



# Evaluation チュートリアル

## Family Tree Modeling Language

( 家系図モデリング言語 )

### はじめに

このチュートリアルでは、MetaEdit+ Workbench で提供される機能について紹介しています。段階を追ったエクササイズを通して、モデル言語の作り方、その言語の使用法が解るようにしています。また、モデル言語の機能を拡張する方法についても言及しています。

このチュートリアルを実施するために必要なものは、以下の通りです。

- MetaEdit+ Workbench
- ウェブ・ブラウザ。MetaEdit+ と、このチュートリアルを開いた状態で、過不足無く使用できるもの  
がお勧め。
- 1 ~ 4 章までで、1 時間程度。5 , 6 章はオプションとして、更に時間を要します。

MetaEdit+ に関する詳細は、 MetaEdit+ User's Guide 、 MetaEdit+ Workbench User's Guide 、  
MetaCase社サイト <http://www.metacase.com> 、  
富士設備社サイト <http://www.fuji-setsu.co.jp/products/MetaEdit/index.html> を参照ください。

## 目 次

1 . 家系図モデリング言語の紹介 .....	3
2 . MetaEdit+へのログイン .....	5
3 . 家系図モデリング言語の実施 .....	7
3.1. オブジェクトの作成 .....	7
3.2. シンボルの作成 .....	11
3.3. 作業の保存 .....	13
3.4. リレーションシップの作成 .....	14
3.5. ロールの作成 .....	15
3.6. Graphの作成 .....	16
3.7. 家系図ダイアグラムの作成 .....	21
3.8. 家系図ダイアグラムにオブジェクトを追加 .....	22
3.9. 家系図ダイアグラムにリレーションシップを追加 .....	23
3.10. 家系図ダイアグラムからレポートを生成 .....	26
3.11. MetaEdit+の終了 .....	27
4 . 家系図モデリング言語の改良 .....	28
4.1. 男女としてのPerson .....	28
4.2. Port (ポート) を使った描画精度の向上 .....	34
4.3. ジェネレータの作成 .....	41
4.4. 家系図モデリング言語を改良するための更なるアイデア .....	48
5 . 終わりに .....	49

## 1. 家系図モデリング言語の紹介

MetaEdit+ がどういったものであるかのイメージを得るために、ドメインスペシフィックモデリング言語の作成を行いましょ。各評価者によって良く知るドメインは異なるため、誰もが知っている“家族”を取り上げます。家族をモデリングするのに相応しいモデリング言語はどんなものか？ もちろん、UMLにトライすることもできますが、予想以上に複雑となり、セマンティックに対してかなりの変形が求められることになるでしょう。そうする代わりに、従来の“家系図 (Family Tree)”に近い、我々自身の言語を作りましょ。

最初の疑問は、我々の例：家系図ドメインで、どんな種類のコアコンセプトが存在するかということです。家系図は単純なドメインで、必要なことはすぐに理解できます。最初に、人物 (Person) のコンセプトがなければなりません。各人物は、親 (Parent) として二人の他の人物を必要とします。人物は、彼 or 彼女の子供 (Child) の親になります。両親と子供は、共に家族関係を築きます。

基本的なコンセプトに加えて、各コンセプトでどんな情報が格納されているか知る必要があります。人物は、氏名 (First name と Last name) を必要とします。最後に、図が実際どのように見えるか知っておく必要があります：コンセプトに対してシンボルを持ちたい。例えば、概観は、図 1-1 のようになります。

図 1-1

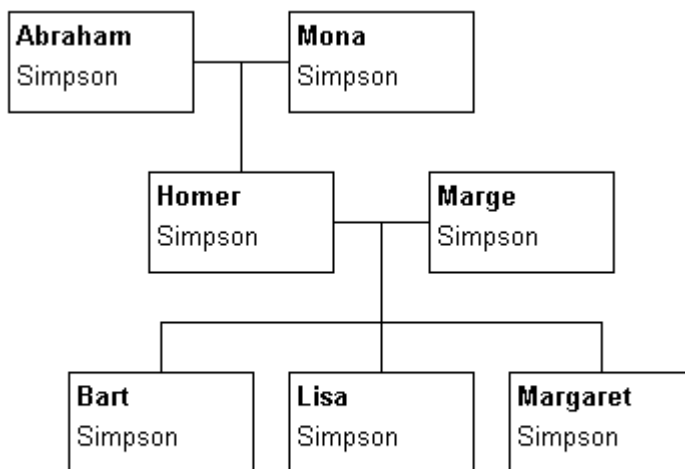


Figure 1-1. Sketch of a graphical notation for the Family Tree.

この概観から、実際の言語実装を作り、それをサポートするツールを提供するために、よりフォーマルな方法で自身の言語構造を示さなければなりません。これを行う1つの方法は、言語のためにメタモデルを作ることです。その名のとおり、メタモデルは、正当な言語要素とルールを定める“モデルのモデル”です。

MetaEdit+において、メタモデルは、GOPRR というメタモデリング言語を使って定義しています。名前の由来は、Graph、Object、Property、Port、Relationship と Role を表す頭文字です。これらは、モデリング言語を記述する時に使用するメタタイプです。図 1-2 は、メタタイプを例で説明しています。

図 1-2

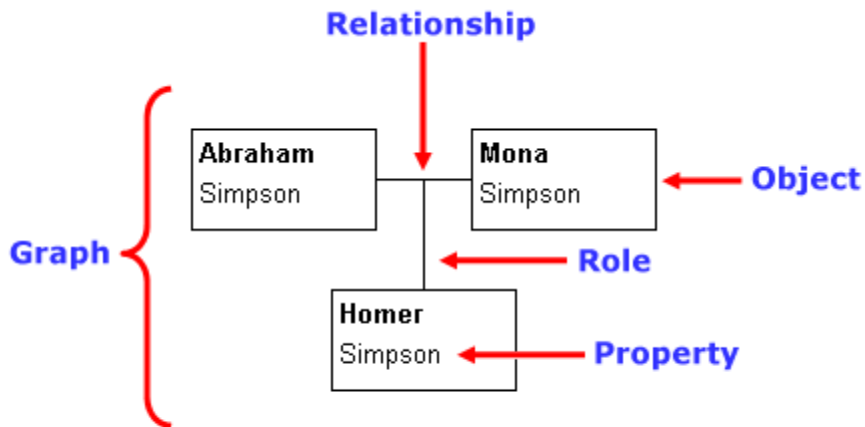


Figure 1-2. GOPRR metatypes as found in Family Tree.

各メタタイプには、下記のように、その働きと意味があります。

- Graph (グラフ) は、ある 1 つのモデルです (通常、ダイアグラムとして表されます)
- Object (オブジェクト) は、Graph の主要な要素です (例でいう人物のような)
- Relationship (リレーションシップ) は、Object 同士を接続します (人物オブジェクト間の家族関係のような)
- Role (ロール) は、Relationship 内でオブジェクトを接続します (人物は、家族関係で、親 or 子ロールのどちらかになります)
- Port (ポート) は、Role がどのように Object につながるかの可能な追加の意味を定義します
- Property (プロパティ) は、上記を特徴づける属性です (例の First Name や Family Name のような)

これらのメタタイプを使って、モデリング言語の構造と基本的な意味論を記述するメタモデルを組み立てることができます。図 1-3 は、家系図モデリング言語のメタモデルを示します。

図 1-3

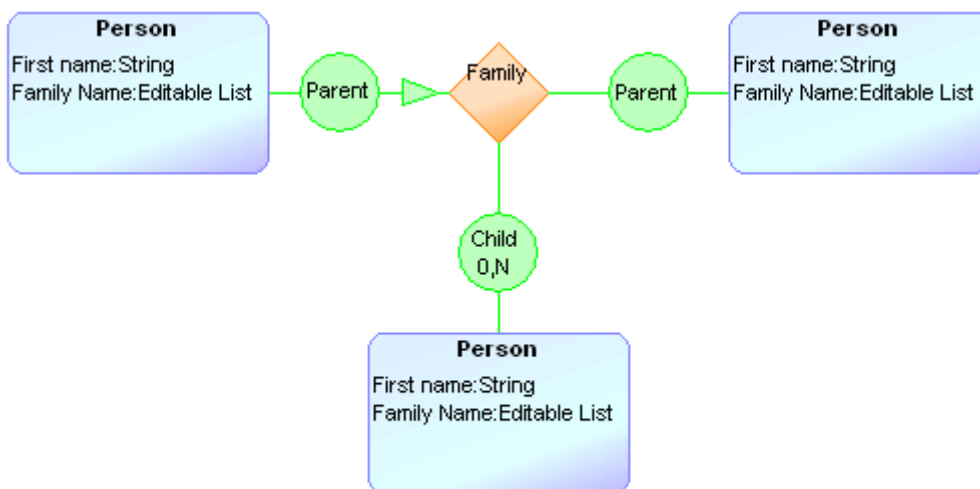


Figure 1-3. Metamodel for the Family Tree modeling language.

上記メタモデルは、下記の通りです：

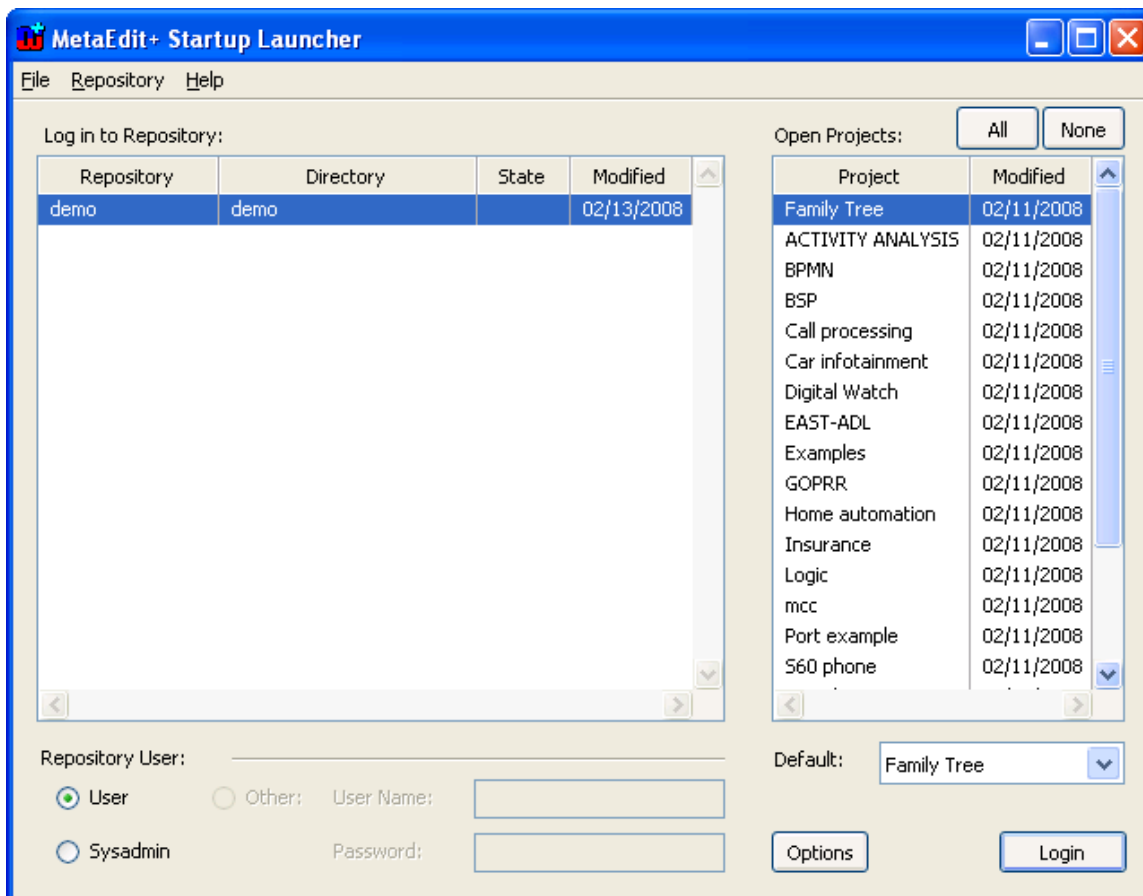
Person オブジェクト（青の長方形）は、他の Person オブジェクトと家族関係（Family Relationship：橙のひし形）にあり、各家族関係に対して、少なくとも二人の人物が Parent ロール（緑の円）になければなりません。任意で、Child ロールに関係する追加の人物がいます（0～N）。Person オブジェクトはまた、Name と呼ばれる識別プロパティを持っています。

スケッチとメタモデルから、モデリング言語を試みるにあたり必要となる全ての基本構築用ブロックを得ることができました。次に行うことは、コンセプトの証明です。それでは、これらが MetaEdit+ でどう扱われるかを調べてみましょう。

## 2 . MetaEdit+へのログイン

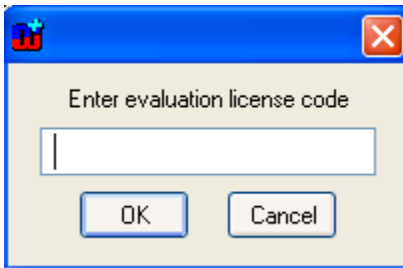
MetaEdit+は、リポジトリベースのアプリケーションで、リポジトリとの接続は、起動中に確立されます。起動した時、MetaEdit+は、図 2-1 のような起動画面が開きます。

図 2-1



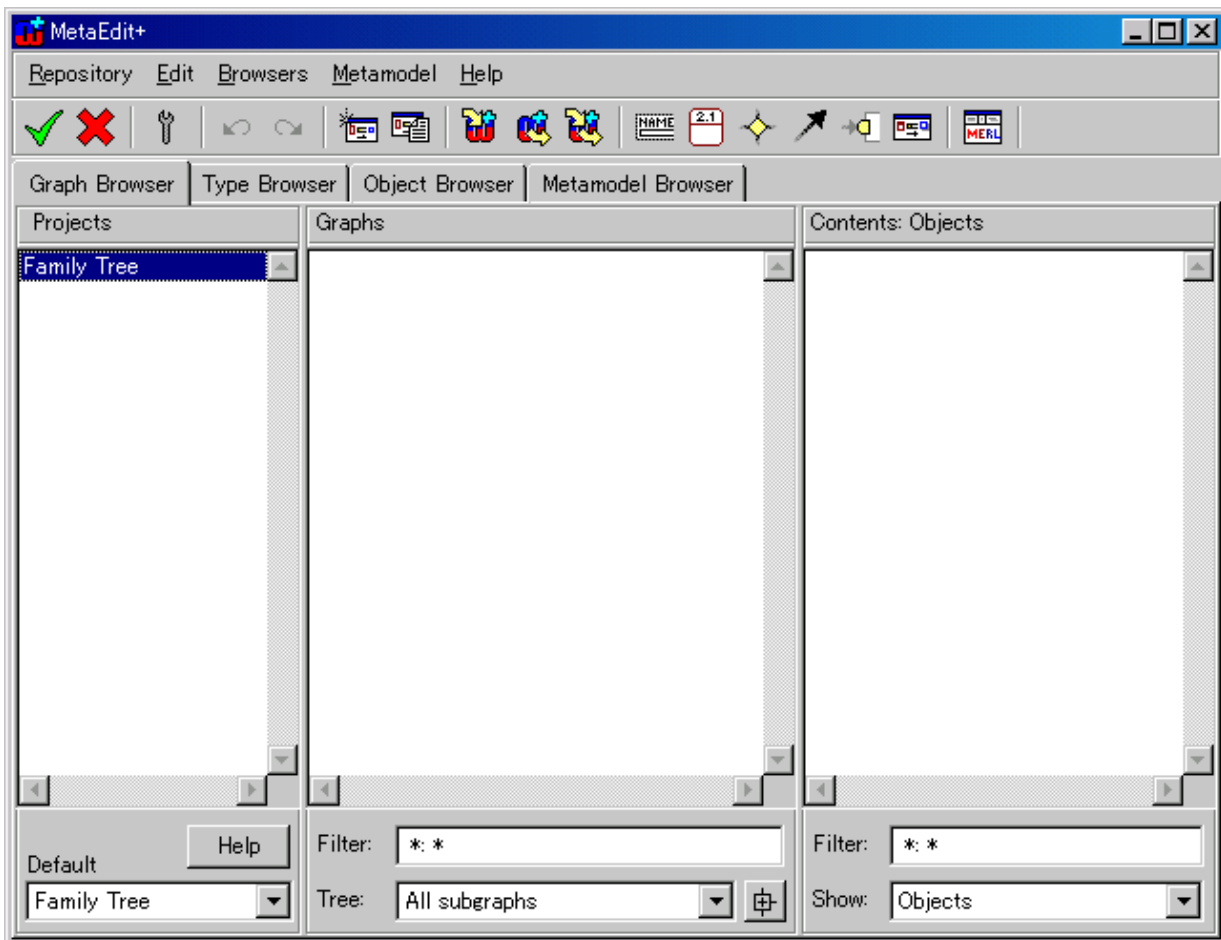
“demo” リポジトリが、“Log in to Repository” から選択され、“Repository User” として “User” が選択されていることを確認してください。“Open Projects” リストから “Family Tree” を選択し、Login ボタンをクリックします。MetaEdit+の評価版を使っている場合、最初のログインダイアログで、評価用ライセンスコードを入力するダイアログが開きます。ライセンスコードを正確に入力し、OK ボタンをクリックします。

図 2-2



ログインに成功すると、起動画面が閉じ、MetaEdit+メイン画面（図 2-3）が開きます。

図 2-3



MetaEdit+ is now ready to be used. If you want to quit MetaEdit+ before completing this tutorial, please see Section [3.11](#) for exit instructions.

MetaEdit+の、使用準備が整いました。（このチュートリアルの中で MetaEdit+を終了する場合、本資料のセクション 3.11 を参照してください）

### 3 . 家系図モデリング言語の実施

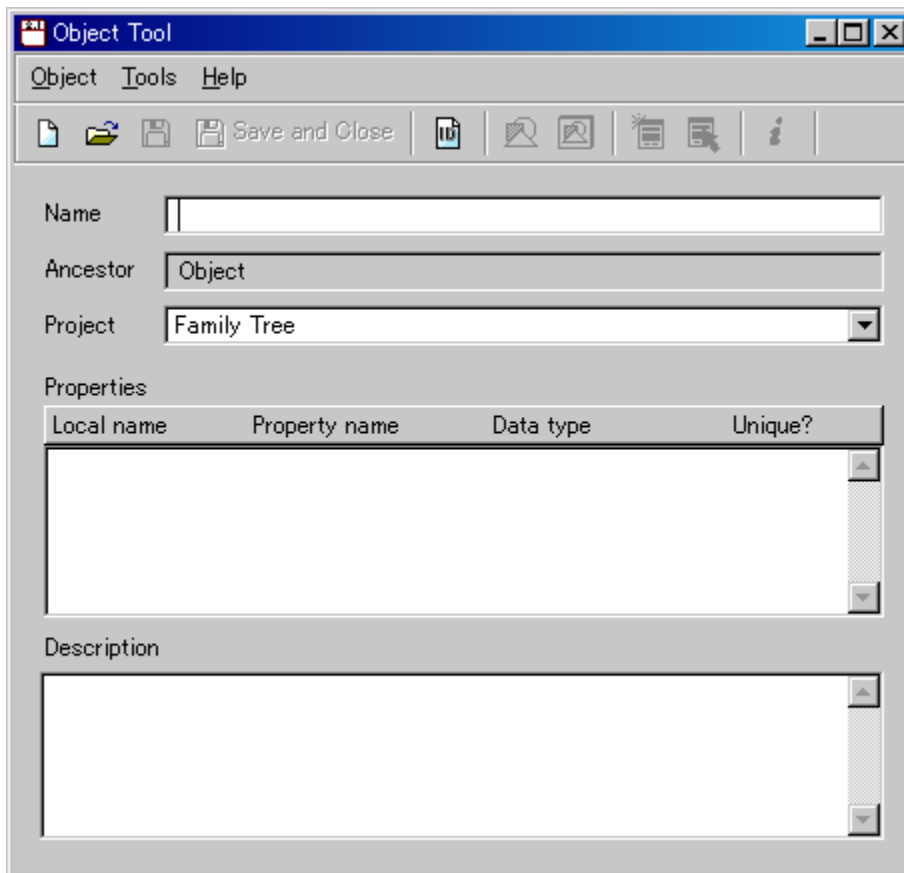
開いた “ Family Tree ” プロジェクトは、空です。まだ何のモデルや定義手法も含んでいません。全くの白紙状態で、家系図モデリング言語の実装ができるようになっていきます。その前に、MetaEdit+に、家系図メタモデルを入力しなければなりません。これに対して MetaEdit+は、2つの異なるアプローチを提供します（グラフィカルメタモデリングとフォームベースメタモデリング）。

前者を選んだ場合、メタモデルは、グラフィカル GOPRR 表記（図 1-3）を使って MetaEdit+で描かれ、ツール内に読み込まれます。後者の場合、MetaEdit+は、各メタタイプを作成し管理するツールを提供します。グラフィカルメタモデリングは、おそらく初心者レベルのメタモデラーやメタモデルを文書化することに対して適しています。このチュートリアルでは、フォームベースメタモデリングツールを用います。なぜなら、チュートリアルの後半のいくつかの作業は、フォームベースツールの使用を必要としていますので、意図的に同一のツールセットのみにしたほうが良いからです。

#### 3.1. オブジェクトの作成

もっとも明らかな GOPRR メタタイプは、Object です。モデリング言語における図の主要な要素は、概ね Object になります。この場合、人物 (Person) のドメインコンセプトは、このカテゴリーに分類されます。ツールバーの Object Tool ボタンを押すか、MetaEdit+の Metamodel -> Object Tool を選択することで Object Tool を起動できます（図 3-1）。

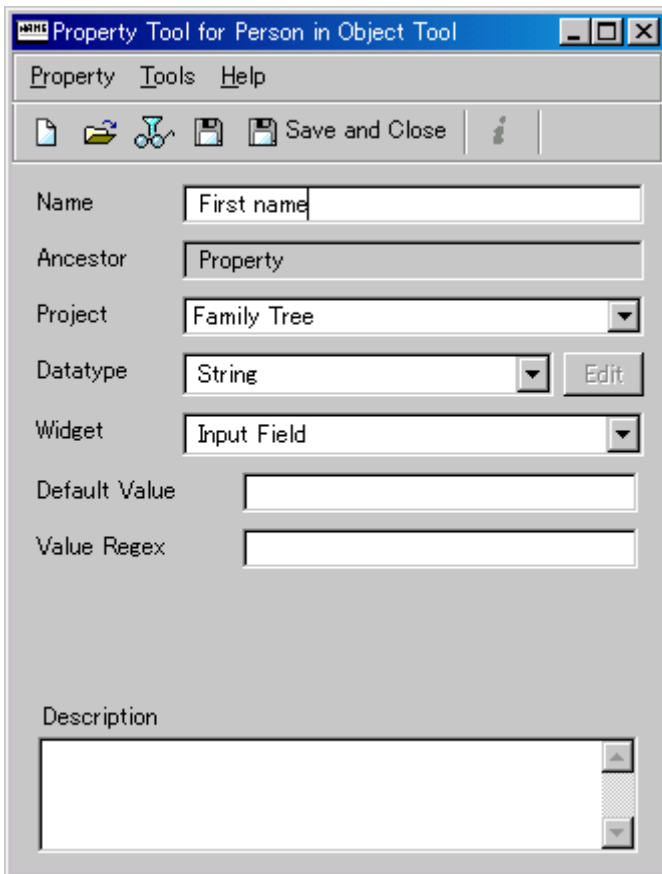
図 3-1



Object Tool の最上位のテキストフィールド (Name) で、その名前 “ Person ” を入力し、Person タイプを作ります。また、オブジェクトの Project は、“ Family Tree ” です（もし違うなら、変更すること）。

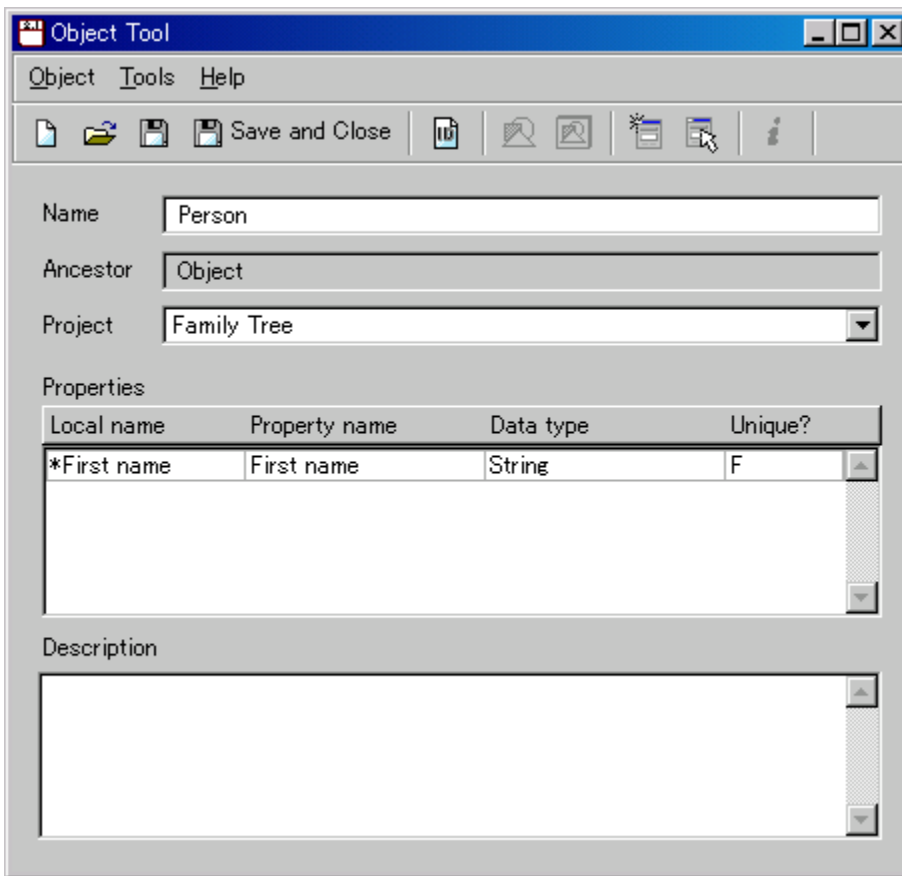
次に、どんな情報が各 Person に格納できるか示す必要があります。GOPRRR で、情報を格納するための各スロットは、property と呼ばれています。Person オブジェクトに property を加えるために、Object Tool で Properties リストの上にマウスを動かし、右クリックします。開いたポップアップメニューから、[Add Property] を選択します。Property タイプ定義が無い場合、MetaEdit+ は自動的に新しい property タイプを作って選びます (すでにいくつかタイプがあれば、開いたダイアログから “New Property Type ” を選びます)。新しい property タイプを定義するための Property Tool が開きます (図 3-2)。

図 3-2



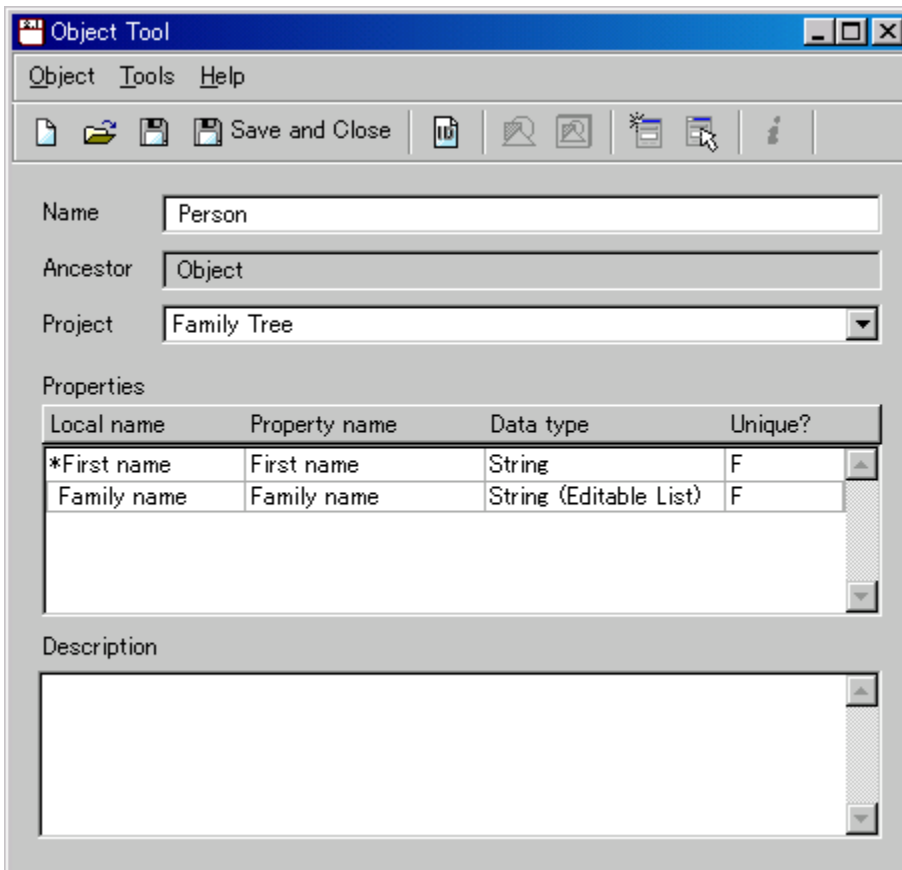
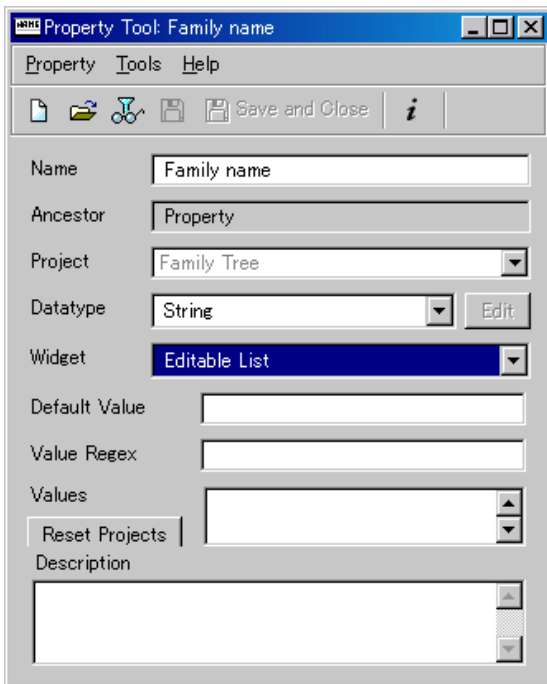
Property Tool で、Name に “First name ” を入力します。他の値は、この場合デフォルトでよく、ツールバーで Save ボタンを押して先へ進みます。このセクションでタイプを生成したのはこれが初めてで、MetaEdit+ は、データベースで全てのタイプのチェックを実施します。多くのタイプが “demo ” リポジトリにあり、数秒かかります。しかし、最初だけです。“Do you really want to add the property First name to Person?” と尋ねるダイアログが出たら、追加を承認し下記ダイアログでローカル名として “First name ” を承認するために、Yes を押します。自動的に閉じない場合、Property Tool を閉じ、Object Tool へ戻ります。Person オブジェクトの Properties リストで、作った property を確認できます (図 3-3)。

図 3-3



次に、“Family name” property を作ります。先程と同様に、Object Tool の Properties リスト上で右クリックし、[Add Property] をクリックします。“New Property Type” を選択して OK ボタンを押します。Property Tool で、Name フィールドに “Family name” を入力します。Property Tool の Widget リストで、ユーザがこの property の値をどのように入力するかを設定することができます。同じ Family Name で、数人の人物があり、ユーザがプルダウンリストから、これらあらかじめ入力された値を選択できるようにすると、いいでしょう。この Widget は、Editable List と呼ばれます：Widget リストから、可能な widget タイプからそれを選択します。Save して、Property Tool を閉じます。

図 3-4



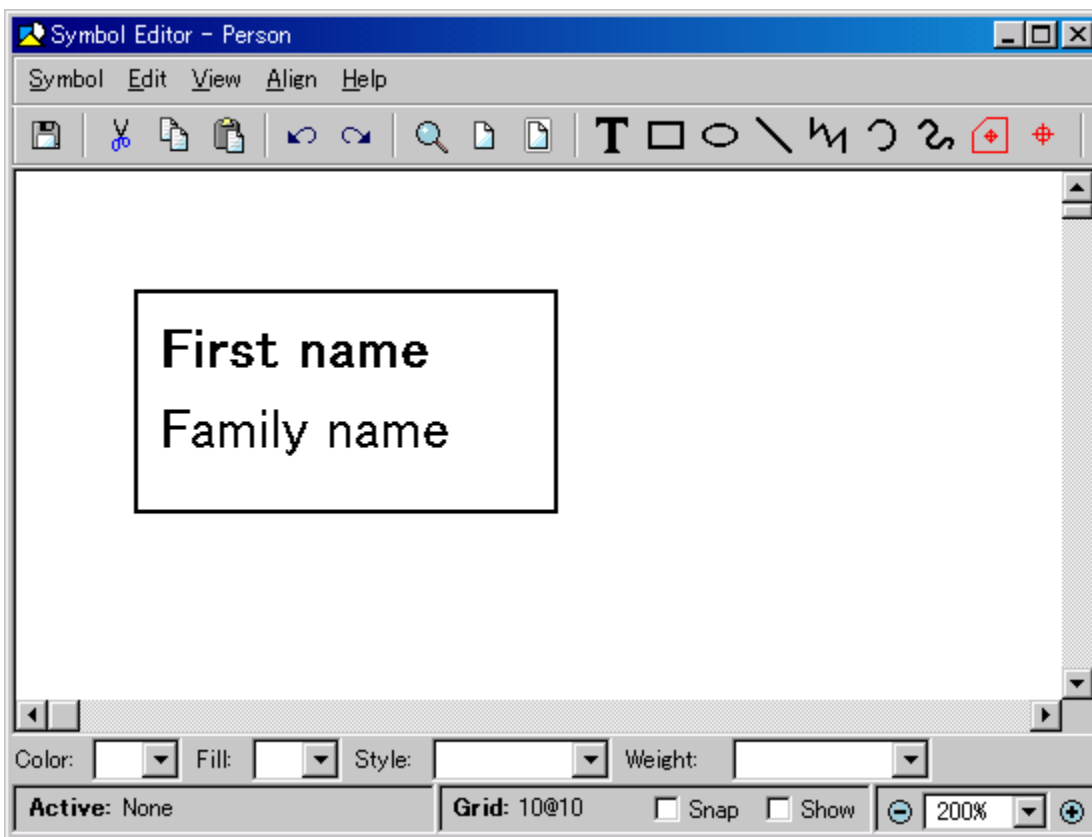
Object Tool は、上記のようになっています。ツールバーで Save ボタンを押します。Person コンセプトの定義を完了しました。

### 3.2. シンボルの作成

Person は、Family Tree のグラフ表示で使用するコンセプトで、そのためにグラフィカルシンボルを定義する必要があります。Object Tool のツールバーで、Symbol Editor ボタンを押すことで Symbol Editor を起動できます。ゼロからきれいなシンボルを作るのは時間がかかるので、ここではいくつかの作業をスキップし、シンボルライブラリからシンボルをロードします。

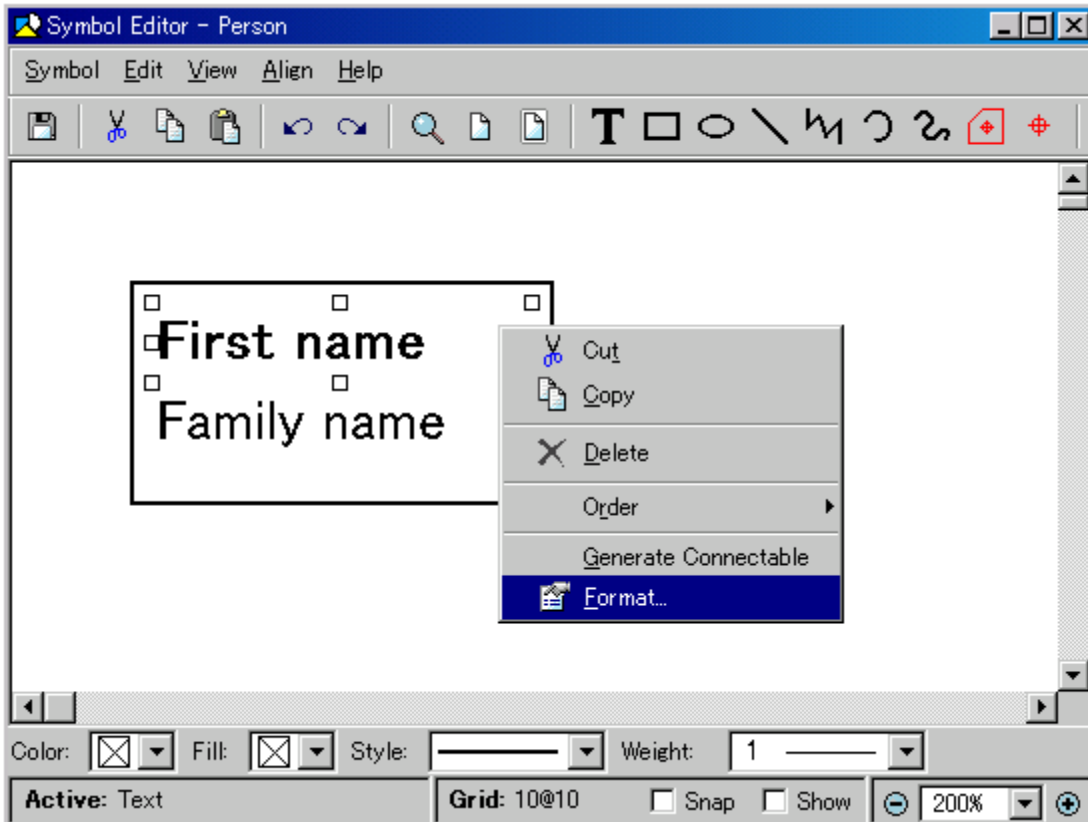
Symbol Editor で、Symbol->Browse Library を選択し、リストから“ Person ”のシンボルを選び、Past & Close ボタンを押します。準備されたシンボルテンプレートがエディタの中に貼り付けられます。その要素は、現在選択されているので、マウスであちこちに動かすことができます。位置を決めて、左マウスボタンを押します。要素の選択はまだ残っており、描画エリア上で左マウスボタンをクリックすることで、非選択になります。エディタは現在、図 3-5 のようになっています。

図 3-5



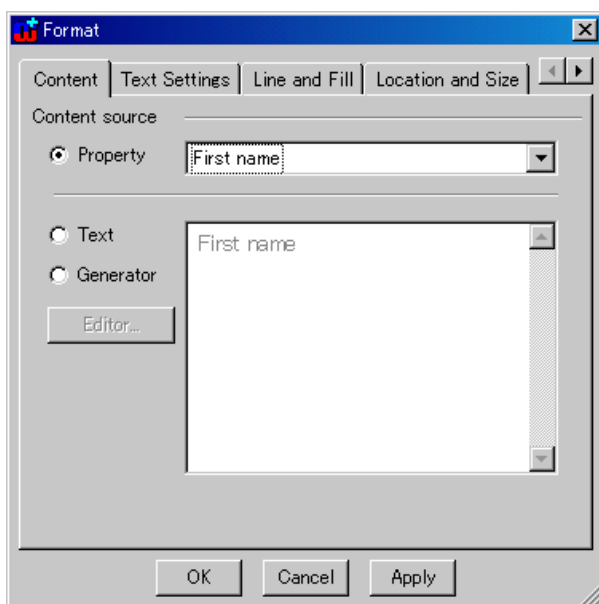
しかし、“ First name ”と“ Family name ”のテキストフィールドに、Person オブジェクトに追加した対応する property を関連付けなければなりません。テキスト“ First name ”をクリックし、右マウスボタンをクリックします。テキストフィールドが選択され、ポップアップメニューが表示されるので、[Format...] を選択します (図 3-6)。

図 3-6



テキストフィールドの Format ダイアログが開きます。現在、テキストフィールドは、ダイアログ中央のテキストボックス内に入力されている固定テキストで示されています。しかし、ここで行いたいのは、“First name” property の代わりに値をテキストフィールドに示すことです。そのために、“Property” ラジオボタンをクリックし、プルダウンリストから “First name” を選択します (図 3-7)。OK ボタンを押します。選択された property タイプの名前は、現在テキストフィールドに表示されています。同様に、“Family name” のテキストフィールドも設定します。

図 3-7



テキストフィールドに property を割当てた後、まだやることがあります。シンボルに対して、connectable エリアを定義しなければなりません。この場合、デフォルトの接続は、目的に適しているため、Symbol->Save or Symbol Editor ツールバーの Save ボタンを押すことで、生成できます。connectable は、図 3-8 のようにシンボル要素の周りに表示されます。

図 3-8

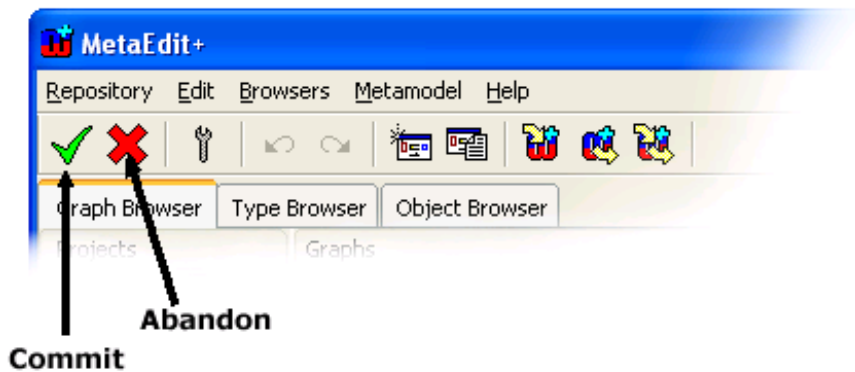


シンボルは準備され、保存されており、Symbol->Exit or エディタウインドウを閉じることで、Symbol Editor を閉じます。また、Object Tool も閉じます。

### 3.3. 作業の保存

ドメインコンセプトの大部分を定義したので、作業を保存します。MetaEdit+の画面に戻り、Commit と Abandon ボタンがあることに気付くでしょう (図 3-9)。

図 3-9

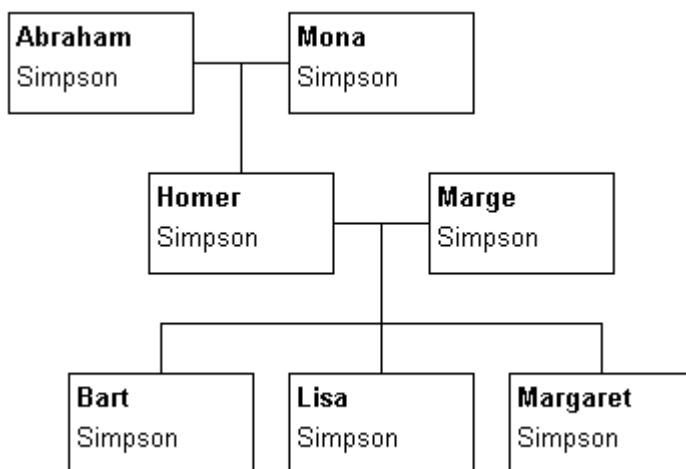


Commit ボタンを押せば、変更内容をリポジトリに保存し、Abandon を押せば、前に Commit によって保存した状態のままです。Commit を押して、操作が完了するのを待ちます。

### 3.4. リレーションシップの作成

次に何を必要とする必要があるか？もう一度、ドメイン仕様を見ましょう。Person のコンセプトを作りました。図 3-10 と図 1-3 のメタモデルによると、親と子のリレーションシップである Family を定義しなければなりません。

図 3-10



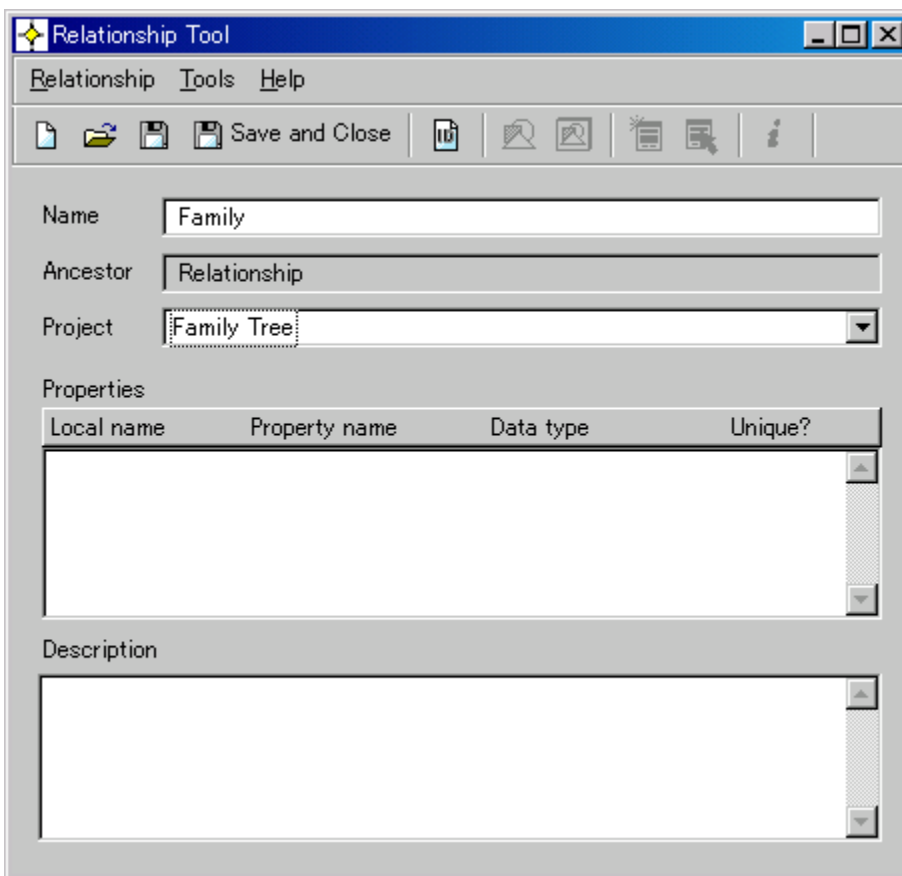
Family リレーションシップに加えて、Person がそのリレーションシップで行うことができるロールを表すためのコンセプトが必要です。単純に、Person が与えられた Family リレーションシップで、Parent ロールか Child ロールのどちらかになります。見た目には、Person に向かう各線が Role で、線の中心が Relationship です。

もう少しよく見てみると、特定のルールが、どのように Family リレーションシップで Persons を結び付けているかがわかります。例えば、与えられた Family リレーションシップで、Parent ロールには、きっちり 2 つのオブジェクトが常になければなりません。しかし、Child ロールは、いくつオブジェクトがあっても構いません。

これらの論点を取り扱うために、binding(バインディング)として知られる新しいメタモデリング構造を必要とします。バインディングは、オブジェクトがどのロールのリレーションシップに関係しているか、各ロールが何回現れるかの情報を含んでいます。各バインディングは、1つのリレーションシップ、2つ以上のロール、各ロールに対して1つ以上のオブジェクトから成ります。

最初に、リレーションシップとロールを定義する必要があります。ツールバーの Relationship Tool ボタンを押すか、MetaEdit+画面で Metamodel->Relationship Tool を選択することで、Relationship Tool を起動します。Relationship Tool は、Object Tool のようになっています。新しい Family リレーションシップタイプを作るために、テキストフィールドで、“Family”を入力します(図 3-11)。

図 3-11

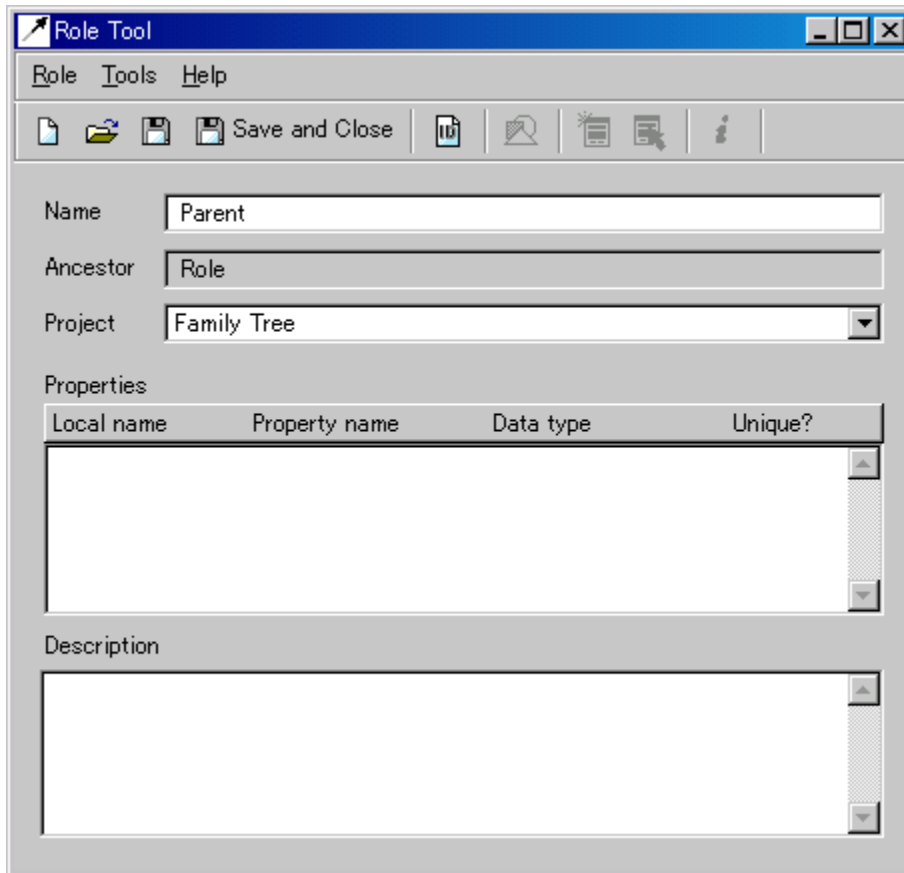


現時点では、Family リレーションシップにどんな property も含めたくないで、Save を押して新しいリレーションシップタイプの生成を完了し、ツールバーの Save and Close ボタンを押します。

### 3.5. ロールの作成

次に、必要なロールを定義しなければなりません。ツールバーの Role Tool を押すか、MetaEdit+の Metamodel->Role Tool を選択することで、Role Tool を起動します。Property やシンボルを必要としないので、Family リレーションシップのロールは、簡単に作れます。従って、Parent ロールを作るために、一番上のフィールドで、“Parent”を入力し、Save を押します(図 3-12)。

図 3-12



Child ロールも、同様に作られます。Role Tool のツールバーで New ボタンを押すか、Role->New を選択します。一番上のフィールドに “Child” を入力し、Save and Close を押します。

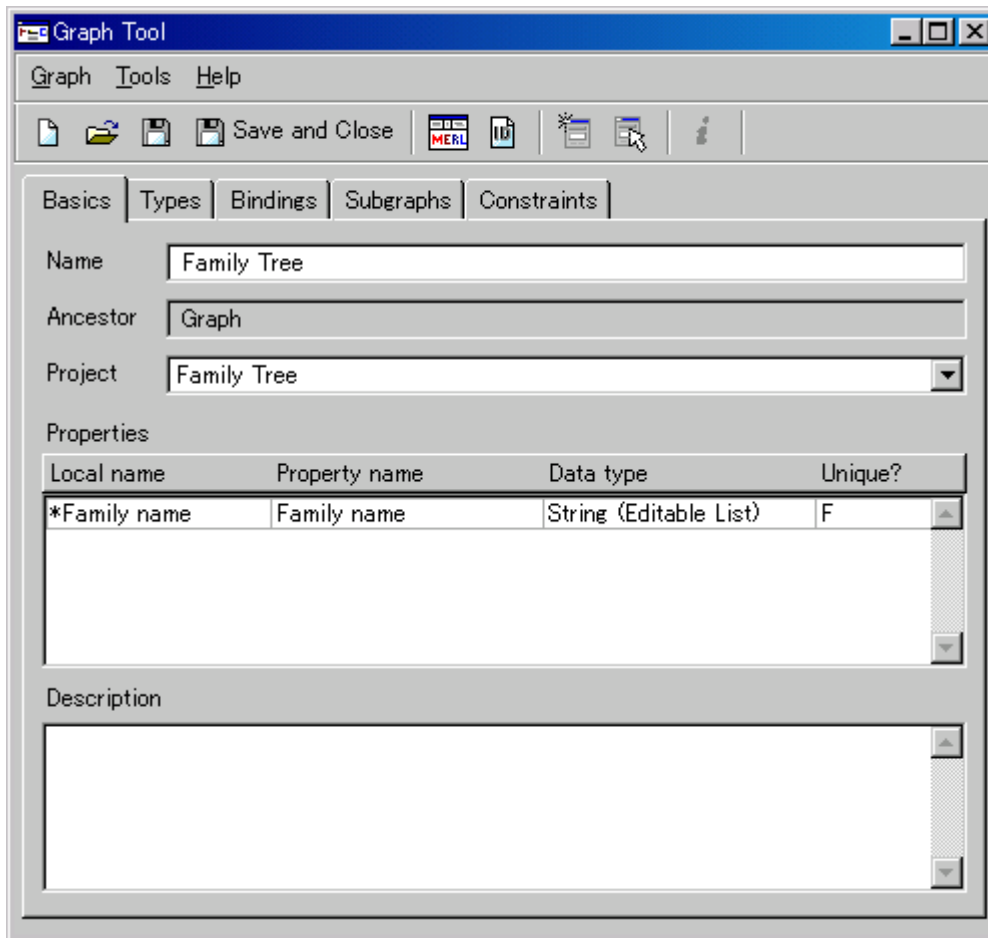
### 3.6. Graph の作成

家系図モデリング言語の最初のバージョンに必要な全てのメタモデル部品が用意できたので、再度 Commit します。次に行うことは、完全なモデリング言語にこれらの部品を結集することです。しかし、最初に我々が何をしようとしていたか考えてみましょう。家系図モデリング言語は、実際どのようになっていたでしょうか？ それは家系図を描くための簡単なダイアグラムでした。では、家系図モデリング言語のダイアグラムタイプを定義しましょう。

GOPRR で、メタタイプグラフは、ダイアグラムタイプに相当します。新しいダイアグラムタイプを作るために、Graph Tool で新しいグラフタイプを作らなければなりません。ツールバーの Graph Tool ボタンを押すか、MetaEdit+の Metamodel->Graph Tool を選択することで、Graph Tool が開きます。

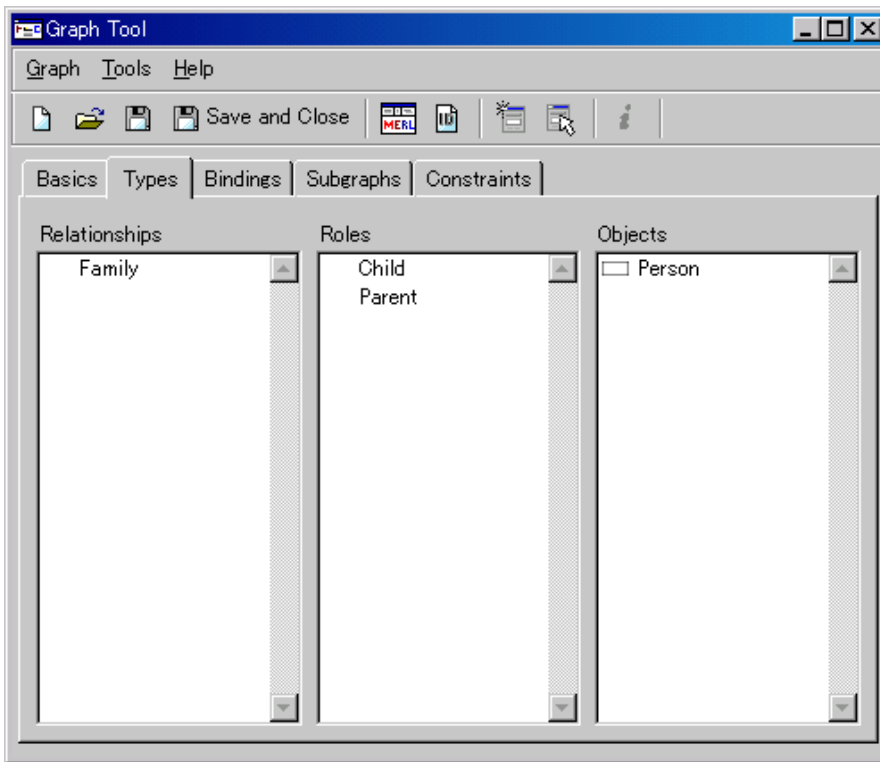
まず、Name フィールドにモデリング言語 (Family Tree) の名前を入力します。Properties リストで、右クリックし、[Add Property]を選択します。今回は、完全に新しいpropertyタイプを作りたくはありません。その代わりに、ダイアグラムの名前として、既存の “Family Tree” property を再利用します。各ダイアグラムの名前は、家名になります。よって、リストから “Family name” を選びます。Graph Tool は、図 3-13 のようになります。

図 3-13



Graph Tool の Type タブを開きます。これは、モデリング言語で使いたいタイプを選ぶ場所です。Relationships リストで、右クリックし、[Add]を選択します。その開いたリストから、“Family”を選び、OK ボタンを押します。“Family”リレーションシップは、Relationships リストに表示されています。同様に、Roles リストに“Parent”と“Child”ロールを追加し、Objects リストに“Person”を追加します。Graph Tool は、図 3-14 のようになります。

図 3-14

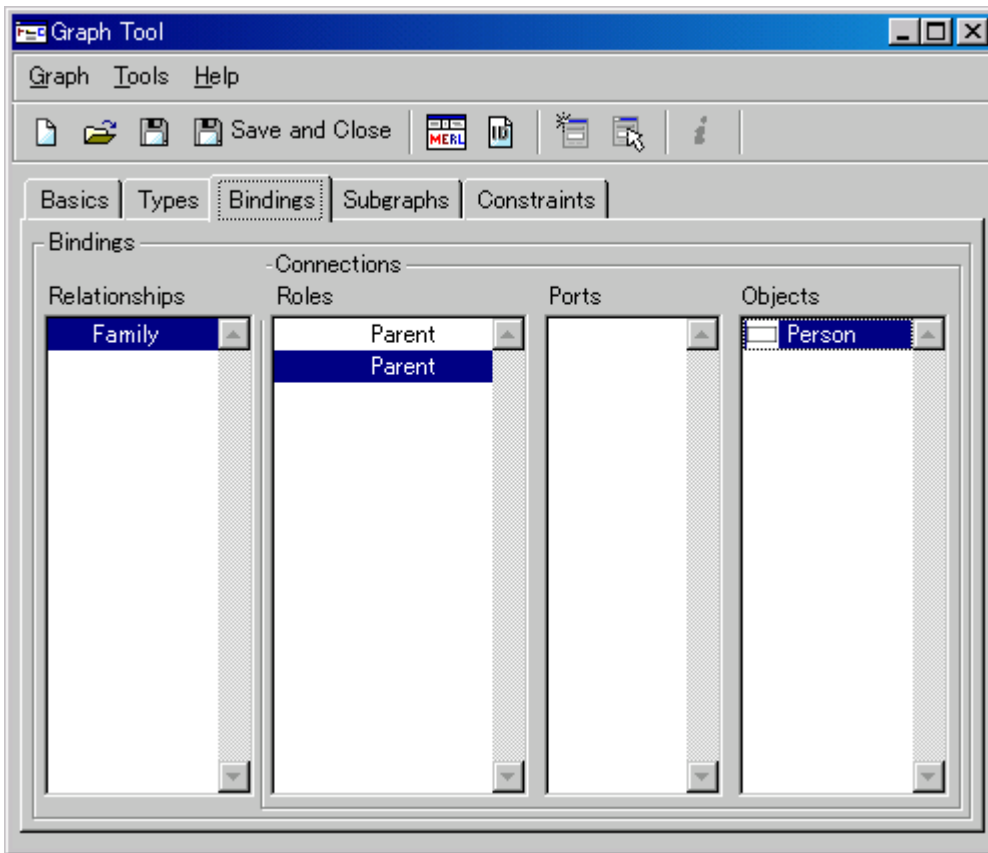


Bindings タブへ行きます。ここでは、完全なモデリング言語仕様として、メタモデル部品をつなぐバインディングを作成します。最初に、Relationships リストで右クリックして、[Add]を選び、“Family”を選択します。

“Family”が、Relationships リストで選択されていることを確認して下さい。Roles リストで右クリックします。[Add]を選びリストから“Parent”を選択します。今、“Family”と“Parent”が選択されています。Objects リストに“Person”を追加します。今、“Family”リレーションシップの Parent ロールは、Person オブジェクトにつなげなければならない”という一部のバインディングを作りました。

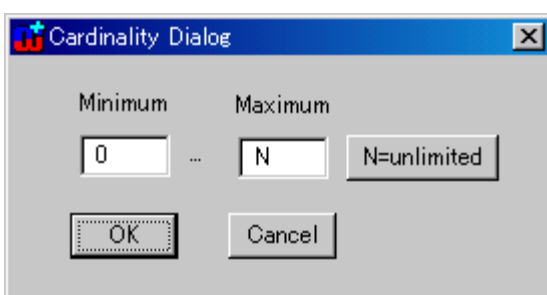
Person に対して常に 2 人の親がいるので、同様に他の Parent ロールを追加し、Objects リストに Person を追加します。Graph Tool は、図 3-15 のようになります。

図 3-15



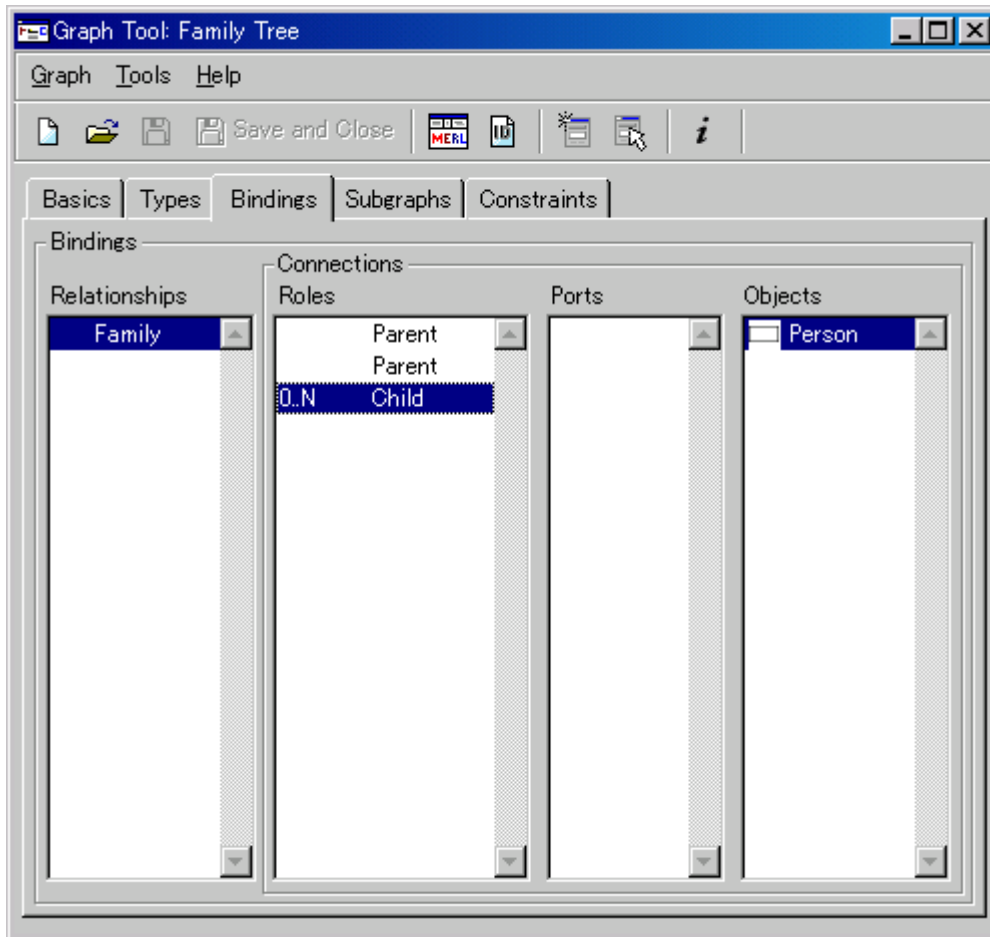
Parent ロールに加えて、Family リレーションシップには Child ロールもあります。それを作るために、Roles リストで右クリックし、[Add] を選びリストから “Child” を選択します。Objects リストが現在空白になっているので、再度 Person オブジェクトを追加します。今、“Family リレーションシップの Child ロールは、Person オブジェクトにつなげなければならない” という一部のバインディングを作りました。ドメイン仕様やメタモデルの課題が含まれるので、一連の可能な Family の組合せを制限する特定の制約があります：家族には、親は 2 人だけだが、子供は 0 ではありません。GOPRR で、これらの制約は、ロールによって取り扱われます。2 つの Parent の必要条件は、すでに 2 つ別々の Parent ロールを備えています。しかし、“子供は 0 ではない” の基数を設定しなければなりません。Roles リストから Child ロールを選択し、右クリックし、[Cardinality] を選びます。図 3-16 のように、Cardinality ダイアログで、Minimum に “0”、Maximum に “N” を設定し、OK ボタンを押します。

図 3-16



Graph Tool は、図 3-17 のようになります。Save and Close ボタンを押し、Graph Tool を閉じます。また、Commit しましょう。

図 3-17



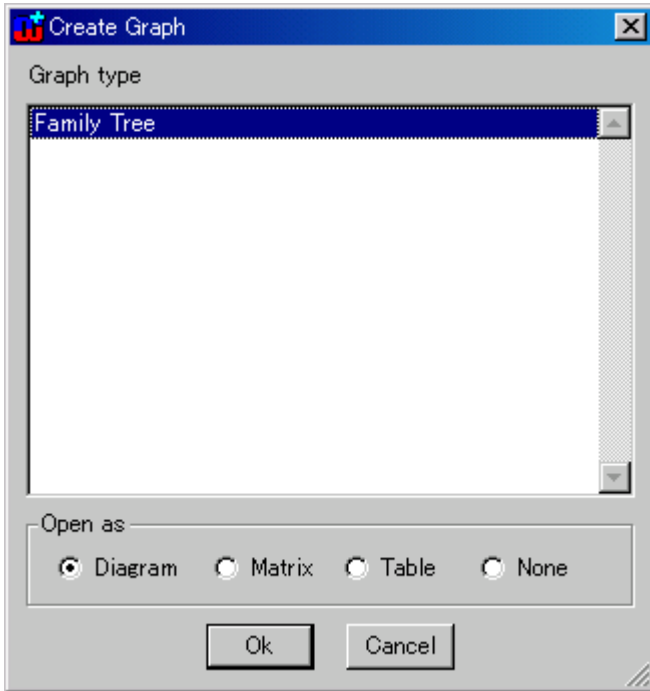
家系図モデリング言語の最初のバージョンが完成しました。

これまで、MetaEdit+のメタモデリングツールを掘り下げてきました。これに加えて、MetaEdit+は、我々が作ったモデリング言語を使って、完全な機能的 CASE 環境も提供します。家系図モデリング言語を試してみることで、CASE ツールの機能性に慣れてみましょう。

### 3.7. 家系図ダイアグラムの作成

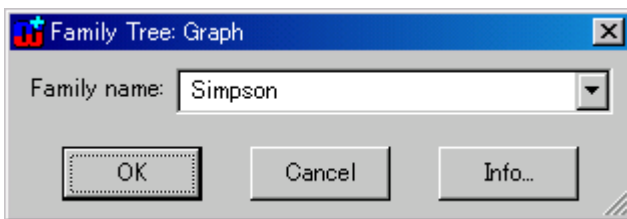
新しい家系図ダイアグラムを作るために、MetaEdit+で Edit->Create Graph を選択します。MetaEdit+が、作ろうとするグラフタイプを選択するよう聞いてくるので、“Family Tree” を選択して OK を押します（図 3-18）。

図 3-18



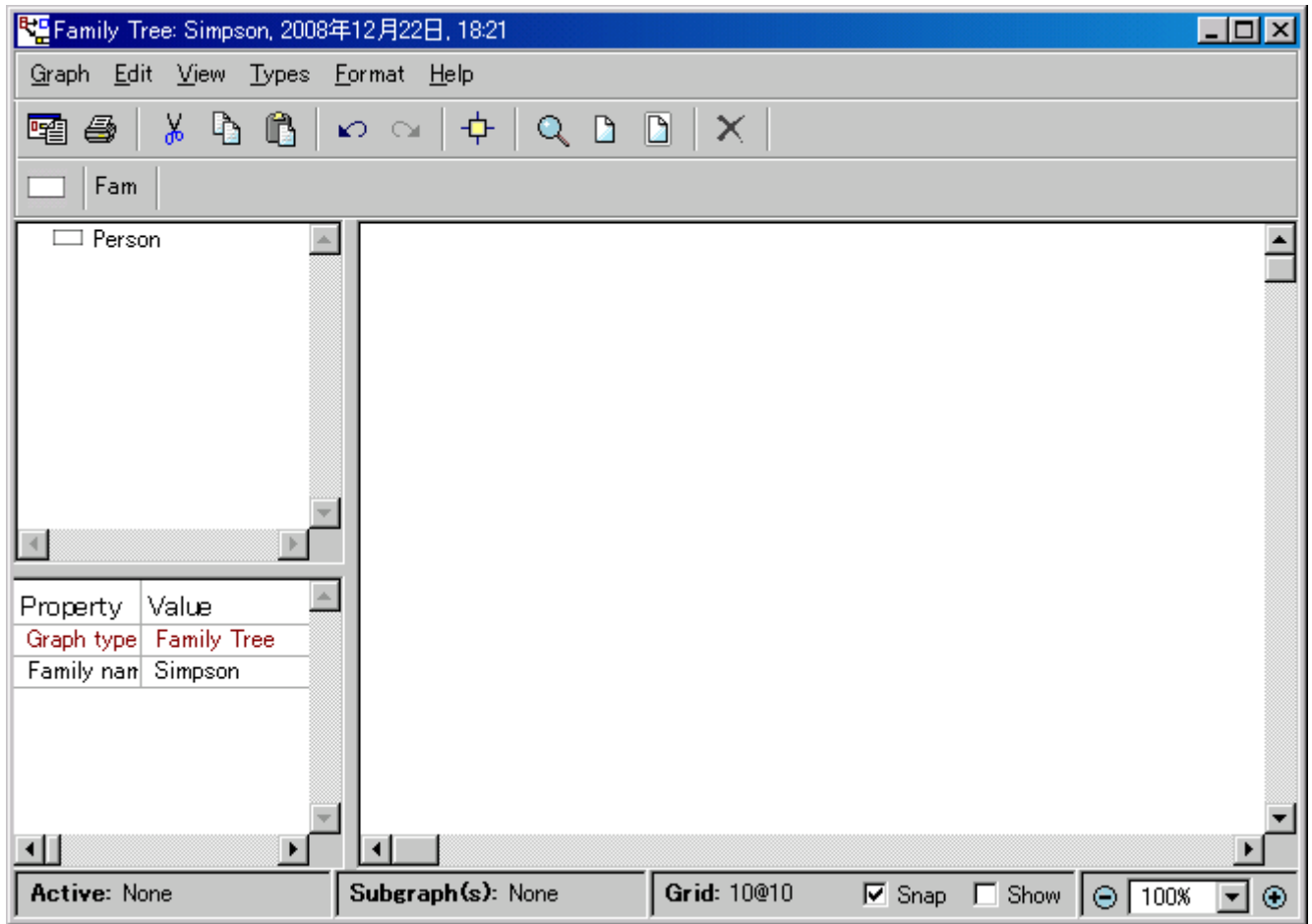
次に、作成する家系図ダイアグラムの家系名を入力するよう聞いてくるので、“Simpson” と入力します（図 3-19）。

図 3-19



OK を押すと、MetaEdit+は、家系図を描くための空のダイアグラムエディタを開きます（図 3-20）。ダイアグラムエディタには、メニューバーがあります。その下に、アクションツールバーがあり、さらにその下に、もう一つのツールバーがあり、そこには使用するモデリング言語のタイプがあります（左側がオブジェクトで、右側にリレーションシップ）。ウィンドウの主要部分は、描画エリアです。グラフでは、左側のバーに、全てのオブジェクトタイプとインスタンスのツリー表示があり、選択された要素のプロパティを表示するプロパティシートもあります。ウィンドウの下部のステータスバーは、選択された要素に関する情報を示し、グリッドやズームの制御があります。

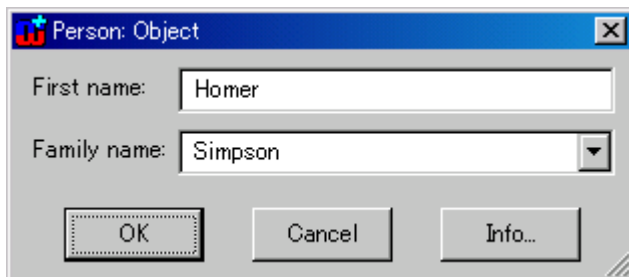
図 3-20



### 3.8. 家系図ダイアグラムにオブジェクトを追加

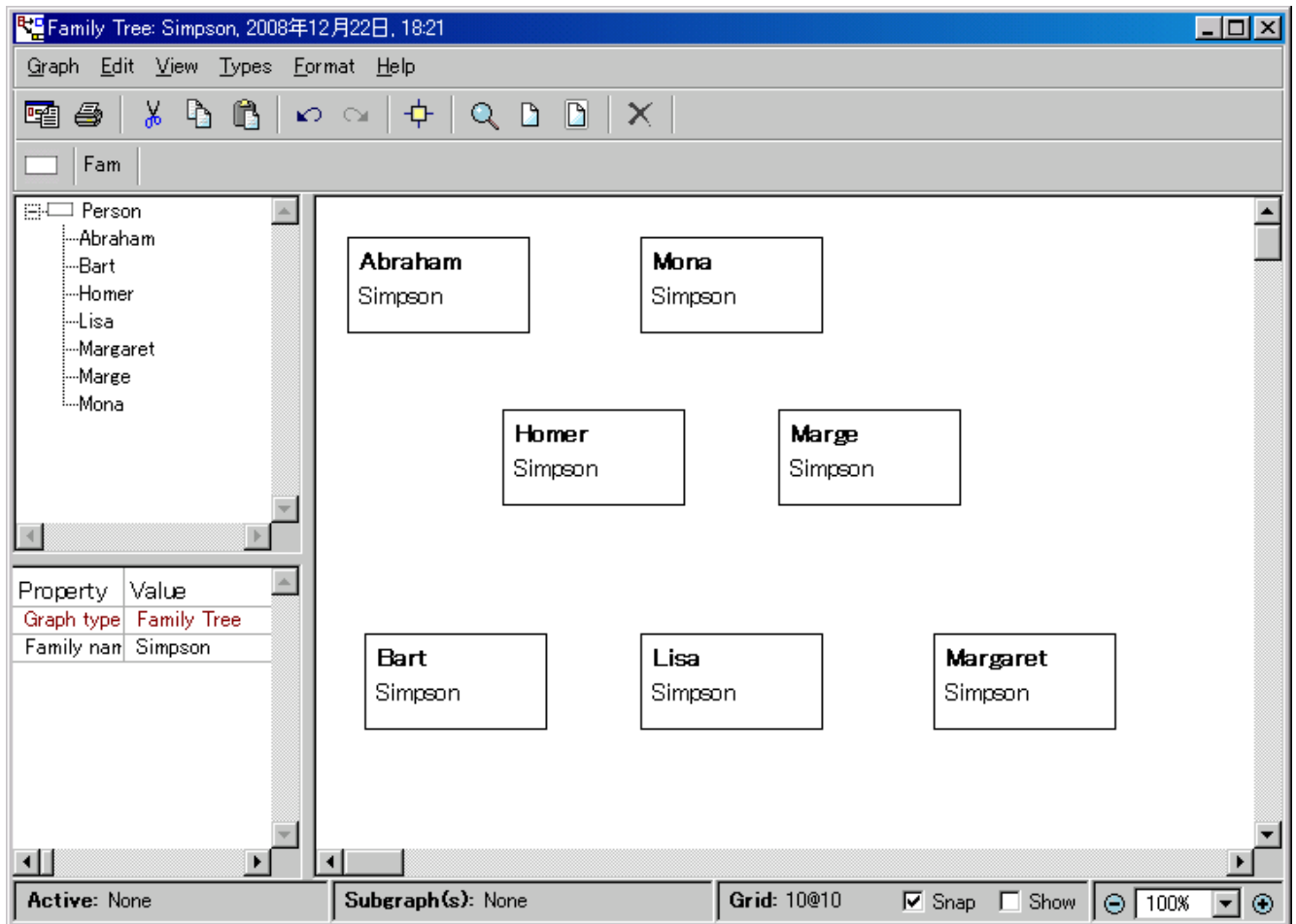
ダイアグラムエディタで、オブジェクトやリレーションシップタイプを含むツールバーの左端のボタンを押します (Person オブジェクトタイプを選択しています)。ダイアグラムの中に Person オブジェクトを配置するため、描画エリア上にマウスで動かし、左クリックします。開いたダイアログで、作成したいオブジェクトの名前 (Homer) を入力します (図 3-21)。OK ボタンをクリックします。

図 3-21



オブジェクトは、ダイアグラムエディタの描画エリアで、そのプロパティを示します。同様に、いくつかの新しいオブジェクトを作成します。シンボルのプロパティ値を変更するには、プロパティダイアログを表示するために、シンボルをダブルクリックします。いくつかのオブジェクトを描き、図 3-22 のようなダイアグラムを作成しましょう。ダイアグラムエディタは以下のようになります。

図 3-22



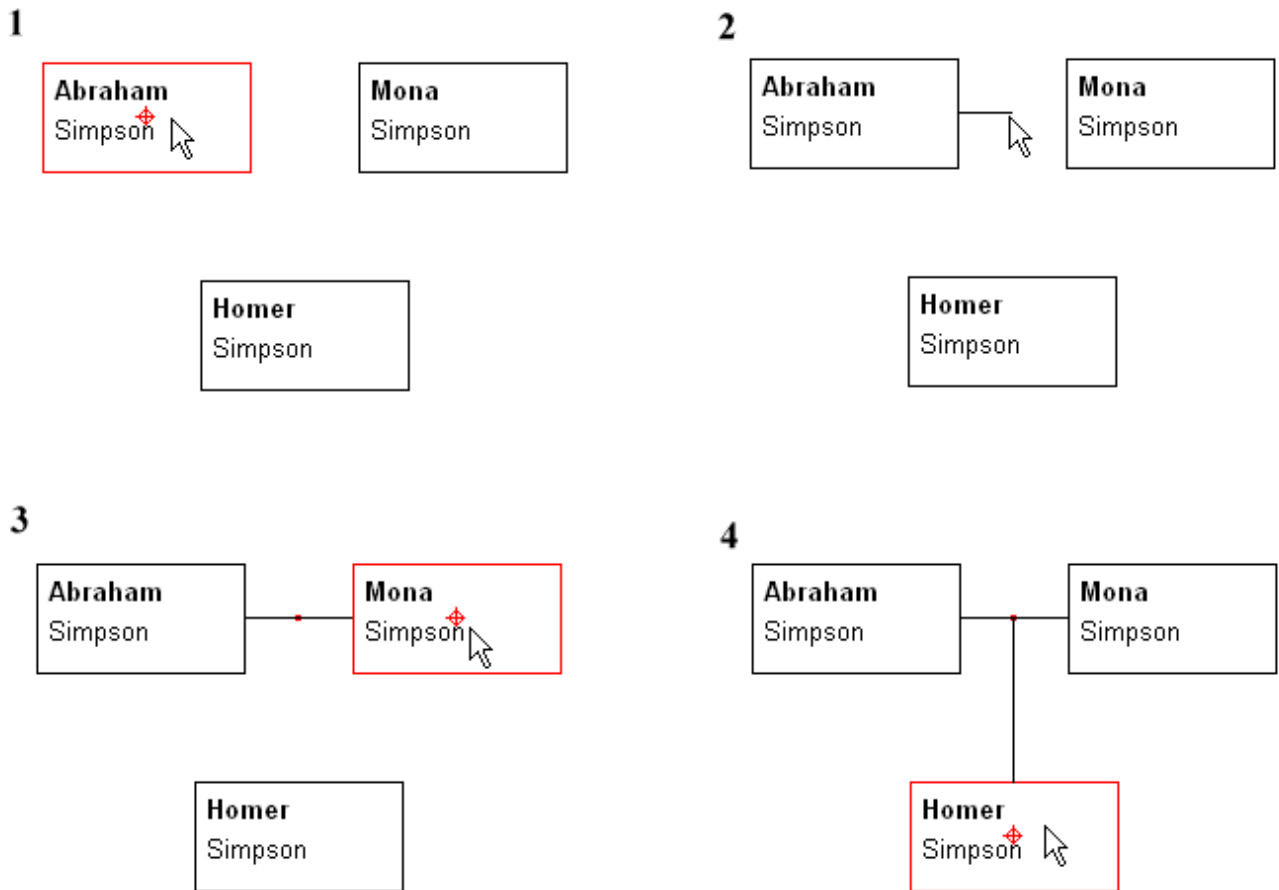
### 3.9. 家系図ダイアグラムにリレーションシップを追加

次に、両親と子供の間に関係を追加します。リレーションシップを描画するのに、2つの方法があります。1つ目は、3つ以上のオブジェクト間のリレーションシップを描きたい時に適用する方法です。2つ目は、2つのオブジェクトだけが関連する時の簡単な方法です。1つ目の方法で、Abraham、Mona と彼らの息子 Homer の家系リレーションシップを作ってみましょう。

リレーションシップ作成中に、ダイアグラムエディタの外にウィンドウフォーカスをすると、操作がキャンセルされることに注意して下さい。

まず、“Family” リレーションシップの作成を行うために、リレーションシップツールバーで、“Fam”(フォントサイズによって表示名は異なります)というラベルが貼られたボタンを押します。最初の Parent ロールである Person オブジェクト (Abraham) の上にカーソルを動かすと、接続可能でハイライトされるので、左クリックします (図 3-23 の step1)。それから、リレーションシップの真ん中をセットするために、両親間で再びクリックします (step2)。次に、もう片方の親 (Mona) をクリックし (step3)、それから子供 (Homer) をダブルクリックします (step4)。選択されたオブジェクト間のリレーションシップがダイアグラムエディタで表示されています。

図 3-23



リレーションシップを作成する最初の方法を要約すると、4 回クリックしました - まず親 (Abraham)、次にもう片方の親 (Mona) までの空きスペース、次にもう片方の親 (Mona)、最後に子供 (Homer) でダブルクリック (Homer をクリックして、右クリックでもリレーションシップの関係を終わることはできます)。

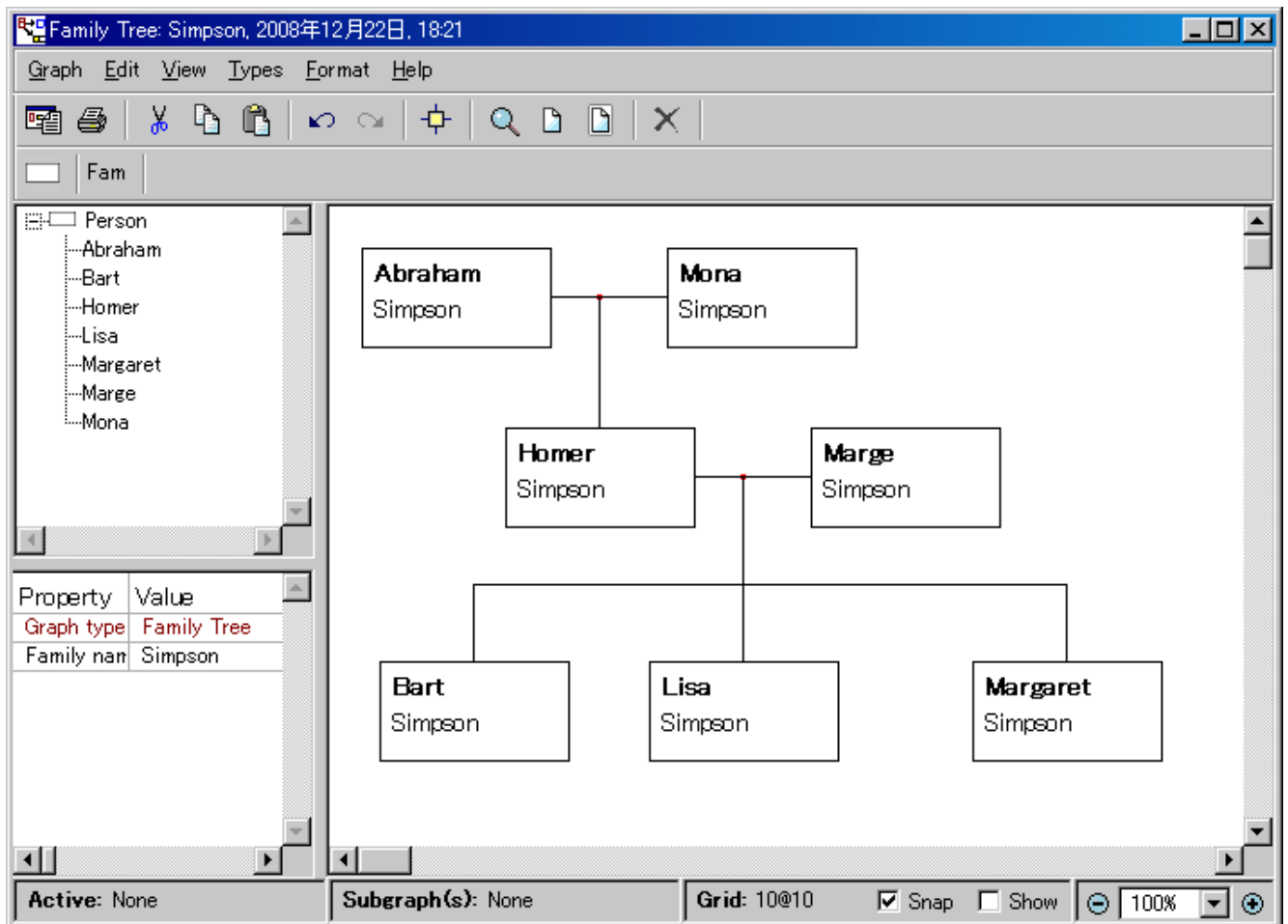
2 つ目の方法は、2 つのオブジェクトだけが関連するリレーションシップを描きたい時に使われます。ここで良い例は、子供のいない家系です。例において、まるで結婚しているかのように、Homer と Margeをつないでみましょう。再び、リレーションシップツールバーで、“Fam” ボタンを押すことで、“Family” リレーションシップの作成を開始します。リレーションシップの最初のオブジェクト (Homer) にカーソルを移動します。左マウスボタンを押して、もう一方のオブジェクト (Marge) までカーソルをドラッグし、ボタンを離します。オブジェクト間の 2 項関係が表示されます。

後で、子供をこの関係に追加することもできます。ちょうど今描いたリレーションシップを見ます。小さな赤い点が、線の中央にあります。その点をクリックすることで、リレーションシップが選択できます。選択ハンドルは、リレーションシップの点の上に現れるべきです。もし、リレーションシップ線の両端にそのようなハンドルが見えるなら、リレーションシップではなくロールを選択したことを意味します。 - この場合、ロールを選択から外して、再度リレーションシップを選択して下さい。

右マウスボタンを押して、ポップアップメニューから [Add a New Role] を選択します。Child ロール (例えば、Lisa) につなげたいオブジェクト上にカーソルを移動し、左マウスボタンをクリックします。ロールがとる経路を定義するために、オブジェクトの途中の空きスペースでもクリックできます。例えば、オブジェクト Bart へのリレーションシップを追加するために、[Add a New Role] を選択し、Lisa の上、Bart の上でクリックします。線を選択し、それをクリックすることで、後でロール線に区切り点を追加することができます。それから、適切な位置に新しく作られた区切り点をドラッグすることで、後で区切り点をロール線に加えることもできます。

図 3-24 のダイアグラムのように、Margaret にロールを加えることで、家系図ダイアグラムを完成することができます。完成したら作業を Commit しましょう。

図 3-24



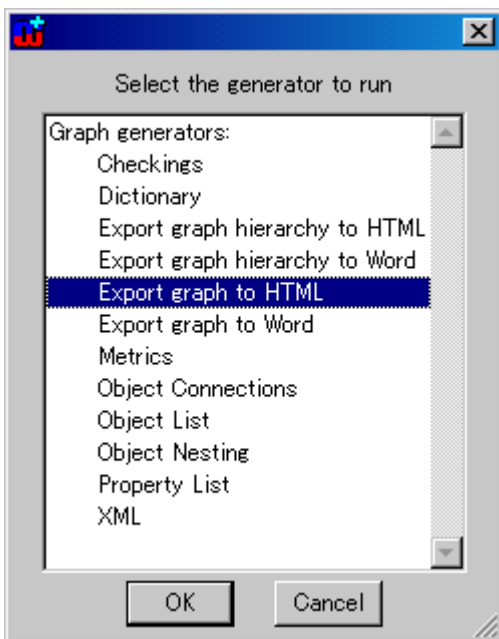
### 3.10. 家系図ダイアグラムからレポートを生成

家系図を形に表すためにグラフィカル言語を作成し、それを使って最初の家系図ダイアグラムを描くことができました。他にできることはあるか？

多くの場合、家系図ダイアグラムに格納される様々な情報の出力を生成できることが望まれます（例：ウェブページや誰が登録されているかなど）。これは、MetaEdit+のレポートジェネレータで実現させることができます。例えば、家系図ダイアグラムから、ウェブページを生成してみましょう。

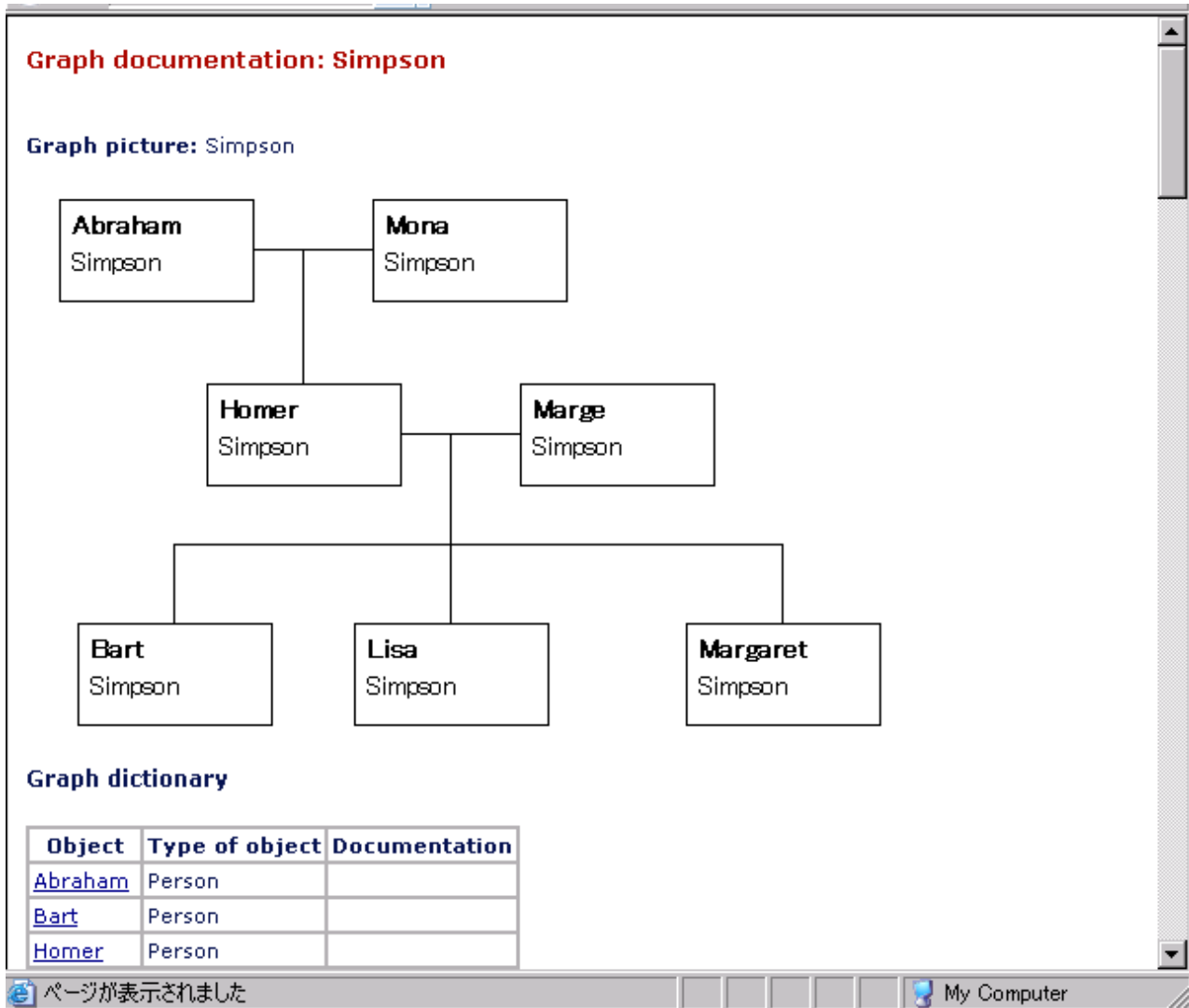
“Simpson”家系図のダイアグラムエディタのメニューバーから、Graph->Generate を選択します（もしくは、ダイアグラムエディタのアクションツールバーの Generate ボタンを押す）。あらかじめ定義された利用可能なレポートの一覧が現れますので、“Export graph to HTML”を選んでOKボタンを押します（図3-25）。

図 3-25



MetaEdit+は、HTML の出力を生成し、デフォルトブラウザで生成されたウェブページを開きます（図 3-26）。

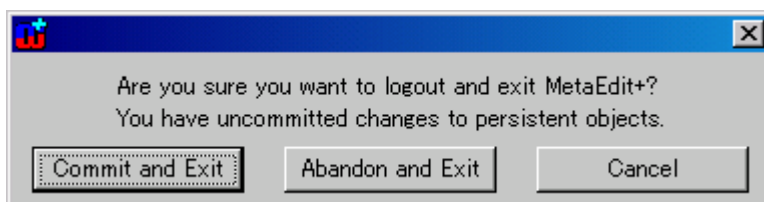
図 3-26



### 3.11. MetaEdit+の終了

家系図モデリング言語の開発を通して、コンセプトの実証を得ることができました。現時点で、このチュートリアルを中止する場合、MetaEdit+の Repository ->Exit を選択することで MetaEdit+を終了できます。MetaEdit+は、終了する前に確認を求めてきます(図 3-27)。最新の変更を保存するかどうかに応じて、“Commit and Exit” または “Abandon and Exit” を選んでください。

図 3-27



家系図モデリング言語のさらなる開発、進化のアイデアに興味があるなら、もう一度 MetaEdit+を起動して、次の章を続けてください。

#### 4 . 家系図モデリング言語の改良

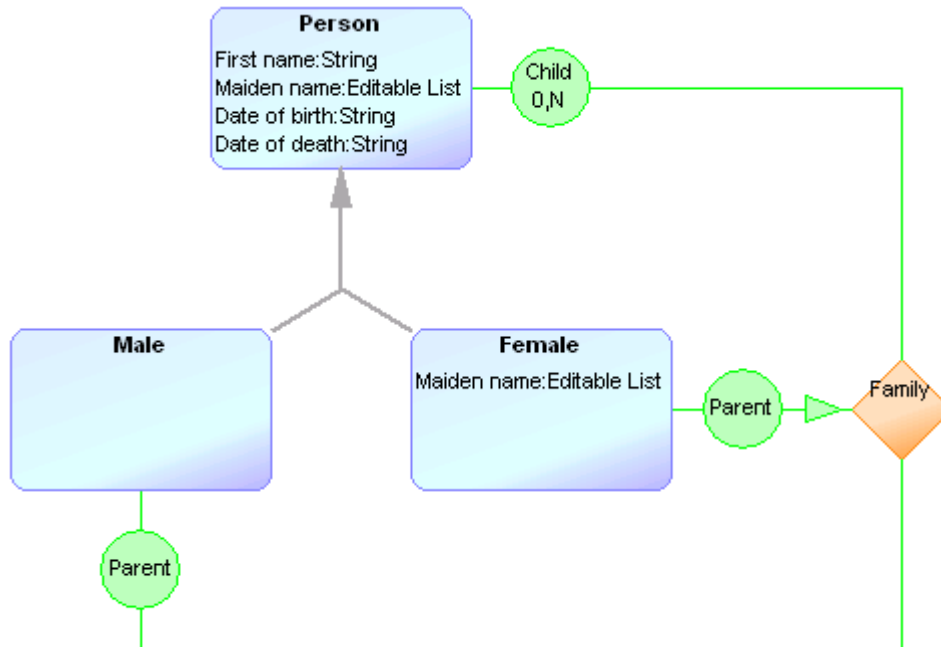
家系図モデリング言語の最初のバージョンをうまくデザインし、実行し、検証しました。しかしこれは、まだ始まったばかりです。通常この段階で、課題領域 ( problem domain ) に対する変更、少なくともモデリング言語を手直しする必要があることについて学ぶことができます。家系図モデリング言語に対して、いくつかの改善を考えてみましょう。そして、MetaEdit+によって提供される、より高度なメタモデリングやモデリングコンセプトに親しんでください。

##### 4.1. Person の男女関係

家系図言語では、First name と Family name のプロパティを使った Person のコンセプトがあります。しかし、ほとんどの場合、各人物に関してより多くの情報を格納したいものです。ほとんどの用途に対して、人物の誕生日や死亡日を知ることは望ましいでしょう。さらに、女性が結婚すると、名字が変わるかも知れないので、女性の旧姓を分けて保管することは、役に立ちます。ダイアグラムにおいて、男女に対して異なるシンボルを持ちたいでしょう。

明らかにこの全ては、男女のコンセプトの中に、Person コンセプトの特殊化を必要とします。しかし、これら 2 つの新しいコンセプトが、Person のきちんとしたサブタイプのように、メタモデルでもこの意味論の事実を示したい。このように、全ての共有資産を人物のコンセプトに割当て、そのサブタイプとして性別を定義し、追加のプロパティに性別を加えます。図 4-1 は、これらの変更を使った家系図モデリング言語メタモデルの新しいバージョンを示しています。

図 4-1

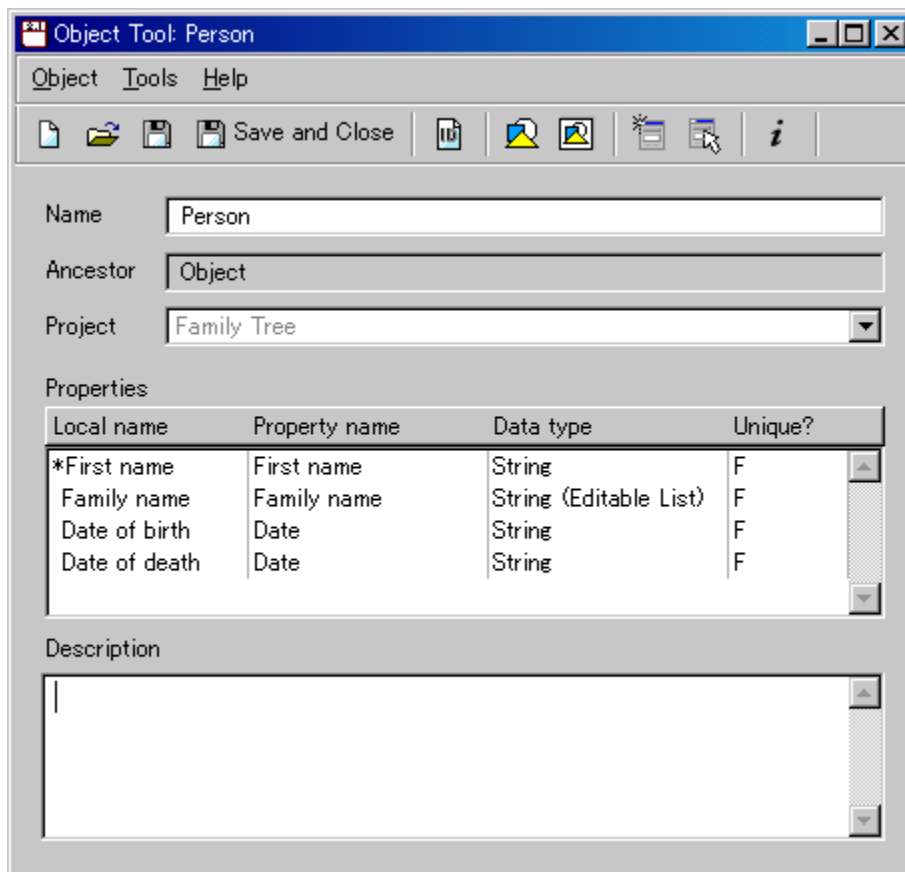


さて、MetaEdit+のメタモデルに同様の変更を行ってみましょう。Object Tool ボタンを押すか、MetaEdit+の Metamodel->Object Tool を選択することで、Object Tool を起動します。Object Tool で、ツールバーの Open ボタンを押すか、Object->Open を選択します。“ Person ” が、現在定義されている唯一のオブジェクトタイプであれば、すぐに開きます。そうでなければ、リストダイアログから選んで下さい。Object Tool

には Person に対する定義があります。Properties リストで、新しいプロパティタイプを作ります。Property Tool では、名前として“ Date ”を入力します。デフォルトの Data type は“ String ”、Widget は“ Input field ”で、Save and Close を押して Object Tool に戻ります。プロパティリストから新しく生成した“ Date ”プロパティを選択し、右マウスクリックでポップアップメニューを開きます。[Local name]を選択し、ダイアログに“ Date of birth ”を入力して OK を押します。

Local name は、再利用のためのメカニズムとして、ここで使用されました：同じプロパティには、様々な目的に対して、より適したものにするために、それぞれの Local name を持ちます。これは、“ Date of death ”プロパティを作成することで、さらに明確になります。Date のプロパティをすでに定義しているので、完全に新しいプロパティタイプを作る必要はありません。しかし、既存のものを再利用することはできます。まず、プロパティリストでプロパティが選択されていないことを確認し（新しいプロパティは最後に追加される）、Property リストポップアップメニューから [Add Property] を選択します。この時、“ New Property Type ”の代わりに、リストから“ Date ”を選択します。“ Date of birth ”と同様に、その Local name を“ Date of death ”に変更します。図 4-2 のようになります。

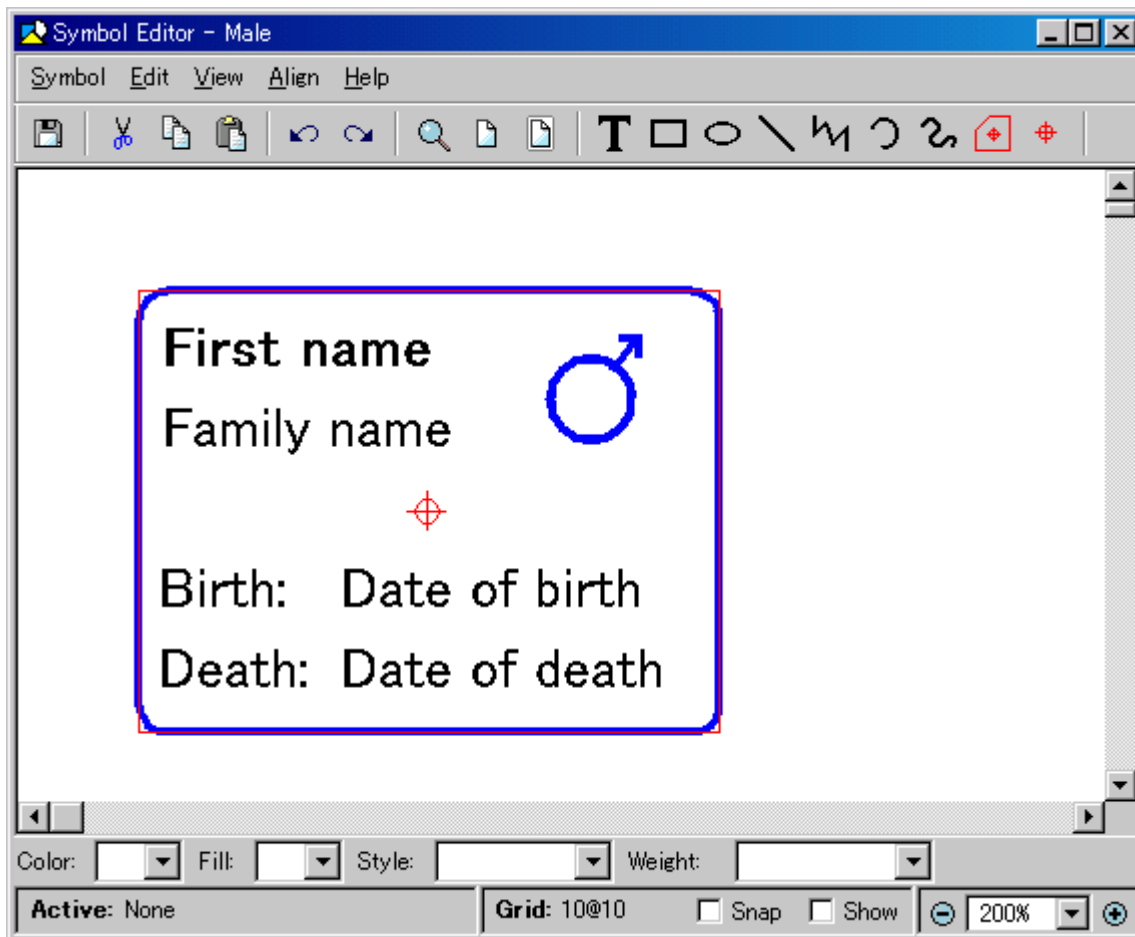
図 4-2



変更を承認するために Save ボタンを押します。次に行うことは、Person のサブタイプとして新しい性別タイプ (Male と Female) を作成することです。Object Tool で、メニューから Object->New Subtype を選択し、リストから“ Person ”を選択します (Object Tool の Ancestor に“ Person ”が設定)。Person オブジェクトからプロパティを継承し (Properties が赤字になる)、新しいオブジェクトタイプを作成しました。Male のコンセプトは、実際に Person コンセプトに追加されず、この新しい Object Tool で名前として“ Male ”を入力することで、Male オブジェクトを作成できます。入力したら Save を押します。

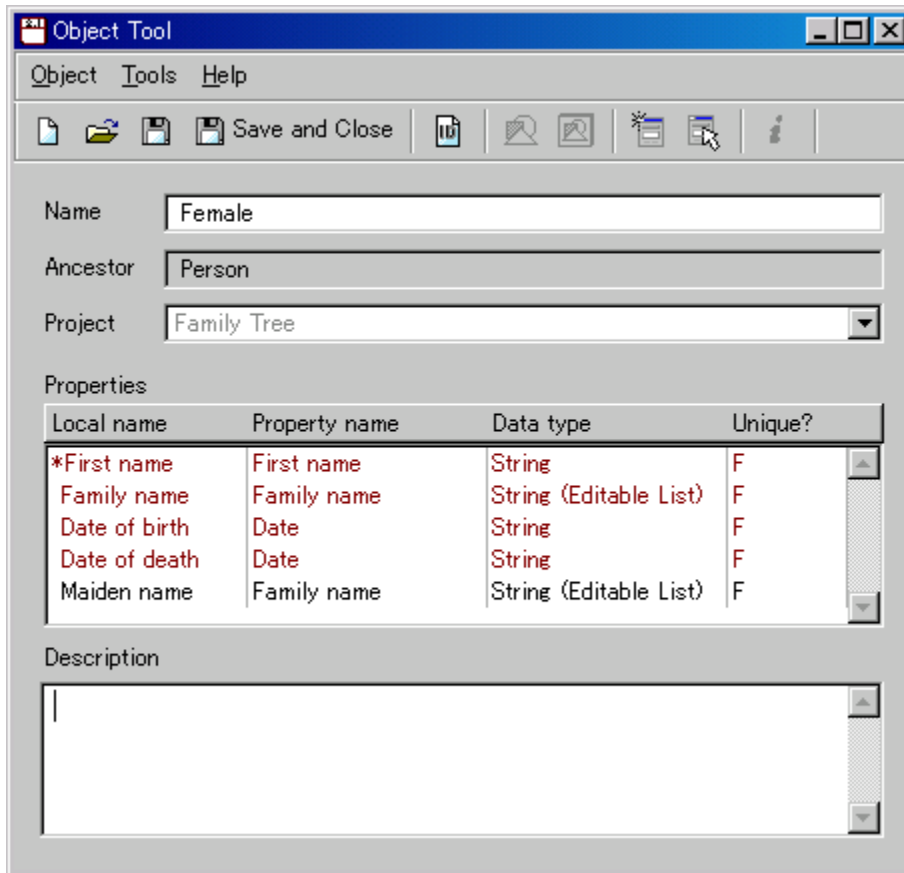
Male は、ダイアグラムで使用されるコンセプトで、グラフィカルシンボルを定義する必要があります。Symbol Editor を起動し、シンボルライブラリから “Male” シンボルを読み込み、“Person” シンボルで行ったように、プロパティに名付けられた 4 つのテキストフィールド (“Birth: ”、“Death: ” を除く) に対して、固定されたテキストの代わりに、実際のプロパティを使用するよう修正します (3.2. シンボルの作成を参照)。シンボルを保存し、Symbol Editor を閉じます。

図 4-3



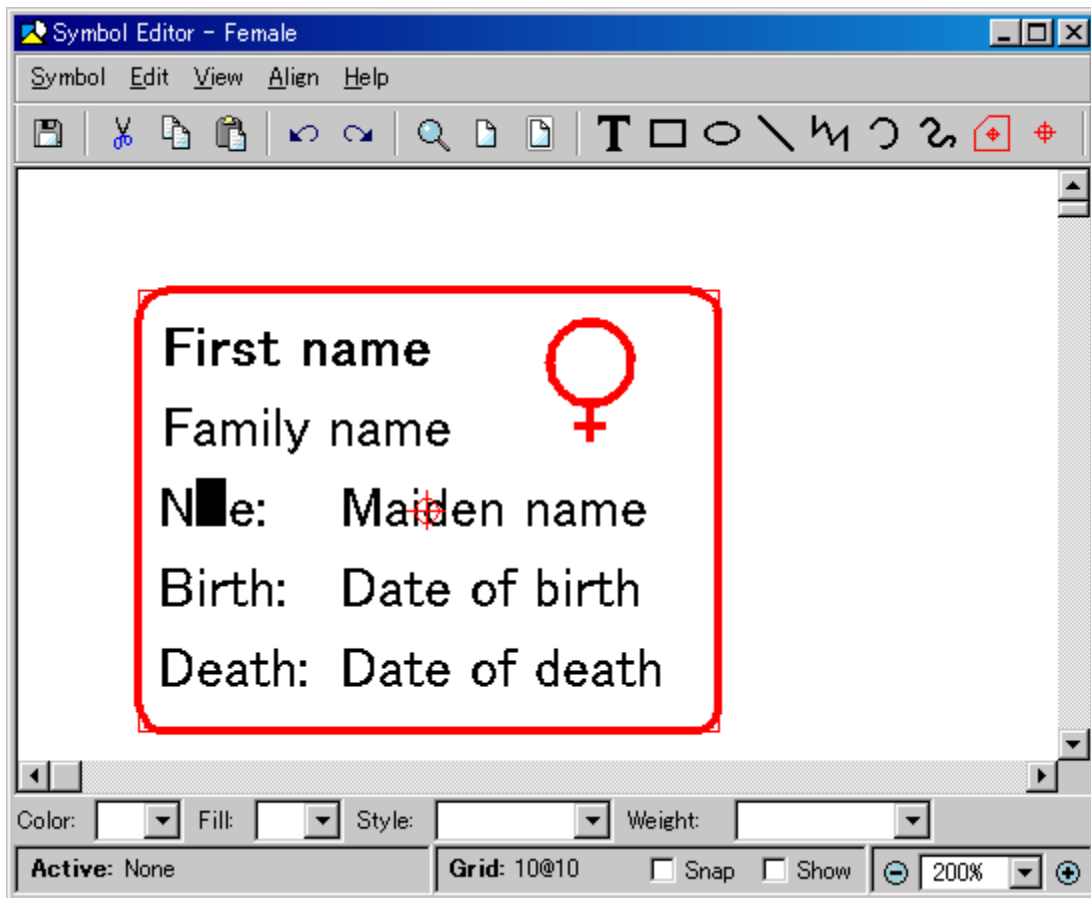
“Female” オブジェクトを作るには、もう少し作業が必要です。もう一度、Person のサブタイプとして新しいタイプを作るところから始めましょう (Object Tool->New Subtype を選択して、“Person” を選択)。この名前として “Female” を入力します。そして、特別なプロパティ (“Family name” タイプを再利用する) を追加し、“Maiden name”(旧姓) という Local name を与えます。そして、Female オブジェクトの Object Tool は、図 4-4 のようになります。

図 4-4



Save を押して、Male と同様に Female オブジェクトのシンボルを作成します。シンボルライブラリから “Female” シンボルを読み込み、5 つの適切なテキストフィールド (“Nee: ”、“Birth: ”、“Death: ”を除く) に対して実際のプロパティを割当てます。シンボルを保存します( 図 4-15)。Symbol Editor と Object Tool を閉じます。

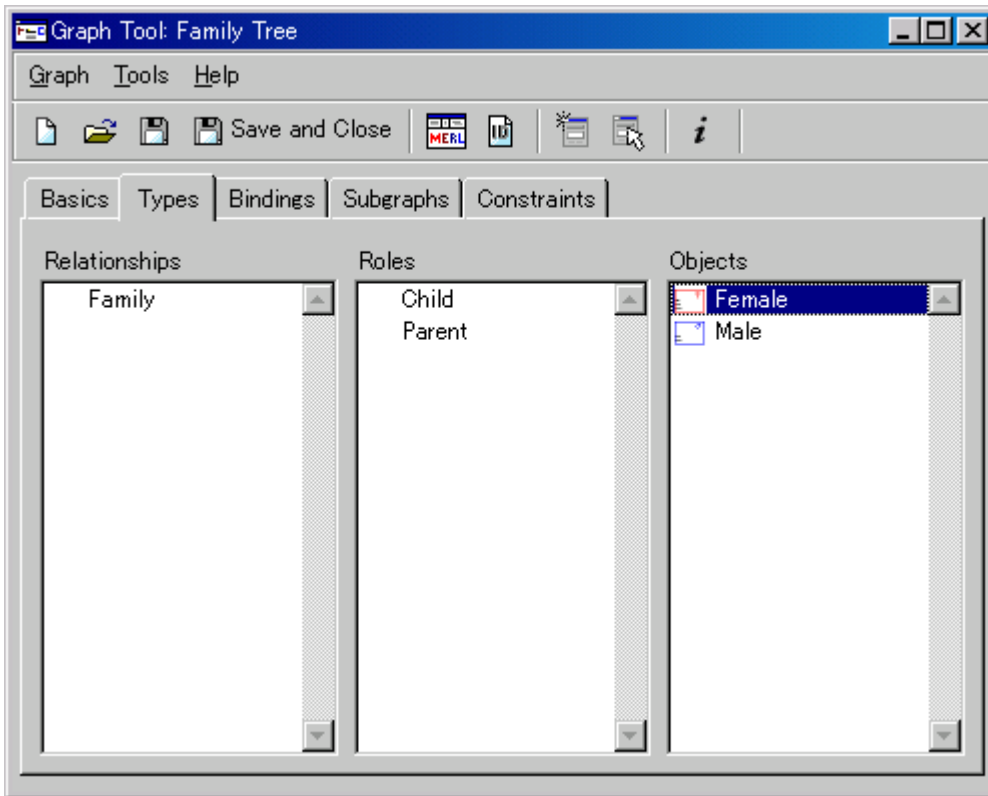
図 4-5



作業を Commit します。ツールバーの Graph Tool をクリックするか、MetaEdit+の Metamodel->Graph Tool を選択することで、Graph Tool を起動します。Graph Tool で、Open ボタンを押します。“Family Tree”だけが定義されており、自動的に読み込まれます。しかし、もし他にも定義があるなら、その開いたオプションから “Family Tree” を選んでください。

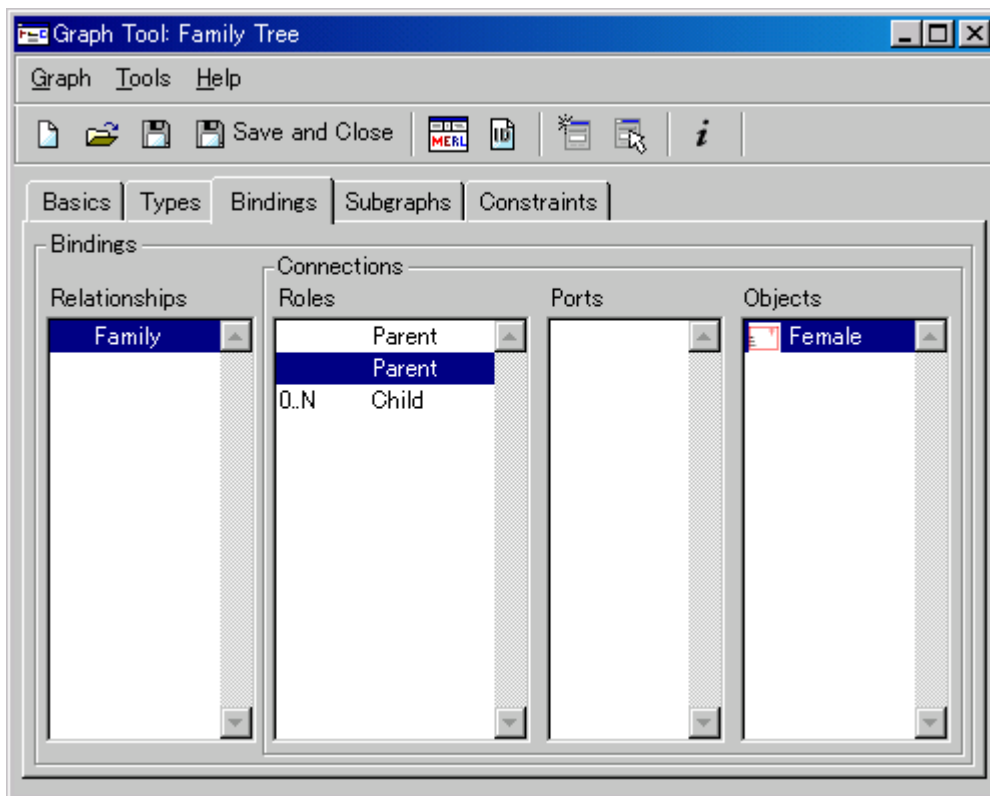
Types タブの Objects リストで、“Person” を選択し、右マウスボタンをクリックしてポップアップメニューから [Delete] を選択します。再度、右マウスボタンをクリックし、[Add] を選んで “Male” を選択します。同様に “Female” も追加します (図 4-6)。

図 4-6



Bindings タブで、Roles リストから最初の Parent ロールを選びます。Objects リストで、“Person” を削除して、そこに “Male” を追加します。同様にして、2 つ目の Parent ロールを選び、“Person” オブジェクトを削除して、“Female” を追加します（図 4-7）。

図 4-7



Child ロールについては、Person オブジェクトへの接続をそのままにしておいてください。Person が実際に一連の家系図モデリング言語から削除される間、その参照はなく、ここで使うことができます。上位タイプとしての Person は、Male と Female 両方の使用を許容しています。

変更を承認するために Save and Close を押し、Graph Tool を閉じます。モデリング言語の改良による変更が行われたので、再びそれを試すことができます。この変更がかなり急進的だったので、全ての古い例を削除して、新しいものを作らなければなりません：古いオブジェクトは、全て Person タイプでしたが、代わりにそれらを “Male” と “Female” にしましょう。

#### 4.2. Port (ポート) を使った描画精度の向上

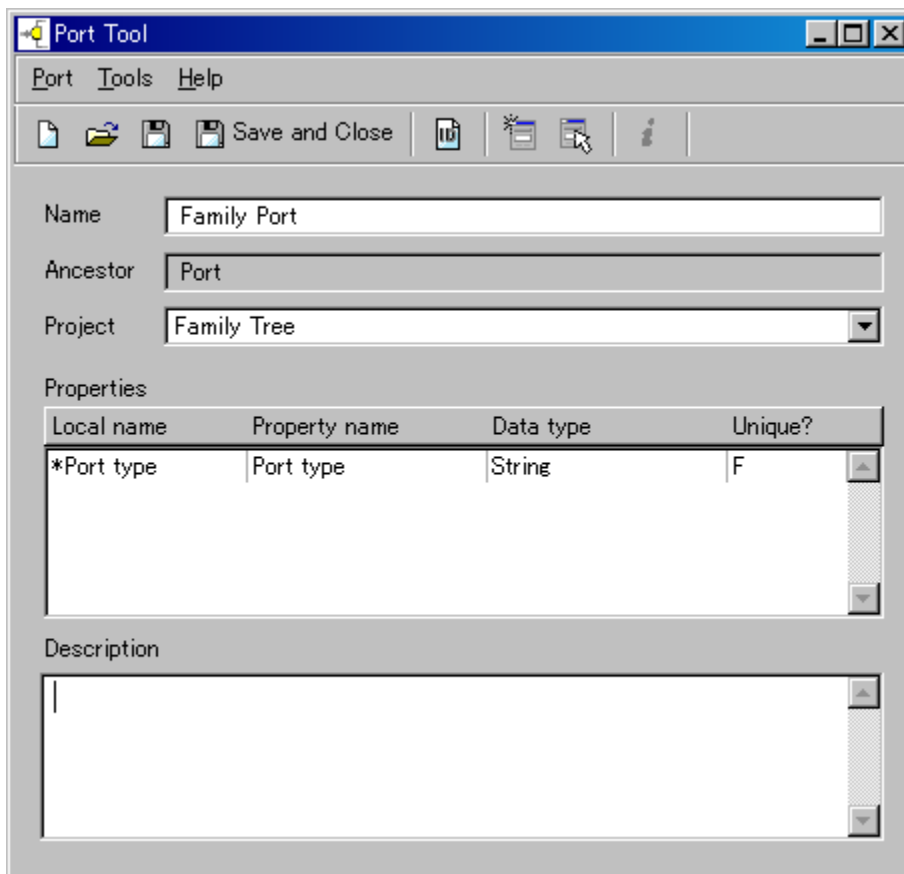
Object、RelationshipやRoleは、様々なモデル要素間で意味論（セマンティックス）となる、かなり強力なメカニズムです。しかし時々、さらに正確な意味論または振る舞いルールが、バインディングに必要です。この目的に対して、GOPRRRメタモデリング言語は、Portのコンセプトを定義します。Portは、Roleが接続されるObjectの一部として考えられるコンセプトです。Portを使って、RoleとObjectがどのように接続されるかの、追加の情報や制約を加えることができます

Port は、たいてい概念上のレベルでの使用を目的とし、進化した描画構造に対して利用されます。例えば、家系図言語のある視覚化関連要件は、各 Child ロールが、“Person” シンボルの一番上にだけ接続され、各 Parent ロールは、“Person” シンボルの横（男性は右側、女性は左側）にだけ接続されなければならない

ということです。これは、左側の父親と右側の母親間で、両親として線を引き、その線の間接点から子供への下向き線を引く方法で、ダイアグラムを描くための制約です(図 1-1)。現在のバインディング定義で、そのような正確さはない為、どういふ追加が可能か見てみましょう。

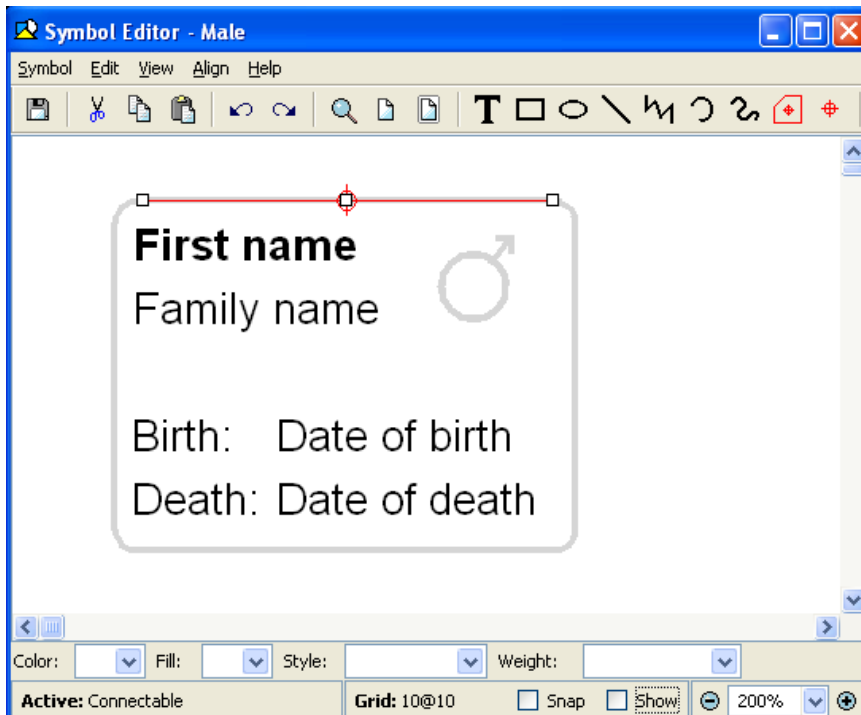
行ふべきことは、シンボルに対して2つの新しい connectable を定義し、それらにそれぞれ Parent と Child のポートを与えます。まず始めに、新しいポートタイプを定義しなければなりません。ツールバーの Port Tool ボタンをクリックするか、MetaEdit+ツールの Metamodel->Port Tool を選択することで、Port Tool を起動します。一番上のフィールド (Name) に “Family Port” を入力し、“Port type” と呼ばれる新しいプロパティタイプを追加します (Data type は “String”、Widget は “Input Field”)。図 4-8 のようになります。Save and Close を押して、Port Tool を閉じます。

図 4-8



Object Tool を開き、“Male” オブジェクトのシンボリエディタを開きます。簡単に新しい connectable を描くために、シンボルの中央にある赤い十字線を選択して右クリックし、ポップアップメニューから [Delete] を選択することで、このデフォルトを削除します。ツールバーから Connectable ボタン (右にある十字線を持った赤い多角形) をクリックします。図 4-9 のように、シンボルの左上隅にカーソルを動かして、左マウスボタンをクリックし、カーソルを右上隅の方へ動かして、左マウスボタンをダブルクリックします。新しい connectable が図 4-9 のようになります (シンボルの青い部分は、新しい connectable を強調するために、図 4-9 ではライトグレイとして表示しています)。

図 4-9



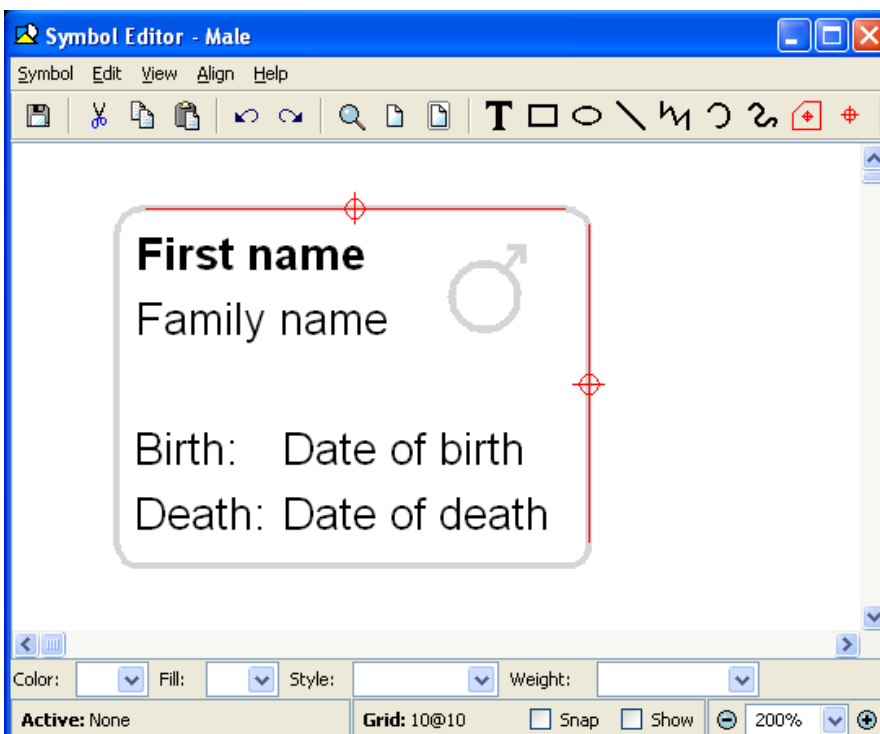
新しい connectable が選択された状態で、右クリックしてポップアップメニューから [Format] を選択します。Port(s) リストで右クリックし、ポップアップメニューから [Add] を選びます。Port type フィールドで “Child” を入力し OK を押します。Child ポートは、図 4-10 のように Port(s) リストに表示されます。OK を押して、Format ダイアログを閉じます。

図 4-10



同様の方法で、シンボルの右側に connectable を作成し、“Parent” と呼ばれる新しい FamilyPort を与えます。シンボル定義は、図 4-11 のようになります。シンボルを保存し、シンボリエディタを閉じます。

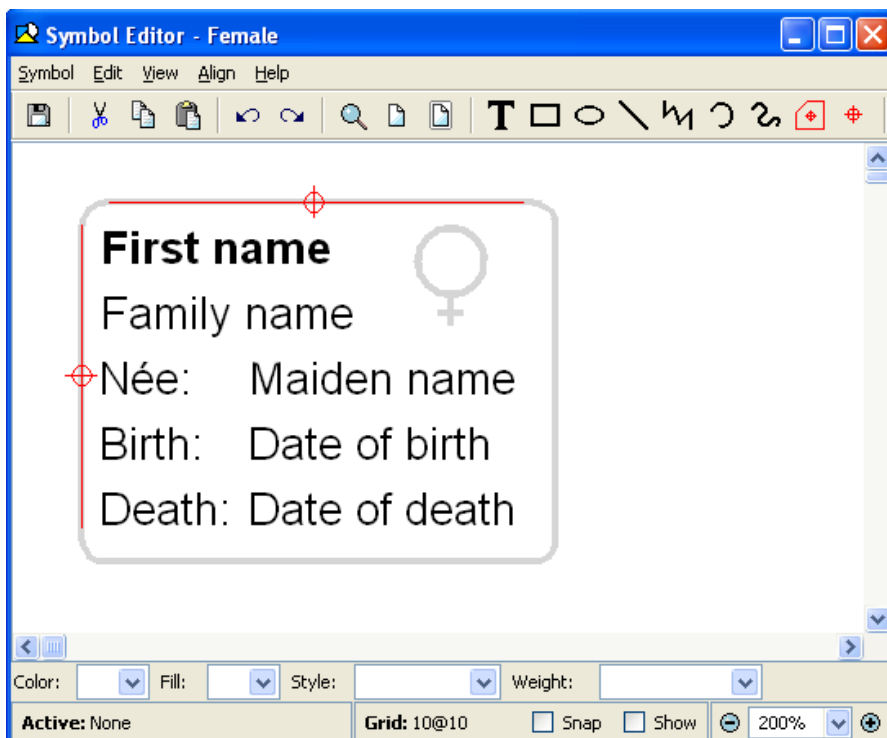
図 4-11



次に、“Female” オブジェクトのシンボリエディタを開きます。“Male” シンボルと同様に、一番上に沿って connectable を作成しますが、新しいポートを作ってはいけません。その代わりに、connectable フォ

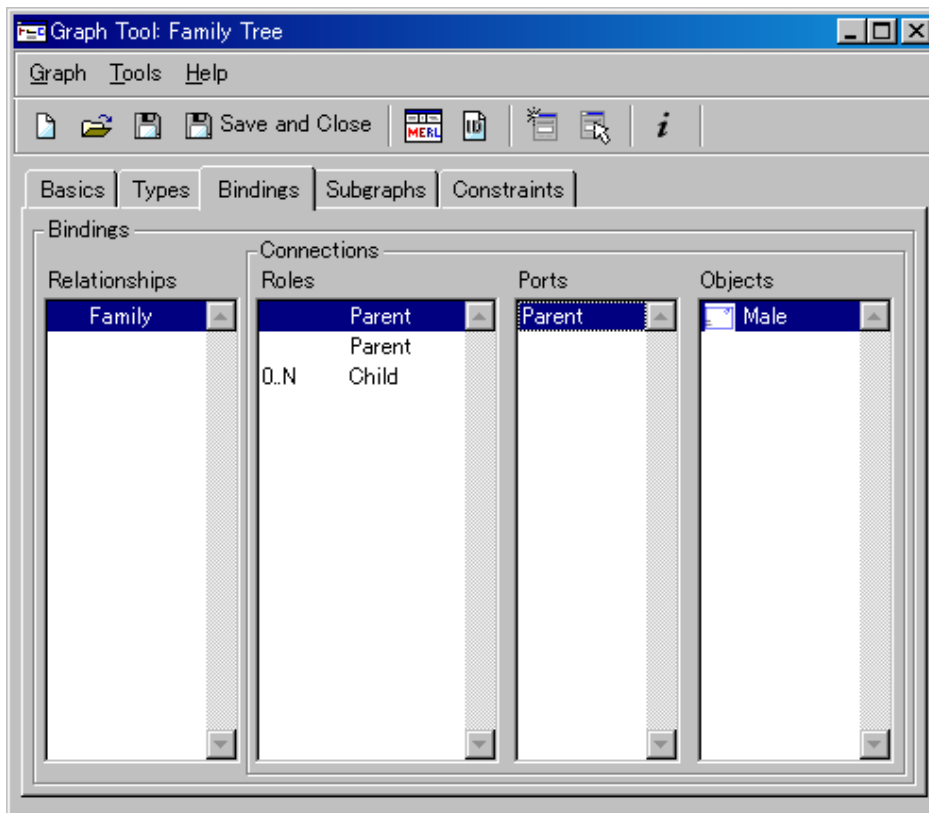
ーマットダイアログのリストのポップアップメニューから [Add Existing] を選択し、既存の Child ポートを追加します。次に、Male と同様に横側に新しい connectable を作成しますが、この時、シンボルの左側に沿って作成し、既存の Parent ポートを与えます。新しいものを作るよりむしろ既存のポートを使うことが重要です。そうしないと、これらのオブジェクト間のリレーションシップを確立することができません。“Female” オブジェクトのシンボルは、図 4-12 のようになります。

図 4-12



シンボルにこれらの追加を作成した後、“Family Tree” グラフタイプの Graph Tool を開き、Bindings タブを開きます。Roles リストから最初の Parent を選び、Ports リストで右クリックし、ポップアップメニューから [Add] を選択します。そのリストダイアログで、“Family Port” タイプをダブルクリックし、次のダイアログから “Parent: Family Port” を選択します。Graph Tool は、図 4-13 のようになります。

図 4-13

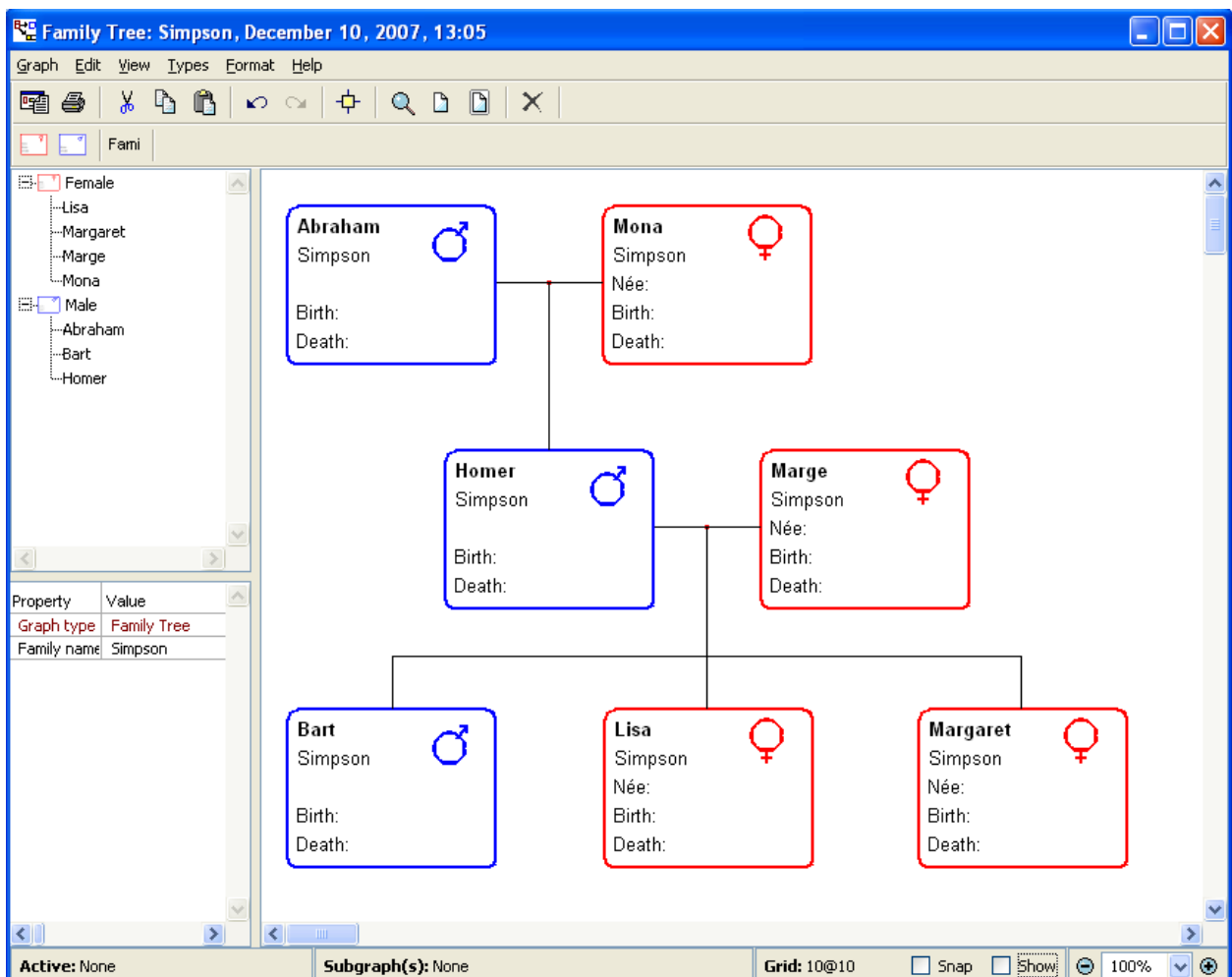


同様に、もう一方の Parent ロールに Parent ポートを、Child ロールに Child ポートを設定します。Graph Tool で Save ボタンを押し、グラフタイプの変更を承認します。

家系図モデリング言語のこの最新バージョンを試してみましょう。前に作ったリレーションシップを削除

し、図 4-14 のように再作成しましょう。リレーションシップを作成する間にオブジェクトをクリックする時、そのポートは、その人物の connectable 十字線がカーソルに一番近いところを選択します。正しいポートが選択されることを確認するために、父親の右側、母親の左側、子供の一番上の近くをクリックします。前とは異なり、関連するオブジェクトを移動すると、ロール線が自由に移動し、全ての Child ロールは、シンボルの一番上に接続されたままであることに気付くでしょう。同様に、Parent ロールは、親シンボルの左 or 右側から離れません。

図 4-14



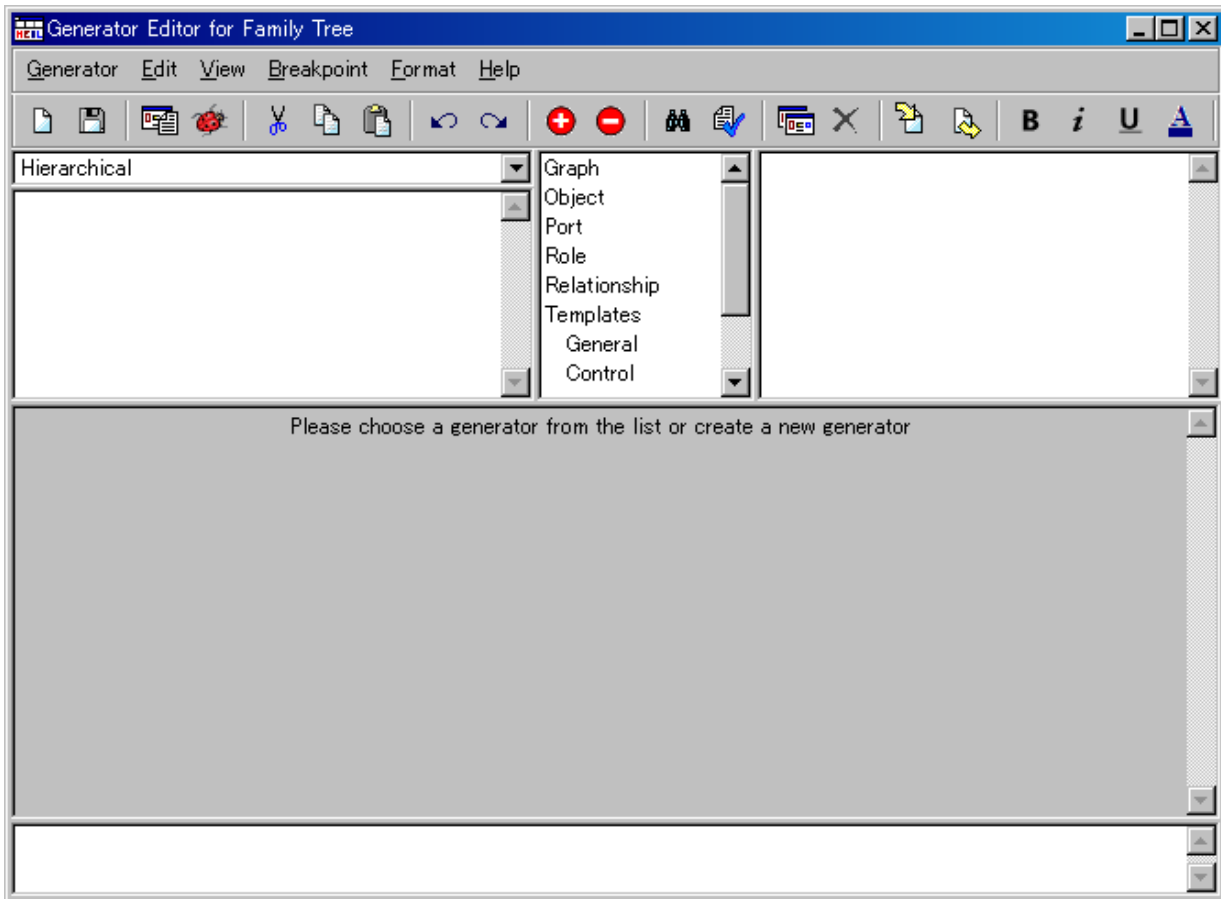
グリッドを使っているが、ロール線が正確に水平または垂直でないなら、connectable の十字線の位置に問題があるからでしょう。これらは、ロール線ポイントへの目標点を形成します。例えば、シンボルの真ん中にデフォルト connectable の十字線によって示されるように、Child の十字線がシンボルの真ん中になら、Child ポートへのロール線は、必ずしも垂直とは限りません。シンボルエディタで、十字線の位置を修正することができます。シンボルの真ん中で十字線を選び、Shift キーを押しながら Child connectable の十字線をクリックします。ツールバーの Align->Align Center を選択します。Parent connectable は、同様に修正することができますが、Align->Align Middle を選択します。

#### 4.3. ジェネレータの作成

以前、家系図のレポートを出力するために、あらかじめ定義されたジェネレータを使用しました。しかし、モデリング言語を最大限に活用するために、新しいドメインスペシフィックジェネレータを作ることができます。自身のジェネレータを作成する方法について、掘り下げて行きましょう。メタモデルを作成するのに、このチュートリアルで示されるメタタイプと異なる名前を使ったなら (“First name” ではなく “Given name” のような)、これらのサンプルジェネレータでも、自身の命名規則を使用しなければならないことに注意してください。

4.2. で作成したダイアグラムエディタにおいて、ジェネレータエディタを開くために、Graph->Edit Generators を選びます。図 4-15 で示す Generator Editor は、ジェネレータを作成、保持、実行するためのツールです。

図 4-15

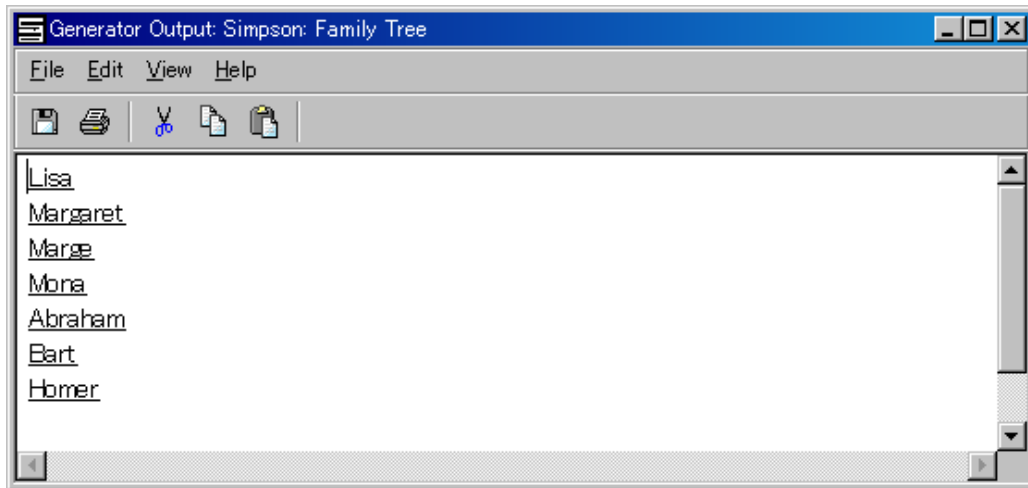


何も書いていない新しいジェネレータ定義を作成するために、ツールバーで New を押すか、Generator Editor で Generator->New を選択し、新しい定義名として “My HTML generator ” を入力します。MetaEdit+ は、編集エリアで基本的な定義テンプレートを表示します。それを以下のように修正します。

```
-----  
Report 'My HTML generator'  
  
foreach .()  
{  
    id  
    newline  
}  
  
endreport  
-----
```

ツールバーで Save を押すか、Generator->Save を選ぶことで、修正された定義を保存します（もしシンタックスエラーがあれば、正しく定義を修正しているか確認してください）。このジェネレータは、ジェネレータエディタ画面の左上に現れます。このジェネレータが選ばれることを確認し、ツールバーの Generate を押すか、Generator->Generate を選択してください。ジェネレータを実行するグラフの選択が促されます。“Simpson”を選び、OK を押します。MetaEdit+は、図 4-16 のような出力を生成します。

図 4-16



ジェネレータが行ったことは、グラフ内の全てのオブジェクトを通してループし（foreach .()）、特定の属性 - この場合 First name をプリントし（id）、各行の末尾にキャリッジリターン（改行）を付加します（newline）。これはジェネレータの基本構造です：ダイアグラムに示したコンセプトを通してループし、それらから情報を抽出して、それをプリントしています。

家系図のメンバーの名前をプリントすることは、自身のジェネレータを書くもっともな理由ではありません。しかしリスト形式で、家系やそれらの親子関係のメンバーを出力するジェネレータは、いろいろな目的に対して役に立ちます。ジェネレータを修正しましょう。行うべきことは、彼 or 彼女がリレーションシップで Parent or Child ロール（or 両方）のどれなのか、各人物をチェックすることです。そのようなロールが存在するなら、リレーションシップの逆の立場から情報を出力します。下記のように、ジェネレータは修正されます（コメントテキスト（/\*...\*/）を入力する必要はありません）。

-----  
Report 'My HTML generator'

foreach .()

```

{
    /* Print out name of person */
    'Person: ' id newline

    /* Print out the parents */
    ' Parents: '
    do ~Child>Family~Parent.()
    {
        id ' '
    }
    newline

    /* Print out the children */
    ' Children: '
    do ~Parent>Family~Child.()
    {
        id ' '
    }
    newline
}

```

endreport

再度、ジェネレータを保存（ツールバーの Save か Generator->Save）し、実行します（Generate か Generator->Generate）。彼らの親や子供に関する情報を持った人物のリストを得るはずですが。現在のオブジェクトのロール（“~Parent” や “~Child”）やリレーションシップ（“>Family”）は、“do” ループによって、それに関係するオブジェクトを探索します。

次に、これを大きく変更します。家系図ダイアグラムの絵と各人物の情報を使って、ウェブページを形成するジェネレータを書いてみます。ダイアグラムで、両親や子供に関する情報を含めて、彼 or 彼女の情報を得るために、誰か人物をクリックできるようにします。さらに、両親と子供は、彼らのそれぞれの人物情報へリンクされます。ジェネレータは、ブラウザで自動的にページを開きます。これら全ての定義は、以下のようになります。

-----  
Report 'My HTML generator'

```

/* Open a HTML file for output */
filename
subreport '_default directory' run
:Family name; '.html'
write

/* Create HTML header tags */
'<html>' newline
'<head><title>The ' :Family name;
' Family Tree</title></head>' newline

/* Create HTML document body */
'<body>' newline
'<h1>The ' :Family name;
' Family Tree</h1>'

/* Create picture (with image map) */
filename
subreport '_default directory' run
oid '.gif'
print
'<img src="" oid
'.gif" border=0 usemap="#"
subreport '_default directory' run
oid '.gif">' newline
'<br><br><hr>' newline

/* Generate Person entries */
foreach .()
{
    /* The name */
    '<a name=' oid '><h3>' id ' '
    :Family name; '</h3>' newline

    /* Date of birth */
    'Date of birth: ' :Date of birth;
    '<br><br>' newline
}

```

```

/* Date of death */
'Date of death: ' :Date of death;
'<br><br>' newline

/* The parents */
'Parents: '
do ~Child>Family~Parent.()
{
    ' <a href=#' oid '>' id
    ' ' :Family name; '</a> '
}
newline '<br><br>' newline

/* The children */
'Children: '
do ~Parent>Family~Child.()
{
    ' <a href=#' oid '>' id
    ' ' :Family name; '</a> '
}
newline '<br><br><hr>' newline
}

/* Create HTML footer */
'</body>' newline
'</html>' newline

/* Close the output file */
close

/* Launch the web browser */
external ''
subreport '_default directory' run
:Family name;'.html''
execute

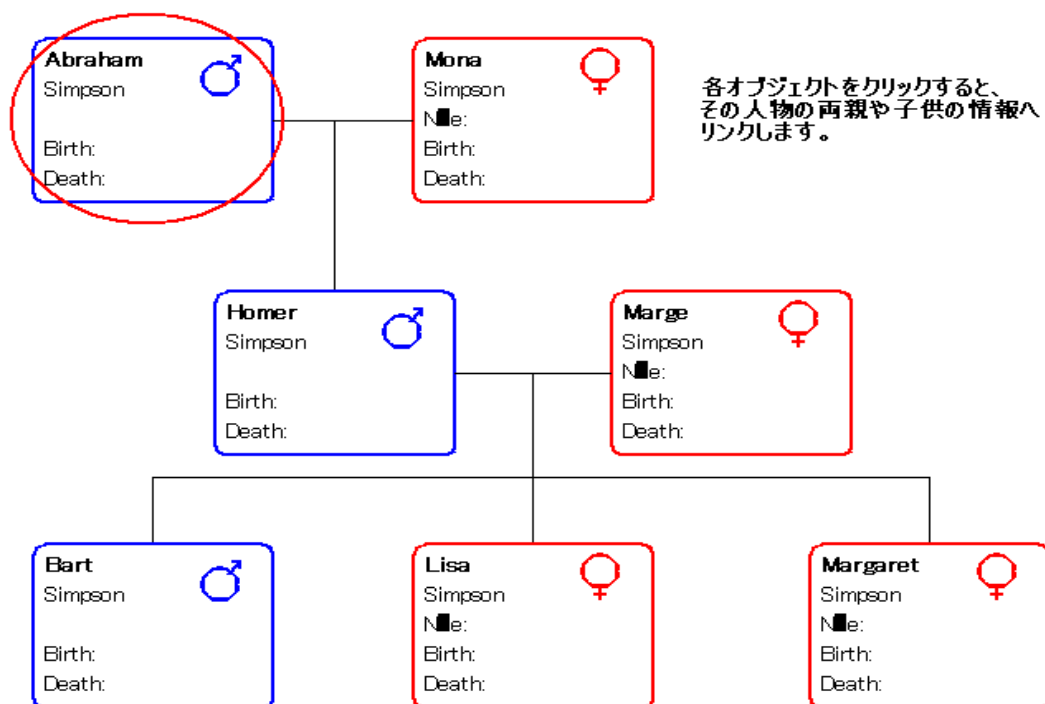
endreport

```

再度、ジェネレータを保存して、実行します。生成されたウェブページを試してみましょう。ご覧のとおり、基本的に、前にあらかじめ定義されたジェネレータで生成したウェブページよりシンプルなバージョンです。しかしながら、1つ重要な違いがあります。新しいウェブページは、前のものよりも良いドメインコンセプトを明示しています。そして、これはまさしく第一にほしかったものです。一般的に、あらかじめ定義されたジェネレータは役に立ちますが、自身のドメインスペシフィックジェネレータを作成することは、自身の Domain-Specific Modeling 言語の中から、最大限の利益を得る秘訣です。

図 4-17

## The Simpson Family Tree



オブジェクト “Abraham” をクリック

---

## Abraham Simpson

Date of birth:

Date of death:

Parents:

Children: [Homer Simpson](#)

---

## Bart Simpson

Date of birth:

Date of death:

Parents: [Marge Simpson](#) [Homer Simpson](#)

Children:

---

#### 4.4. 家系図モデリング言語を改良するための更なるアイデア

家系図モデリング言語の第2の開発サイクルを完了しました。たとえこれが、実際のモデリングで代用できる最初のバージョンであっても、モデリング言語が進化し続けることは極めて普通です。家系図の開発で、次に最もありそうな目標は、Family リレーションシップです。このチュートリアル内でさらにそれを掘り下げつつもりはありませんが、あなた自身でそれを試みたい場合に備えて、いくつかの思考材料をここに示します。

現在、家系図モデリング言語の Family リレーションシップは、意味論的に極めてルーズです。これは、利点であり不利点でもあります。Male と Female が子供を持つ家系を築くという状況に関しては、同じリレーションシップタイプで、簡単に全ての状況をモデル化できます。ここでは、自然な血統に重点を置いています。しかしながら、多くの場合、法的存在として家族の血統は関係しません。例えば、両親が既婚中である家族から、両親が既婚中ではない家族を区別したいかもしれません。これは、Male と Female に関係するのと同様に、Family リレーションシップの限定を必要とするでしょう。

他にも、養子の関係があるでしょう。血統関係だけを示す家系図で、彼らは示せません。しかし、法的な家系図では、彼らは重要なロールを持つことになります（例えば、ローマ皇帝の家族において、養子は王位の継承者になれるなど）。おそらく、これを行うための最も良い方法は、実際の子や養子として Child 口

ールを専門にすることです。

さらなる段階としては、修正することは必要ですが、他の問題領域に家系図モデリング言語を適用することです。例えば、動物血統。動物（例えば、馬や犬）を育てる人々は、動物に関する情報を保管する必要があります。彼らはまた、血統線や世代を通じて動物のいくつかの属性を追跡して分析したい。これは、家系図に含まれる全く新しい属性を必要とします。（または、すでに血統線モデリング言語があるか？）

以上、モデリング言語の構築に必要なこと、そしてどのようにその問題領域（problem domain）を基にし実装するかは、言語の利用意図、などをみてきました。モデリング言語を作成するには、必要となるドメインのコンセプトを見つけ出すことがとても大切です。そして厄介なのは、進化させて運用し続けることです。これらに対し、MetaEdit+ を用いれば、柔軟なメタモデリング環境が提供されるので、アイデアを自由に実装できるようになります。

## 5. 終わりに

MetaEdit+を適応する方法例をざっと通してみました（まず、自身のドメインスペシフィックモデリング言語を作成するためのツールとして、それからその言語をサポートするツールとして）。この後の最終的な議題は、あなたの組織において、ここで提示されるアプローチを採用することで、どういう利点を期待するのか？ MetaCase 社や顧客の経験によれば、MetaCase 技術を採用することで、3つの重要な利点があります。

### 1. 10倍の生産性向上。

従来ソフトウェア開発は、ある一連のコンセプトから他のコンセプトへの、エラーの基となる複数のマッピングを必要としました（要求仕様書内のドメインコンセプトから、デザインコンセプトへ（UMLなども）、そしてデザインのコンセプトから、プログラミング言語のコンセプトへ（C++など））。これは、何度となく同じ問題を解決することに相当します。Domain-Specific Modeling 言語では、ピュアなドメインコンセプトへの取組みのみで問題は解決されます。最終製品が、これらの高水準な仕様書から自動的に生成されるので、エラーを起こしやすいコンセプト間のマッピングの必要はありません。研究では、この種のアプローチは通常、現在の手順より5~10倍速いことを証明しています。

### 2. 変更に対する、より良い柔軟性と応答性。

コードではなくデザインにフォーカスすることで、変更の要求に対する対応が速くできるようになる。ドメインコンセプトレベルで変更ことはより簡単であり、そして1つのドメインコンセプトから、複数のプラットフォームや製品バリエーションに対してコードを生成させることができるようになる。

### 3. 全開発チームで共有される専門知識。

チーム内で各開発者のドメインの知識が欠如しているため、ドメインのアイデアをコードに落とすことが、問題となっています。新入りの開発者が一人前になるまでに、多くの時間が掛かります。また、

より上級の開発者でさえ、頻繁にドメインのエキスパートと相談する必要があります。このチュートリアルで示されるアプローチでは、エキスパートが、ドメインコンセプトやコードへのマッピングを定義しています。そして開発者は、ルールによって導かれるコンセプトを使ってモデルを作成し、コードは自動的に生成されます。コードは、熟練の開発者の専門知識で生成されているので、品質は飛躍的に向上します。

このチュートリアルの内容や、上記に挙げたリスト以外にも多くの情報があります。このアプローチを導入することについてもっと学び、Domain-Specific Modeling言語のより広範囲な例を見るには、Watchモデリング言語例のチュートリアルを推奨します。また、この技術を利用したノウハウに興味がある場合は、ウェブページ( <http://www.metacase.com> )に、事例紹介、ユーザリファレンス、ホワイトペーパーやFAQがあります。MetaCase社の技術についての質問、または組織が利益を得るための方法について考えている場合など、ご遠慮なく連絡くださると幸いです。



富士設備工業株式会社 電子機器事業部

〒591-8025 大阪府堺市北区長曾根町1928-1

Tel: 072-252-2128 [www.fuji-setsu.co.jp](http://www.fuji-setsu.co.jp)