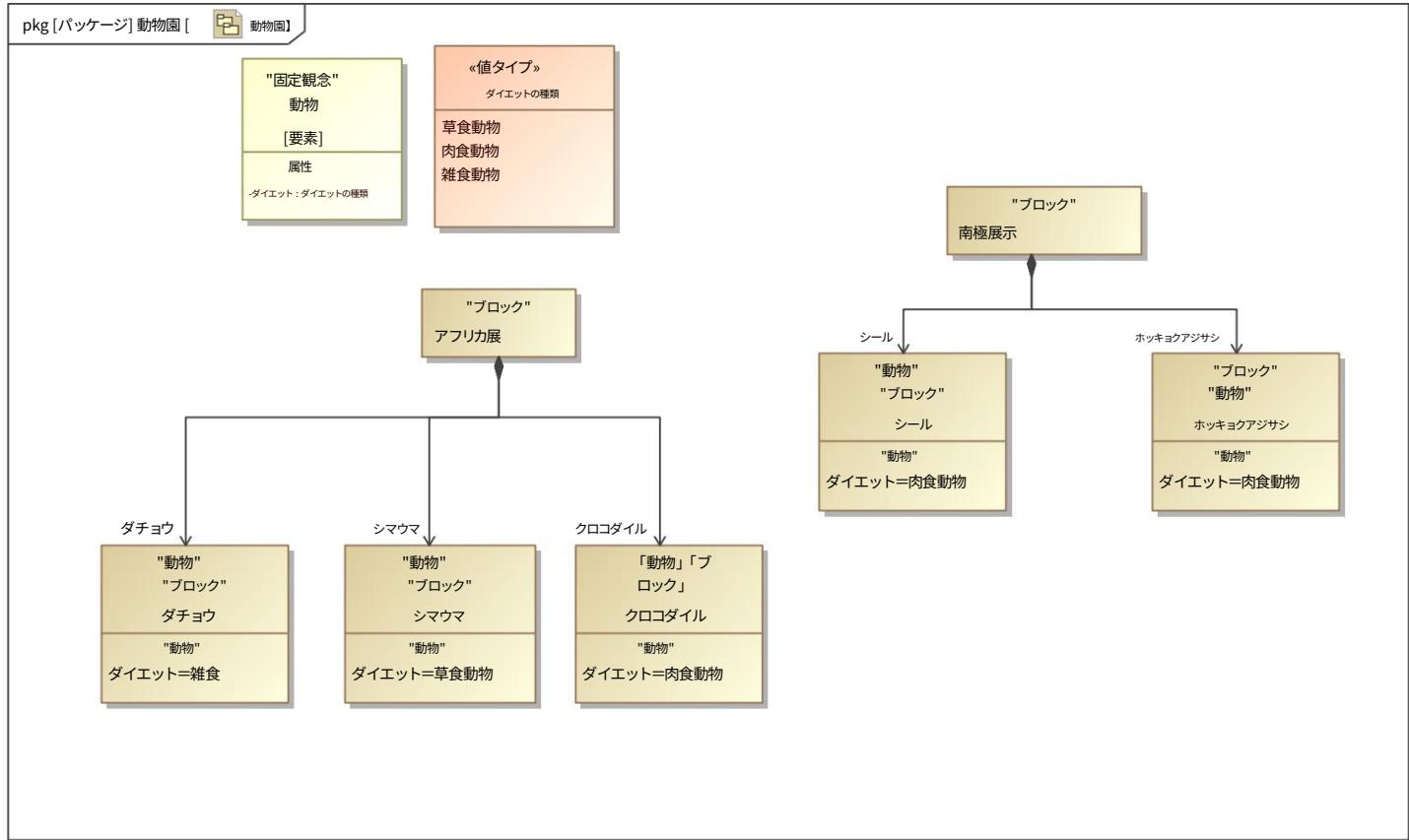


図 79. 動物園

#	名前
1	ホッキョクアジサシ
2	クロコダイル
3	ダチョウ
4	シール
5	シマウマ
6	アフリカ展
7	南極展示
8	タイトルブロックあり

図 80. 動物園の動物

1.3.2.5 動的セクショニング

動的セクショニングは、視点メソッドで定義されたセクションを作成することです。これらは、ビューポイント メソッド図の「構造化クエリ」アクティビティを使用して作成されます。

このセクションでは、単一セクションの作成と複数セクションの作成という 2 つの主なタイプのダイナミック セクショニングについて説明します。

新しい動的セクションは、目次内の小さなページ記号 § によって標準の <>view></> と区別できます。動的セクションも親ビューと同時にビューにロードされることに注意してください。

これらの例では、段落の例で最初に紹介されたパッケージ 「Animals」が公開されました。5 つの動物要素はそれぞれ、要素のタイトルとドキュメントのテキストで構成されます。

これらのセクションには主に 2 つの用途があり、基本的な構成によりセクションを分割できます。正式には、動的セクションにより、正規のビュー階層を維持しながら、ビューをさらに組織化できます。

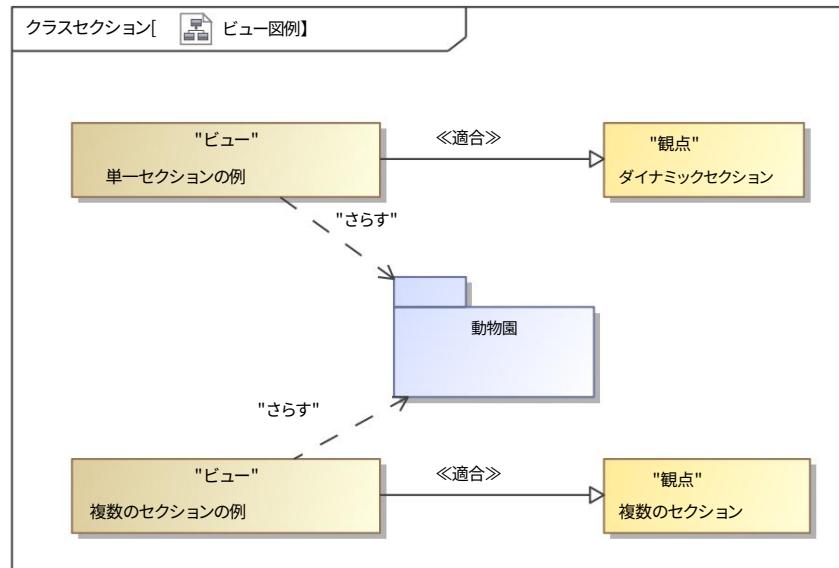


図 81. ビュー図の例

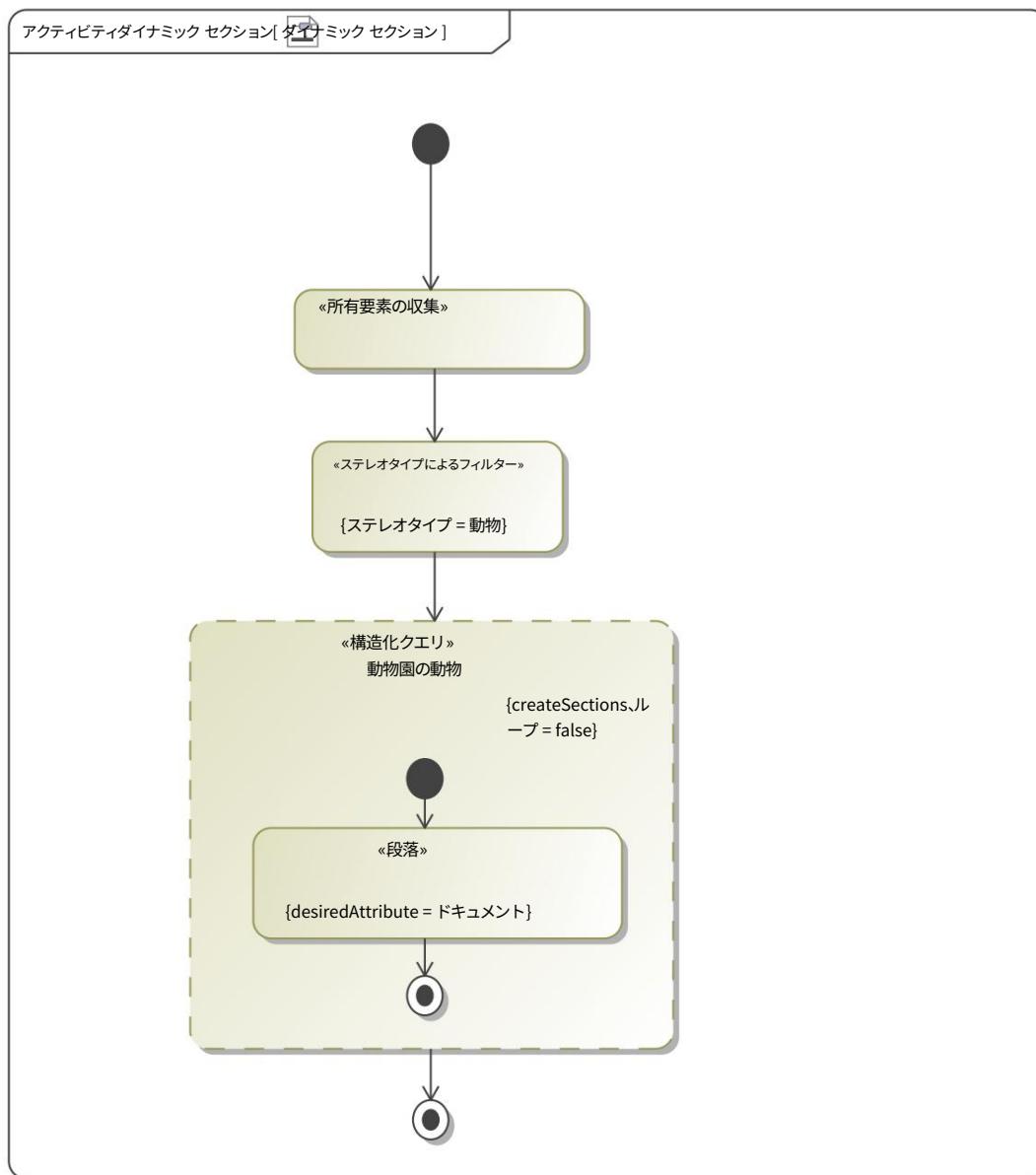


図 82. 動的セクション

「動的セクショニング」とは、視点方式で定義されたセクションを作成することです。これらは、ビューポイント メソッド図の「構造化クエリ」アクティビティを使用して作成されます。

この特定の例は、単一のセクションの作成を示しています。

参考:[動的セクショニング](#)

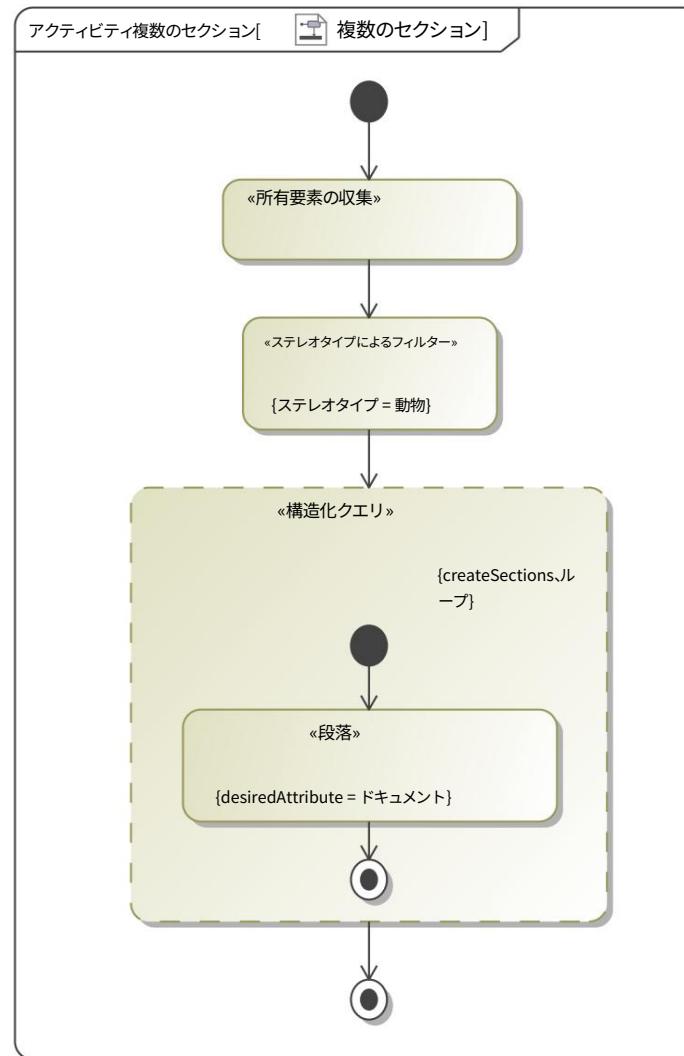


図 83. 複数のセクション

「動的セクショニング」とは、視点方式で定義されたセクションを作成することです。これらは、ビューポイント メソッド図の「構造化クエリ」アクティビティを使用して作成されます。

この特定の例は、複数のセクションの作成を示しています。

参考:[動的セクショニング](#)

1.3.2.5.1 単一セクションの例

1.3.2.5.1.1 動物園の動物

ダチョウは時速 70 km で走ることができ、これは鳥の中で最速の地上速度です。

基本的に恐竜です。

シマウマはほぼ完全に草を食べます。

ほとんどが脂肪です。

ホッキョクアジサシはどんな鳴き声を出しますか？

1.3.2.5.2 複数のセクションの例

1.3.2.5.2.1 ダチョウ

ダチョウは時速 70 km で走ることができ、これは鳥の中で最速の地上速度です。

1.3.2.5.2.2 クロコダイル

基本的には恐竜です。

1.3.2.5.2.3 ゼブラ

シマウマはほぼ完全に草を食べます。

1.3.2.5.2.4 シール

ほとんどが脂肪です。

1.3.2.5.2.5 キョクアジサシ

ホッキョクアジサシはどんな鳴き声を出しますか？

1.3.2.6 トムソーヤ図

DocGen は Tom Sawyer Perspectives と統合して、ダイアグラムを動的に生成し、ビューに表示します。

注: この機能は、ビュー エディターと組み合わせてのみ機能します。ローカルドキュメント生成ではサポートされていません。

1.3.2.6.1 トムソーヤ図の作成

トムソーヤ図は、Tom Sawyer Diagram アクションを DocGen アクティビティに追加することによってモデルに実装されます。

次の目的は、パッケージ内のモデル要素からトムソーヤ図を生成することです。この例では、作成されたトムソーヤ図はブロック定義図ですが、次の図タイプもサポートされています。

- 内部ブロック図・パッケージ図・
- パラメトリック図・要件図
- シーケンス図・ステートマシン図・ユースケース図

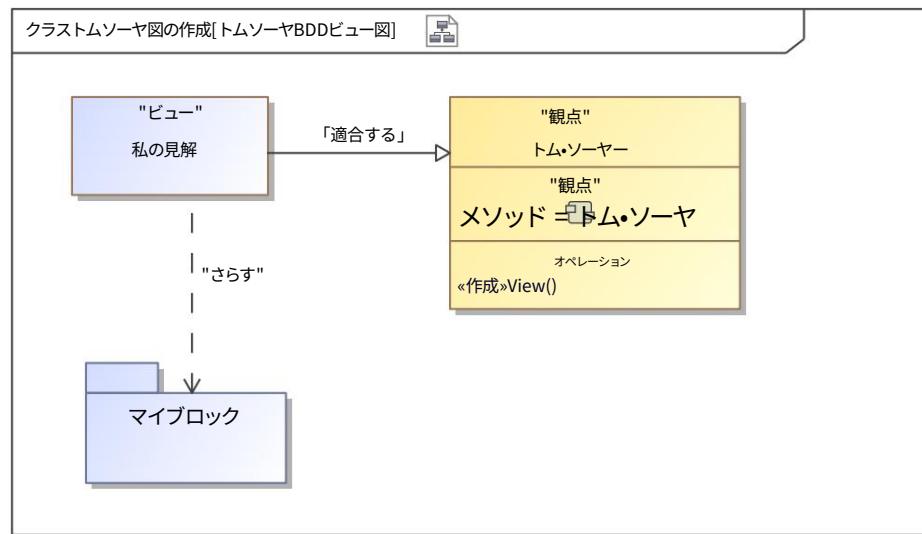


図 84. Tom Sawyer の BDD ビュー図

上に示したように、トムソーヤ図が作成される myView という名前のビューがあります。これを実現するには、どの要素を含めるか、またトムソーヤ図を生成する方法を myView に指示する必要があります。

どの要素に関心があるかを myView に伝える方法は複数あり、それは Viewpoint メソッドの構築と必要な図のタイプの両方によって異なります。この場合、パッケージ myBlocks は 4 つのブロックを含むパッケージであり、1 つは他の 3 つのブロックで構成されます (指定された構成により)。この場合、対象となるすべてのブロックが同じパッケージ内にあり、図上で不要な要素が myBlocks にならないため、上に示したように、単純にビュー (myView) からパッケージ (myBlocks) を公開します。

次に、視点トムソーヤの視点メソッドを構築する必要があります。

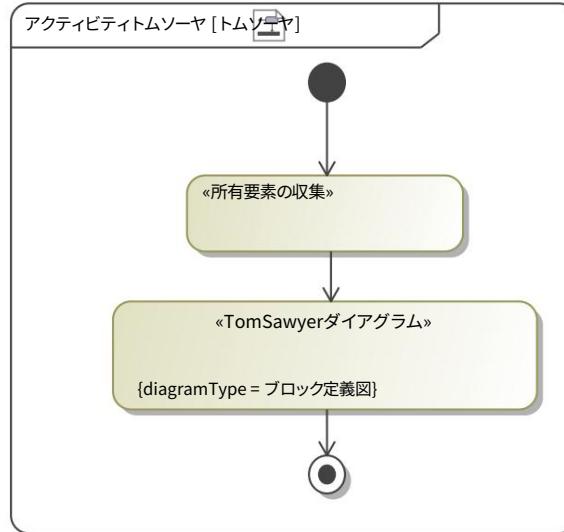


図 85. トム・ソーヤ

ビューポイントを作成したら、まずビューポイント メソッドを指定する必要があります。これを行うには、ビューポイントの View() 操作を右クリック >> 「メソッドの作成」 >> 「ダイアグラム」 >> 「ビューポイント メソッド ダイアグラム」を選択します。

この単純な例では、トムソーヤ図を生成するために必要なアクションは 2 つだけです。まず、CollectOwnedElements アクションを追加します。これにより、ビューによって公開された要素が所有するすべての要素が収集されます (この例では、パッケージ myBlocks が myView によって公開されます)。詳細については、ユーザー ガイドの「収集」セクションを参照してください。

次に、TomSawyerDiagram アクションを追加します。その名前が示すように、このアクションは Tom Sawyer ダイアグラムを生成します。ただし、最初に 1 つのオプションを定義する必要があります。 TomSawyerDiagram アクションの仕様では、図のタイプを任意の図に設定します。この場合は、ブロック定義図に設定します。

この後、ビューが生成され、Tom Sawyer ブロック定義図がビュー エディターに表示されます。この図は、ビュー エディターのインターフェイスを通じて操作できます。

1.3.2.6.1.1 myView

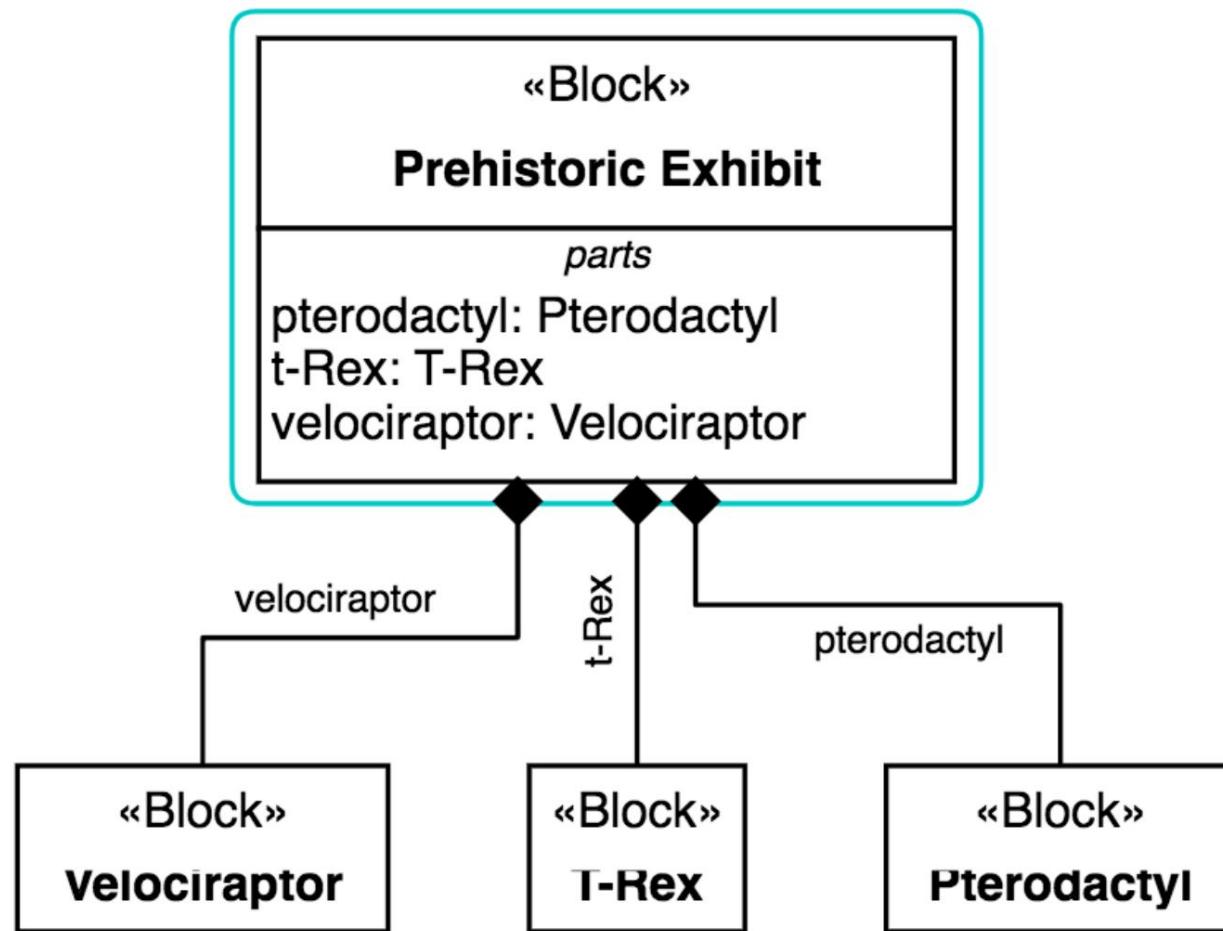


図 86. tomSawyerDiagram

1.3.2.6.2 よくある間違い

前のセクションで説明したように、パッケージを公開し、メソッドが CollectOwnedElements および TomSawyerDiagram アクションのみを使用するビュー ポイントに準拠するビューが、これらの図を生成する唯一の方法ではありません。より複雑な収集/フィルター/並べ替え操作をアクティビティ図に追加して、必要な要素を含むトムソーサー図を生成できます。以下は、トムソーサー図を使用する際に注意すべきよくある間違いの例です。

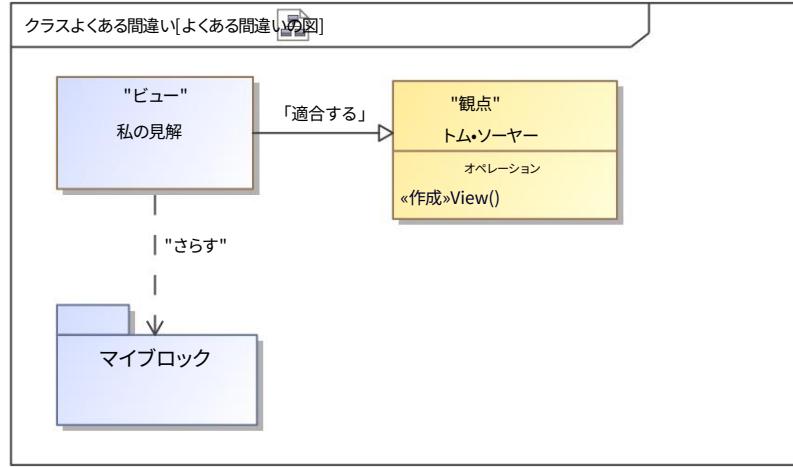


図 87. よくある間違いの図

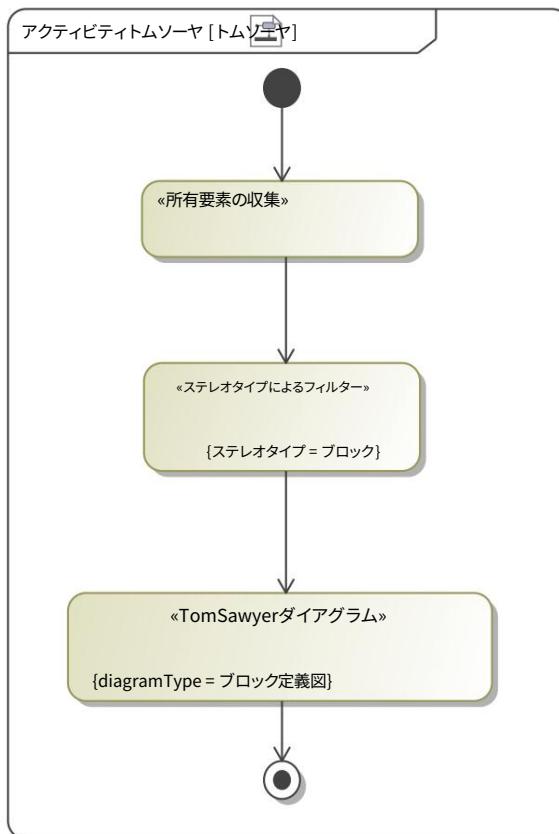


図 88. トム・ソーヤ

パッケージを公開し、内容を収集、並べ替え、またはフィルター処理してトムソーヤ図を生成しようとする場合、注意すべき点がいくつかあります。上記の myBlocks のようなパッケージには、ブロック間の関係を持ついくつかのブロックが含まれていますが、トムソーヤ図では望ましくない他の要素も含まれていると考えてください。上に示したように、単純にビューからパッケージを公開してから、Viewpoint メソッド図で CollectOwnedElements、FilterByStereotype を実行して、Tom Sawyer 図を生成することを誘惑します。ブロック以外のすべてをフィルターで除外すると、モデルはそれらのブロック間の関係を TomSawyerDiagram アクションに渡さず、結果として得られるトムソーヤ図にはブロックのみが含まれ、関連付けは含まれないため、これは間違いです。

1.3.2.6.3 内部ブロック図

以下は、Tom Sawyer の内部ブロック図を生成する方法の例です。

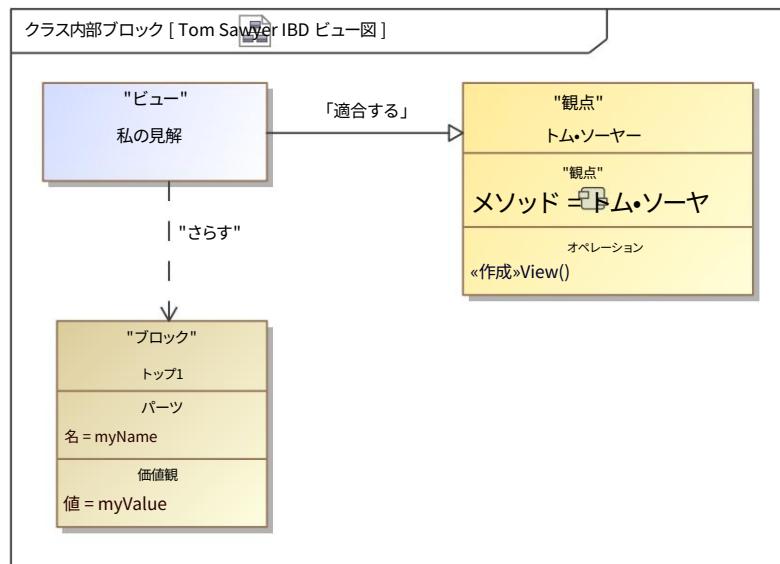


図 89. Tom Sawyer の IBD ビュー図

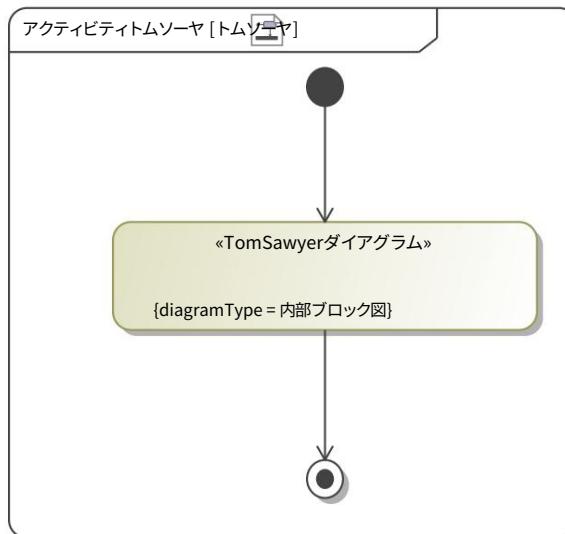


図 90. トム・ソーヤ

Tom Sawyer 内部ブロック図を作成するプロセスは、ブロック定義図の作成プロセスとは少し異なります。Tom Sawyer 内部ブロック図を生成する最も簡単な方法は、まず、図が作成されるビューから目的のブロックを公開することです。上の例では、top1 という名前のブロックが myView という名前のビューによって公開されているため、top1 の内部ブロック図が myView 上に作成されます。

次のステップでは、myView が準拠するビューポイントのビューポイント メソッド ダイアグラムを作成します。この場合、ビューポイントの名前は Tom Sawyer です。Viewpoint Method Diagram で必要なアクションは TomSawyerDiagram だけです。TomSawyerDiagram アクションの仕様では、図のタイプを内部ブロック図に設定し、関連要素の収集* を True に設定する必要があることに注意してください。

*内部ブロック図の場合、関連要素の収集がオンになっている必要があることに注意してください。ビューによって公開される要素がブロックである場合に、CollectOwnedElements が TomSawyerDiagram に接続されている Viewpoint メソッド ダイアグラムを使用しようとすると、Collect 関連要素が True か False かに関係なく、空のダイアグラムが生成されます。このため、モーダーは前の段落で説明した方法を使用して内部ブロック図を生成することをお勧めします。

1.3.2.6.4 前の図からの生成

トムソーヤ図を生成する別の方法は、最初にモデル内に図を作成し、それからトムソーヤ図を生成することです。以下は簡単な例です。

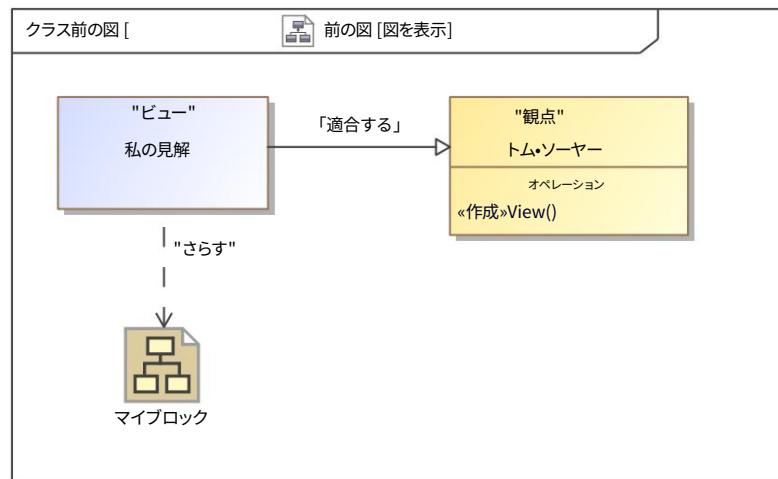


図 91. 前の図の図の表示

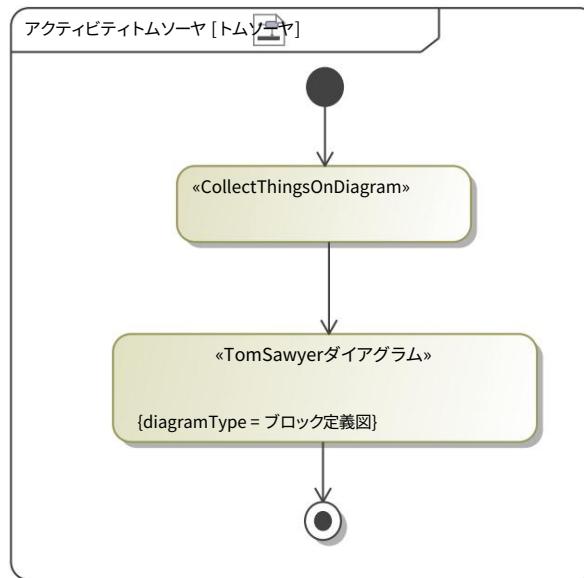


図 92. トム・ソーサー

以前に作成した図からトムソーサー図を生成するには、まずトムソーサー図が生成されるビューから目的の図を公開します。上の画像では、myView という名前のビューがダイアグラム myBlocks を公開しています。図 myBlocks は、モデル内で以前に作成されたブロック定義図です。

次に、ビューガが準拠する視点の視点方法図を作成する必要があります。この場合、ビュー myView は Tom Sawyer という名前のビューポイントに準拠します。ビューポイントメソッドダイアグラムでは、2つのアクションのみが必要です。まず、CollectThingsOnDiagram が、公開されたダイアグラム myBlocks にあるすべての要素を収集します。次に、Tom Sawyer Diagram は、指定されたタイプの図を生成します。Tom Sawyer Diagram アクションの仕様のオプション Diagram Type が、必要な図に設定されていることを確認してください。

1.3.2.7 プロット

プロットは、MagicDraw の SysML モデルから Plot アクションを介して生成できます。プロット アクションは、ビューポイントメソッドダイアグラムで使用できます。ビューポイントメソッドとビューの設定の詳細については、「[ビューポイントメソッドの作成](#)」を参照してください。

プロットでは、テーブル構造を使用して、生成されたチャートで使用されるデータを解釈します。テーブル構造の作成の詳細については、「[テーブル](#)」を参照してください。
[プロット] アクションは、テーブルデータを次のように解釈します。

- テーブルの最初の行(列ヘッダー)は、グラフの X 軸のマークとして使用されます。テーブル構造で「ヘッダーを非表示」オプションが True に設定されている場合、X 軸のマークは数値 0、1, 2, 3, ... になります。注: 最初の行の最初のセルは使用されません。。
- 表の最初の列は、プロットされる各線の名前として機能します。

- 各行（ヘッダー行を除く）はプロットされる行に対応し、その行の名前がその最初のセルにあります。
- 行とそれに続くすべてのセルには、その行の y 値が含まれます。以下の例を参照してください。

例

表 9. ヘッダーのあるテーブル

	x値1	x値2	x値3	x値4
ライン1	2	1	3	4

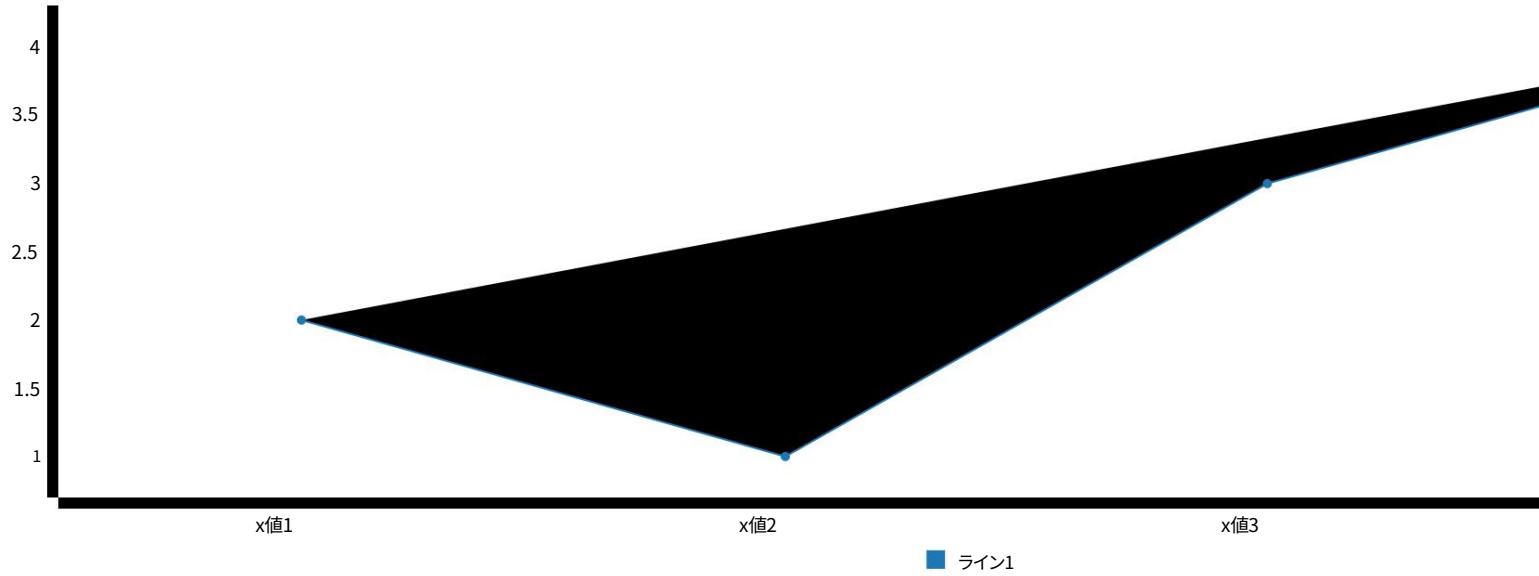


図 93. ヘッダー付きのデモ プロット

上に示したようにテーブル ヘッダーが含まれる場合、列ヘッダーは X 軸上の点として使用されます。注意すべきこと
ここでは、プロットの生成元のテーブル構造を示すために、テーブルがプロットとともに表示されますが、テーブルはそうではありません。
プロットと一緒に表示する必要があります（これについては例で詳しく説明します）。

表 10. ヘッダーのないテーブル

ライン1	2	1	3	4
------	---	---	---	---

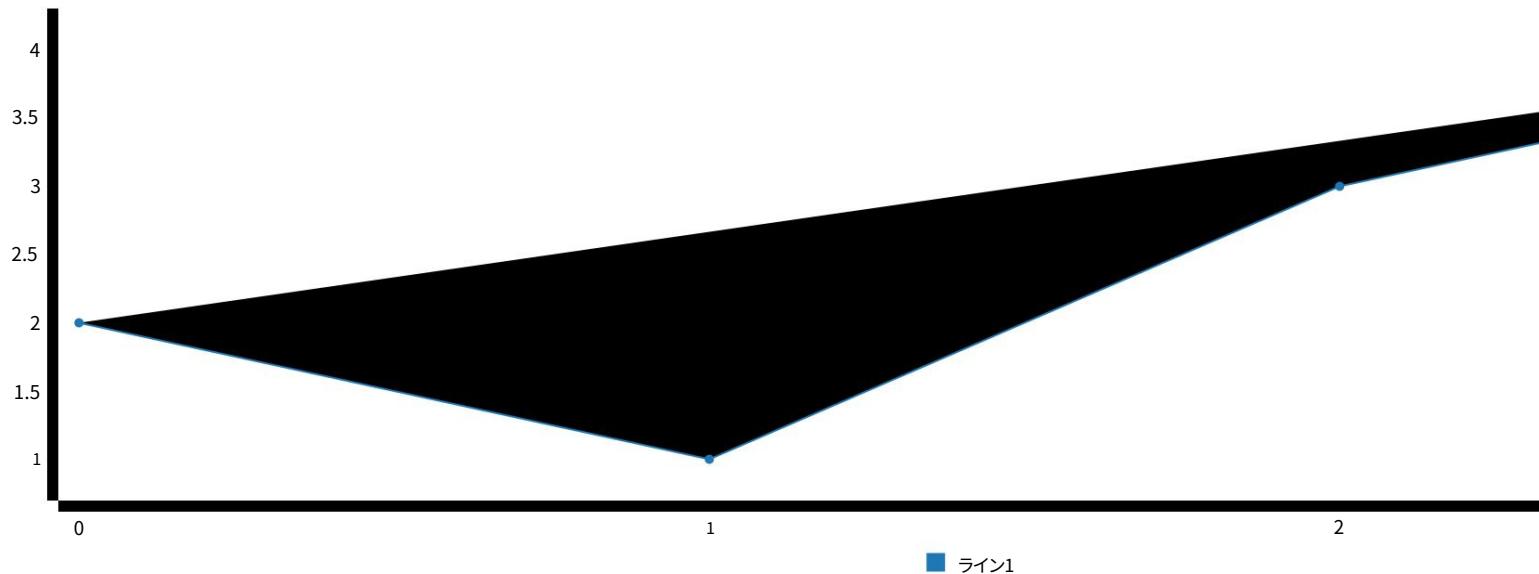


図 94. ヘッダーのないデモ プロット

テーブル ヘッダーが含まれていない場合、デフォルトの X 軸ラベルは、上に示したように、数字 0,1,2,3, ... になります。ここでは、プロットの生成元となるテーブル構造を示すためにテーブルがプロットとともに表示されていますが、テーブルはプロットと一緒に表示する必要はないことに注意してください（これについては例で詳しく説明します）。

1.3.2.7.1 プロットの作成

この例では、視点メソッドでプロット アクションを使用して折れ線グラフを作成します。値プロパティを持つブロックを使用してプロットのテーブル構造を作成しますが、テーブル構造を作成するにはさまざまな方法があり、そのどれもプロットの作成に使用できます。

視点手法図

この例では、視点メソッドでプロット アクションを使用して折れ線グラフを作成します。値プロパティを持つブロックを使用してプロットのテーブル構造を作成しますが、テーブル構造を作成するにはさまざまな方法があり、そのどれもプロットの作成に使用できます。



図 95. 折れ線グラフのブロック

上記の 2 つのブロック、リンゴとオレンジは、Line Plot Blocks パッケージ内の 2 つのブロックです。各ブロックには、プロットの作成に使用される値を含む 10 個の値プロパティがあります。たとえば、これらの値は、毎日販売される果物の数である可能性があります。

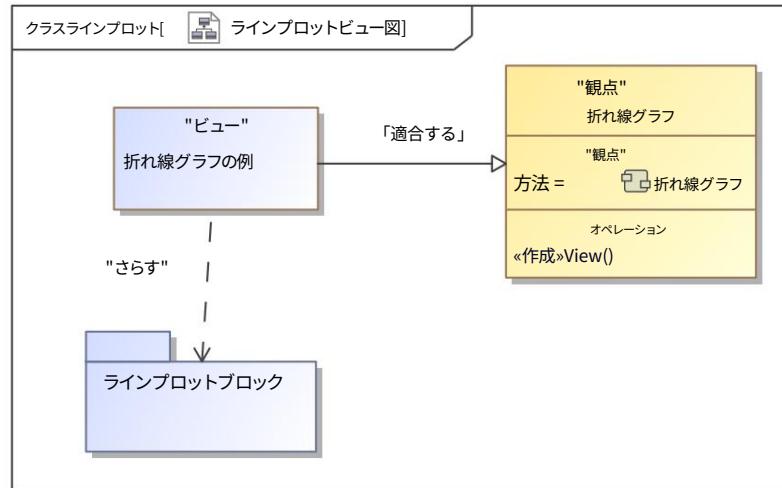


図 96. 折れ線グラフ図

プロットが表示されるビュー（この場合はLine Plot の例）を設定するには、[Line Plot Blocks パッケージ](#)を公開し、ビューポイントに準拠します。

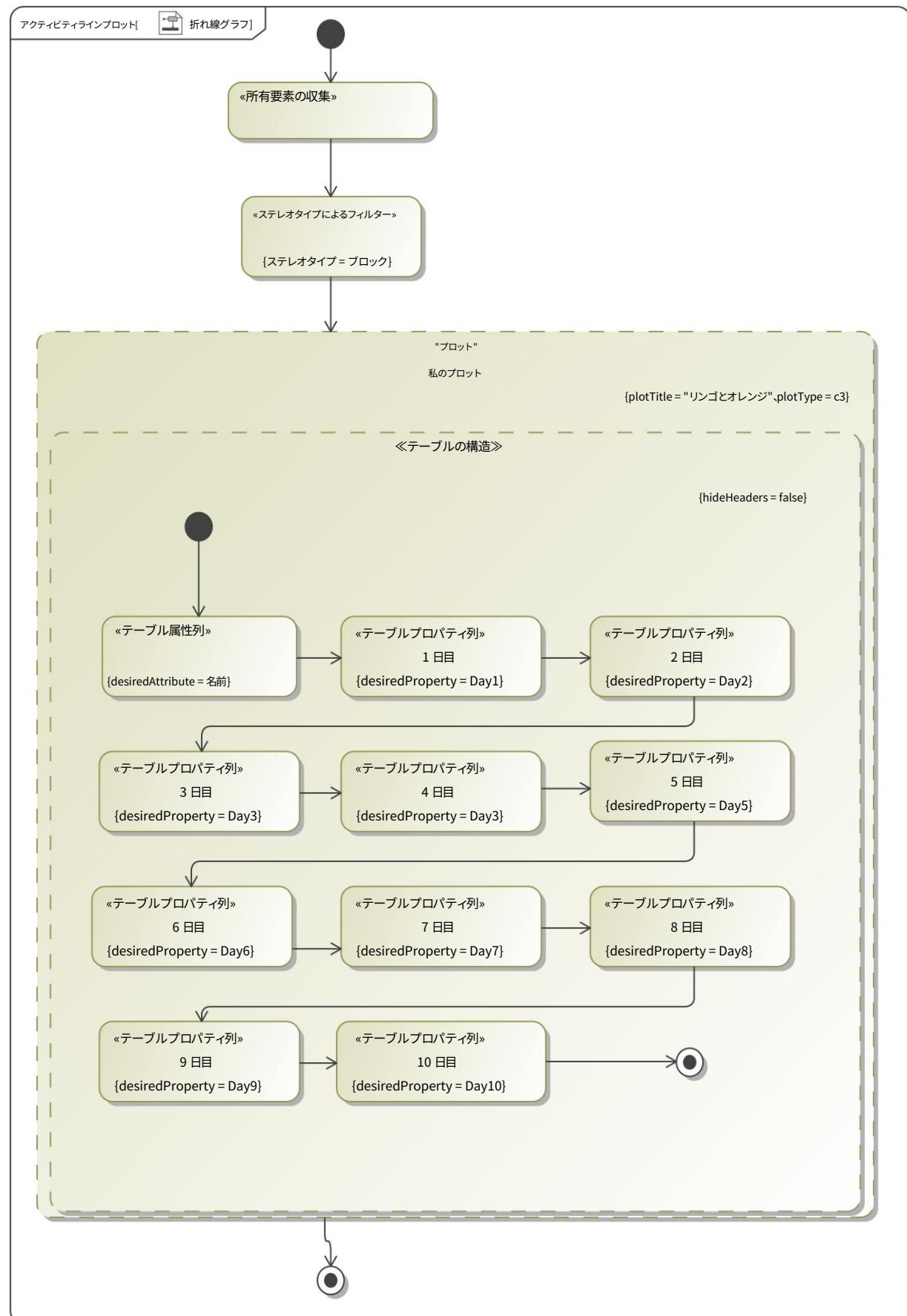


図 97. 折れ線グラフ

次に、上記の視点メソッド図を使用して、ラインプロット視点のメソッドを定義します。まずブロックを収集します
CollectOwnedElements と FilterByStereotype (ブロック) を含む Line Plot Blocks パッケージ。次に、Plot 構造を作成し、その中に
仕様では、プロット タイトルとプロット タイプを定義します (線プロットの場合はプロット タイプを c3 に設定します)。その Plot 構造内に Table 構造を作成します。
このテーブルはプロットの作成に使用され、テーブルとして表示されません。

テーブル構造では、最初の列のエントリが行の名前として機能するため、最初の列をプロックの名前に設定します。
プロットされることになります。次に、X 軸に日ごとのラベルを付けたいので、ヘッダー付きの 10 列を作成します (1 日目、2 日目、 ...)。覚えておいてください
ヘッダーは TablePropertyColumn アクションの名前です。最後に、列の値をプロパティの値に設定します。
Line Plot Blocks パッケージ内のブロックに Day1、Day2、... という名前が付けられています。テーブル構造の詳細については、「[「テーブル」を参照してください。](#)

上記のモデルで生成された結果については、[「折れ線グラフの例」を参照してください。](#)さらに、例については「[「レーダー プロットの例」を参照してください。](#)
同じデータを使用したレーダー プロットと、平行軸プロットの例については「[「平行軸プロットの例」を参照してください。](#)

1.3.2.7.1.1 折れ線グラフの例

以下は、「プロットの作成」のモデルから生成された折れ線グラフです。

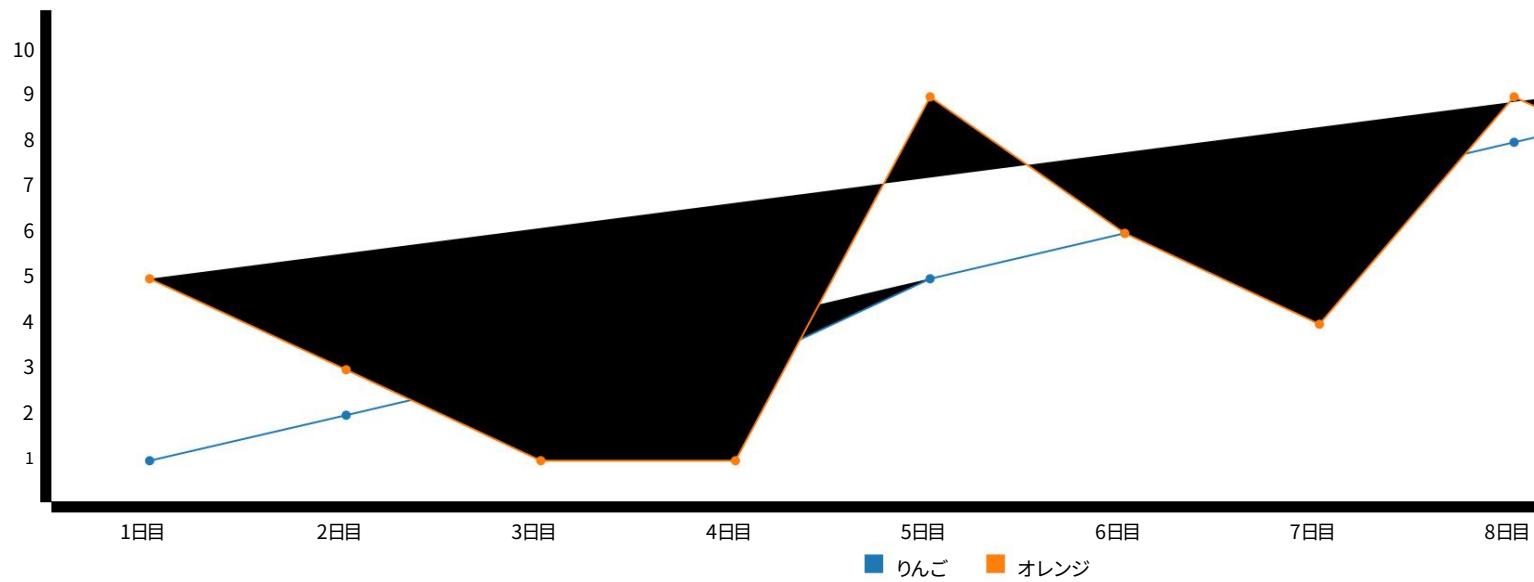


図 98. リンゴとオレンジ

1.3.2.7.1.2 レーダープロットの例

次のレーダー プロットは、「[「プロットの作成」](#)の折れ線グラフとまったく同じ方法で生成されました。唯一の違いは、
プロット構造の仕様では、プロット タイプが d3-radar に設定されました。同じブロックのパッケージが使用されました。

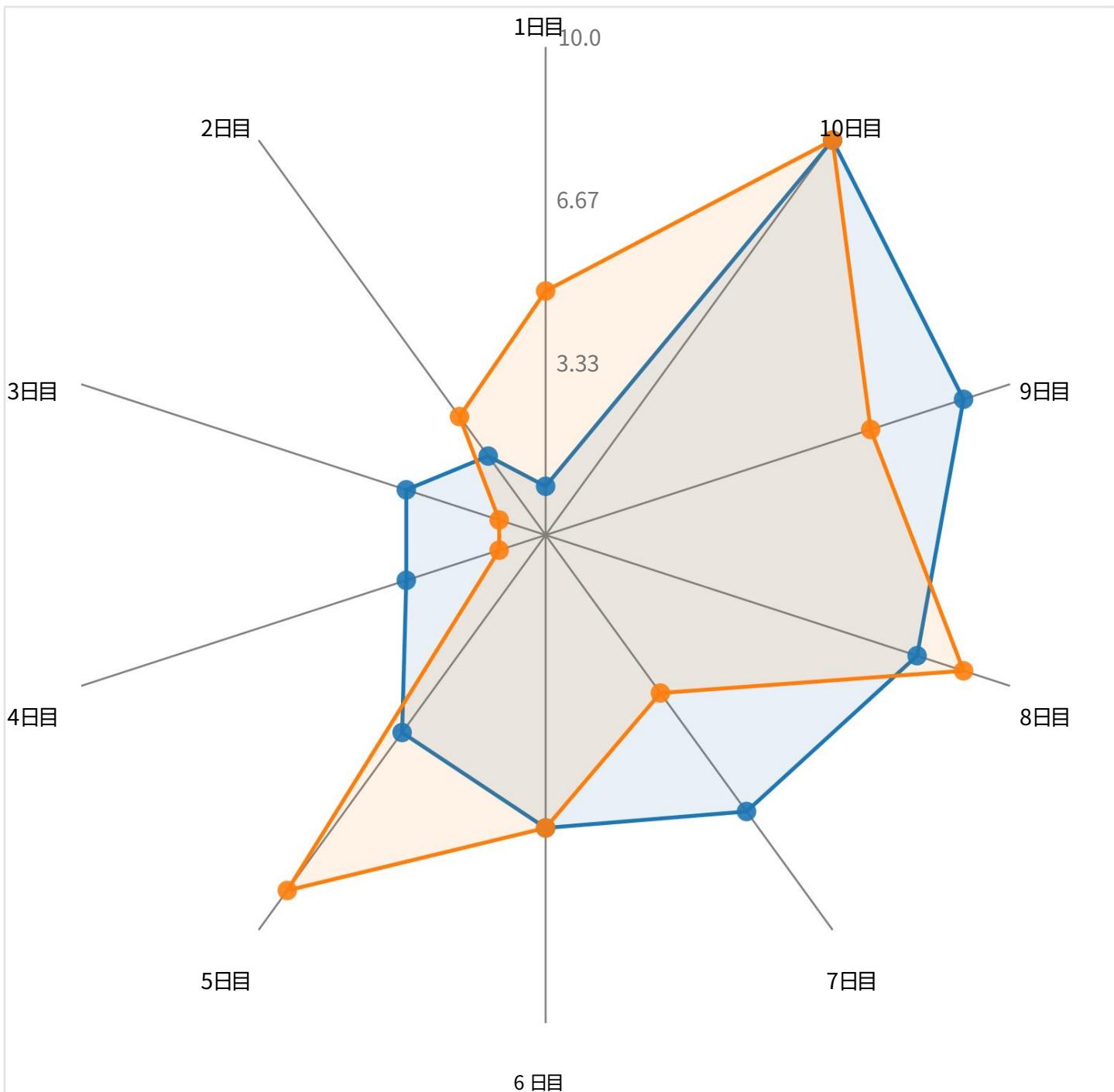


図 99. リンゴとオレンジ

1.3.2.7.1.3 平行軸プロットの例

次のレーダー プロットは、[「プロットの作成」](#)の折れ線グラフとまったく同じ方法で生成されました。唯一の違いは、プロット構造の仕様でプロット タイプが d3-平行軸に設定されていることです。同じプロックのパッケージが使用されました。

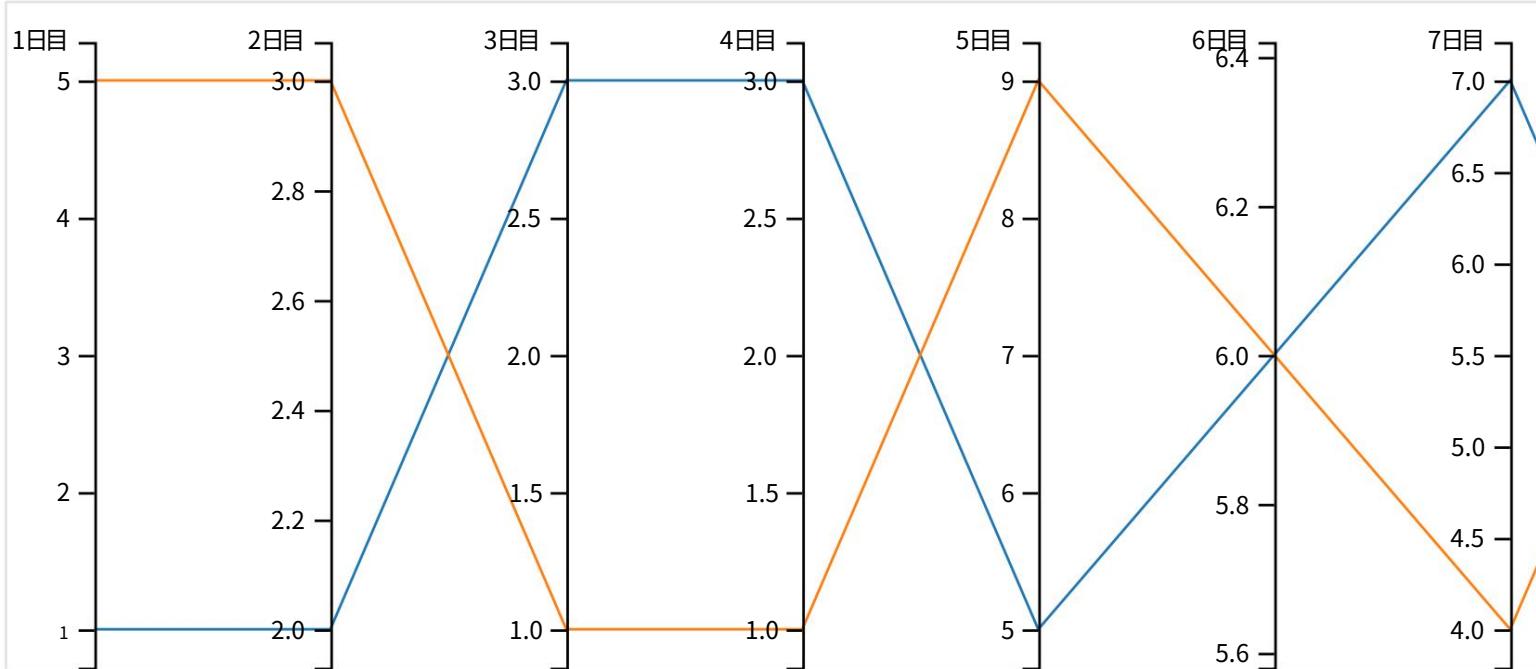


図 100. リンゴとオレンジ

1.3.2.7.2 プロットのカスタマイズ

プロットは、他のプロット タイプ、色、ラベルなど、さまざまな方法でカスタマイズできます。カスタム プロパティを使用してプロットを生成するには、Plot プロット構造の仕様の構成フィールドには、カスタム オプションを含む JSON を入力する必要があります。

以下にいくつかの例を示します。これらの例は、「[作成](#)」で示したものと同じ方法と要素を使用して作成されました。

以下で説明するいくつかの小さな変更を加えた[プロット](#)。

これらのプロットは、多くのオプションを備えた C3.js JavaScript ライブラリを使用して作成されていることに注意してください。完全なドキュメントを参照してください
<https://c3js.org/>で。

表 11. リンゴとオレンジのデータ

	1日目	2日目	3日目	4日目	5日目	6日目	7日目	8日目	9日目	10日目
りんご	1	2	3	3	5	6	7	8	9	10
オレンジ	5	3	1	1	9	6	4	9	7	10

参考までに、この表は次のプロットを作成するために使用されるデータです。繰り返しますが、テーブルを次のように生成する必要はありません。

プロット、ここでは明確するために使用されています。

ドーナツチャート

ドーナツ チャートを作成するには、プロット構造の仕様でプロット タイプを c3 に設定し、プロットで次の JSON を使用します。

構成フィールド (仕様にもあります)、

```
{
  "オプション": {
    "データ": {
      "タイプ": "ドーナツ"
    },
    "ドーナツ": {
      "title": "リンゴ vs. オレンジ"
    }
  }
}
```

注: 上記の JSON では、「タイトル」はドーナツ円の内側に配置されるテキストを示しますが、プロット構造の仕様で定義されたプロット タイトルはドーナツの下に太字のテキストを作成します。

プロット構造にネストされたテーブル構造の仕様には、ヘッダーを非表示と呼ばれるオプションがあり、この例では、そのオプションは True に設定されています。この例で Hide Headers を False に設定すると、追加の凡例エントリが生成されます。詳細については、[「一般的な問題」](#)を参照してください。

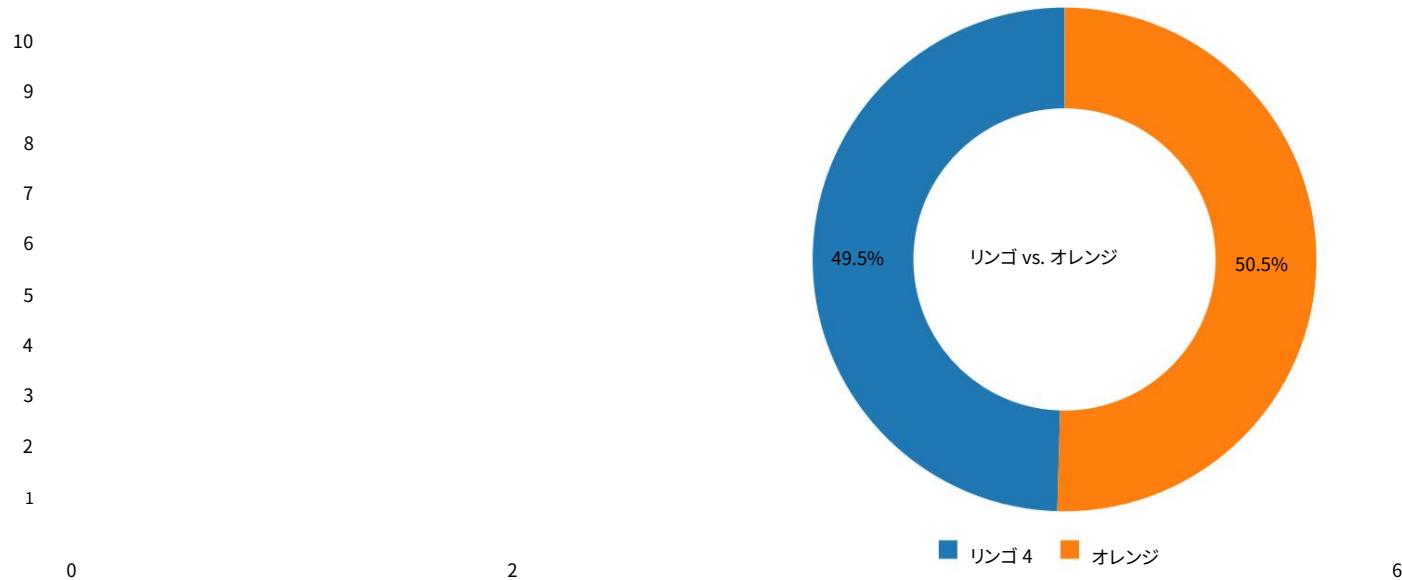


図 101. リンゴとオレンジ

上のプロットは、[プロット構成] フィールドで上記の JSON を使用した結果です。

散布図

散布図の作成は、ドーナツの作成に似ています。プロット タイプを c3 に設定し、プロット構成で次の JSON を使用します。

```
{
  "options": {
    "data": {
      "type": {
        "scatter": {
          "axis": {
            "x": {
              "label": "Days"
            },
            "y": {
              "label": "販売数"
            }
          }
        }
      }
    }
  }
}
```

上記の JSON では、複数のカスタム機能が表示されます。散布図の作成に必要なのは「type」を「scatter」に設定することだけです。「axis」の下にある他の 2 つのエントリは、x 軸と y 軸のラベルを設定します。

プロット構造にネストされたテーブル構造の仕様には、ヘッダーを非表示と呼ばれるオプションがあり、この例では、そのオプションは True に設定されています。この例で Hide Headers を False に設定すると、追加の凡例エントリが生成されます。詳細については、[「一般的な問題」](#)を参照してください。

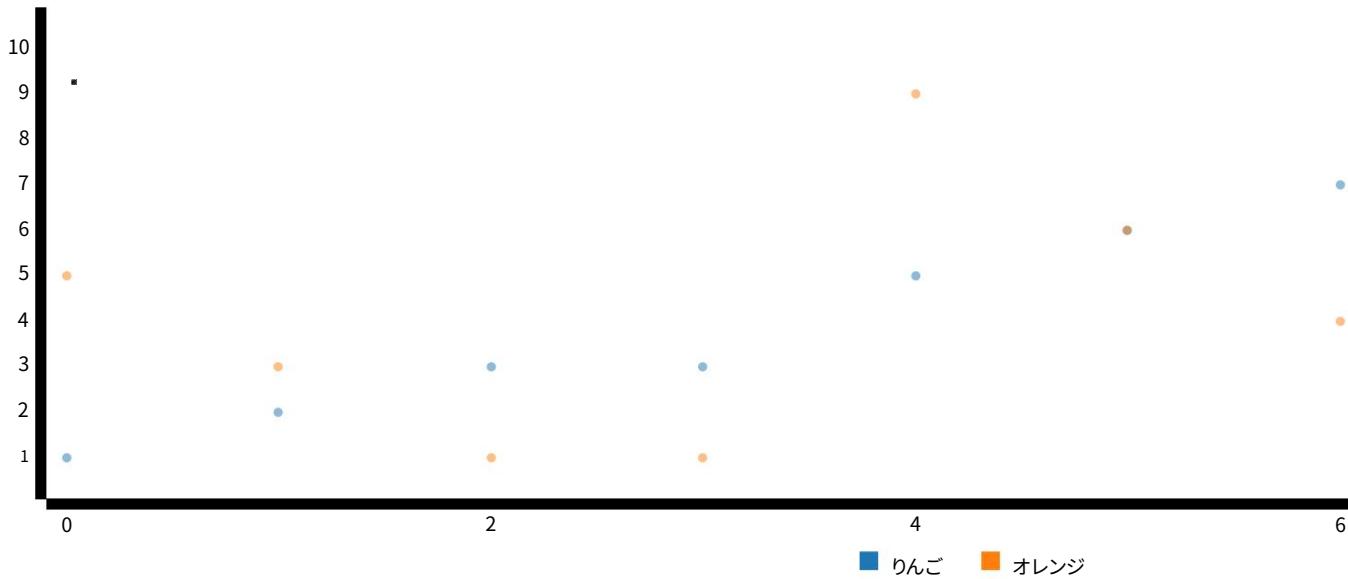


図 102. リンゴとオレンジ

上のプロットは、[プロット構成] フィールドで上記の JSON を使用した結果です。

1.3.2.7.3 一般的な問題

以下は、プロットを扱うときによく発生するいくつかの問題です。

ヘッダー

プロットが表示されない理由は数多くありますが、プロットのタイトル（プロット構造の仕様で定義されている）は生成されるが、プロット自体は生成されない場合、ヘッダー設定に問題があることがあります。最も一般的なヘッダーの問題は次の 2 つです。

- 1.プロットが生成されない:プロットの生成に使用されるビューポイント メソッド ダイアグラムでは、「x 座標」がテーブル列の名前として使用されますが、列を作成するアクション（TablePropertyColumn など）に名前が付けられていない場合は、この問題が発生する可能性があります。プロットが空になったり、すべての y 値が同じ x 値を持つなど、奇妙な動作が発生します。これを回避するには、すべての列アクションに名前が付いていることを確認してから、テーブル構造の仕様で [ヘッダーを非表示] オプションを使用して、X 軸にヘッダーナーを使用するか数値を使用するかを選択します。注: Hide Headers を True に設定すると、X 軸に数値が表示されますが、False に設定すると、X 軸に列の名前が使用されます。
- 2.追加の凡例エントリ:特定の種類のプロット、特にカスタム プロットでは、X 値を凡例のデータ セットとして表示することができます。これを修正するには、Hide Headers を True に設定します。

1.3.3 その他

表 12. DocGen メソッド

メソッド名	メソッドの説明
シミュレートする	Cameo Simulation Toolkit シミュレーションを実行し、結果をテーブルとして表示します。

1.3.3.1 シミュレーション

DocGen には、ドキュメントの生成時に Cameo Simulation Tool Kit のシミュレーションを実行する機能があります。これは、最新の値とインスタンスに基づいてシミュレーションを実行し、これらの値を参照するドキュメントを生成する場合に便利です。

1.3.3.1.1 構成のシミュレート

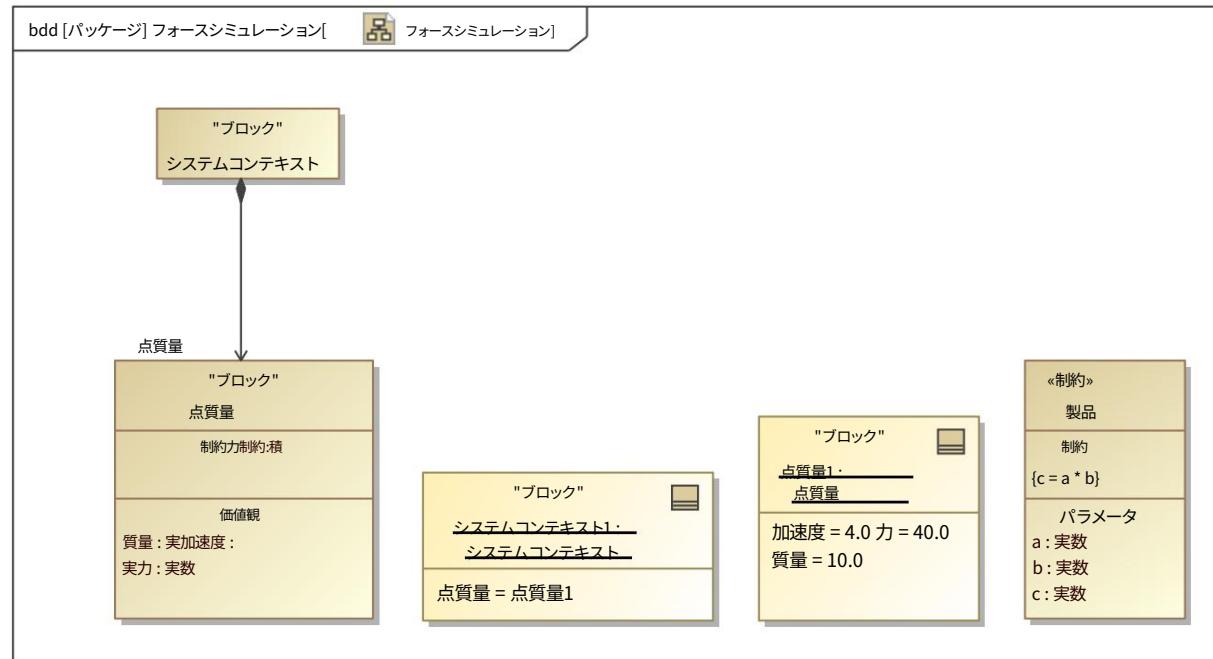


図 103. 力のシミュレーション

このシミュレーション例では、質量と加速度を考慮して、点質量にかかる力を計算します。

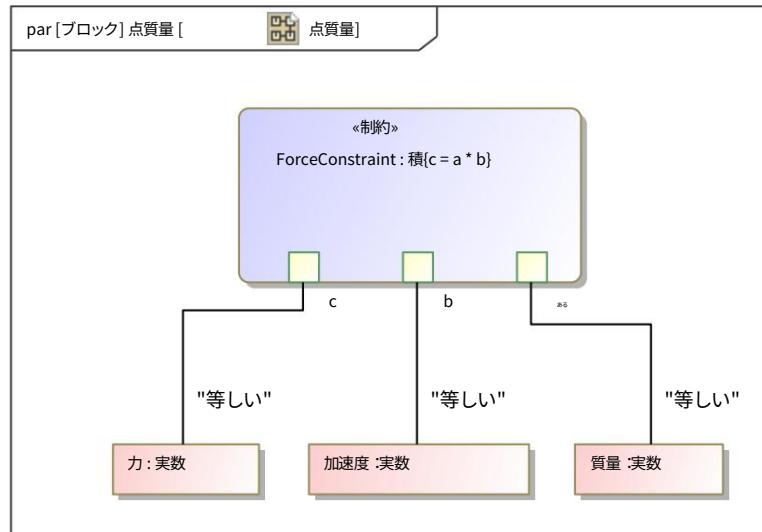


図 104. 点質量

この制約は、ニュートンの運動の第 2 法則を点質量に適用します。

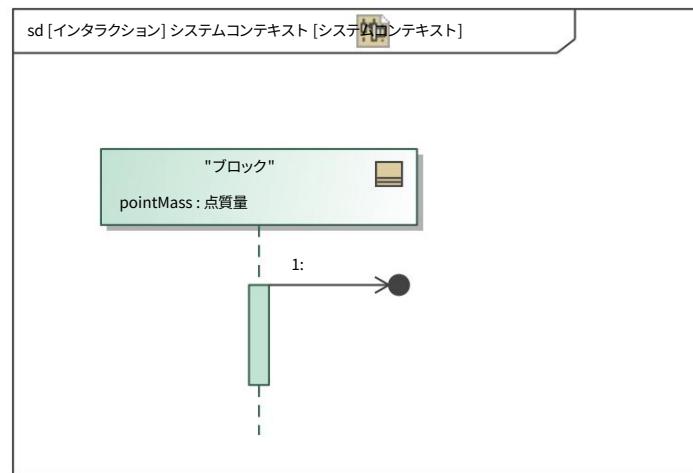


図 105. システムコンテキスト

このシーケンス図は、点質量システムがどのように実行されるべきかを指定します。



図 106. 力のシミュレーション

SimulationConfig は、制約を解決し、解決されたシステムが保存されるように、実行ターゲットと結果の場所を定義します。

インスタンスの仕様にあります。

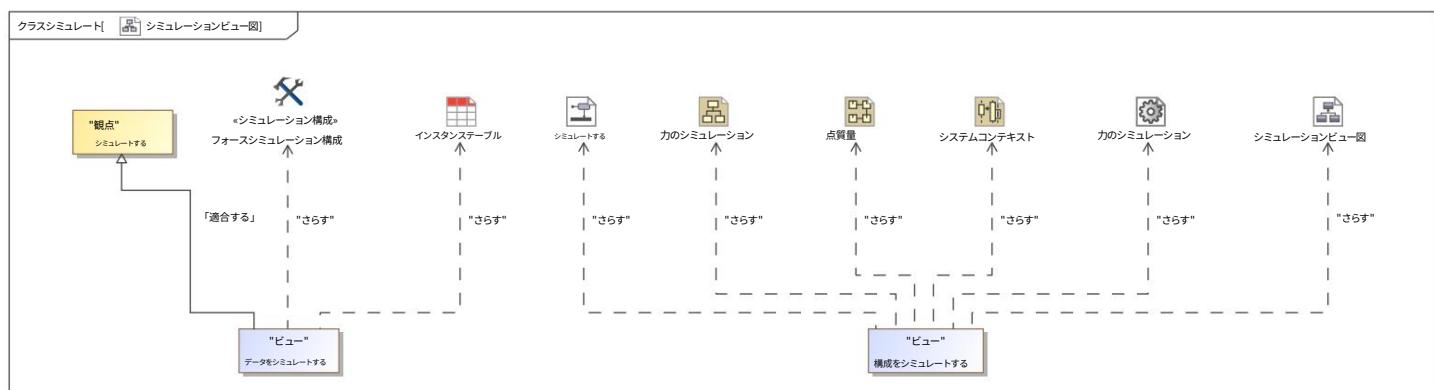


図 107. シミュレート ビューの図

シミュレーションを実行するビューは、シミュレーション ターゲット（この場合は SimulationConfig）を公開します。インスタンス テーブルは、シミュレーションの結果を表示するために公開されます。ビューポイントは、StructuredQuery 内のダイアグラムをフィルターで除外し、フィルターされたセットをシミュレートします。これは、シミュレーション例の場合は単なる SimulationConfig です。次に、ビューポイントは StructuredQuery 内のダイアグラムのみをフィルターし、フィルターされたセットをテーブルとして表示します。これは、シミュレーション例の場合は単なるインスタンス テーブルです。

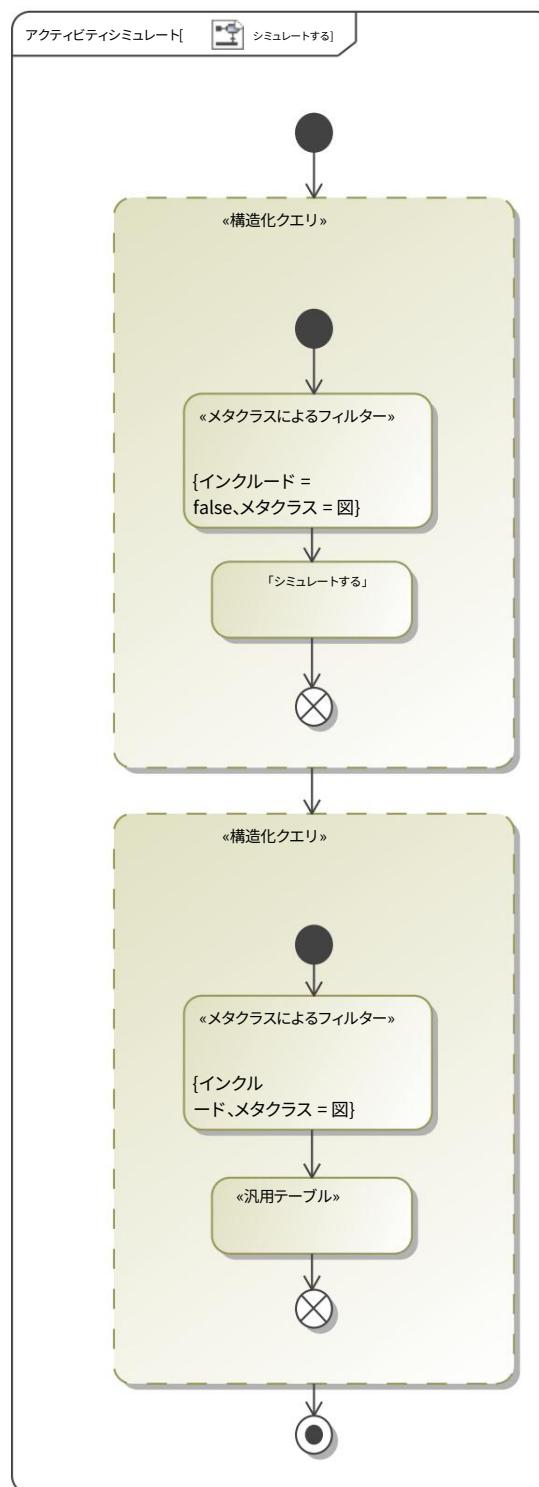


図 108. シミュレーション

Cameo Simulation Toolkit シミュレーションを実行し、結果をテーブルとして表示します。

1.3.3.1.2 データのシミュレート

表 13. インスタンス テーブル

名前	質量 :実数	加速度 :実数	力 :実数
点質量1	10.0	4.0	40.0

1.3.3.2 不透明な動作

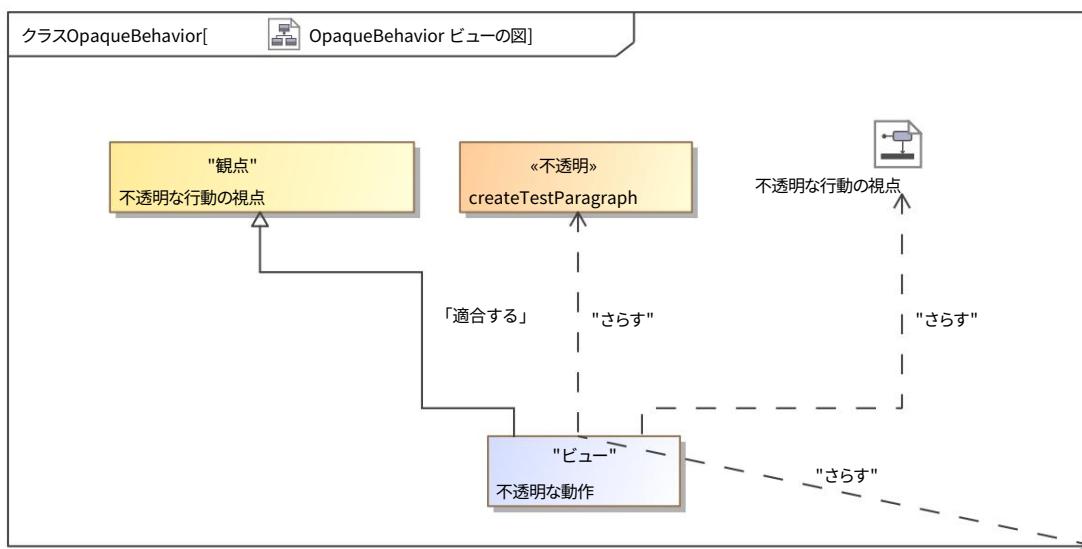


図 109. OpaqueBehavior ビューの図

このサンプル ビューは、DocGen で OpaqueBehaviors を使用して任意のコードを実行し、オプションでどのように実行できるかを示すために使用されます。任意の[JSR 223](#)でプレゼンテーション要素を生成するGroovy, Scala, Pythonなどの準拠したスクリプト言語。OpaqueBehaviors にはモデル要素内のコードが含まれるため、バージョン管理され、モデルと一緒に転送されるため、ビューを生成するには外部依存関係が必要です（プラグイン、ユーザースクリプトなど）。

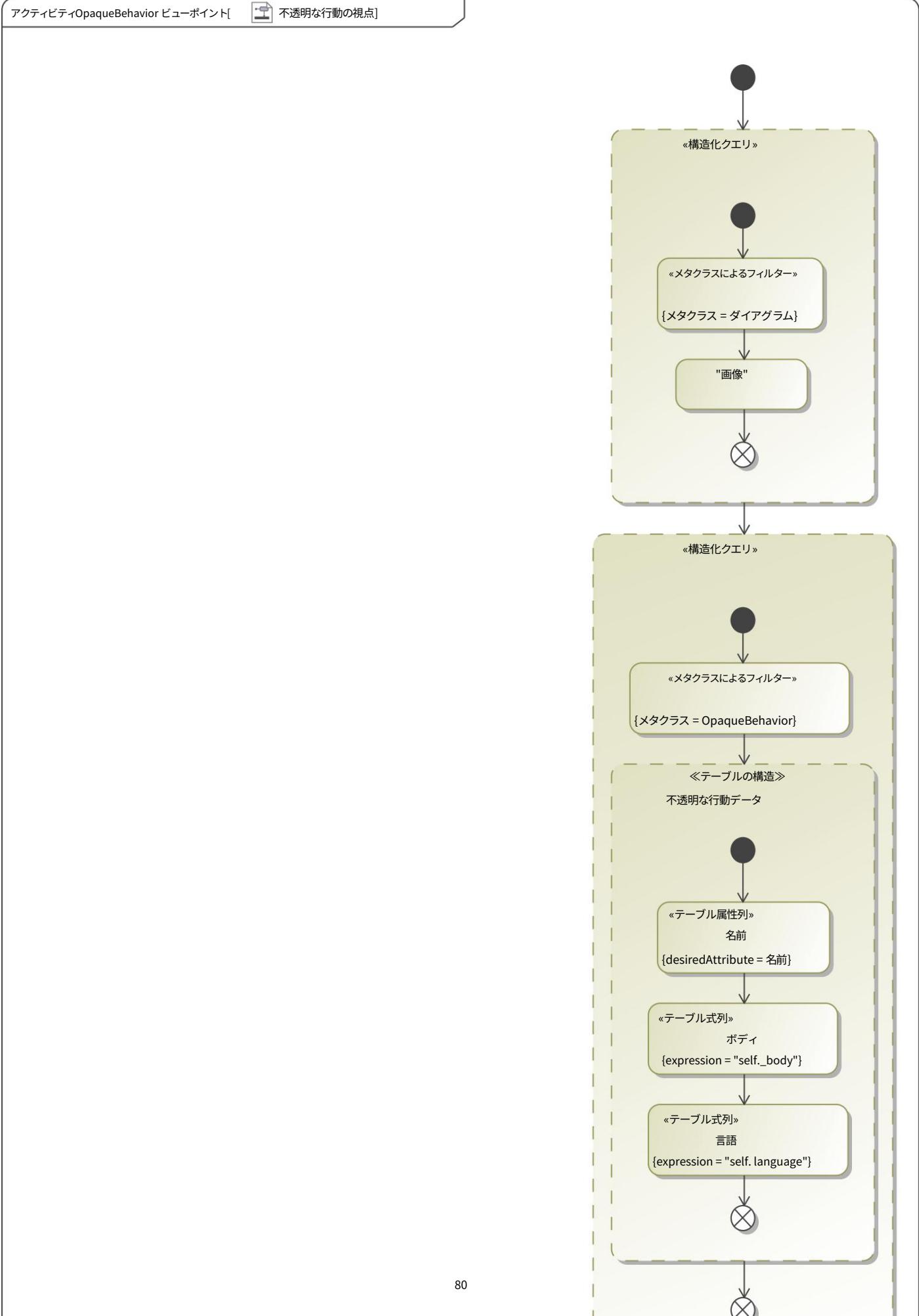


図 110. OpaqueBehavior ビューポイント

このビューポイントは、createTestParagraph という名前の OpaqueBehavior の例の使用パターンを示します。DocGen で使用されるすべての OpaqueBehavior には、実行時に 4 つの入力が提供されます。

- ExposureElements:ビュー生成の結果としてビヘイビアに公開される要素のリスト。
- forViewEditor:生成のターゲットがビュー エディターである場合、つまりローカル エディターではない場合に true となるブール値。
- 世代。
- OutputDirectory:ローカル生成を使用する場合の出力ディレクトリへのパスを含む文字列。
- context:コードを実行している gov.nasa.jpl.mbee.mdk.model.BehaviorQuery オブジェクト。この入力は次のように公開されます
- 上級ユーザー向けですが、ほとんどのユースケースでは必要ありません。

この例では、OpaqueBehavior は Groovy を使用して変数のすべての値を出力する段落を作成します。OpaqueBehaviors は、単一の gov.nasa.jpl.mbee.mdk.docgen.docbook.DocumentElement またはそれらのコレクションを返すことができます。他の出力は無視され、ユーザーには通知ウィンドウで警告が表示されます。コード内のエラーも通知ウィンドウに表示され、完全なスタックトレースがログ ファイルに出力されます。

[OpaqueBehaviors については、MagicDraw ドキュメント](#)を参照してください。実装の詳細については、

表 14. 不透明な動作データ

名前	体	言語
createTestParagraph	<pre>return new gov.nasa.jpl.mbee.mdk.docgen.docbook.DBParagraph('context は "' + context + '"', exportedElements は forViewEditor は + ExposureElements + "'." + '"outputDirectory は forViewEditor + "'. "' + outputDirectory + "'.")</pre>	グローバル

コンテキスト

は「BehaviorQuery(),callBehaviorAction=com.nomagic.uml2.ext.magicdraw.actions.mdbasicactions.impl.CallBehaviorActionImpl@aa1951dd」です。公開要素は

「[com.nomagic.uml2.ext.magicdraw.commonbehaviors.mdbasicbehaviors.impl.OpacityBehaviorImpl@ca48b6ad,
com.nomagic.uml2.ext.magicdraw.classes.mdkernel.impl.DiagramImpl@45bdb0c2,com.nomagic.uml2.
ext.magicdraw.classes.mdkernel.impl.DiagramImpl@74dfc5b7]」。forViewEditor は「true」です。出力ディレクトリが
「null」です。

1.4 OCL制約の作成と評価

このセクションでは、オブジェクト制約言語 (OCL) と、それが MDK 内でどのように使用されるかを簡単に紹介します。これは OCL の包括的な説明ではないことに注意することが重要です。このチュートリアルのどの時点でも、言語のより詳細な説明 (つまり、特定の構文ルール) が必要な場合は、参照できるリソースが豊富にあります。いくつか見つかりましたので、次のセクションでいくつか紹介します。

1.4.1 OCL とは何ですか?なぜ使用するのですか?

OCL 入門書 ブラッドリー・クレメント著

OCLとは何ですか?

[オブジェクト制約言語 \(OCL\)](#)テキスト言語 (QVT のサブセット)これを使用すると、Jython、QVT、Java などで外部コードを記述する必要があるさまざまな方法でビューとビューポイントをカスタマイズできます。OCL は、オブジェクトの不变条件と、指定された操作の前後の条件を定義するために使用されます。これにより、モデル要素とそのプロパティのより高度なクエリが可能になります。

このセクション全体を通じて、「式」と「操作」という言葉が頻繁に使用されます。OCL の「式」は、必要なものを得るために組み合わせができる OCL の「操作」のステートメントです。これらの式は、必要に応じて非常に単純にも複雑になります。

例えば：

1.n ()

```
2. r('analysis:characterizes').oclAsType(Dependency).source.m('mass').oclAsType(Property).defaultValue.oclAsType(LiteralString)
.value.toInteger()=eval(self._constraintOfConstrainedElement->asSequence()->at(1).oclAsType(Constraint)
.specation.oclAsType(OpaqueExpression)._body->at(1))
```

1と2はどちらもOCL式ですが、2がかなり複雑なことがあります。番号1は、式として独立した単一の演算です。これらの式は、慎重に選択された演算を文字列化し、それらを適切な構文で区切ることによって構築されていることに注意することが重要です。上の複雑な例では、ピリオド(.)と矢印(->)が多数あることがあります。これらの記号は両方とも、式の1つの部分を次の部分から分離します（これら2つだけではありません）。それをいつ使用するかを理解するには、ある程度の練習が必要です。OCL構文のより広範かつ高度な説明については、この[リンク](#)を確認してください。

OCLを使用する理由

OCLを使用すると、モデルデータの収集、フィルター、並べ替え、制約、表示の方法を指定できます。モデル要素に対する制約はMDKの背後にある原動力であり、このドキュメントの前半で説明した特定の要素の相互作用については、OCLを介して説明します。

ビューポイント内でOCL式を処理するために特別に作成されたいくつかの新しい要素も導入されます。

モデル要素の日常的な収集とフィルターの場合、OCLはおそらく最も良い選択ではありませんが、モデルのより高度なクエリの場合はゴールデンチケットになる可能性があります。このアイデアを概説する例を次に示します。

OCL構文は、UMLメタモデルを通じてモデル情報にアクセスするため、注意が必要な場合があります（Magicdrawインストールディレクトリのマニュアルの下にあるUMLメタモデルマニュアルを参照してください）。ここでは、Jython、Java、QVTなどで外部スクリプトを作成することなく、ビューをより簡単にカスタマイズするためのOCL式を作成する方法に関するヒントをいくつか紹介します。このトピックは、このドキュメントで前述した以前のトピックよりも高度であることに注意してください。この時点では、視点メソッド図がどのように機能するかを理解し、1つのアクションから次のアクションに段階的に進み、最終結果でどのような要素が期待されるかを理解できることを前提としています。

OCL式も同じように機能するため、これは重要です。

OCLリソース

OCLについてのヘルプが必要な場合は、[OCL Cheat Sheet](#)を試してください。または、OCLなどのWebを検索します。以下のセクションでもいくつかの例を示します。OCLチートシートは、OCLがどのように機能するかを基本的に理解すると非常に役立ちますが、ある程度理解するまでは混乱する可能性があります。次のセクションでは、このリソースへの参照と、それがどのように役立つかについて説明します。

NoMagicのUMLメタモデルは、さまざまなUMLタイプのOCL式で参照できるオブジェクトを指定します。メタモデルのPDFマニュアルは、MagicDrawインストールのマニュアルフォルダにあります。Eclipseを使用している場合は、メニューから「[ウィンドウ] -> [ビューの表示] -> [その他]」からMetamodel Explorerビューを開いてみてください。多くのメタモデルがある可能性がありますが、com.nomagic.uml2メタモデルを探してください。ビューにはタイプを検索できるボタンがありますが、複数のUMLメタモデルが存在する可能性があるため注意してください。MagicDrawのUMLメタモデルが必要です。OCLで参照できる継承されたメンバーを表示するボタンもあります。

1.4.2 OCLブラックボックス式とは何ですか？

ブラックボックス操作は、式の構造を単純化するためによく使用されます。次のセクションでは、それらを認識できるように、これらのいくつかについて説明します。ただし、ブラックボックスの操作について説明する前に、ビューポイントメソッドのアクション<<TableExpressionColumn>>について説明することが重要です。これは、次のビューポイントメソッドの例で使用されるためです。

<<TableExpressionColumn>> OCL式を<<TableStructure>>内のターゲット要素に適用します。これにより、ビュー エディターに表示されるテーブルの列が設定されます。これを使用すると、他の方法では困難または不可能な要素や関係に対する操作を連鎖させることができます。

例ではこのアクションに直接注目してください。OCL式がどのように処理されるかを理解することが重要です。

次に進む前のもう1つの重要なトピックは、OCL式での「キャスト」の考え方です。多くの場合、OCL操作から結果が返された場合、その後の操作を実行するには、その結果をクエリから期待される要素としてキャストする必要があります。これはOCLの性質であり、慣れるのに時間がかかります。多くの場合、返された要素を忘れずにキャストすることでエラーを修正できます。前に説明したチートシートには、キャストに必要な操作が記載されています。それらは以下に示されており、次の例で実装および説明されます。例を確認しながら、必ず戻ってこれらの操作を参照してください。

以下は、式の短縮表現として使用される OCL ブラックボックス操作です。これらについては、次のセクションで説明します。

- `m()`、`member()`、または `members()` は、所有されている要素を返します。
- `r()`、`relationship()`、または `relationship()` は、所有するすべての関係、または要素がターゲットであるすべての関係を返します。 • `n()`、`name()`、または `names()` は、名前を返します。 • `t()`、`types()` は、返します。すべてのタイプ（ステレオタイプ、メタクラス、Java クラスとインターフェース）。 • `type()` は、選択された要素のタイプのみを返します。 • `s()`、`stereotype()`、または `stereotypes()` はステレオタイプを返します。これは基本的に `applyStereotypeInstance.classifier` の短縮形です。 `s()` と `Stereotypes()` はすべてのステレオタイプを返す必要があり、`stereotype()` は 1 つだけを返す必要があります。
- `e()`、`evaluate()`、または `eval()` は、要素から取得した ocl 式を評価します。 • `value()` は、プロパティまたはスロットの値を返します。 • `owner()` は、所有者と所有者の所有者を再帰的に返します。 • `log()` は、次のように出力します。通知ウィンドウ • `run(View/Viewpoint)` は、指定された入力を使用して単一のビューまたはビューポイントで DocGen を実行し、結果を取得します。

これらのビューの多くは、視点例の結果を参照しています。これらの結果は、[現在のモデルデータ] ビューで見られるものと似ています。

図 111. 動物

1.4.2.1 値()

`value()` 式は、プロパティの値を取得するのに便利なツールです。以下の例では、`Animals` パッケージが、示されている視点メソッド図に公開されています。式 [`self.get('noise').v().v()`] を調べると、`get()` が「noise」という名前の要素（この場合はすべてのプロパティ）を収集していることがわかります。次に、`v()` が 2 回使用され、1 回目はプロパティのデフォルト値であるリテラル文字列を取得します。その後、リテラル文字列のテキストを取得するために再度使用されます。

図 112. `value()` ビューの図

図 113. 値のブラックボックスのデモ

1.4.2.1.1 ビューの例

表 15.<>

動物の鳴き声	
ニヤー	
インチキ	
もー	
ウサギ	
吠える	

1.4.2.2 リレーションシップ「`r()`」

`r()` の演算概要は以下の通りです。以下のビューポイント メソッドは、`r()` を単独で使用するだけでなく、より複雑な式の一部としても使用するように設計されています。以下の箇条書きリストでは、以下の `<<TableExpressionColumn>>` で使用される OCL 式について説明します。

- `r()`、`relationship()`、または `relationship()` は、所有されているすべての関係を返します。 • 最初の `<<TableExpressionColumn>>` の `r('likes')` は、名前またはタイプ 'likes' のすべての関係を取得します。 • 2 番目の `<<TableExpressionColumn>>` の `r('likes').oclAsType(Dependency).target` は取得します。 「好き」関係を取得し、`Dependency` としてキャストし、関係のターゲット (Cat) を収集します。 `r()` が関係「likes」を返す場合、操作の実行を続けるためには、返されたものをキャストする必要があることに注意してください。これは `oclAsType()` によって行われます。これらの操作に慣れるために、前に説明したチートシートに目を通してください。
- 注：ビューポイント方法図のフィルターは無視してください。これらは、この記事で説明する概念にとって重要ではありません。

- <<TableExpressionColumn>> アクティビティは、ビューポイントで OCL 式が定義されている場所であることに注意してください。
メソッド図。

図 114. relationship() ビューの図

図 115. 関係のブラックボックスのデモ

1.4.2.2.1 ビューの例

表 16. 関係 好き 関係

要素の名前		いいね対象
犬	好き	猫

1.4.2.3 「n()」の名前

- n()、name()、または names() は名前を返します。

この例では、式「n()」はアクションに渡された要素の名前を返します。この場合は「犬」です。

図 116. 名前ビューの図

図 117. 名前ブラックボックスのデモ

1.4.2.3.1 ビューの例

表 17. <>

要素の名前
犬

1.4.2.4 ステレオタイプ「s()」

- s()、stereotype()、または stereotypes() はステレオタイプを返します。これは基本的に applyStereotypeInstance.classifier の短縮形です。 s() と Stereotypes() はすべてのステレオタイプを返す必要があります、 stereotype() は1つだけを返す必要があります。

この例では、Dog が公開され、そのステレオタイプがテーブルに入力されます。

図 118. ステレオタイプ ビューの図

図 119. ステレオタイプのブラックボックスのデモ

1.4.2.4.1 ビューの例

表 18. 固定観念

要素の名前	要素のステレオタイプ
犬	ブロック

1.4.2.5 メンバー「m()」

- m()、member()、または members() は、所有されている要素を返します。
 - m('text') は、名前またはタイプが 'text' のメンバーを取得します

例：

- 要素 Dog が再び公開されるため、 m('noise') は名前が 'noise' のメンバーを取得し、その名前を特定のテーブルの属性列。

- ・ビュー・ポイント・メソッドの <<TableExpressionColumn>> アクションでは、異なる OCL 式が使用されていることに注意してください。の「属性」列には、名前が「noise」であるメンバーが返されます。「属性のデフォルト値」アクションの式は、同じプロパティを収集し、それをプロパティとしてキャストして、デフォルト値を返します。これは、`m('noise').oclAsType(Property).defaultValue` によって行われます。

図 120. メンバービューの図

図 121. メンバー ブラック ボックスのデモ

1.4.2.5.1 ビューの例

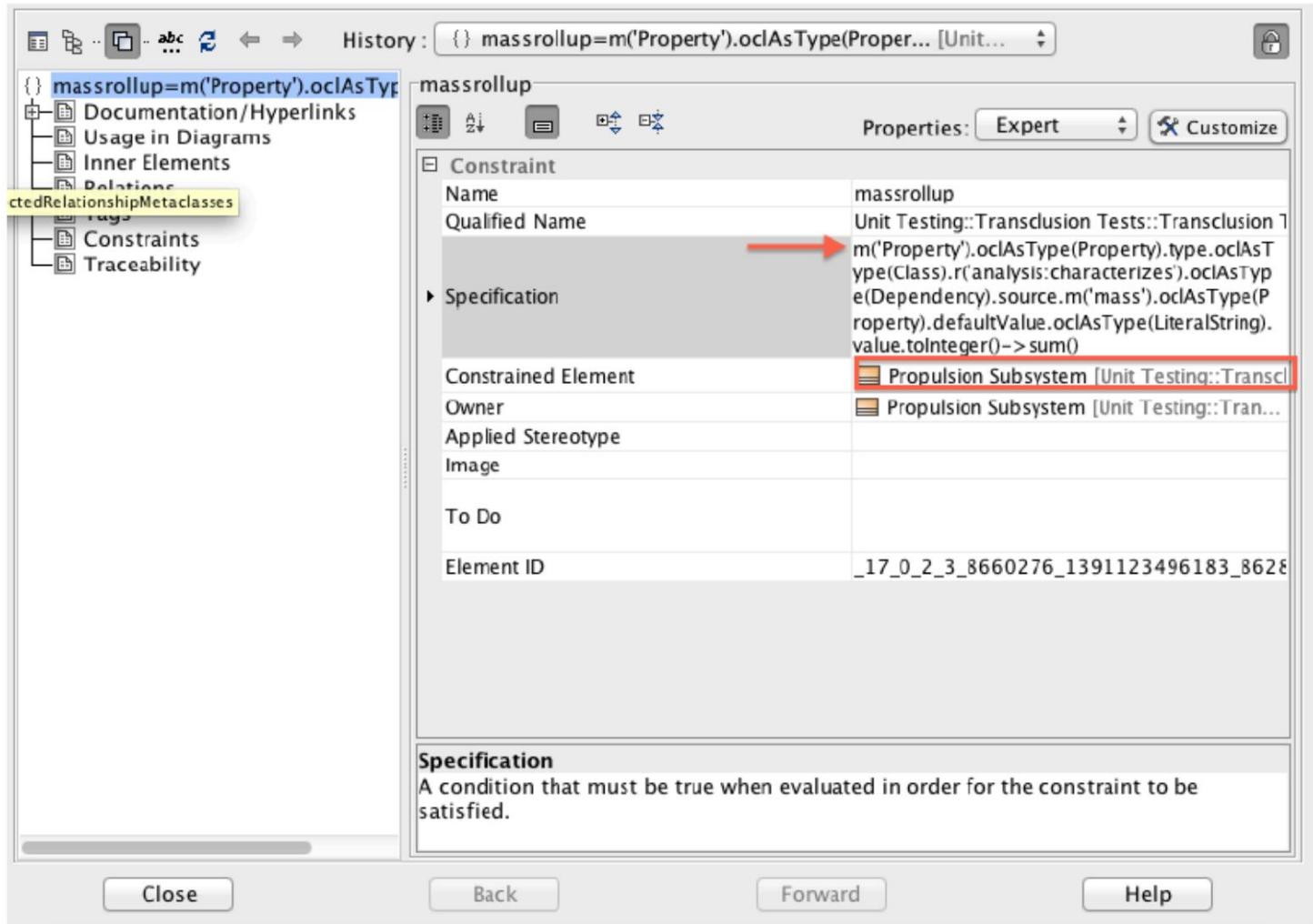
表 19. メンバー

要素の名前	属性	属性のデフォルト値
犬	ノイズ	吠える

1.4.2.6 「eval()」を評価する

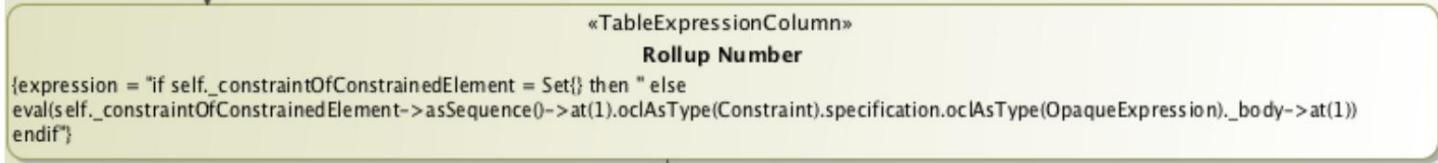
`eval()`または`e()`は、要素から取得した OCL 式を評価します。この例は非常に複雑に見えるかもしれません、重要な点はブラックボックス式の使用を示すことです。下の画像は、制約が設定されている（赤い矢印）要素（推進サブシステム）を示しています。

これは、制約仕様を持つ制約された要素です。



式の例は長いですが、理解を助けるために時間をかけてまとめてみてください。役立つヒントは、`oclAsType()` オペレーションから始めることです。これらのオペレーションにより、前のオペレーションで何が返されたのかがわかります。

これは、制約された要素を取得して制約を評価する eval() です。



これは完全な視点メソッド図です。

図 122. 評価ビューの図

図 123. ブラック ボックス デモの評価

図 124. トランスクルージョン テスト モデル

1.4.2.6.1 ビューの例

表 20. <>

コンポーネント名	CBE マス	ロールアップ番号
推進サブシステム		0
打ち上げマス		
定常状態の電力		
の上		
オフ		
待機する		
ヒドラジンタンク		
打ち上げマス		
定常状態の電力		
の上		
オフ		
待機する		
スラスター		
打ち上げマス		
定常状態の電力		
の上		
オフ		
待機する		
トランスクルージョンテストモデル		

1.4.2.7 型「t()」

t() は選択された要素のタイプを返しますが、type() は公開された要素のタイプのみを返します。要素「Dog」はクラス型であり、他の型が関連付けられています。要素 Dog の型だけを返したいので、<<TableExpressionColumn>> でtype()を使用します。

注: Class で返されるその他の情報は、MagicDraw のアーティファクトです。

図 125. タイプ ビューの図

図 126. タイプ ブラック ボックスのデモ

1.4.2.7.1 ビューの例

表 21. タイプ

要素の名前	要素の種類
犬	インターフェイス com.nomagic.uml2.ext.magicdraw.classes.mdkernel.Class

1.4.2.8 所有者 「owners()」

所有者と所有者の所有者を再帰的に返します。以下の例では、OCL 評価ツールを使用して要素「Dog」の所有者を収集します。評価ボックスの結果には、所有するパッケージとモデルが表示されます。注：式の最初の部分(self.)はターゲット要素を指定します。これは公開されている 1 つの要素にすぎないため、これがなくても操作は機能します。

下の赤い矢印は、収集されたパッケージ階層を示しています。

図 127. 所有者 ビューの図

図 128. オーナーズ ブラック ボックスのデモ

1.4.2.8.1 ビューの例

表 22.<>

所有者
動物
現在のモデルデータ
ビュー ポイント作成 メソッド の例
モデル
ドクゲン
データ

1.4.2.9 log()

工事中

log() ブラック ボックス式は、OCL 式の任意の時点で収集された要素を取得し、Magic Draw の通知 ウィンドウに出力します。この例では、OCL エバリュエーター（後述）に示されている OCL クエリが「Animals」パッケージに対して実行されました。この式は、パッケージの所有要素を収集します。OCL 評価 ウィンドウ（上）と通知 ウィンドウ（下）を比較すると、表示される要素が同じであることに注意してください。赤と青の矢印は同じ要素に対応します。

図 129. ログ ビューの図

図 130. ログ ブラック ボックスのデモ

1.4.2.9.1 ビューの例

1.4.2.10 実行(ビュー/視点)

run() を使用する 1 つの方法は、最初にビューポイントを収集してから、選択した要素に対してそれを実行することです。以下の最初のビューポイント メソッドには、<<TableExpressionColumn>> アクティビティに OCL 式 self.get('testforrun').run(self) があります。式を分析すると、「testforrun」というタイトルのビューポイントが get() を使用して収集されたことがわかります。（注：get() は、特定の要素を選択するときに非常に便利な式です）。次に、操作 run() がビューポイントを取得し、前にフィルターで除外されなかったすべての要素に対して実行しました。この時点では、

メソッドでは、これらの要素は「self」として分類されます。以下に示す 2 番目の視点メソッド図は「testforrun」です。各要素 (self) が run() 経由でビューポイントに渡されると、その名前が収集され、<<TableExpressionColumn>> アクションに返されます。

図 131. RunViewViewpoint ビューの図

図 132. ブラック ボックス デモの実行

1.4.2.10.1 ビューの例

表 23. <>

動物

1.4.3 ビューポイントで OCL 式を使用するにはどうすればよいですか？

OCL 式は、ビューポイント メソッドでさまざまな方法で使用して、さまざまなことを行うことができます。次のセクションでは、これらの機能のいくつかについて概要を説明します。

1.4.3.1 式による収集/フィルタ/並べ替えの使用

OCL 式をビューポイント メソッドにさらに簡単に導入するために、いくつかのカスタム アクションが提供されています。これらのアクションは、前のセクションで説明した他の収集およびフィルター アクションとともにビューポイント メソッド図のツール バーにあります。それについて簡単に説明します。各アクションの仕様ウィンドウには、目的の OCL 式を入力するための指定されたタグがあります。

視点要素

<<CollectByExpression>> OCL 式を使用して要素を収集します。これは、他の収集ステレオタイプと同様に機能します。

<<FilterByExpression>> OCL 式を使用してコレクションをフィルタリングします。これは他のフィルターのステレオタイプと同様に機能します。

<<SortByExpression>> OCL 式を使用してコレクションを並べ替えます。これは他の種類のステレオタイプと同様に機能します。

<<制約>> このステレオタイプをアクティビティ図のコメントまたはアクションに追加して、アクションの結果に対する OCL 式を評価できます。式は true または false を返す必要があります。 false の場合、制約に違反しており、違反が検証結果パネルに追加されます。制約コメントを複数のアクションに固定して、すべてのアクションの結果に個別に適用できます。

<<TableExpressionColumn>> OCL 式をターゲット要素に適用します。これを使用すると、他の方法では困難または不可能な要素や関係に対する操作を連鎖させることができます。

<<ViewpointConstraint>> アクションに渡される任意の要素のビューポイント メソッド ダイアグラム内の任意のポイントで制約を評価できるようにします。

zz<<CustomTable>> CustomTable を使用すると、1 つのビューポイント要素ですべての列を OCL 式として指定できます。ターゲット要素にはそれぞれテーブル内に行があります。タイトル、見出し、キャプションは他の表と同様に指定します。

1.4.3.2 高度なトピック

現在建設中を見る

1.4.3.2.1 反復フラグの使用

1.4.4 OCL エバリュエーターとは何ですか？また、それを使用する理由は何ですか？

望む結果を得るために OCL クエリを結合する必要があることが多いため、このツールは非常に役立ちます。

上記のセクションで説明したすべての OCL 式は、Magic Draw の OCL Evaluator で直接再現できます。のために

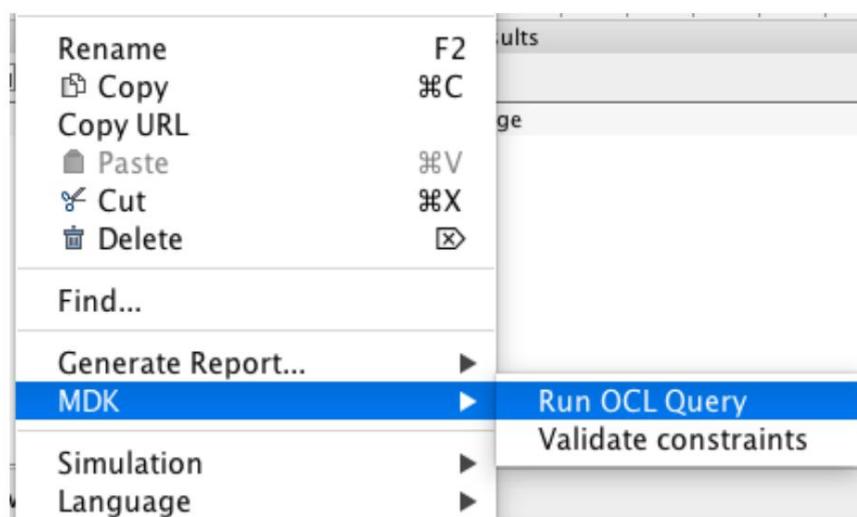
たとえば、以下の画像は、セクション「8.2.10 value()」で使用される式が OCL Evaluator ツールでどのように動作するかを示しています。パッケージの要素を収集するために.ownedElementが追加されたことに注意してください。8.2.10 の視点法では、要素がすでに収集およびフィルタリングされているため、これは必要ありませんでした。

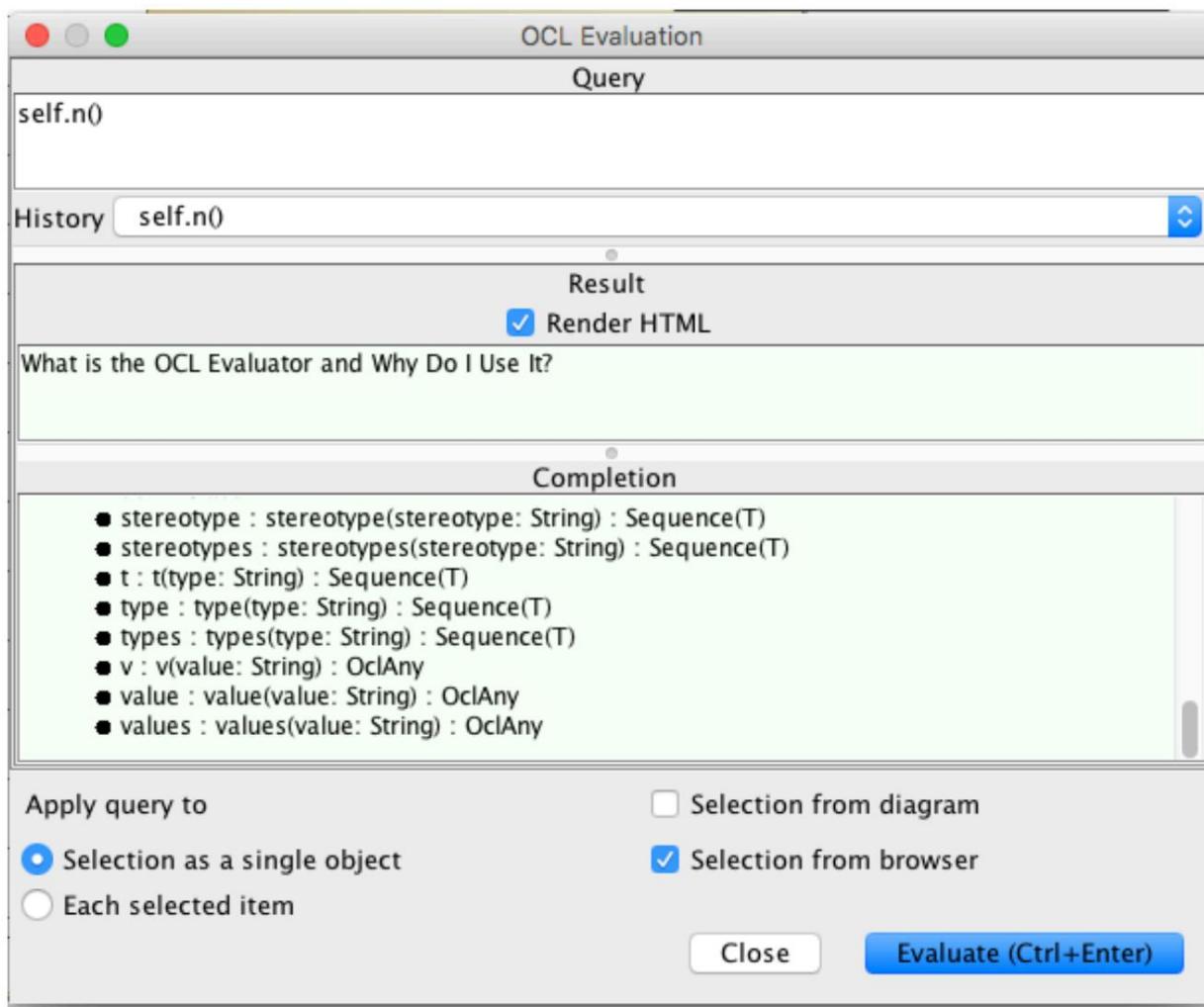
ご覧のとおり、この機能を使用すると、ビューをビュー エディターにエクスポートすることなく、式がどのような結果を返すかを確認できます。これにより、時間を大幅に節約できます。

自分で試してみるには、次の手順に従ってください。

1. 図または包含ツリーでいくつかのモデル要素を選択し、2. MDK メニューを見つけて、3. [Run OCL Query] を選択します。
4. 入力フィールドを表示するには、ポップアップ ウィンドウのサイズを変更する必要がある場合があります。
5. OCL に式を入力し、「評価」をクリックして、結果（またはエラー）を確認します。

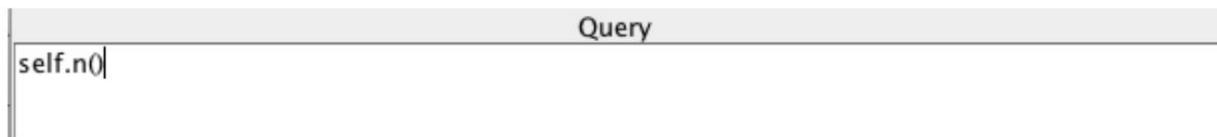
注: 以下の例の「self.n()」は、OCL 式のターゲットを設定しています。





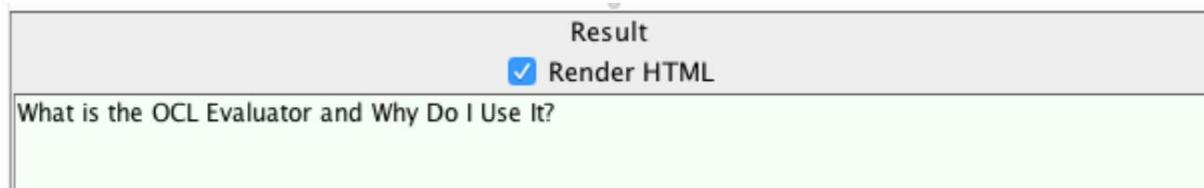
OCL エバリュエーターには 4 つの部分があります。

1. 式の入力



この領域では、OCL クエリを入力し、フィールドの右側にある矢印をクリックして、以前に評価したクエリを呼び出すことができます。

2. 結果



ここにクエリの結果が表示されます。

3. クエリ補完の提案

Completion

- stereotype : stereotype(stereotype: String) : Sequence(T)
- stereotypes : stereotypes(stereotype: String) : Sequence(T)
- t : t(type: String) : Sequence(T)
- type : type(type: String) : Sequence(T)
- types : types(type: String) : Sequence(T)
- v : v(value: String) : OclAny
- value : value(value: String) : OclAny
- values : values(value: String) : OclAny

このフィールドは、クエリの現在の結果に対する潜在的なオプションを提案します。

4. 選択場所

Apply query to

Selection as a single object Selection from diagram

Each selected item Selection from browser

これにより、「self.n()」またはクエリのターゲットをどの場所から取得するかを選択できます。

1.4.5 OCL 視点制約を使用するにはどうすればよいですか？

この例では、OCL Viewpoint Constraints パッケージには、ステレオタイプ化された 2 つの依存関係を持つ <<requirement>> が含まれています。これらの関係の 1 つは、要件で指定できる内容の設定されたパターンに違反しています。以下に示すビューポイント メソッドには、指定された OCL 式で渡された要素を検証する <<ViewpointConstraint>> アクションが含まれています。これは使用される表現です：

```
r('ミッション:指定').oclAsType(Dependency).target.oclisKindOf(Relationship).validationReport
```

式の内訳：

- ミッションの選択から始まります：r()を使用して関係を指定します
- 次に、.oclAsType(Dependency)を使用して依存関係として返された値をキャストします。
- 次に、.targetを使用して依存関係のターゲットを選択します。
- 次に、.oclisKindOf(Relationship)を使用して、返された値をリレーションシップとしてキャストします。
- 最後に、.validationReportを使用して検証レポート（最初の 2 つの表を以下に示します）が返されることを通知します。

検証レポートは自動的に生成され、エラー データが入力されます。1 つは概要でもう 1 つは詳細です。

OCL Viewpoint Constraints パッケージの要件にはターゲットとして別のタイプのリレーションシップとの依存関係があるため、エラーがスローされます。この制約に違反する要素はテーブル構造に送信され、以下に示す最後のテーブルに移入されます。この場合、「要件名!!」最終的な結果になると予想されます。

1.4.6 OCL ルールを作成するにはどうすればよいですか？

1.4.6.1 特定のモデル要素に対するルールを作成するにはどうすればよいですか？ (制約評価者)

以下の図には、「コメント付きブロック」という名前の <<ブロック>> が含まれています。この要素には 2 つの制約が付加されています…

1. oclIsKindOf(コメント)

2. oclIsKindOf(クラス)

図 133. 特定のモデル要素にルールを作成するにはどうすればよいですか?

これらの制約をモデル内で直接検証するには、要素を選択して右クリックし、「MDK」の「制約の検証」に移動します。下の画像を参照してください。明らかに、これらの制約の 1 つは true、もう 1 つは false であるため、警告は 1 つだけスローされる必要があります。これを下の図に示します。赤い矢印は、Magic Draw 通知ウィンドウで失敗した制約を示しています。

明らかに、これらの制約の 1 つは true、もう 1 つは false であるため、警告は 1 つだけスローされる必要があります。これを下の図に示します。赤い矢印は、Magic Draw 通知ウィンドウで失敗した制約を示しています。

1.4.6.2 ビューポイント内でルールを作成するにはどうすればよいですか?

1.4.6.3 モデル内の OCL ルールを検証するにはどうすればよいですか?

モデル内の OCL クエリを検証するには、右クリックと MD 検証ウィンドウを使用する 2 つの方法があります。

右クリック メニューを使用して OCL ルールを検証するには、検証する要素を含むパッケージを選択し、[MDK] -> [MDK] を選択します。
制約の検証

1.4.7 式ライブラリを作成するにはどうすればよいですか?

1.4.8 クエリで正規表現を使用するにはどうすればよいですか?

http://www.vogella.com/tutorials/Java_RegularExpressions/article.html <http://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html>

1.4.9 OCL クエリでトランスクルージョンを作成するにはどうすればよいですか?

表 26.<>

名前	ドキュメンテーション
OCL クエリでトランスクルージョンを作成するにはどうすればよいですか?	

表 27.<>

名前 (構築されたトランスクルージョンから)	ドキュメント (構築されたトランスクルージョンから)