
1. モデルデータジェネレーターとは

CommonMP (Ver1.0) を用いて、特定の河川に対して その振る舞いを記述する「水理／水文モデル」を作成する場合、地域（河川）毎にそれぞれ「演算要素」を配置し、それらを相互に接続する事で、特定の河川モデルを製作します。 この場合、CommonMP では、モデル管理画面上で 各要素を配置／接続し、また各要素のプロパティ画面からパラメータ入力によって モデルを開発する事が可能です。 しかしながら、現実には、河川毎に、配置されたこれら演算要素に対する、川幅、傾斜、その他の地形情報、植生等、調整すべきパラメータの数は膨大なものになります。 そこで、CommonMP では、半自動的に 要素モデルの接続、パラメータの調整を行う為のフレームワークを準備しております。 これを、CommonMP では モデルデータジェネレーターと呼びます。

実際に 如何なる演算要素を、どの様に接続し、それらの演算要素にはどんなパラメータを設定するかは、 使用する演算要素モデルや、作成しようとするモデルによって異なる為、CommonMP として、汎用的に使用できる モデルを自動生成するツール本体を提供する現実的ではありません。

そこで、 モデルを自動生成するツールは、モデル開発や研究者等によって開発・作成される事になりますが、 それを、各開発者／研究者が独自の仕様で作成すると、せっかく開発したツールが 一般に広く普及する事が出来ません。 そこで、CommonMP は、モデルを自動（半自動）的に生成するツール（以下モデルジェネレーションツールと呼びます）を 組み込むフレームワークを提供し、 各開発者／研究者は そのフレームワーク上で開発した モデルジェネレーションツールを提供することで、CommonMP を利用して 自らが開発したモデルジェネレーターを広く一般に 使用して頂く事が可能となります。

CommonMP とモデルジェネレーションツールの関係を 図 1. 1 に示します。

モデルジェネレーションツールとしては 主に 下記が必要となります。

1) ネットワーク構築するツール

どの演算要素をどのように配置し、また、各演算要素をどのように接続するかを決定する 言わばモデル構築の核となる部分です。

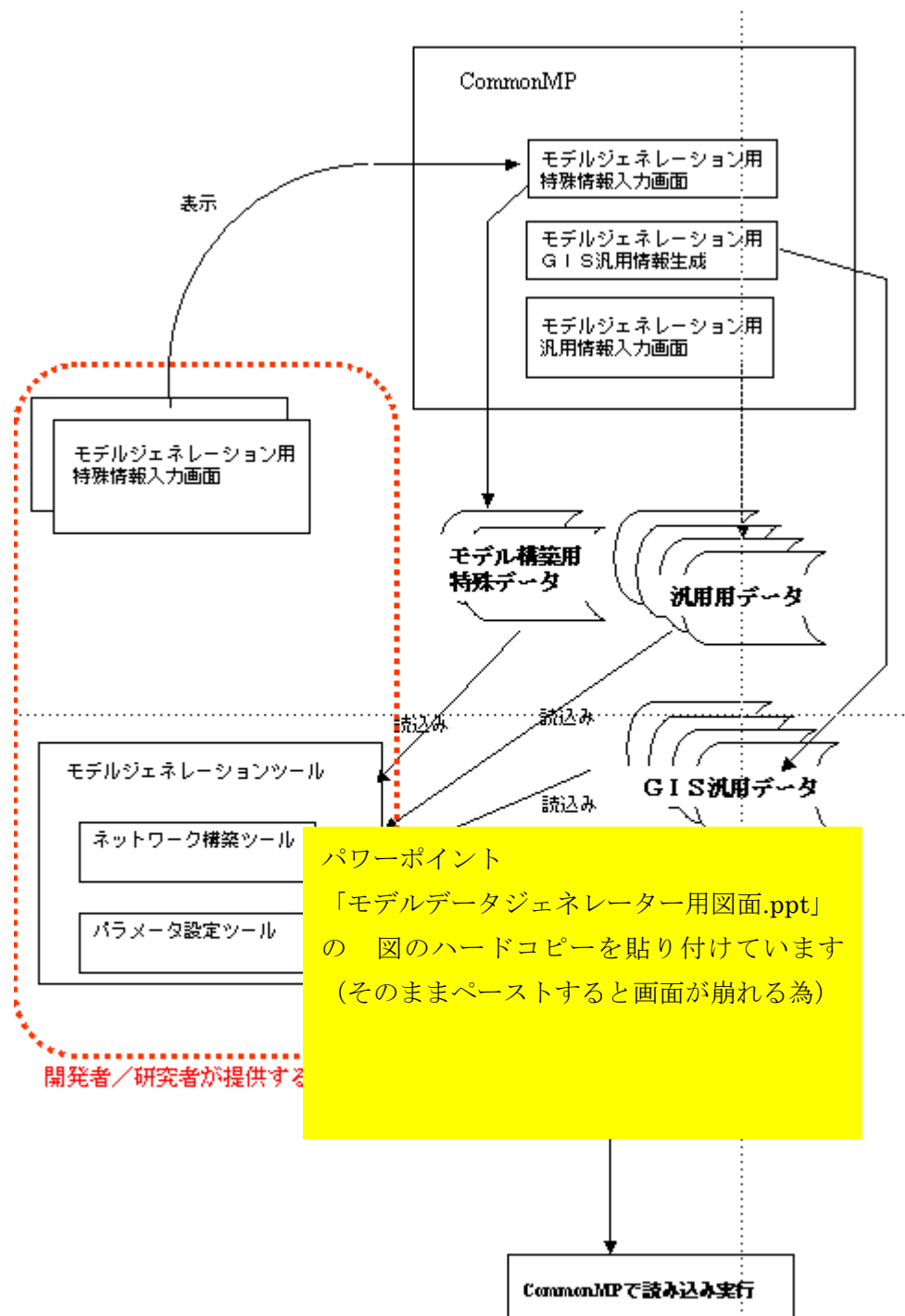
2) パラメータ設定ツール

配置された演算要素毎に 動作条件を設定していきます。

3) 独自情報入力画面

モデル生成の際、CommonMP が標準で提供する画面から出力された情報では不足する場合、独自の画面を提供し、その画面から必要な情報を設定できます。

（CommonMP 標準の画面で事足りる場合には、不要です。）



CommonMP では、

モデルデータジェネレーター関連サンプルとして

¥CommonMP¥Source¥HYMCO¥OptionImpl¥McModelGeneratorSample¥下に

- ・ネットワーク構築するツール
- ・パラメータ設定ツール

¥CommonMP¥Source¥HYMCO¥OptionImpl¥McModelGeneratorScreenSample¥下に

- ・独自情報入力画面

コーディングサンプルを準備しております。

以後、説明はこれらのコーディングサンプルを元に説明を行います。

尚、これらのサンプルは、モデル開発者向けの

CommonMP¥Source¥HYMCO¥OptionImpl¥ModelDeveloperExpressEdition¥下の

TestModelDeveloperMainExp.sln

または、

CommonMP¥Source¥HYMCO¥OptionImpl¥ModelDeveloperStandardEdition¥下の

TestModelDeveloperMainStd.sln

を起動して マイクロソフト社のビジュアルスタジオを立ち上げてた後、

デバッグモードで動作させる事が出来ます。

・ 2 モデルデータジェネレーター の提供方法

通常 演算要素を作成する場合 演算要素開発者／研究者は、

- ・ 演算要素を生成するファクトリ ([McBasicModelFactoryBase 派生クラス](#))
- ・ 演算要素固有プロパティ画面を生成するファクトリ ([McPropertyScreenFactoryBase 派生クラス](#))
(但し固有プロパティ画面が必要な場合のみ)

を提供する必要があります。

演算要素開発者／研究者が、モデルデータジェネレーター関連クラスを提供しようとする場合にも上記クラス ([McBasicModelFactoryBase 派生クラス](#)、[McPropertyScreenFactoryBase 派生クラス](#)) を使用します。

但し、全ての演算要素開発者／研究者が、モデルジェネレーター関連クラスを開発し、提供するとは限らない為、親クラスでは、virtual メソッドとして 何もしない下記メソッドを実装しております。このため、派生クラス側でそれらのメソッドを **override** しなければ、モデルデータジェネレーター関連クラスは生成しません。

図 2. 1 は、演算要素を生成するファクトリクラス上で モデルデータジェネレーター用に
オーバーライド すべきメソッドを示します。

また、図 2. 2 は、演算要素固有プロパティ画面を生成するファクトリ上で、モデルデータジェネ
レーター用にオーバーライドすべきメソッドを示します。

<注>

モデルと画面は、別々の DLL で供給する為に 敢えてファクトリも分離してある。
これは、例えばサーバー等、画面を使用しない環境で モデルを使用したい時、
同一 DLL 内に画面関連の処理まで含めると、画面を表示しないにも関わらず、
画面表示用の ソフトウェアもインストールしなければならない事を防ぐ為である。
(将来拡張への配慮)

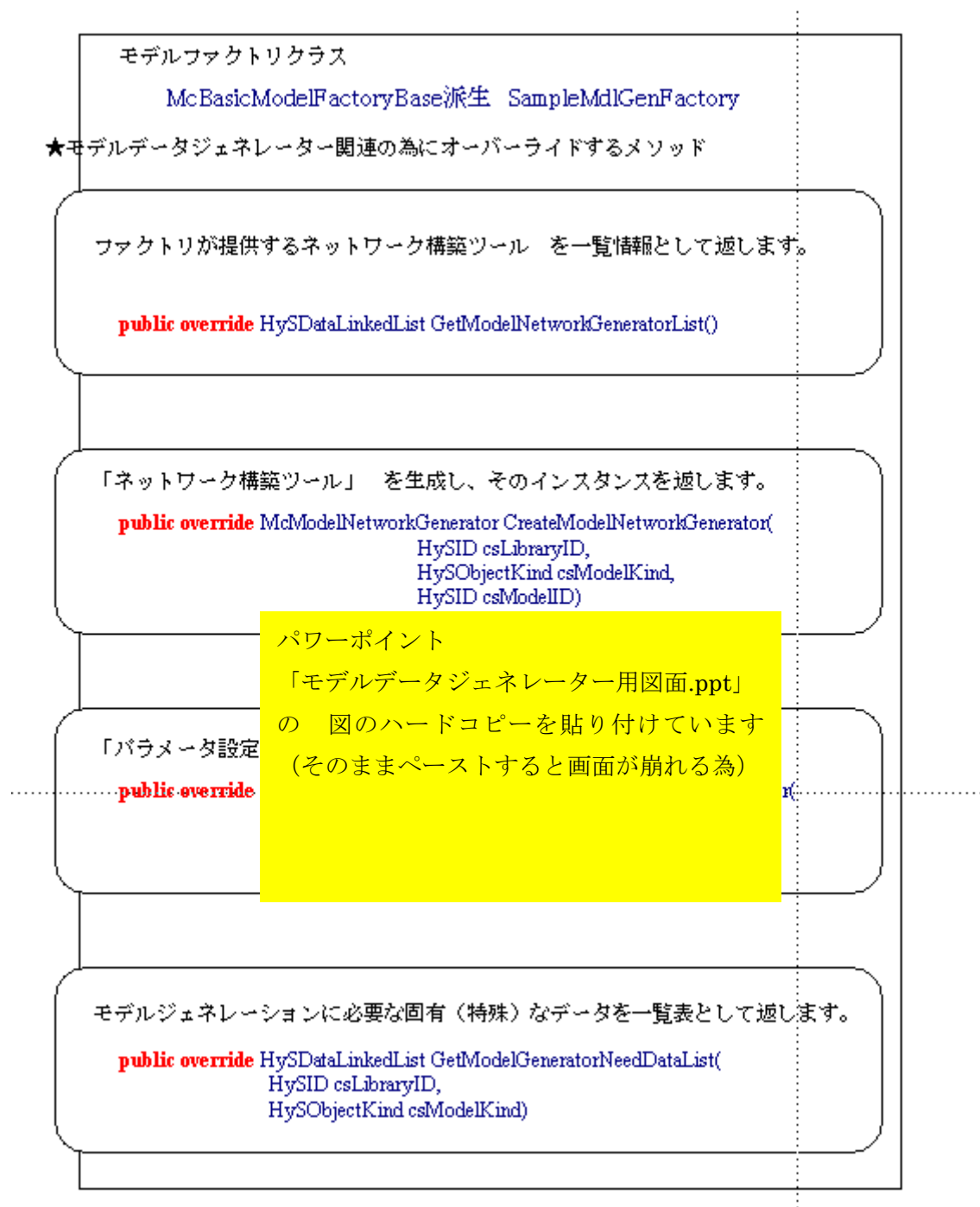


図 2. 1

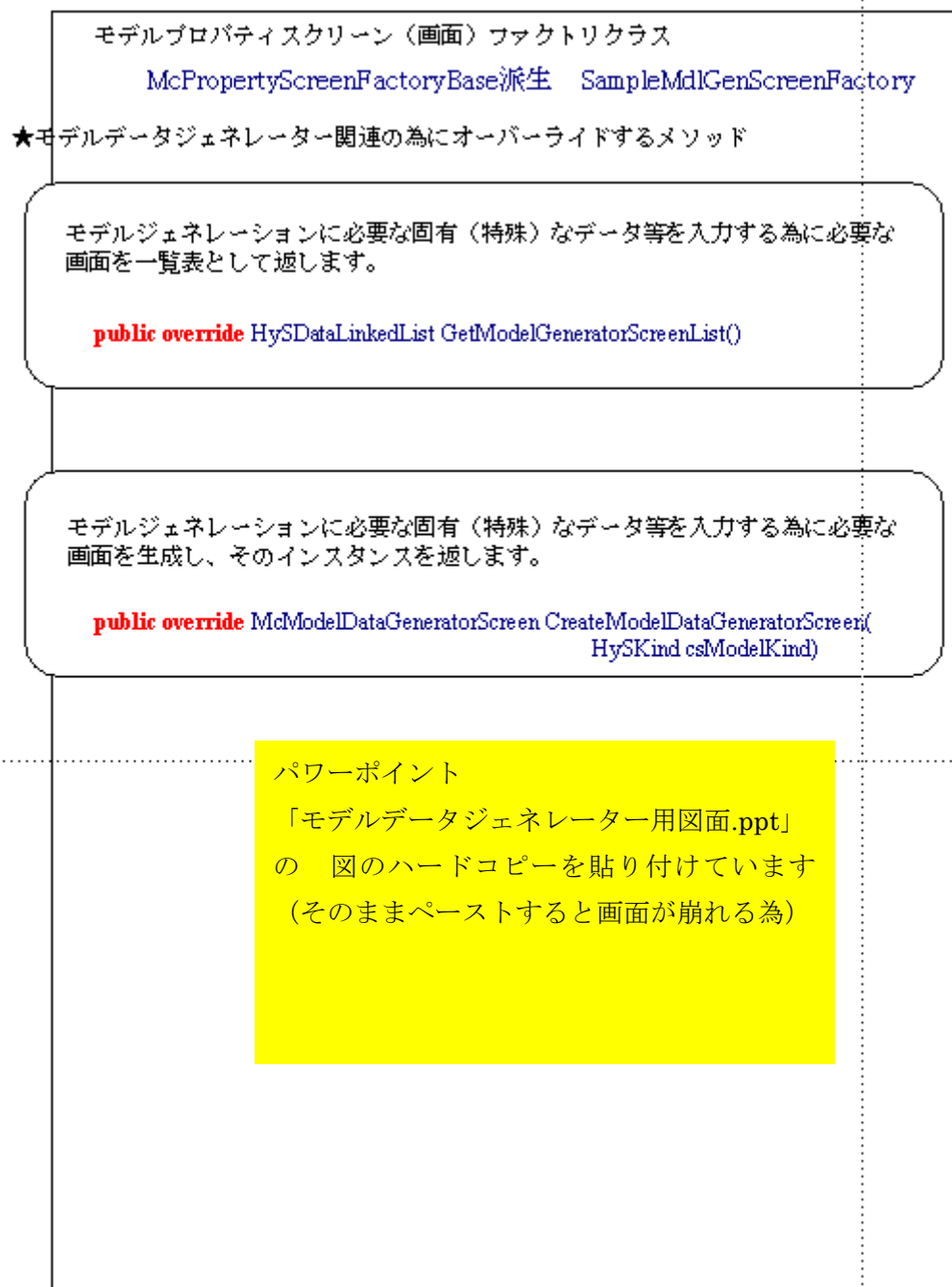


図 2. 2

・ 3 モデルデータジェネレーター の動作概要

CommonMP 上では、モデルデータジェネレーターの制御は 図 3. 1 に示すモデルデータ生成制御画面から行います。

画面の具体的操作説明は、画面からの「？」ボタン押下によるヘルプを参照して下さい。

本章では、画面とモデルデータジェネレーター内部処理の関連について述べます。

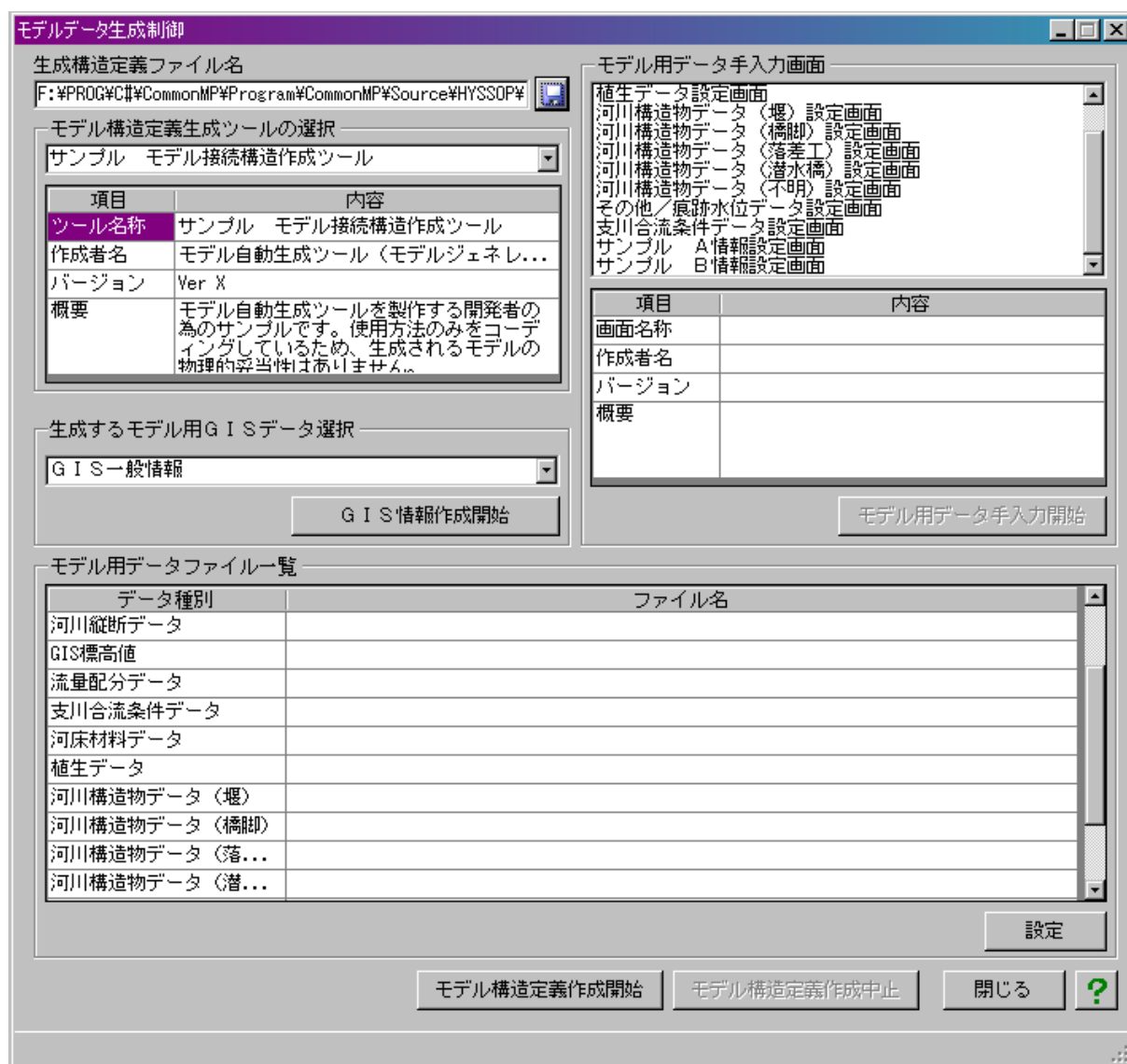


図 3. 1 モデル半自動生成用制御画面（モデルデータ生成制御画面）

図 3. 2に モデル半自動生成用制御画面（モデルデータ生成制御画面）とモデルジェネレーションツールの 動作の関連を模式的に示します。

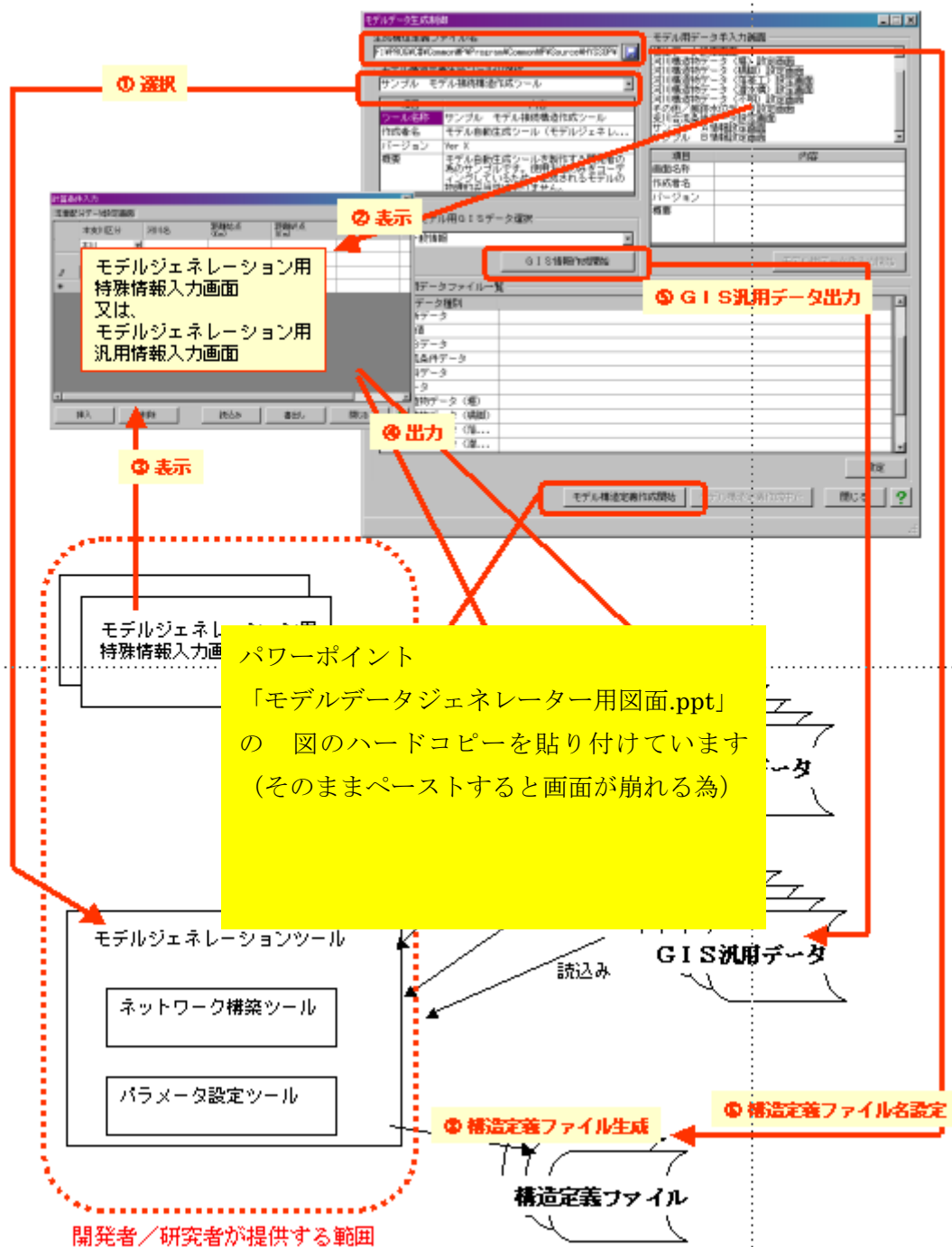


図 3. 2 モデル半自動生成用制御画面とモデルジェネレーションツールの関係概要

図3. 2 において オペレーターは モデルジェネレーションツールを使用し 以下に示す様な操作によって、モデルを構築（半自動構築）します。

1. オペレーターは、モデル開発者／研究者が提供した モデルジェネレーションツールの中から、今回使用したいツールを選択します。（図中①） これで、どのようなモデルを構築するのかを決定します。

2. 選択したモデルジェネレーションツールが必要とするデータを生成します。（図中②、③、④） モデル固有の情報が必要な場合には、 モデル開発者／研究者が提供した画面を使用して情報を入力します（②） また、汎用画面を使用する場合には、システムが提供する画面を使用して必要情報の入力を行います（③）。 G I Sが提供する汎用データ出力機能を利用する為には、「G I S情報作成開始」を押下して G I S関連汎用データを作成します（⑤）。

3. これらの情報を元に モデルジェネレーションツールに対して、 構造定義ファイル生成を指示します（⑦）。 （但し、指示前に 出力する構造定義ファイル名称を設定しておく必要があります（⑥））

4. 指示を受けたモデルジェネレーションツールは、既に作成されている各種データファイルを読み込んで、モデルの構造定義ファイルを作成します（⑧）。

作成される構造定義ファイル（例として SampleMdl.xml とした場合）は、下記の構造定義ファイルが生成されます。

- a. モデルの接続情報本体ファイル（SampleMdl.xml）
接続情報を記述
- b. 演算要素プロパティ情報ファイル（SampleMdlProperty.xml）
各演算要素別のプロパティ情報を記述
- c. 演算要素初期化情報ファイル（SampleMdlInitial.xml）
各演算要素別の初期化情報を記述
- d. CUI で動作する際の プロジェクト情報（SampleMdlProjectFile.xml）

上記ファイルは、CommonMP の メニュー

「ファイル」－「構造定義ファイル」－「書き出し（詳細情報付き：分割）」によって構造定義ファイルを出力した場合と同様の形式です。

以上述べてきた「モデルモデルジェネレーションツール」、「モデルジェネレーション用特殊情報入力画面」は、2章で述べた ファクトリークラスによって供給され、CommonMP 内に取り込まれて モデル半自動生成用制御画面（モデルデータ生成制御画面）上で動作します。

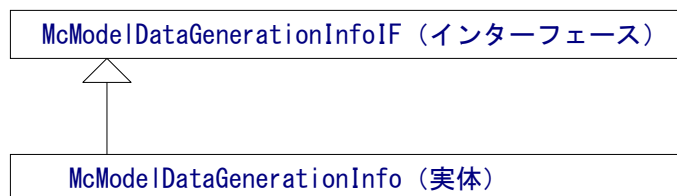
・ 4 モデルジェネレーションツール

モデルジェネレーションツールは、
モデルネットワークを構築するツール（`McModelNetworkGeneratorBase` 派生クラス）と パラメータ設定ツール（`McModelParameterGeneratorBase` 派生クラス）から構成されます。

これらのツールに対して、モデルデータジェネレーション用に必要な情報ファイルは、メソッドの引数として渡されます。モデルデータジェネレーション用に必要な情報ファイル群を格納するクラスとしては、

モデルデータジェネレータ作成条件格納クラス（`McModelDataGenerationInfo`）を使用します。

モデルジェネレータ作成条件格納クラス派生関係



モデルデータジェネレータ作成条件格納クラス（`McModelDataGenerationInfo`）は、モデル半自動生成用制御画面（モデルデータ生成制御画面）でオペレーターが設定した各種ファイルをハッシュテーブルとして管理しています。

必要とする情報（ファイル名：フルパス付き）は

```
public virtual bool GetInfo(HySObjectKind csKey, ref HySString csParameterFileName)
```

メソッドを用いて取得します。

データ種別を表すキーワード（`csKey`）を与えると それに対応するデータファイル名（`csParameterFileName`）が取得できます。

データ種別を表すキーワードは

G I S 関連であれば

`CommonMP.HYSSOP.CoreImpl.HSGIS.HySGISDataKindDefine` クラス内に定義されています。

（例：標高値データ： `KIND_GIS_DRAINAGE_DIRECTION_MATRIX`）

また、汎用データであれば

`CommonMP.HYMC0.CoreOptions.McViewer.MdGenScreenCommonLIB.McMdGenCommonScreenDefine` クラス内に定義されています。

（例：河川構造物データ（堰）： `MODEL_GENERATOR_DATA_KIND_4`）

固有（独自）データの場合には、固有画面側で

```
public virtual void SetInfo(HySObjectKind csKey, HySString csParameterFileName)
```

メソッドにより、キー情報と共にモデルデータジェネレータ作成条件格納クラス (McModelDataGenerationInfo) に設定しておくことで、GIS 情報や、汎用情報と同様の方法でその値を取得する事ができます。

尚、独自情報を使用する場合には、演算要素を生成するファクトリクラス (McBasicModelFactoryBase 派生クラス) の、

```
public override HySDataLinkedList GetModelGeneratorNeedDataList (HySID csLibraryID,
HySObjectKind csModelKind) メソッドをオーバーライドしておく必要があります。
```

この情報は モデル半自動生成用制御画面 (モデルデータ生成制御画面) が表示される時に、予め通知されます。その中の 固有 (独自) データのキー情報を元に、モデル半自動生成用制御画面 (モデルデータ生成制御画面) は、固有 (独自) 画面の表示制御を行います。

・ 4. 1 モデルネットワークを構築するツール (McModelNetworkGeneratorBase 派生クラス)

さて、モデルネットワークを構築するツール (McModelNetworkGeneratorBase 派生クラス) は、GIS 汎用データファイル、モデル構築用特殊データファイル、汎用データ等のファイルを読み込んで、それらの情報を元に 自らが提供する演算要素モデル・あるいは、他者が提供する演算要素モデルを配置し、それらを接続するツールです。構築されたモデルは、構造定義ファイルとして保存されます。実際の 構造定義ファイル (XML 形式) への書出しは、親クラス側で行う為、派生クラスでは、記述すべき情報を XML のノードクラス (HySXmlNode) へ設定することで、構造定義ファイルを生成できます。

本クラスで、モデル開発者／研究者がオーバーライドすべきメソッドを下記に述べます。

1. 演算要素モデルネットワーク生成前のチェック

```
public override bool CheckBefoModelGeneration(ref McModelDataGenerationInfoIF csMDGInfo)
```

引数で与えられた、モデルジェネレータ作成条件格納クラスを元に、必要な情報が全て設定されているか否かのチェックを行います。

情報に不足があれば、リターン値 : false を返します。

2. 演算システム全体を制御するプロパティ情報の生成

```
public override HySXmlNode CreateSystemPorperty(
    ref McModelDataGenerationInfoIF csMDGInfo,
    ref HySXmlNode csXmlParent,
    ref HySXmlWriter csXmlWriter)
```

モデル系全体を制御するプロパティ情報

例えば、 全体を 同期型で動作させるか／非同期型で動作させるか 等

の設定を行います。

(コーディング例)

```
public override HyXmlNode CreateSystemPorperty(  
    ref McModelDataGenerationInfoIF csMDGInfo,  
    ref HyXmlNode csXmlParent,  
    ref HySXmlWriter csXmlWriter)  
{  
    // 構造定義ファイル  
    HyXmlNode csRtnNode = csXmlWriter.CreateElement(McDefine.xml_PROPERTY );  
    // 非同期型  
    csRtnNode.SetAttribute(McDefine.xml_DATAFUSIONTIMING, "ASYNCHRONOUS");  
    // 系全体を進める時間間隔  
    csRtnNode.SetAttribute(McDefine.xml_TIMESTEP, "60");  
    return csRtnNode;  
}
```

上記設定により、構造定義ファイルには

```
<Property DataFusionTiming="ASYNCHRONOUS" TimeStep="60" />
```

個の様なプロパティタグが 生成されます。

3. 演算要素／接続の作成

```
public override bool CreateModels(  
    ref McModelDataGenerationInfoIF csMDGInfo,  
    ref HyXmlNode csXmlParent,  
    ref HySXmlWriter csXmlWriter)
```

引数で与えられた、モデルジェネレータ作成条件格納クラスから取得された各データファイルの内容を元に 必要な演算要素を生成し、それらの結ぶ接続線を生成します。

本部分は、モデル開発者／研究者が自由に 自らの開発したアルゴリズムに従って要素を 現実
に即したモデルを配置する部分です。

親クラスは、モデル開発者／研究者が 構造定義ファイルを直接認識しなくてもそれらの接続が
行える様に ツールメソッドを準備しております。

- ① **class** McElementTagInf 演算要素の識別子、種別等を格納するクラスです。
- ② **class** McPortTagInf 演算要素へ接続される端子情報を格納するクラスです。
- ③ **class** ConnectionTagInf 接続情報(送信側端子と受信側端子の ID 等)を格納するクラスです。
- ④ **class** ConnectionPropertyInf 接続線内部を流れる情報を格納するクラスです。

モデル開発者／研究者は、上記クラスに必要な情報を設定し 下記に示す

HyXmlNode CreateElementTag ()

bool AddPortTag ()

HyXmlNode CreatePortTag ()

HyXmlNode CreateConnectionTag ()

HyXmlNode CreateConnectionPropertyTag ()

等のメソッドを使用して、XML ノードを生成し、モデル全体の構造定義 XML ノードに追加します。詳しい 使用方法は、 サンプル

CommonMP.HYMC0.OptionImpl.ModelGeneratorSample.SampleModelNetworkGenerator

クラスの

```
public override bool CreateModels(  
    ref McModelDataGenerationInfoIF csMDGInfo,  
    ref HyXmlNode csXmlParent,  
    ref HySXMLWriter csXmlWriter)
```

メソッド内をご覧ください。

4. 自らが開発していないモデル（他者モデル）を使用する場合、そのモデルのプロパティ情報等を設定するためのクラスの生成

```
public override McModelParameterGenerator CreateExternalModelParameterGenerator(  
    HySID csLibraryID, HySObjectKind csModelKind, HySID csModelID)
```

自らが開発したモデルに対するパラメータ設定等は、後述する 自演算要素専用のパラメータ設定ツール（McModelParameterGeneratorBase 派生クラス）を使用します。

しかしながら、他者が作成したモデルに対するパラメータ設定等を行いたい場合には、他者が パラメータ設定ツール（McModelParameterGeneratorBase 派生クラス）を提供しているか否か不明である為、 使用したい他者演算要素用の パラメータ設定ツール（McModelParameterGeneratorBase 派生クラス）を自前で準備する必要があります。 このメソッドは、その為に使用します。

サンプルでは、

コーディングサンプルとしての

CommonMP.HYMC0.OptionImpl.McSampleModelForDeveloper.McSampleKinematicWave モデルと、

画面へのモニター出力用演算要素

CommonMP.HYMC0.CoreImpl.Model.McLineGraphScreen

を使用する場合を例として記述しております。

・ 4. 2 パラメータ設定ツール（McModelParameterGeneratorBase 派生クラス）

モデルジェネレーションツールのもう一つのツールである パラメータ設定ツール（McModelParameterGeneratorBase 派生クラス）は、先に述べた、モデルネットワークを構築するツール（McModelNetworkGeneratorBase 派生クラス）が配置した各要素に 演算条件や、初期条件を設定するツールです。 設定した値は、構造定義ファイルに出力されます。

モデルネットワークを構築するツールは、各要素を配置する際に、それぞれ ID を付与するため、パラメータ設定ツールは、その ID を調べて、各演算要素を区別し、演算要素毎にパラメーターを変更することが可能です。

本クラスで、モデル開発者／研究者がオーバーライドすべきメソッドを下記に述べます。

1. プロパティ情報設定

```
protected override bool CreateModelPropertyDetailData(  
    ref McModelDataGenerationInfoIF csMDGInfo,  
    ref HyXmlNode csPropertyInfoParent,  
    ref HySXmlWriter csXmlWriter)
```

引数で与えられた `McModelDataGenerationInfoIF csMDGInfo` と、
自演算要素 ID (`HySID csElmID = (HySID) this.GetID();` として取得可能) によって プロパティ情報を設定します。

設定した情報は、XML の親ノードである `csPropertyInfoParent` (引数で与えられる) に追加することで、親クラス側の処理により、XML ファイル出力されます。

2. 初期化情報設定

```
protected override bool CreateInitialDetailData(  
    ref McModelDataGenerationInfoIF csMDGInfo,  
    ref HyXmlNode csInitialInfoParent,  
    ref HySXmlWriter csXmlWriter)
```

引数で与えられた `McModelDataGenerationInfoIF csMDGInfo` と、
自演算要素 ID (`HySID csElmID = (HySID) this.GetID();` として取得可能) によって 初期化情報を設定します。

設定した情報は、XML の親ノードである `csInitialInfoParent` (引数で与えられる) に追加することで、親クラス側の処理により、XML ファイル出力されます。

・ 5 モデルジェネレーション用特殊情報入力画面

`CommonMP` は、モデルデータジェネレーションに必要なと思われる汎用データを生成することが出来ますが、モデルによっては、それら情報では不足する事が考えられます。

この場合、それらの情報を作成するために、`EXCEL` 等の `COTS` 品使用する、あるいは、独自に画面を作成してそれを使用する等の作業が考えられます。そこで、`CommonMP` では、独自に画面を作成する場合、それを、`CommonMP` から起動できる仕組みを設けました。

独自画面は、`CommonMP` が提供するモデルジェネレーター個別モデル用設定画面スクリーン親クラス (`McDotNetModelDataGeneratorScreenBase`) から派生して作成して下さい。

CommonMP の基本思想として、スクリーンクラスには、`.net` に依存した処理（`.net` が提供するコンポーネント）を入れないで、それらをモデルジェネレータ個別モデル用設定画面フォーム（`McDotNetModelDataGeneratorFormBase` の派生クラス）クラスに隠蔽する方法をとっています。

（理由：`.net` の仕様等が変更されても、他のクラスへの影響を極力局所化する為）

したがって、モデル開発者／研究者は モデル用設定画面スクリーンクラス（`McDotNetModelDataGeneratorScreenBase` から派生）を作成 内部で、モデルジェネレータ個別モデル用設定画面フォーム（`McDotNetModelDataGeneratorFormBase` から派生）クラスを生成して下さい。 画面操作処理は、モデルジェネレータ個別モデル用設定画面フォーム（`McDotNetModelDataGeneratorFormBase` から派生）クラス内に記述してください。

CommonMP として

`CommonMP¥Source¥HYMCO¥OptionImpl¥McModelGeneratorScreenSample¥下`に

モデルジェネレーション用特殊情報入力画面 作成用のサンプルプロジェクトを提供しています。 コーディングの詳細はそのソースを参照して下さい。