



IBM Software Group

# データモデルとUMLモデルの変換について

*RDAとRSMとRequisiteProの統合*

**Rational** software

2007年5月24日  
IBM Rational 事業部 藤田一雄



**ON DEMAND BUSINESS**

# 前提

- ここではモデルの変換について以下のツールを前提に説明する
  - ▶ Rational Software Modeler v7.0.0.2 (以降RSMと記す)
  - ▶ Rational Data Architect v7.0.0.1 (以降RDAと記す)
  - ▶ Rational RequisitePro v7.0(以降ReqProと記す)
  - ▶ ここでいうデータモデルとはRDAで扱うデータモデルのことを言う
  - ▶ ここでいうUMLモデルとはRSMで扱うオブジェクトモデルのことを言う



# RSMとRDAとReqProの統合

1. RSMとRDAとReqProはインストール時に統合されるので特に統合に関して考慮する必要はない
2. RSMとRDAはインストールすると以下のプログラムメニューにそれぞれ起動メニューが作られる
  1. IBM Software Development Platform
  2. ReqProはIBM Rationalフォルダーに起動メニューが作られる
3. RSMとRDAはそれぞれ起動できるが同じワークベンチを参照している場合は同時に起動することは出来ない
4. RSMとRDAは統合されることで同じパースペクティブを持ちそれぞれの機能をお互いに利用できる

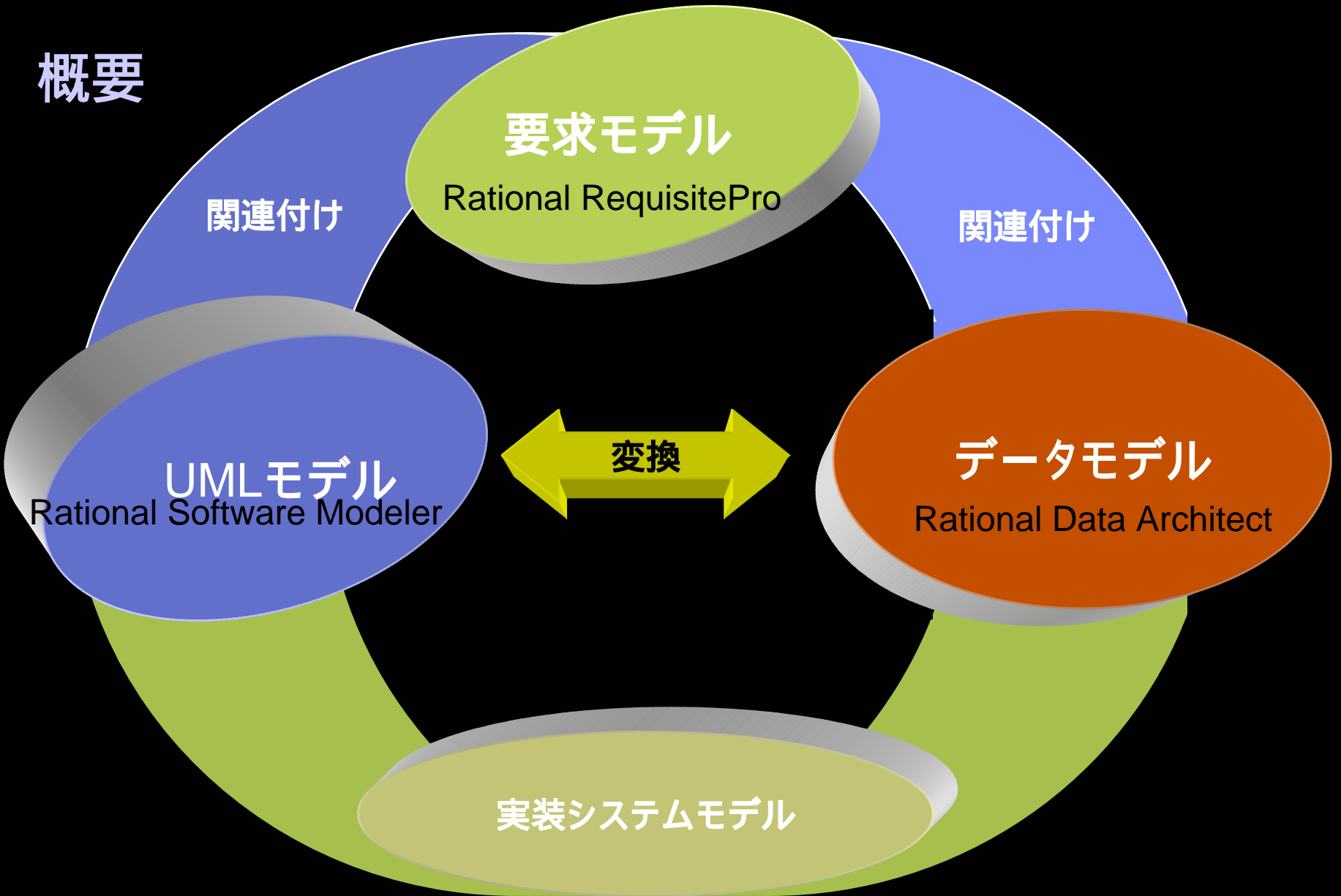


# アジェンダ

- 概要
- RDAにおけるデータモデリング
- データモデルからUMLモデルへの変換
- UMLモデルからデータモデルへの変換
- 変換の問題点
- 要求を捉える
- 要求をReqProで管理
- ReqProとRDAの統合
- 要求モデルとデータモデルの関連付け
- ReqProとRSAの統合
- 要求モデルとUMLモデルの関連付け
- 要求モデルとデータモデルとUMLモデルの関連



# 概要



## データモデリングとUMLモデリング

# 「UMLでデータモデリングはできない！」

- × UMLには主キーがない！
- × UMLには論理設計/物理設計という概念がない！！
- × UMLは絵がいっぱいあって複雑だ！？
- × UMLでは主キーの自動移行がサポートされていない??
- × UMLで業務モデルは書けない??



# データモデリングとUMLモデリング

## UMLの強み

- あらゆる開発者のニーズに対する標準表記法
- ライフサイクルを通じ、業務プロセスを追跡

## その他必要なもの

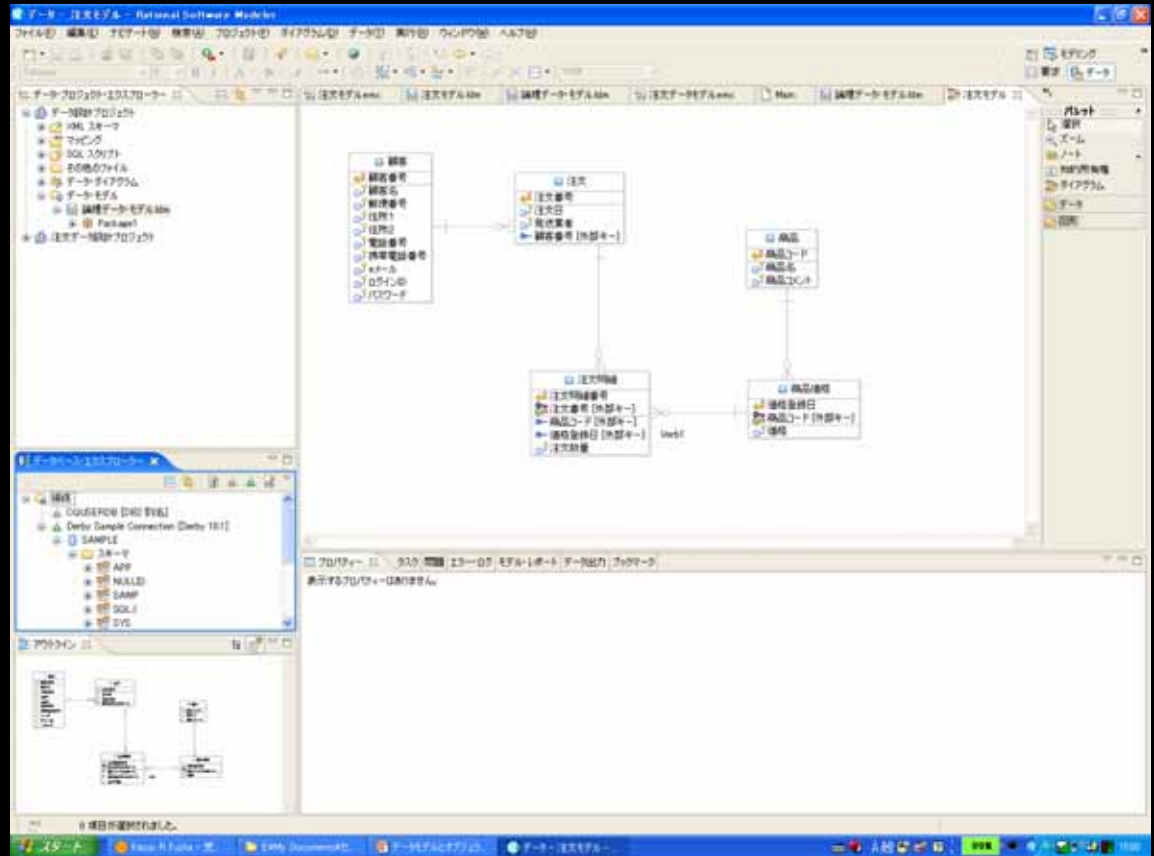
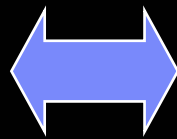
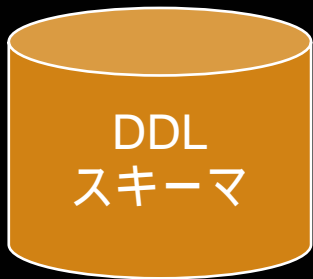
- データベース要素: テーブル、カラムなど...
- 組み込みのデータベース要素...



# RDAにおけるデータモデリング



論理データモデル設計  
物理データモデル設計

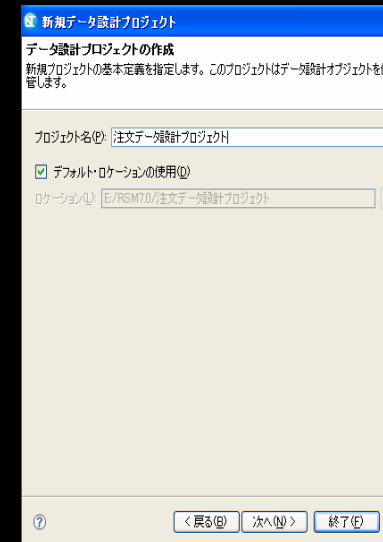
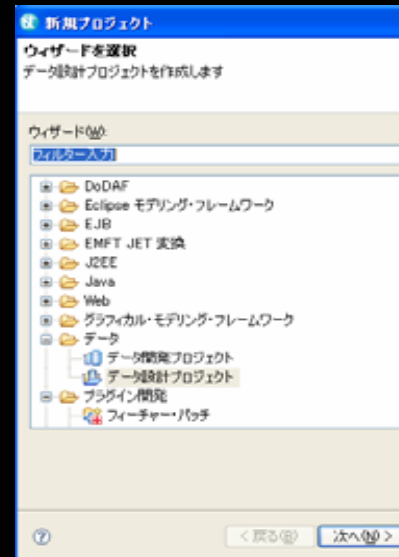
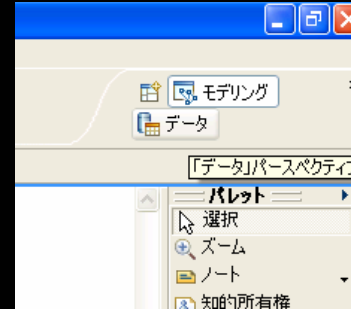


物理データモデルからDDL,スキーマの作成  
DDL,スキーマから物理データモデルの作成



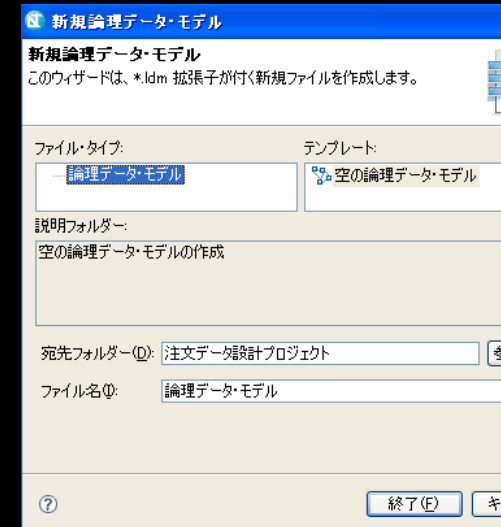
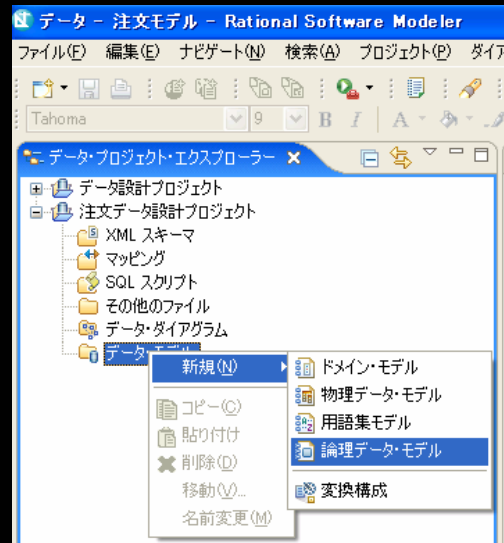
# データモデルの作成 ( )

1. データパースペクティブを選択する
2. ファイルメニューの新規から「プロジェクト」を選択
3. 「新規プロジェクト」ウィザードからデータフォルダを展開
4. データフォルダの「データ設計プロジェクト」を選択
5. 「次へ」ボタンをクリック
6. 「新規データ設計プロジェクト」ウィザードでプロジェクト名を入力
7. 「終了」ボタンをクリック



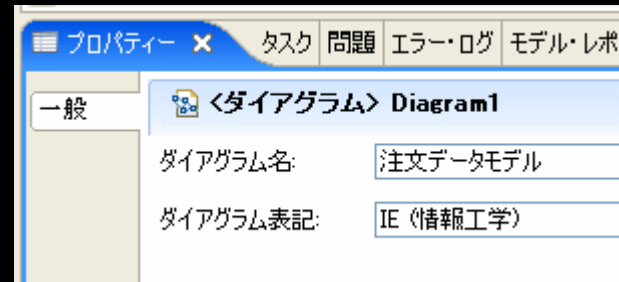
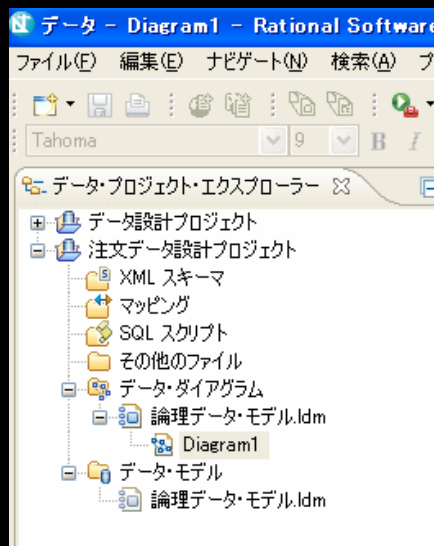
# データモデルの作成 ( )

1. 「データ・プロジェクト・エクスプローラー」に指定したプロジェクト名のプロジェクトが作成される
2. 作成されたプロジェクトを展開しデータモデルフォルダーを右クリックする
3. 新規を選択し「論理データ・モデル」をクリックする
4. 「新規論理データ・モデル」ウィザードでファイル名を入力し「終了」ボタンをクリックする



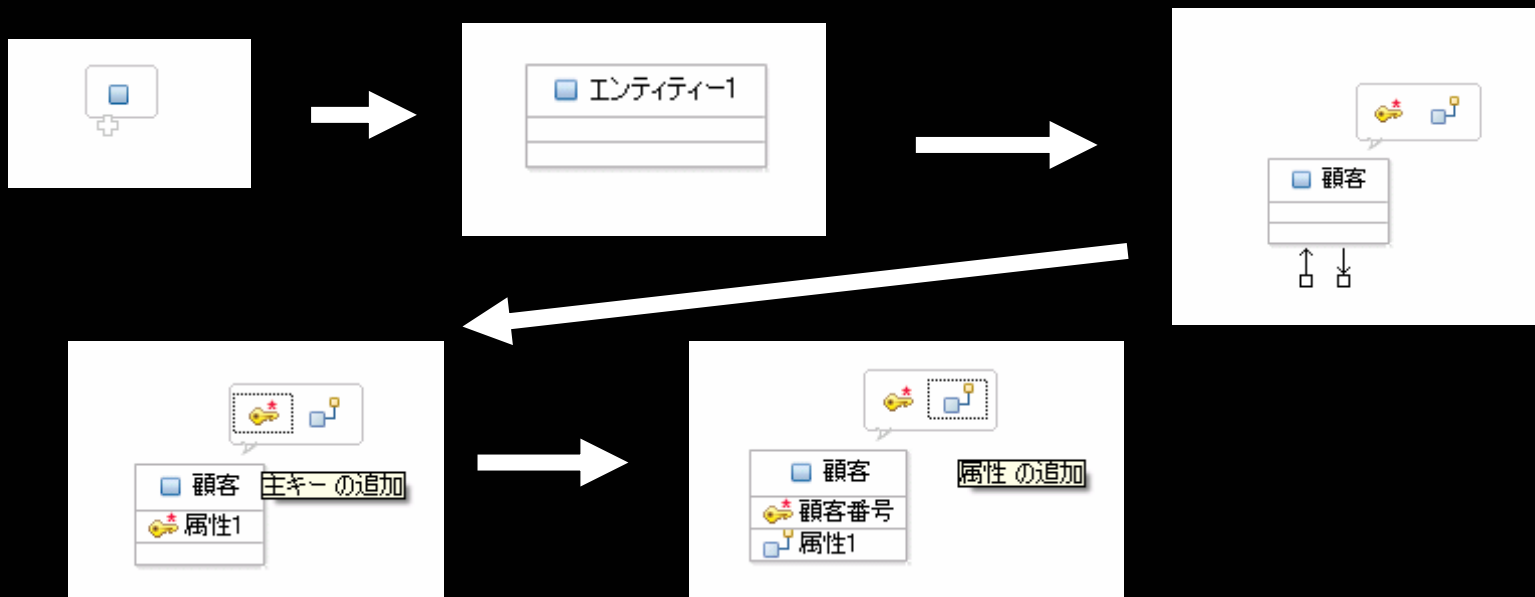
# データモデルの作成 ( )

1. 「データ・ダイグラム」と「データ・モデル」フォルダーの下に指定したファイル名のモデルファイルが出来、「データ・ダイグラム」フォルダー下のモデルファイルのした下に「Diagram1」が出来、データモデルを作図できる
1. このDiagram1はこのモデル要素のプロパティで名前を変更できる

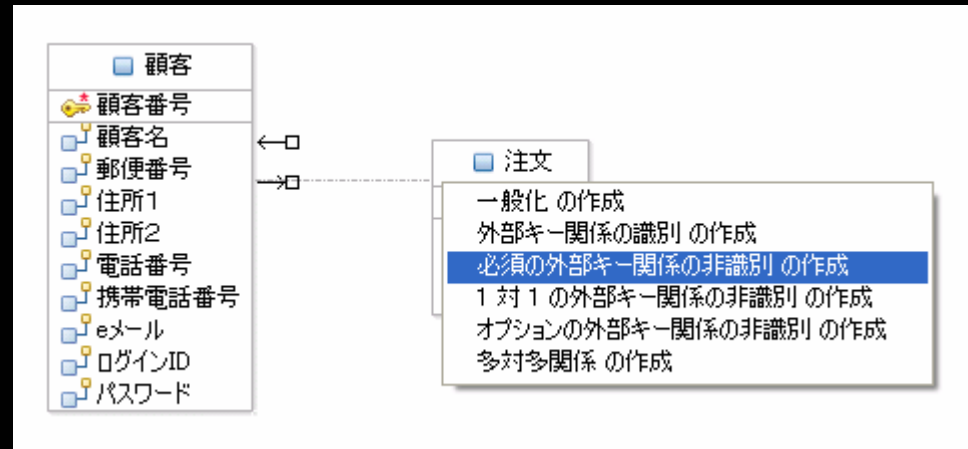
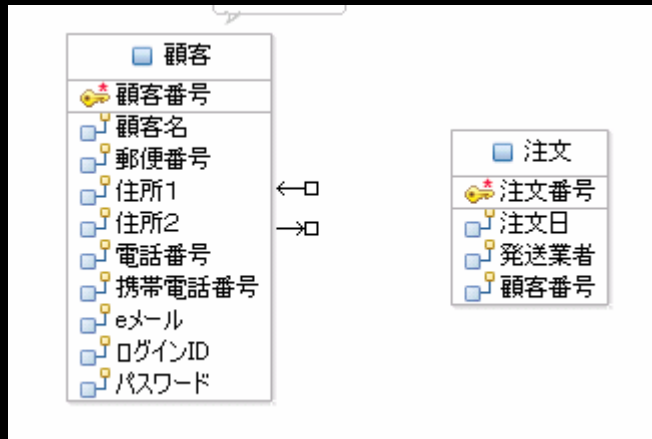


# データモデルの作成 ( )

1. 作成されたダイアグラムをダブルクリックしダイアグラム上にマウスを置くとモデルアシスタント機能によりモデル要素が表示される
2. 表示されたモデル要素をクリックしエンティティを作図する
3. 作図したエンティティには名前を入力し主キーとその他の属性を追加し、エンティティ間に関係の線を引きデータモデル図を作成する



# データモデルの作成 ( )



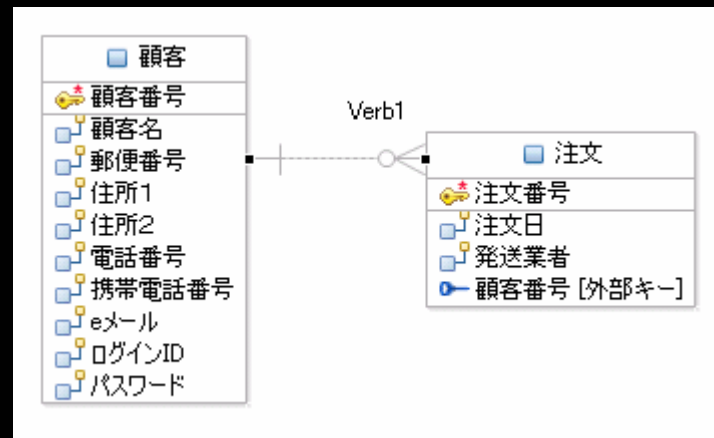
**キーのマイグレーション**

キー・オプションのマイグレーション  
子エンティティ/テーブルが類似した名前の属性/列を含んでいます

既存の子属性/列を使用  
 マイグレーションした FK 属性/列で置換  
 新規子属性/列の作成

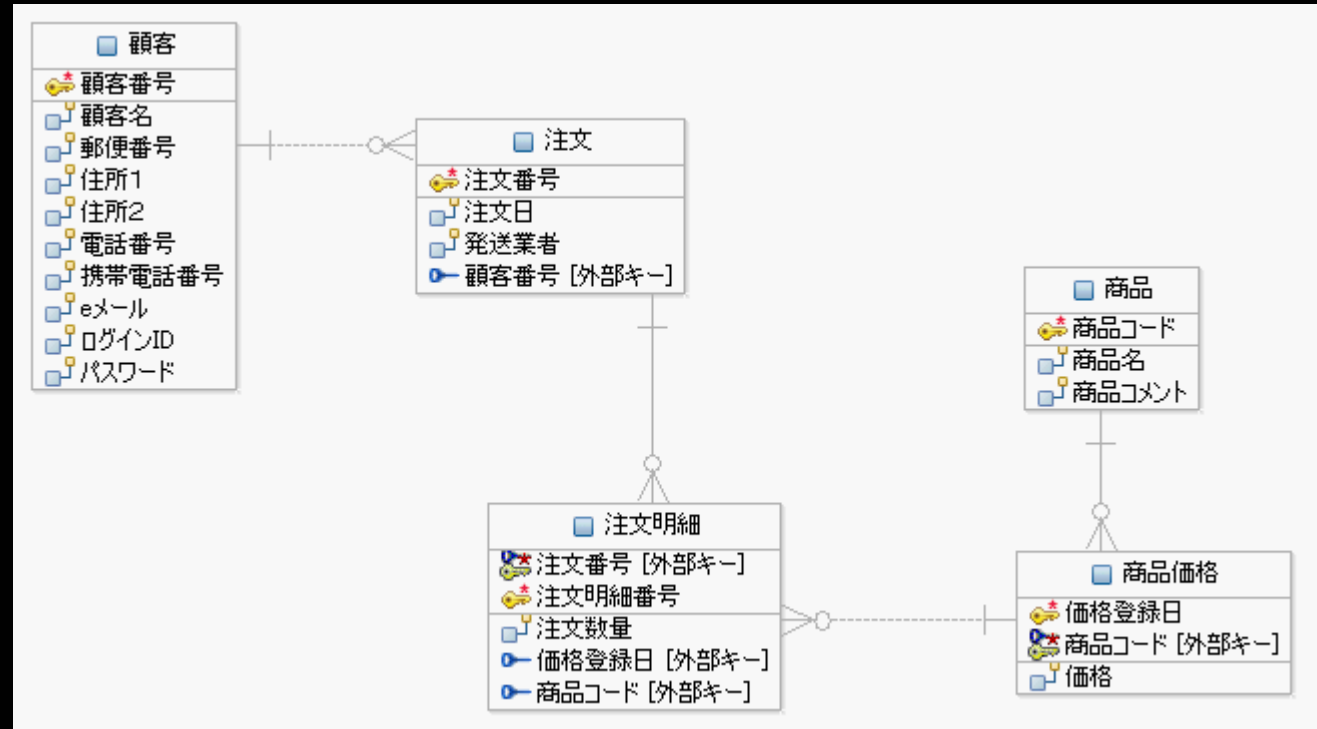
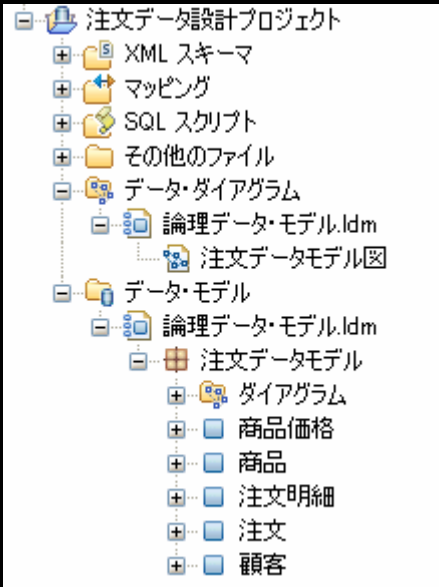
今後この質問を表示しない

OK



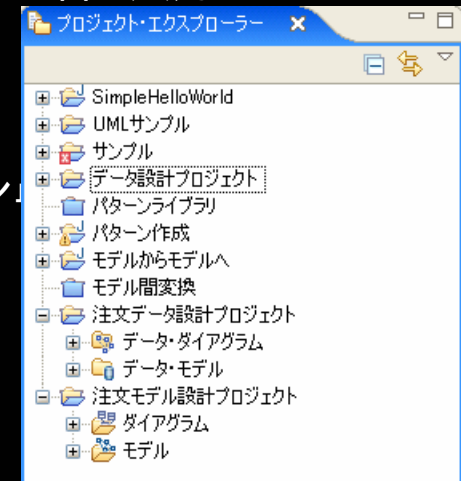
# データモデルの作成 ( )

1. 各エンティティ間に関係の線を引きデータモデル図を作成する
2. データ・モデルフォルダーにパッケージが作成されるので適切な名前を付ける
3. そのパッケージの中にモデル要素が定義される



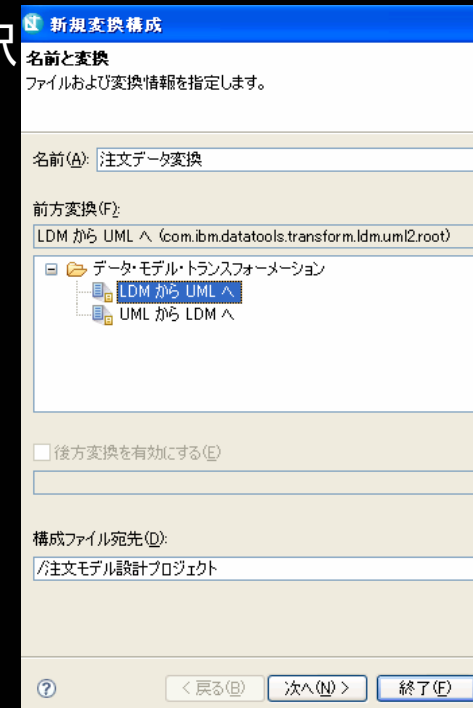
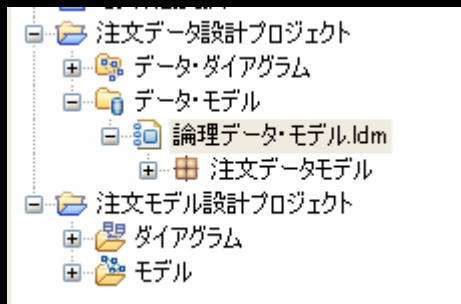
# データモデルからUMLモデルへの変換( )

1. モデルパースペクティブに切り替える
  1. プロジェクト・エクスプローラの中にデータモデルを作成したプロジェクトが見える
2. オブジェクトモデルを作成するプロジェクトを定義する
  1. ファイルメニューの新規から「プロジェクト」を選択
  2. 「新規プロジェクト」ウィザードの「モデリング」フォルダーを展開する
  3. 「モデリング」フォルダーの中の「UMLプロジェクト」を選択し「次へ」をクリック
  4. 「UMLモデリング・プロジェクト」ウィザードでプロジェクト名を入力
  5. 「次へ」をクリック
  6. ファイル名を入力し「終了」をクリック
  7. 作成されたプロジェクトには「ダイアグラム」と「モデル」フォルダーが作られている
  8. フォルダが作られている



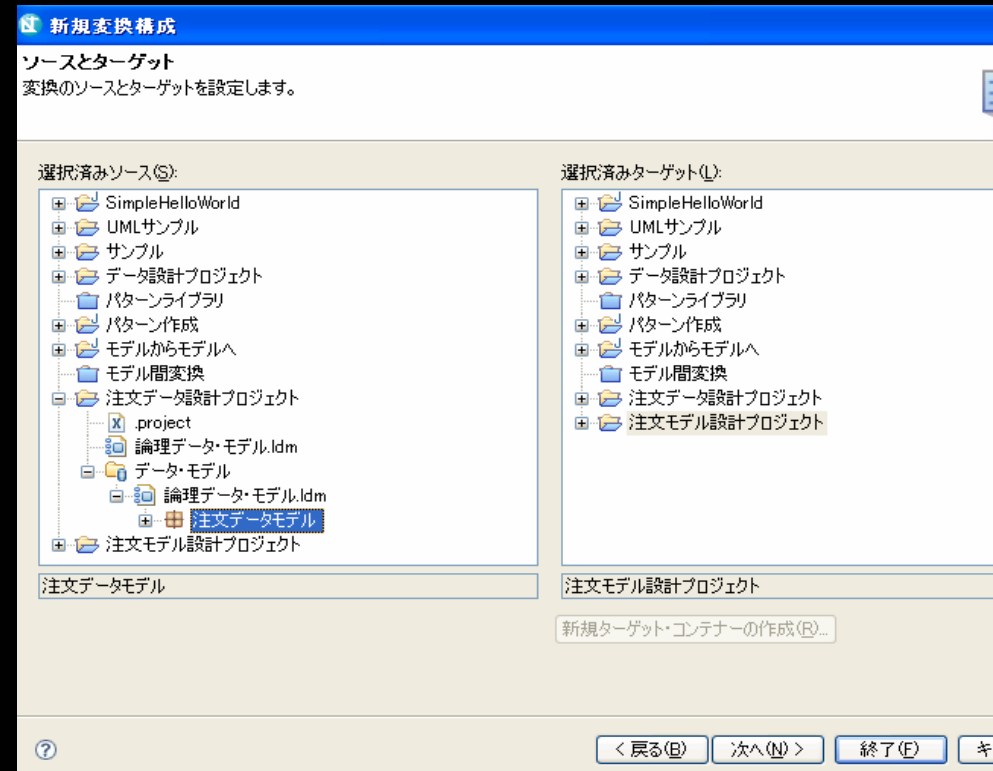
# データモデルからUMLモデルへの変換 ( )

1. プロジェクト・エクスプローラからデータモデルの作成プロジェクトを展開
2. 「データ・モデル」フォルダーを展開し「論理データ・モデル.ldm」をクリック
3. ファイルメニューの新規を選択し「変換構成」をクリック
4. 「新規変換構成」ウィザードで「LDMからUMLへ」を選択
5. 名前を入力し構成ファイル宛先にUMLのモデルプロジェクトを選択して「次へ」をクリック
6. 名前を入力し構成ファイル宛先にUMLのモデルプロジェクトを選択して「次へ」をクリック



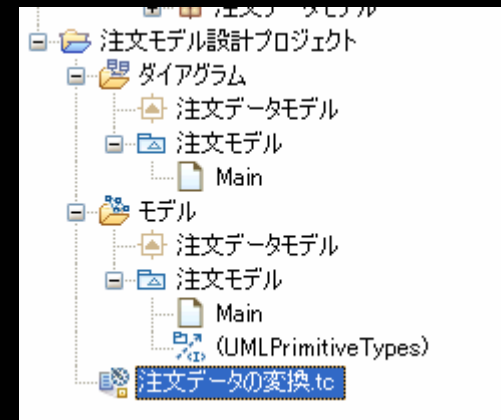
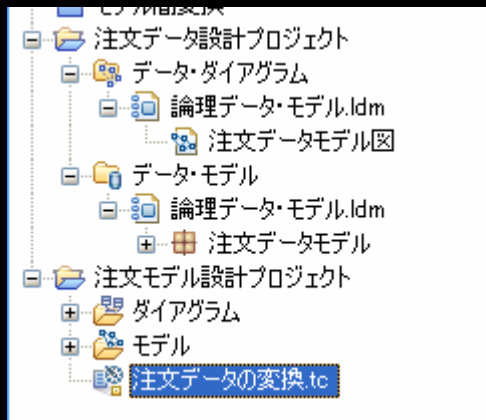
# データモデルからUMLモデルへの変換( )

1. 「選択済みソース」の中でデータモデルのモデル要素が定義されているパッケージを選択
2. 「選択済みターゲット」の中では変換後のUMLモデルを格納するプロジェクトを選択
3. 「次へ」を2回クリックし「終了」をクリック



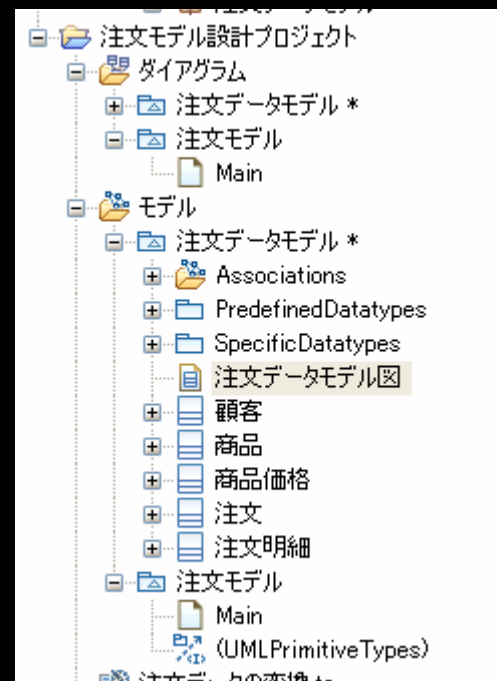
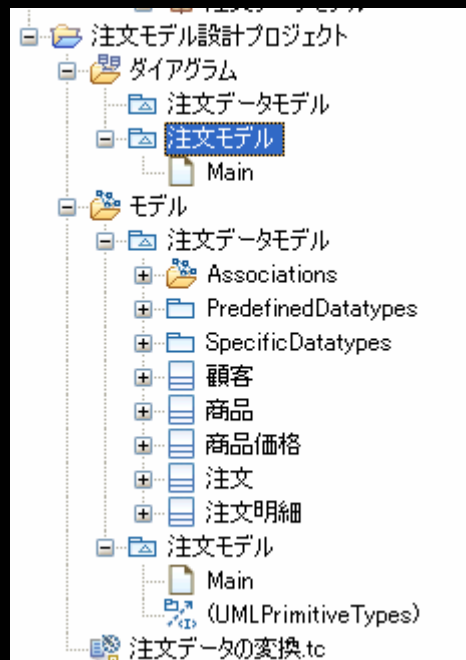
# データモデルからUMLモデルへの変換 ( )

1. モデル変換の機能がUMLモデルプロジェクト内に作成される
2. そのモデル変換機能をクリックし、メインメニューのモデリングをクリック
3. モデリングメニューの中の「変換」をクリックしその中の「LDMからUMLへ」をクリック
4. データモデルと同じ名前でUMLモデルプロジェクトに変換モデルが作成される



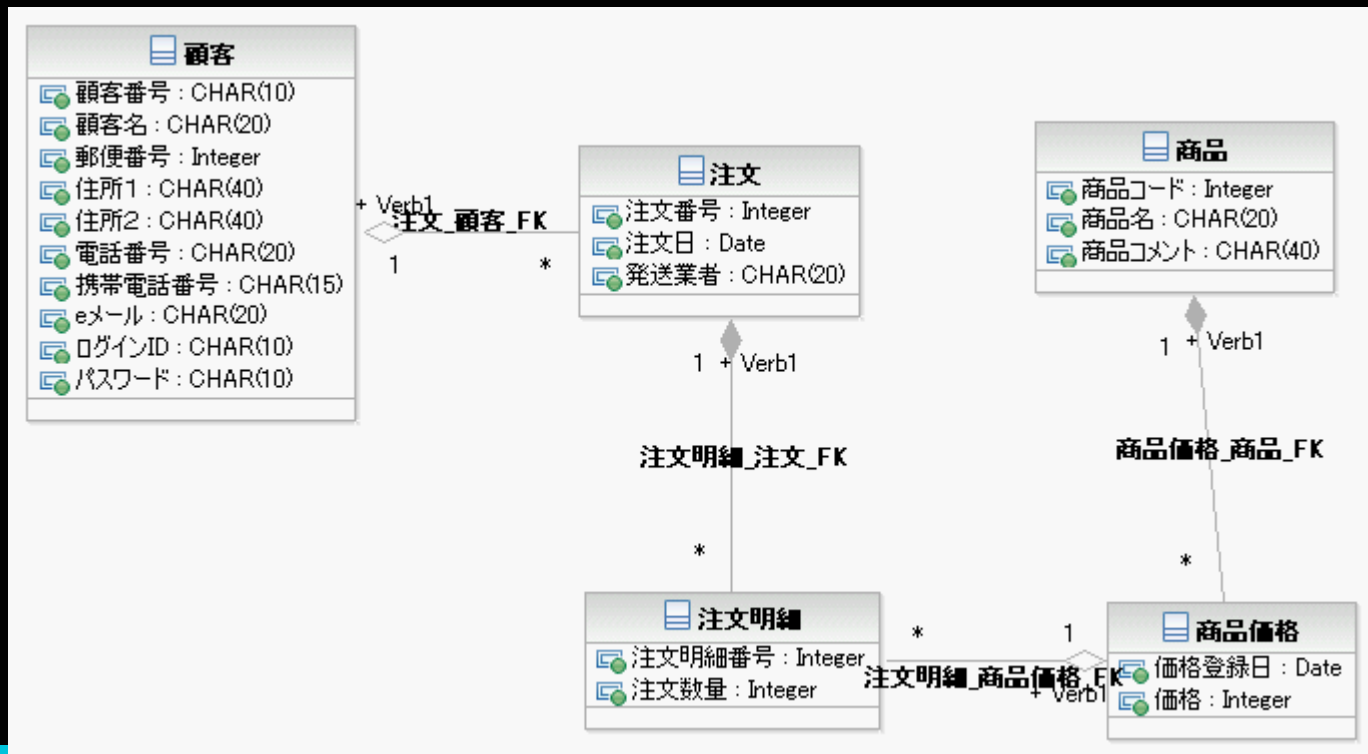
# データモデルからUMLモデルへの変換( )

1. 作成されたモデルをダブルクリックして変換後のモデルを表示する
2. クラス図を定義する
3. クラス図に作成されたモデルをドラッグ&ドロップしてクラス図を作成する



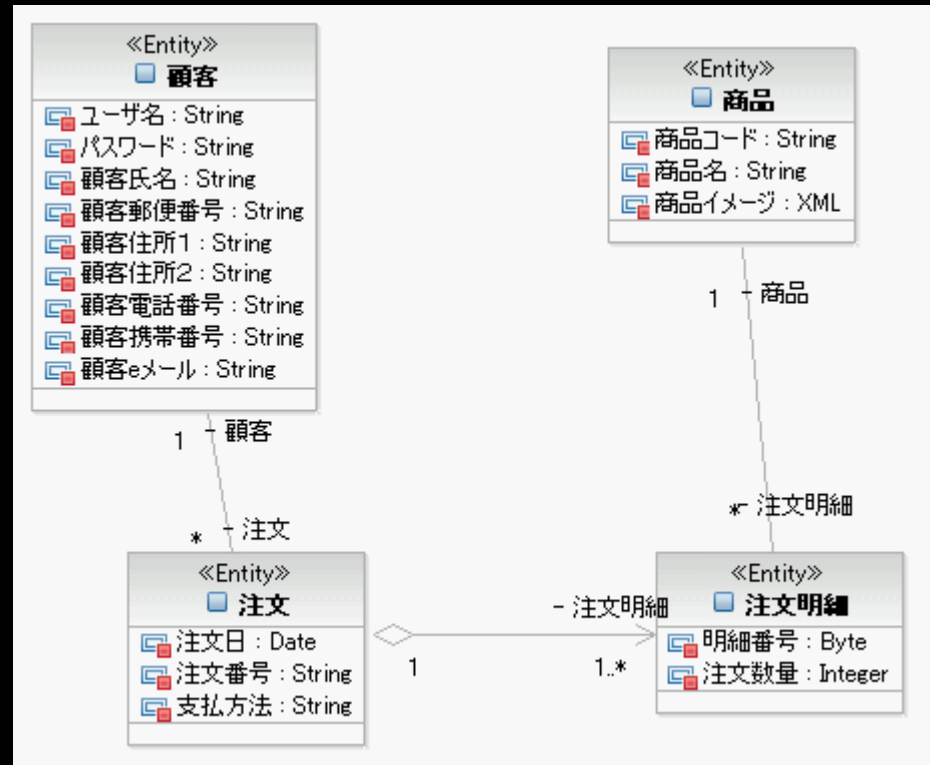
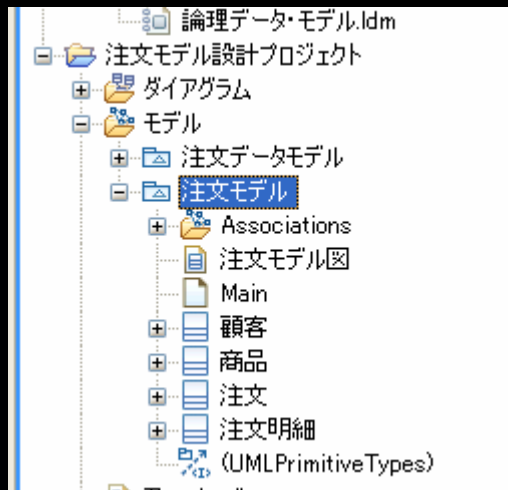
# データモデルからUMLモデルへの変換( )

1. データモデルと変換されたUMLモデルの間にはリンクは特にない
  1. データモデルの構成に関係なく変換されたモデルをアプリケーションのアーキテクチャに最適なUMLモデルに変換する
  2. データモデルを修正後再度( )の2からを繰り返すと修正後のモデルに置き換えることができる



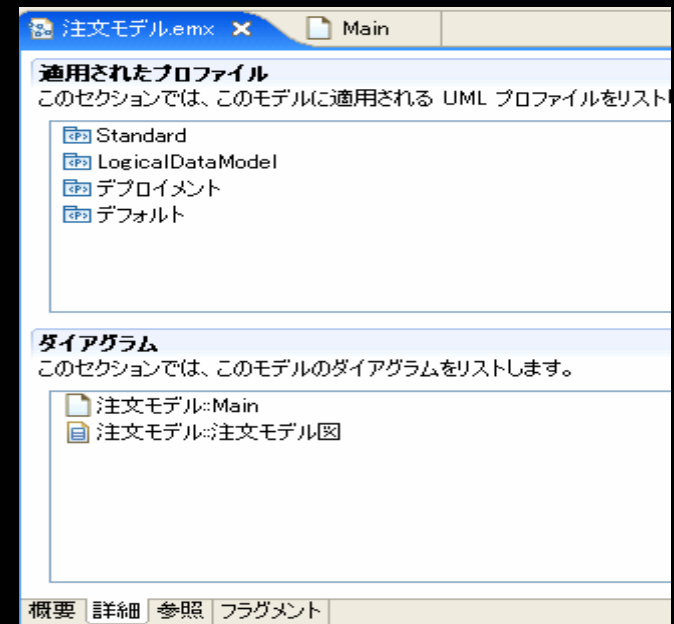
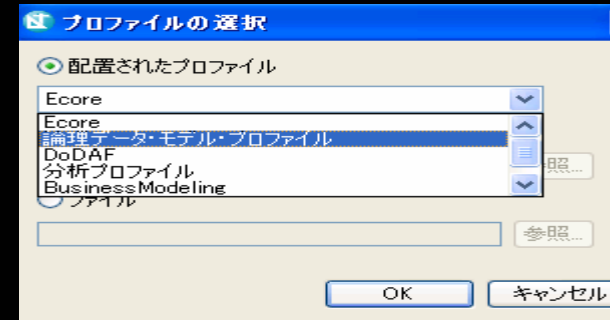
# UMLモデルからデータモデルへの変換( )

- 既に作成してあるUMLモデルをデータモデルに変換する
  - ▶ このときUMLモデルのクラスはステレオタイプが指定されていないといけない
  - ▶ 指定できるステレオタイプはRSM又はRADのヘルプを参照

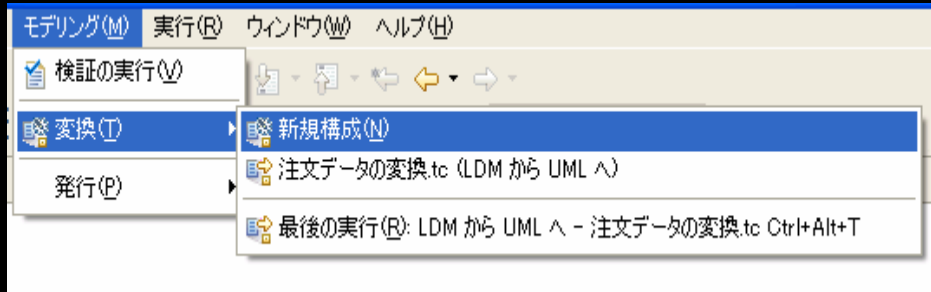


# UMLモデルからデータモデルへの変換 ( )

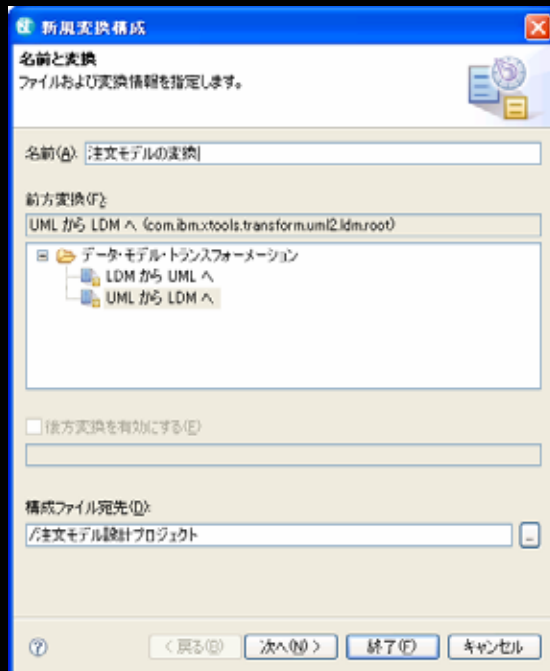
1. プロジェクト・エクスプローラのUMLモデルプロジェクトにあるモデルフォルダーに作成したモデルを変換
  1. モデルフォルダーに作成したモデルパッケージをクリック
  2. .emxファイルのプロパティの詳細タブをクリック
  3. 「適用されたプロファイル」で「追加」ボタンをクリック
  4. 「プロファイルの選択」ウィザードの「配置されたプロファイル」で「論理データ・モデル・プロファイル」を選択し「OK」をクリック
  5. 「適用されたプロファイル」枠に「LogicalDataModel」が追加される



# UMLモデルからデータモデルへの変換 ( )

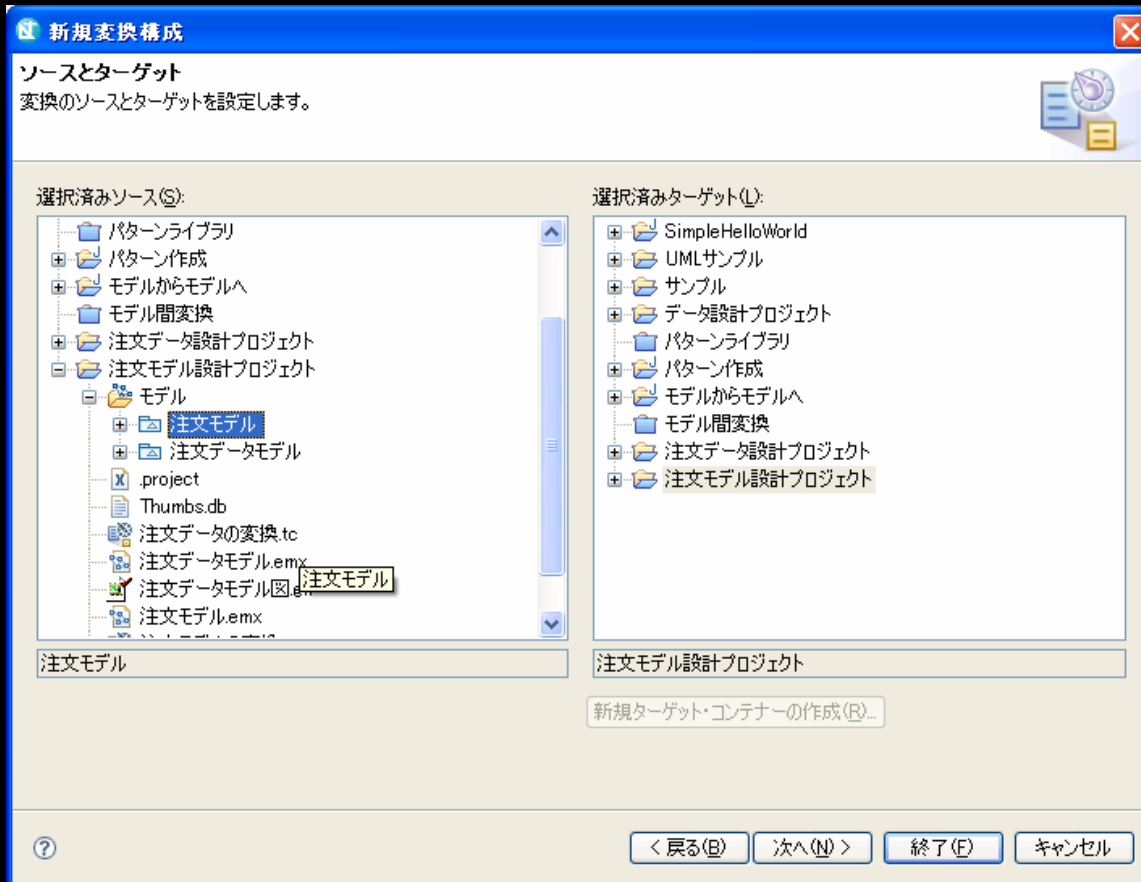


1. プロジェクト・エクスプローラのUMLモデルプロジェクトにあるモデルフォルダーに作成したモデルを変換



1. メインメニューの「モデリング」から「変換」「新規構成」をクリック
2. 「新規変換構成」ウィザードの「名前と変換」ページで「前方変換」の枠で「UMLからLDMへ」をクリック
3. 名前を入力し「次へ」をクリック

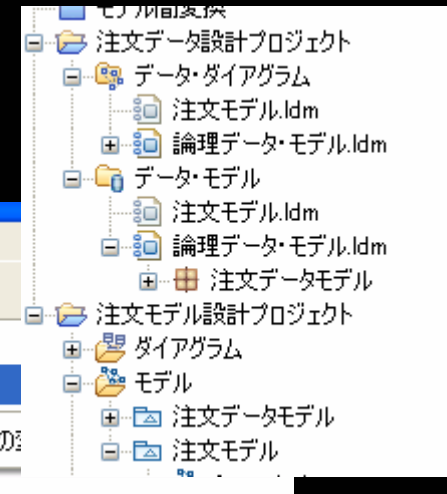
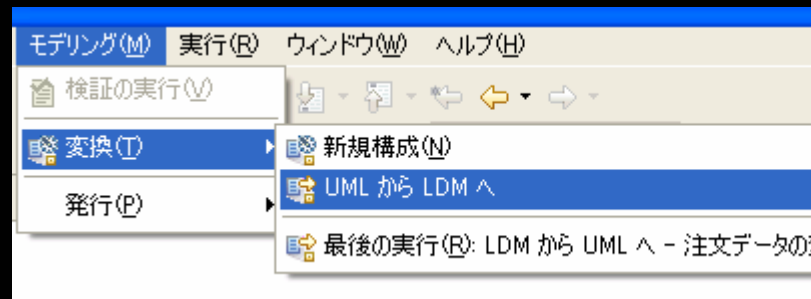
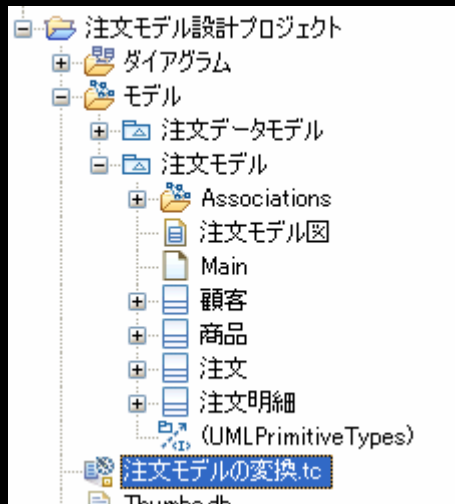
# UMLモデルからデータモデルへの変換( )



1. プロジェクト・エクスプローラのUMLモデルプロジェクトにあるモデルフォルダーに作成したモデルを変換
1. 「新規変換構成」ウィザードの「ソースとターゲット」ページの「選択済みソース」で変換対象のモデルを選択
2. 「選択済みターゲット」で変換後のデータモデルを格納するプロジェクトを選択
3. 「次へ」「終了」をクリック

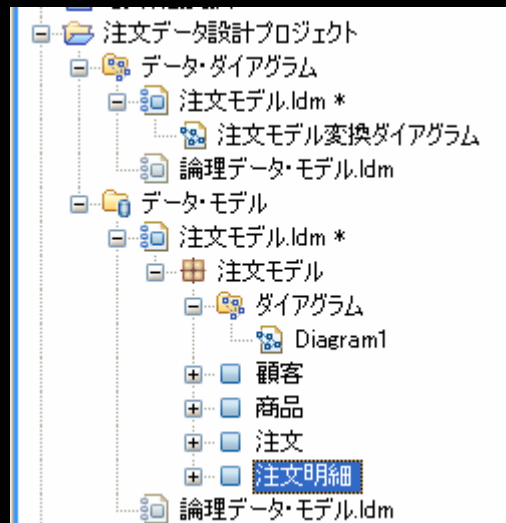
# UMLモデルからデータモデルへの変換 ( )

1. プロジェクト・エクスプローラのUMLモデルプロジェクトにあるモデルフォルダーに作成したモデルを変換
  1. UMLモデルプロジェクトに作成された変換をクリック
  2. メインメニューの「モデリング」から「UMLからLDMへ」をクリック
  3. ターゲットのプロジェクトに変換後の.ldmファイルが作成される



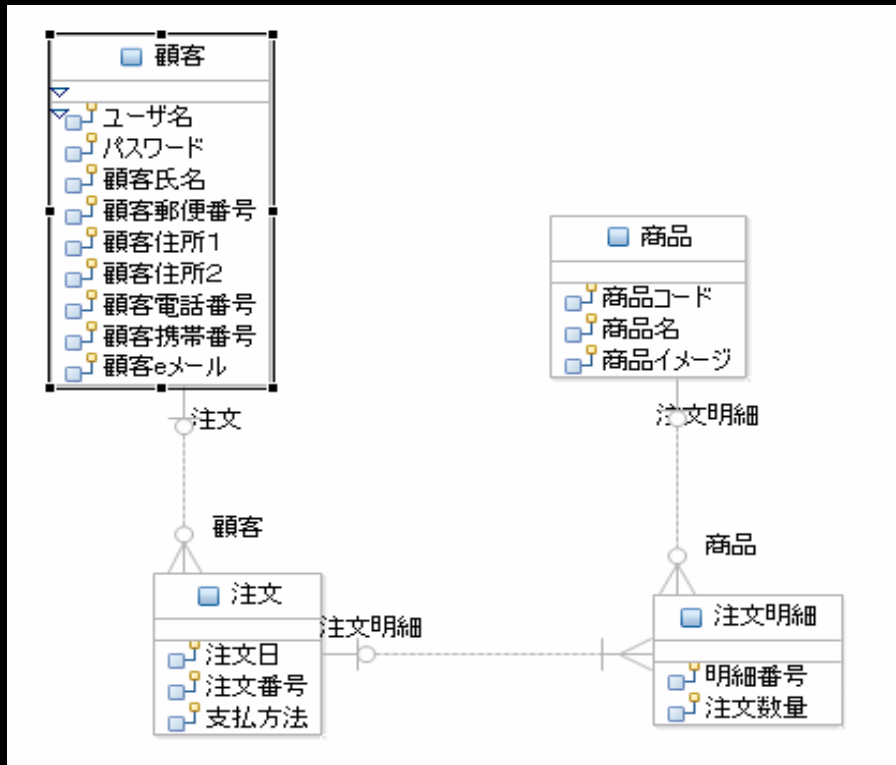
# UMLモデルからデータモデルへの変換( )

1. プロジェクト・エクスプローラのUMLモデルプロジェクトにあるモデルフォルダーに作成したモデルを変換
  1. 変換されたデータモデルをダブルクリックする
  2. 「ダイアグラム」を右クリックし「新規空白・ダイアグラム」をクリック
  3. ダイアグラムの名前はプロパティで変更する
  4. 開いたダイアグラムに変換後のモデル要素をドラッグ&ドロップ



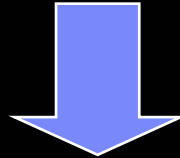
# UMLモデルからデータモデルへの変換( )

1. 変換後のデータモデルはUMLモデルとはリンクしていない
2. UMLモデルとは独立に正規化、非正規化を行うことができる



## 変換の問題点

- 変換については変換規則がありあらかじめヘルプを見ておくこと
- 変換直後は名前は同じ名前になっておりモデル間の対応がわかりやすい
- 変換後にUMLモデルはアプリケーションアーキテクチャに応じて変更され、データモデルは正規化、非正規化が行われ変更されていく
  - ▶ 変換後のモデルが変更されることによってモデル要素の名前や個数などが変わりモデル間の対応がとりづらくなるので、保守性が悪くなる



2つのモデルの関連を維持するためのプロセスを確立することが望ましい  
このプロセスの確立にRequisiteProの活用がある



# 要求を捉える

- 注文処理の一部を例にとって要求を以下のように捉えたとする

処理上のアクション	欲しい情報(入力)	まとめたい情報(出力)	制約
顧客の注文を受け付ける	・顧客情報 ・注文情報	・顧客情報 ・商品情報 ・注文情報	・応答時間 ・安定度 ・使い勝手
商品在庫の調査	・商品情報 ・在庫情報	・在庫情報	・応答時間 ・安定度 ・信頼性



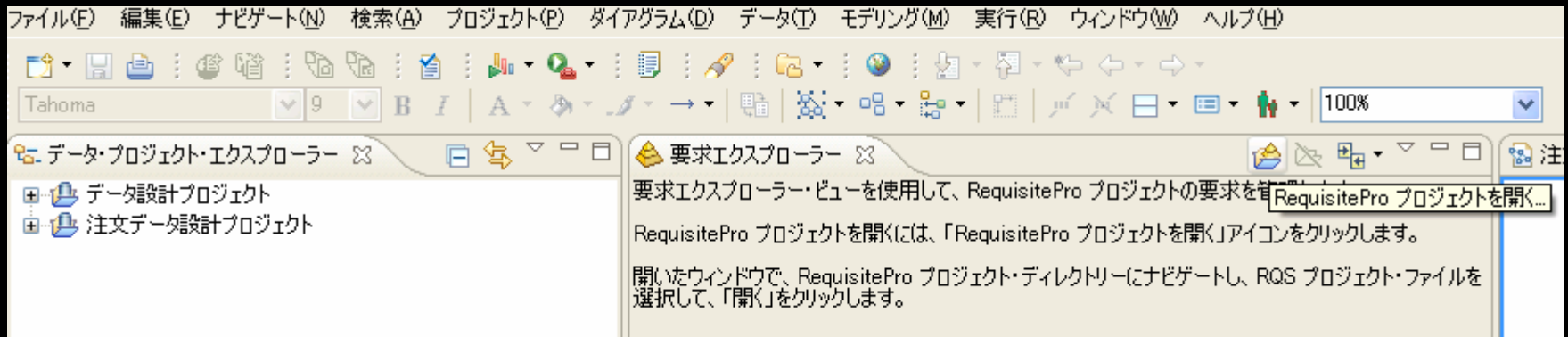
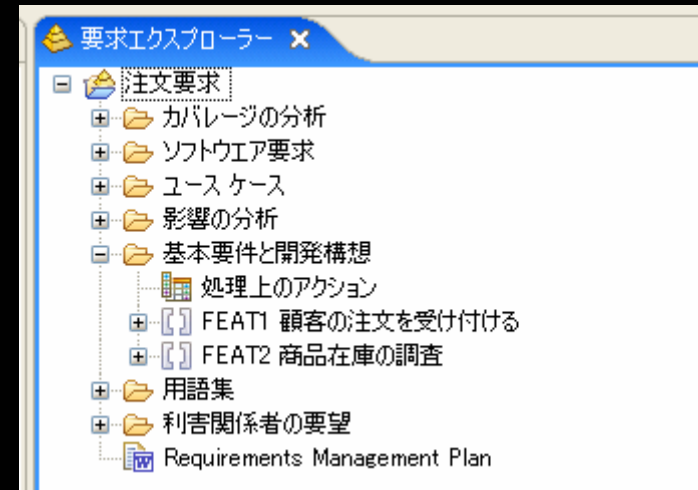
# 要求をReqProで管理

- 各要求項目を基本要件としてReqProで管理できる
- 注文処理の例では右記のようにまとめることができる

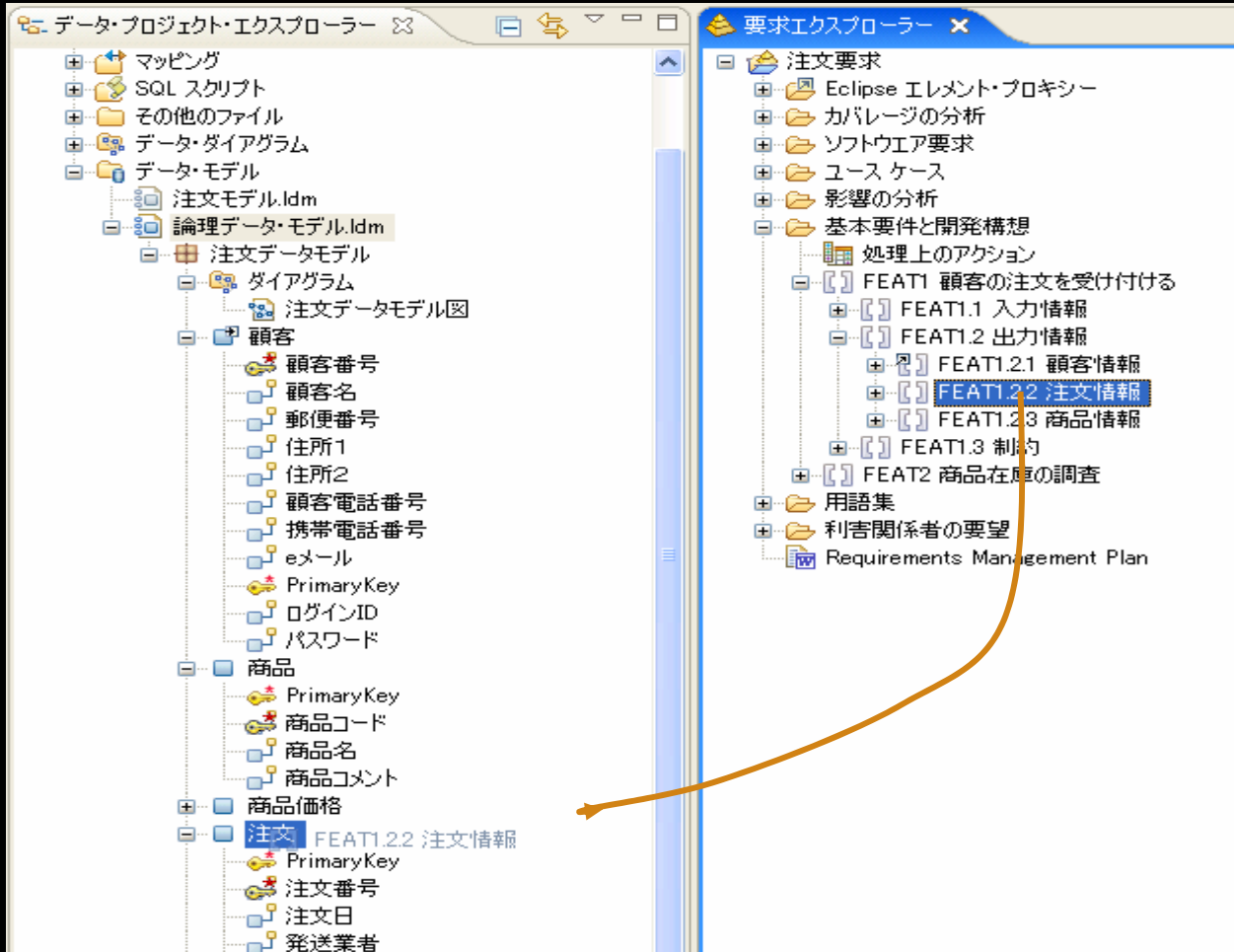


# ReqProとRDAの統合

1. ここではReqProに要求が登録されていることを前提とする
2. RDAを起動し要求パースペクティブを開く
3. 要求エクスプローラからRequisiteProプロジェクトを開く



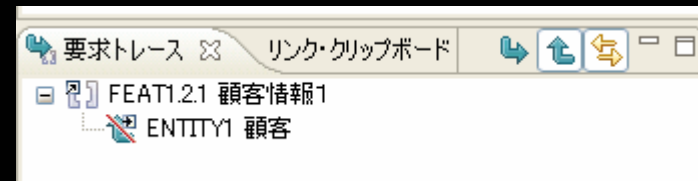
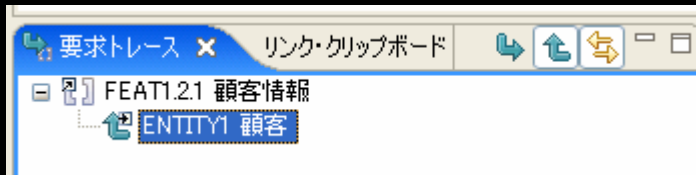
# 要求モデルとデータモデルの関連付け ( )



1. 要求エクスプローラに見える要求をマウスで左クリックしたままデータ・プロジェクト・エクスプローラに見えるデータ・モデルの対応するモデル要素上にマウスを移動させ、そのモデル要素上でマウスを離す(ドラッグ&ドロップ)
  - ▶ これによって要求モデルとデータモデルが関連付けられる

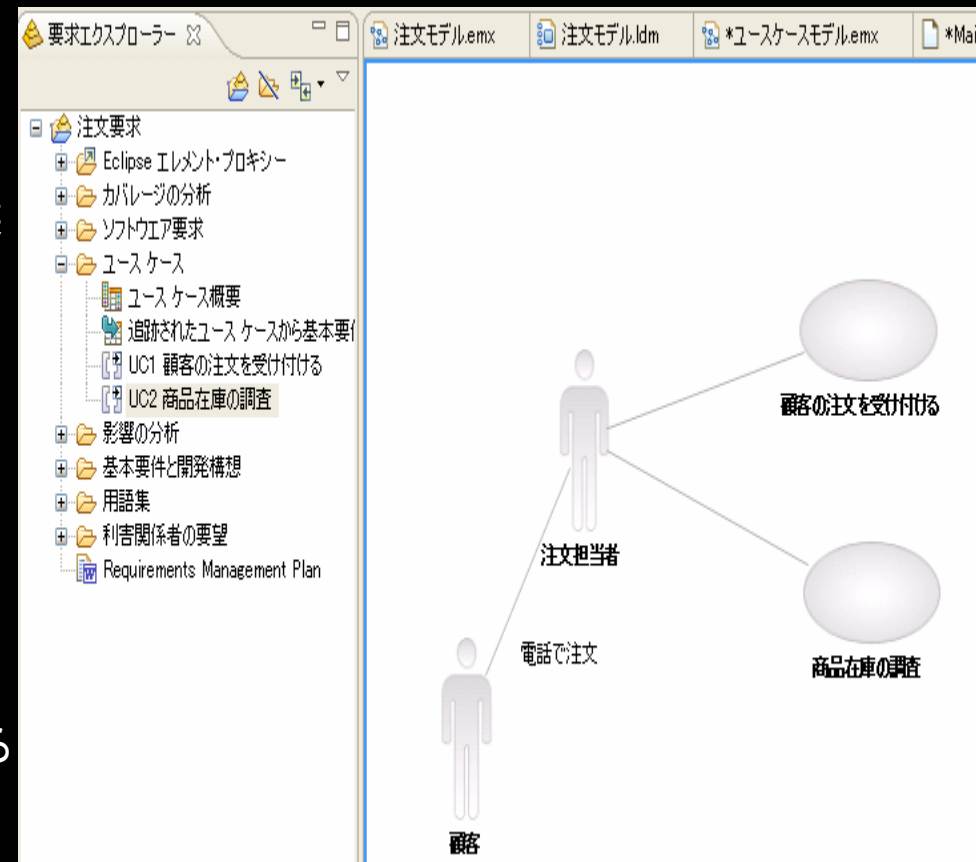
# 要求モデルとデータモデルの関連付け ( )

1. 要求モデルとデータモデルが関連付けられることによってRDAの要求トレースビューでマウスでクリックした要求と関連したデータモデル要素が見えるようになる
2. またReqProで要求モデルが変更されると(下の例では顧客情報 顧客情報1)↑ の上に\が引かれる。これをサスペクトリンクという
3. このサスペクトリンクはリンクされた情報が修正されたので見直せという意味である
4. これによって関連のある情報に変更が必要か見直す
5. 見直しによって関連を削除するか、関連はそのままサスペクトリンクだけを解除する必要があるがこの場合はRDAの要求トレースにあるサスペクトリンクが張られた要求を右クリックすることで削除や解除の操作が出来る

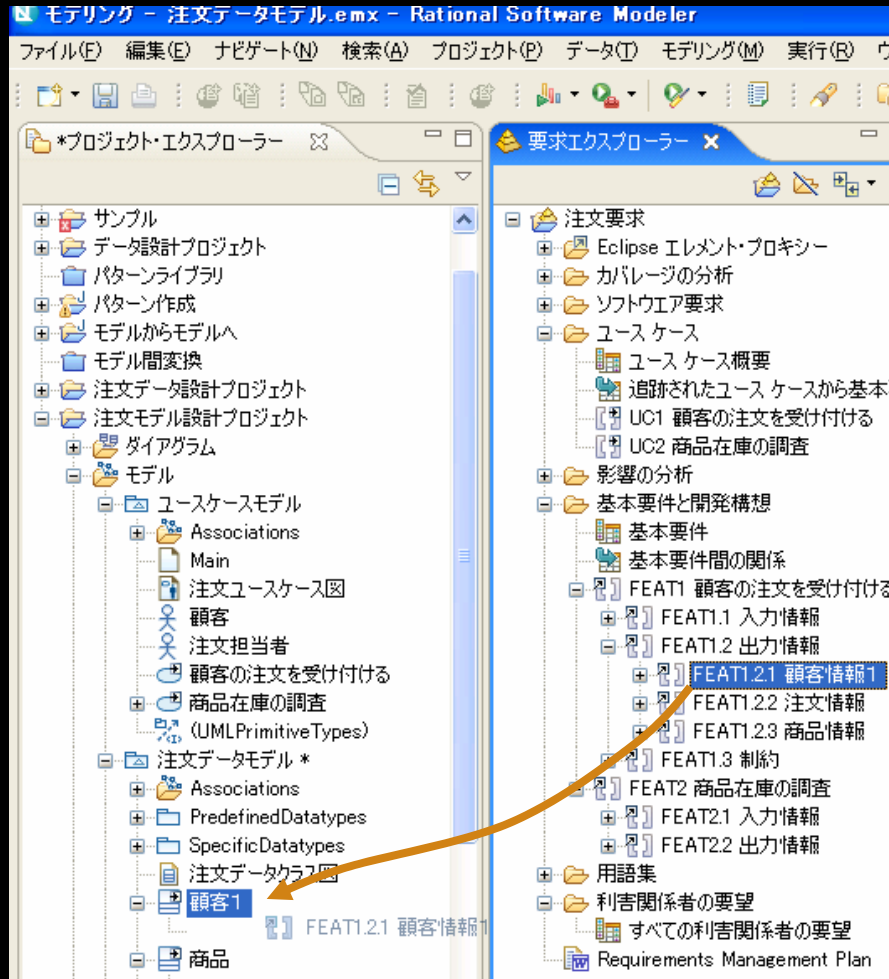


# ReqProとRSMの統合

1. RSMでモデリングパースペクティブを開く
2. RSMのwindowメニューから「ビューを開く」の中の「要求エクスプローラ」を選択しクリックする 要求エクスプローラが開く
3. ReqProで作成したユースケースをRSMのユースケース図にドラッグ&ドロップする
4. ユースケース図にアクターと関連を追加してユースケース図を完成する
5. RSMでユースケースが新たに見つかった場合はRSMのプロジェクトエクスプローラから要求エクスプローラへドラッグ&ドロップする



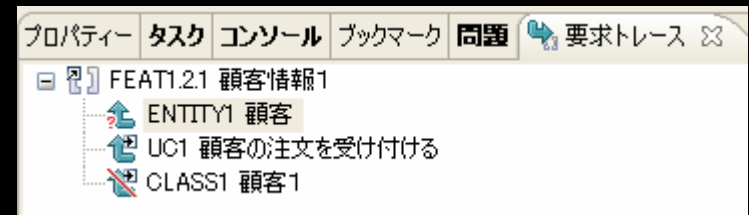
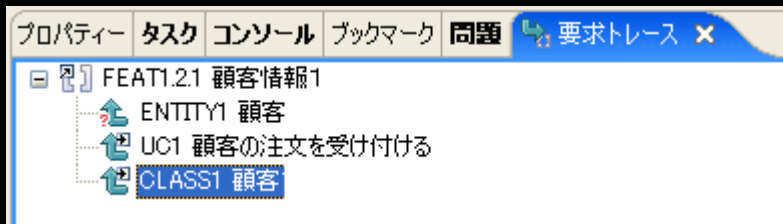
# 要求モデルとUMLモデルの関連付け( )



1. 要求エクスプローラに見える要求をマウスで左クリックしたままプロジェクト・エクスプローラに見えるUMLモデルの対応するモデル要素上にマウスを移動させ、そのモデル要素上でマウスを離す(ドラッグ&ドロップ)
  - ▶ これによって要求モデルとUMLモデルが関連付けられる

# 要求モデルとUMLモデルの関連付け( )

1. 要求モデルとUMLモデルが関連付けられることによってRSMの要求トレースと継承エクスプローラで選択した要求とUMLモデル要素の関連が見えるようになる
2. またモデル(要求モデル、UMLモデル)が変更されるとリンク記号の上に \ が引かれる。これをサスペクトリンクという
3. このサスペクトリンクはリンクされた情報が修正されたので見直せという意味である
4. これによって関連のある情報に変更が必要か見直す
5. 見直しによって関連を削除するか、関連はそのままサスペクトリンクだけを解除する必要があるがこの場合はRSMの要求トレースにあるサスペクトリンクが張られた要求を右クリックすることで削除や解除の操作が出来る



# 要求モデルとUMLモデルとデータモデルの関連

- 要求モデルは開発における最上位モデルとなる
- 要求モデルの最終成果物はユースケースモデルである (RUPより)
- ユースケースから下位モデルへの関連付けをRSMとReqProの統合で行うとその関連はReqProで管理される
- またRSMとRDAの統合のように下位のモデル要素同士で直接的に関連するものについてはRSM,RDAとReqProの統合の中でモデル要素同士を関連付ける
  - ▶ この関係もReqProで管理される
- ReqProを通して関連が定義されるとモデルに変更が発生した場合にサスペクトリンクにより影響するモデル要素が分かる
- これはRSMの要求トレースでも分かるが、ReqProでサスペクトリンクをみるビューを作っておくとその範囲の捉え方が易くなる



# ReqProからみた影響範囲

ReqProの追跡可能性ツリービューでサスペクトリンクが発生した要求全般を表示

これによって変更が発生した要素に関連する要素が全て特定され変更による影響範囲を特定する

Rational RequisitePro - 注文要求 - [FEAT: 基本要件からサスペクトリンクによる影響範囲全...]

ファイル(F) 編集(E) ビュー(V) 要求(R) 追跡可能性(A) ツール(T) ウィンドウ(W) ヘルプ(H)

注文要求

- Eclipse エlement-プロキシ
- カバレッジの分析
- ソフトウェア要求
- ユースケース
- 影響の分析
  - 基本要件とデータモデルの関係
  - 基本要件とモデルの関係
  - 基本要件の変更により影響を受けるユースケー...
  - 基本要件からサスペクトリンクによる影響範囲全...
  - 基本要件への影響範囲
- 基本要件と開発構想
- 用語集
- 利害関係者の要望
- Requirements Management Plan

FEAT1: 顧客の注文を受け付ける

- FEAT1.3.1: 応答時間が2秒以内
  - UC1: 顧客の注文を受け付ける
- FEAT1.3.2: 24時間移動の安定性
  - UC1: 顧客の注文を受け付ける
- FEAT1.1: 入力情報
  - FEAT1.3.3: 入力ミスを防ぐ使い勝手
    - UC1: 顧客の注文を受け付ける
  - UC1: 顧客の注文を受け付ける
- FEAT1.2.1: 顧客情報 1
  - CLASS1: 顧客 1
  - ENTITY1: 顧客
    - UC1: 顧客の注文を受け付ける
- FEAT1.2.1.1: 顧客番号
  - FEAT1.1.1.1: 顧客名
    - UC1: 顧客の注文を受け付ける
  - UC1: 顧客の注文を受け付ける
- FEAT1.2.1.2: 顧客名
  - FEAT1.1.1.1: 顧客名
    - UC1: 顧客の注文を受け付ける
  - UC1: 顧客の注文を受け付ける
- FEAT1.2.1.3: 郵便番号
  - FEAT1.1.1.4: 郵便番号
    - UC1: 顧客の注文を受け付ける
  - UC1: 顧客の注文を受け付ける
- FEAT1.2.1.4: 顧客住所
  - FEAT1.1.1.2: 顧客住所
    - UC1: 顧客の注文を受け付ける
  - UC1: 顧客の注文を受け付ける
- FEAT1.2.1.5: 電話番号
  - FEAT1.1.1.3: 顧客電話番号
    - UC1: 顧客の注文を受け付ける
  - UC1: 顧客の注文を受け付ける
- FEAT1.2.1.6: 携帯番号
  - FEAT1.1.1.5: 携帯番号
    - UC1: 顧客の注文を受け付ける
  - UC1: 顧客の注文を受け付ける
- FEAT1.2.2: 注文情報
  - CLASS3: 注文
  - ENTITY2: 注文
    - UC1: 顧客の注文を受け付ける